RocketSim: Benchmarking Reinforcement-Learning Hover Control of a Reusable Booster

Swarnendu Ghosh*
swarnendu.ghosh@fri.uni-lj.si
FRI, University of Ljubljana
Ljubljana, Slovenia

David Muhič
dm4695@student.uni-lj.si
FRI, University of Ljubljana
Ljubljana, Slovenia

Branko Šter
branko.ster@fri.uni-lj.si
FRI, University of Ljubljana
Ljubljana, Slovenia

Abstract

Reinforcement learning (RL) is increasingly proposed for controlling reusable rockets, but existing studies address landing, where touchdown gives the learner a clear goal. We ask what happens when the task is instead to *hover* at a fixed height, where no such goal event exists. We built RocketSim, a configurable rocket-control simulator in the Unity game engine, and trained proximal policy optimisation (PPO) to release a booster at 80 m and hold it at 30 m. The agent learned to stay upright, stay over the pad and fly until its tanks ran dry — but not to hold the commanded height: with nothing changed but the random seed, its final altitude ranged from the ground to the ceiling. Two causes are responsible. First, the reward ranks obedient flight highest — a hand-written controller that follows the command scores 23% more under it — but far from the target it barely changes with height, so PPO has no gradient to follow. Second, below 33.6 t the engine cannot throttle under its 39% floor, so for the last 55% of the burn no setting holds altitude. We tried to repair both: three reshaped objectives cut the mean final error from 36 to 6 m but still strand one seed in five, and a vehicle that can throttle down sinks instead. Hover has to be fixed in the reward and in the vehicle together. RocketSim and the analysis code will be released open source at https://swa-g.github.io/LANCER/ after publication.

Keywords

reinforcement learning, reusable launch vehicle, hover control, benchmark, reward shaping, Unity ML-Agents, PPO

1 Introduction

A reusable booster must keep its attitude and position while its mass falls, its engine can throttle only within limits, and a single mistake ends the flight. Reinforcement learning (RL) learns such control by trial and error in simulation [10]. Learning-based rocket guidance to date targets *landing* [3, 6], as does classical optimal guidance [1]: the flight ends at touchdown, and success is a clear, rewarded event. *Hover* — holding a fixed height over a pad — has no such event; the agent is rewarded continuously for being close, and nothing ends the flight except a failure. We found no published RL benchmark for booster hover, so our numbers cannot be ranked against earlier results. Instead we show why an approach that suits landing does not automatically transfer to hover.

RocketSim is a rocket-control simulator we built in the Unity game engine. It models the vehicle — body, engine, throttle and gimbal limits, fuel use, atmosphere and drag — defines tasks such as hover together with their reward functions and safety limits, and connects to Unity ML-Agents [4], whose PPO implementation [8] trains the agent. Every setting is recorded and frozen before a run starts, so any run can be reproduced exactly.

Contributions. Our own contributions are: (i) RocketSim itself — the vehicle model, the task and reward definitions, and the configuration workflow; (ii) a benchmark of PPO on the hover task, reporting training cost, flight behaviour and variability across random seeds (Sections 4–5); (iii) an analysis of why the agent does not hold altitude, separating a reward-design cause from a physical one (Section 6); and (iv) four attempted fixes — three reshaped objectives and a vehicle that can throttle down —

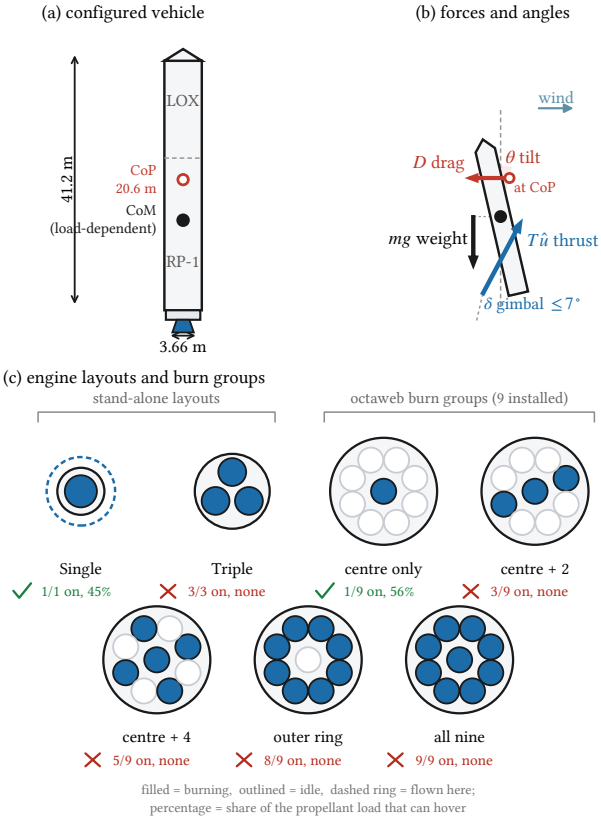


Figure 1: The configured vehicle. (a) Side elevation to scale, with the liquid-oxygen (LOX) and kerosene (RP-1) tanks, the centre of pressure and the load-dependent centre of mass. (b) Forces and angles entering Table 1. (c) Every engine layout and burn group RocketSim offers, active engines filled and the flown configuration ringed, with the share of the propellant load that can hover.

*Corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
Information Society 2026, Ljubljana, Slovenia
© 2026 Copyright held by the owner/author(s).
https://doi.org/10.70314/is.2026.skui.80

Table 1: Governing dynamics, as integrated at $\Delta t = 0.01$ s; standard models from [9, 2].

Quantity	Expression
Translation	$m\,\dot{\mathbf{v}} = T\hat{\mathbf{u}} + \mathbf{D} + m\mathbf{g}$
Thrust axis	$\hat{\mathbf{u}} = R(\theta)\,R(\delta)\,\hat{\mathbf{e}}_b$
Pressure ratio	$p(h) = e^{-h/H}$
Thrust	$T = \tau\,T_0\,[k_v + (1-k_v)\,p(h)]$
Specific impulse	$I_{sp}(h) = I_0\,[k_v + (1-k_v)\,p(h)]$
Propellant flow	$\dot{m} = -\,T/(I_{sp}(h)\,g_0)$
Air density	$\rho(h) = \rho_0\,e^{-h/H}$
Axial drag	$D_a = \frac{1}{2}\rho\,C_a A_a\,v_a\lvert v_a\rvert$
Lateral drag	$D_\ell = \frac{1}{2}\rho\,C_\ell A_\ell\,v_\ell^2$ at the CoP
Throttle limits	$\tau \in [\tau_{\min}, 1],\quad \lvert\dot{\tau}\rvert \le 5\ \mathrm{s}^{-1}$
Gimbal limits	$\lvert\delta\rvert \le 7^\circ,\quad \lvert\dot{\delta}\rvert \le 30^\circ\mathrm{s}^{-1}$
Hover feasibility	$m \ge m_{\mathrm{crit}} = \tau_{\min} T_{\max}(h)/g$
Hover endurance	$t_b = (I_{sp} g_0/g)\,\ln(m_0/m_{\mathrm{dry}})$

h is altitude, m and $\mathbf{v}$ mass and velocity, $g = 9.81\,\mathrm{m\,s^{-2}}$, g_0 standard gravity. Thrust acts along $\hat{\mathbf{u}}$: the body axis rotated by the tilt θ and gimbal δ (Figure 1b). τ is throttle, N the active engines, $T_{\max}(h) = NT_0[k_v + (1-k_v)p(h)]$; m_0, m_{dry} are launch and burnout mass. Values in Table 2.

Table 2: Component specification of the configured vehicle.

Quantity	Value
Body, length × diameter	41.2 × 3.66 m cylinder
Reference area, axial A_a / lateral A_ℓ	10.53 / 150.8 m^2
Drag coefficient, axial C_a / lateral C_ℓ	0.30 / 1.0
Centre of pressure, from the base	20.6 m
Mass: dry m_{dry} / propellant / launch m_0	17,390 / 40,000 / 57,390 kg
Sea-level thrust per engine T_0	845 kN, one engine installed
Sea-level specific impulse I_0, vacuum factor k_v	282 s, 1.10
Throttle τ, range and slew	0.39–1.00, $5\ \mathrm{s}^{-1}$
Gimbal δ, range and slew	$\pm 7^\circ$, $30^\circ\mathrm{s}^{-1}$
Sea-level density ρ_0, scale height H	1.225 $\mathrm{kg\,m^{-3}}$, 8500 m
Step, physics / decision	0.01 / 0.03 s

none of which solves the task (Section 7). The physics equations are standard textbook models [9, 2], and the learning algorithm and its implementation are those of ML-Agents. The outcome is a set of open challenges for future research rather than a solved task.

2 Vehicle Model and Specification

Table 1 lists the equations RocketSim integrates. They are standard models, not our own: the rocket equation and the altitude dependence of thrust and specific impulse follow Sutton and Biblarz [9], and the exponential atmosphere and quadratic drag law follow Anderson [2]. Table 2 gives the values we configured. The model omits structural flexibility, propellant slosh, plume–ground interaction and sensor noise, and is uncalibrated against flight data: it is an algorithmic simulation, not a trajectory predictor.

3 Task, Objective and Training

The vehicle is released at 80 m and must hold 30 m over the pad centre. With e_z the absolute vertical error, e_{xy} the planar error, $u = \frac{1}{2}(\cos\theta + 1)$, v the speed, ω the angular rate in °/s and c the control effort, the configured *Balanced* objective integrates

$$\frac{\mathrm{d}r}{\mathrm{d}t} = 0.12\,e^{-e_z/5} + 0.12\,e^{-e_{xy}/8} + 0.07\,u + 0.06\,e^{-v/4} + 0.04\,e^{-\omega/35} - 0.0015\,v - 0.0008\,\omega - 0.0025\,c, \quad (1)$$

with a cost of 0.25 per ignition and 10 for each safety terminal (Table 3). In words: every second the agent earns a small reward for being near the target height, near the pad, upright and calm, and pays a little for fast motion and control effort. There is no success event and no time limit, so a flight lasts until a safety limit is crossed. Four of the five positive terms pay out whenever the rocket is merely flying steadily — up to 0.29 reward per second — so surviving longer than 34.5 s already outweighs the crash penalty, whatever the height.

Table 3: Task, objective and training configuration, with measured cost.

Item	Value
Target / launch altitude	30 / 80 m
Launch dispersion	±5 m planar, ±2 m/s, ±1°
Safety terminals	ground 0.5 m, tilt 63.26°, planar 80 m, ceiling 200 m, fuel 0
Terminal / ignition cost	−10 each / −0.25
Policy interface	30 observations, 4 actions
PPO [8]	batch 4096, buffer 40,960
	learning rate 2×10^{-4} linear, clip 0.2
	GAE [7] λ 0.98, γ 0.9995
	6 epochs, horizon 1024, net 3 × 512
Environment steps	30,001,665, 16 parallel areas
Training time	7.04 h wall-clock, 1183 steps/s

4 Training Benchmark

In short: the agent quickly learns to survive and its score keeps rising, but its height error grows rather than shrinks. To repeat training over many seeds we use a faster three-degree-of-freedom reproduction of the task. It is not the full simulator: it matches the RocketSim run's endurance to 3%, scores 16% lower, and its seeds straddle that run's altitude error (dashed lines, Figure 2); Figures 2–3 and Table 4 come from it. Across five seeds the final score varies by less than ±10%, but the altitude error ranges from 24 to 105 m: runs that score the same hover four-fold apart, because the reward does not pin down where to hover.

5 Post-Training Evaluation

In short: after training the rocket flies steadily over the pad, but not at the commanded height. In test flights (Figure 3, Table 4) the vehicle stays within about 3 m of the pad and 2° of vertical, yet spends about a fifth of each flight within 5 m of the commanded height. Figure 3 shows five flights of one agent; *which* way an agent misses depends on its seed. Over 200 episodes each, one of the five agents reaches the ceiling in 179 of its episodes, three run into the ground, and one burns its fuel 107 m above the target. Lowering the throttle floor (variant C) removes the climb, but the vehicle then sinks instead.

6 Why the Altitude Objective Is Not Optimised

The Score Comes from Surviving, Not from Height. The agent's whole score can be predicted without the altitude term. Treating the flight as an ideal hover lasting until the tanks are empty (Table 1) reproduces the measured flight time to 0.01%, and Equation (1) evaluated on the measured behaviour reproduces the score to 0.8% — almost all of it from staying near the pad, upright and calm. The altitude term contributes 5×10^{-5} points.

Far from the Target, Height Is Worth Almost Nothing. An RL agent improves by noticing which small changes raise its score. The reward is a flat plateau for surviving with a narrow spike at the target height (Figure 4). Released 50 m away, the agent sees a height signal a thousand times smaller than the ordinary run-to-run noise in its score, so it has nothing to follow; the signal is only usable within about 16 m of the target.

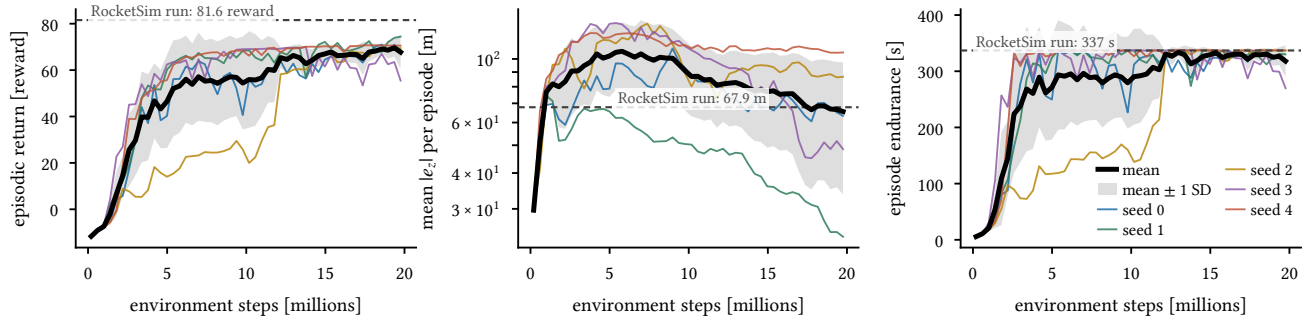


Figure 2: Training benchmark, five seeds, 20M steps each: thin lines are individual runs, the thick black line their mean, the grey band ±1 standard deviation; dashed lines are the RocketSim run. Return and endurance converge across seeds; mean altitude error does not, fanning from 24 to 105 m at equal return.

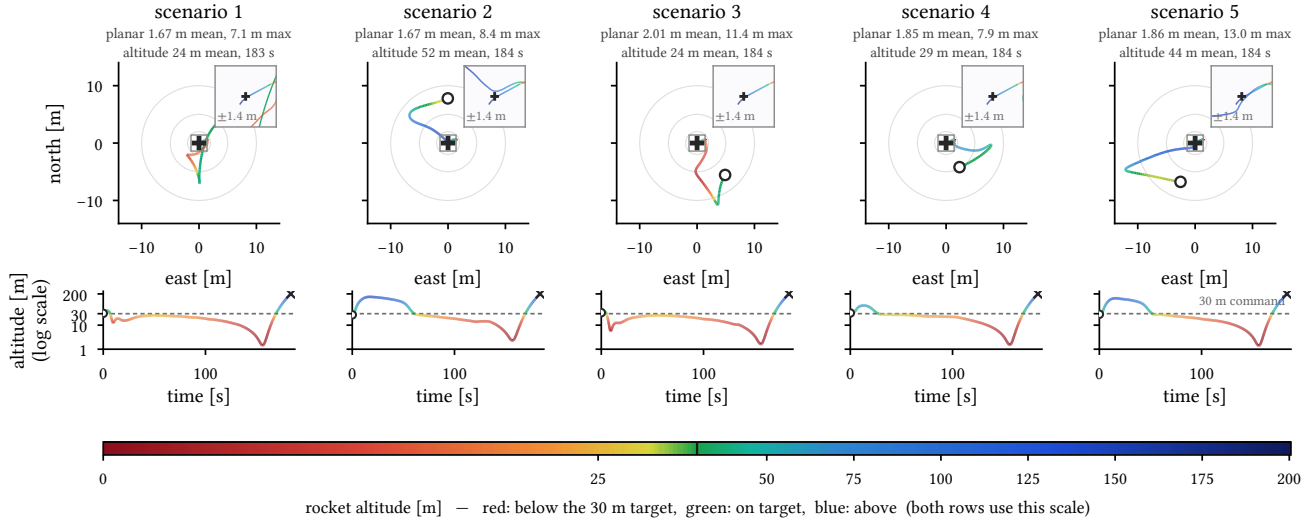


Figure 3: Five deterministic test flights of one trained agent (seed 0). Top: plan view; the cross is the pad centre, the open circle the release point, the inset the region within 1.4 m of the pad. Bottom: altitude against time, *logarithmic* axis — red below the 30 m target, green on it, blue above. Lateral control is tight; every flight of this agent sags below the target, nears the ground at 150 s, then climbs to the ceiling.

Table 4: Post-training benchmark: 200 deterministic episodes per variant. A is the configured task (30 m target, 80 m launch, $\tau_{\min} = 0.39$); B raises them to 100 and 200 m; C also lowers the throttle floor to $\tau_{\min} = 0.20$, which makes hover feasible for the whole burn. The settled rows are signed errors over the final quarter of each flight.

Metric	A	B	C
Mean altitude error [m]	21.4	81.6	40.9
RMS planar error [m]	3.09	1.98	1.20
RMS tilt [deg]	1.61	1.05	0.77
Mean speed [m/s]	2.59	5.71	0.70
Flight duration [s]	161	317	325
Flight on target [%]	21.6	2.5	4.8
Propellant used [kg]	24,215	37,670	38,966
Engine ignitions	0.0	13.4	0.0
Settled altitude error [m]	+3.7	+87.8	−74.7
Settled above target [%]	90	94	0
Dominant terminal	175/200 altitude	188/200 fuel	200/200 ground

For the Last 55% of the Burn, Hovering Is Impossible. A rocket burning fuel gets lighter and needs less thrust, but this engine cannot throttle below 39%. Once the vehicle is lighter than 33.6 t

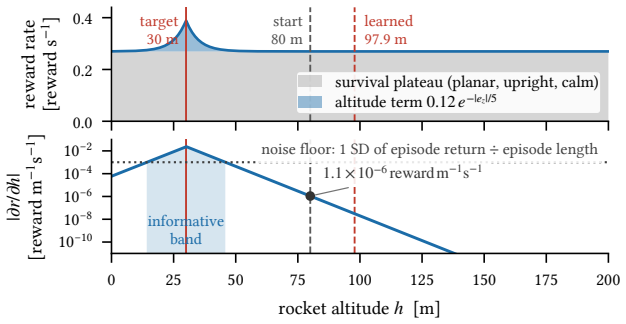


Figure 4: (a) Configured reward rate against altitude: a flat 0.27 reward s^{-1} plateau with a 5 m-wide spike on top. (b) Altitude gradient, logarithmic; below the dotted line one metre is worth less than the return noise.

(Table 1, Figure 5a) even the lowest thrust exceeds its weight, so it can only climb. That happens 151 s into a 337 s burn, so no throttle setting holds altitude for the remaining 55% of the flight. No other engine configuration helps: three or more engines raise the lightest hoverable mass to 100.8 t or beyond, far above the

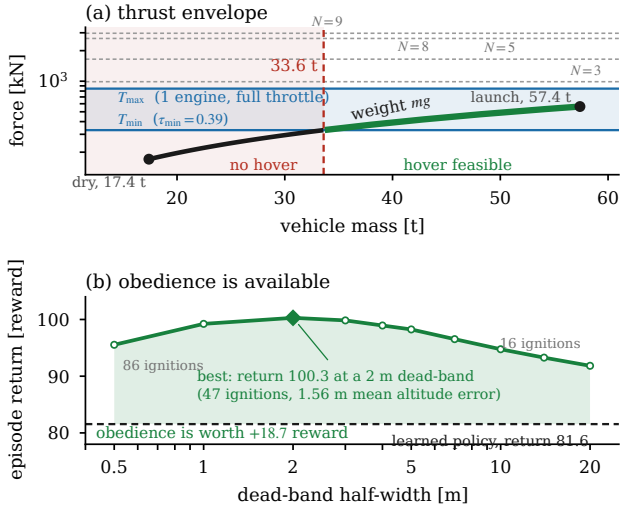


Figure 5: (a) Hover needs the weight ramp to lie between minimum and maximum thrust; it leaves that band at 33.6 t, and no group with $N \geq 3$ reaches the ramp. (b) Return of an obedient controller against its dead-band; every setting beats the learned policy. Return is the undiscounted sum of Equation (1) over one episode.

Table 5: Attempted repairs of the altitude signal: five seeds, 12M steps, vertical-axis surrogate. "In band" counts seeds that ever hold 5 m.

Objective	$\lvert e_z \rvert$ [m]	in band	steps [M]	return
as configured	35.6	3/5	8.5	85.5
wider altitude term	7.8	4/5	6.2	89.5
release inside the band	21.2	3/5	6.6	89.5
potential-based [5]	6.4	4/5	4.6	99.5

57.4 t launch mass (Figure 5a), and changing the target height hardly moves the threshold.

A Controller That Obeys Would Have Scored More. The height can still be held by switching the engine on and off, at a small cost per ignition. A simple hand-written controller that does this — smooth throttle while hovering is possible, 47 on–off pulses afterwards — scores 100.3 against the learned agent's 81.6, for every tolerance we tried (Figure 5b). The objective is therefore not mis-ranked: it already prefers obeying the command. It is uninformative — nothing in the reward landscape leads PPO from the behaviour it found to the one the reward prefers — so the repair is to restore a gradient, not to re-weight the terms.

7 Attempted Fixes and Open Challenges

We did not stop at diagnosing the objective. Table 5 compares the configured reward with a wider altitude term, a release inside the band and potential-based shaping [5], which pays for progress towards the target everywhere. All three alternatives beat the configured reward, and potential-based shaping is best — mean final error 6 m against 36 m, reached in 4.6M steps instead of 8.5M — but each still strands one or two seeds. Re-weighting does not help: a tenfold heavier altitude term still leaves a gradient of order 10^{-5} at launch.

Three problems remain open. *First, the reward must supply a gradient far from the target*, and shaping alone is not enough: the best variant still fails on one seed in five. *Second, the task must be physically feasible:* the vehicle needs throttle authority for the whole burn ($\tau_{min} \leq 0.20$ here), or the objective must say what to do once hover becomes impossible — with a feasible floor our agent sinks instead, so reward and vehicle have to be designed together. *Third, results must be reported over several seeds*, since equal scores can hide four-fold different behaviour.

8 Conclusion

We built RocketSim, a configurable simulator for RL rocket control, and used it to benchmark PPO on booster hover. The agent learns to survive but not to hold the commanded height, and where it settles depends on the seed. The cause is not the learning algorithm: the reward already ranks obedient flight highest — a hand-written controller scores 23% more under it — but far from the target it gives nothing to follow, and for the last 55% of the burn the vehicle cannot hover at all. Reshaping the objective helps without sufficing, and a vehicle that can throttle down sinks instead. A reward with a gradient everywhere, a feasible vehicle and multi-seed reporting are the open challenges this benchmark sets for future work.

Acknowledgement and Disclaimer

This publication is co-funded by the European Union's Horizon Europe research and innovation programme under the Marie Skłodowska-Curie COFUND Postdoctoral Programme, grant agreement No. 101081355–SMASH, and by the Republic of Slovenia and the European Union from the European Regional Development Fund. This work was also supported by the Slovenian Research and Innovation Agency (ARIS), research programme P2-0241 — Synergetics of complex systems and processes. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

References

[1] Behçet Açıkmeşe and Scott R. Ploen. 2007. Convex programming approach to powered descent guidance for mars landing. *Journal of Guidance, Control, and Dynamics*, 30, 5, 1353–1366. DOI: 10.2514/1.27553.

[2] John D. Anderson. 2016. *Introduction to Flight.* (8th ed.). McGraw-Hill Education, New York, NY. ISBN: 9780078027673.

[3] Brian Gaudet, Richard Linares, and Roberto Furfaro. 2020. Deep reinforcement learning for six degree-of-freedom planetary landing. *Advances in Space Research*, 65, 7, 1723–1741. DOI: 10.1016/j.asr.2019.12.030.

[4] Arthur Juliani et al. 2020. Unity: a general platform for intelligent agents. (2020). arXiv: 1809.02627 [cs.LG]. DOI: 10.48550/arXiv.1809.02627.

[5] Andrew Y. Ng, Daishi Harada, and Stuart J. Russell. 1999. Policy invariance under reward transformations: theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML)*. Morgan Kaufmann, 278–287.

[6] Carlos Sánchez-Sánchez and Dario Izzo. 2018. Real-time optimal control via deep neural networks: study on landing problems. *Journal of Guidance, Control, and Dynamics*, 41, 5, 1122–1135. DOI: 10.2514/1.G002357.

[7] John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. 2016. High-dimensional continuous control using generalized advantage estimation. In *4th International Conference on Learning Representations (ICLR)*. arXiv: 1506.02438 [cs.LG].

[8] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. (2017). arXiv: 1707.06347 [cs.LG]. DOI: 10.48550/arXiv.1707.06347.

[9] George P. Sutton and Oscar Biblarz. 2016. *Rocket Propulsion Elements.* (9th ed.). Wiley, Hoboken, NJ. ISBN: 9781118753651.

[10] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction.* (2nd ed.). MIT Press, Cambridge, MA. ISBN: 9780262039246.