2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Rocket Landing Control with Random Annealing Jump Start Reinforcement Learning

Yuxuan Jiang[2], Yujie Yang[2], Zhiqian Lan[2], Guojian Zhan[2], Shengbo Eben Li[*1,2],
Qi Sun[2], Jian Ma[3], Tianwen Yu[3], Changwu Zhang[3]

***Abstract*—Rocket recycling is a crucial pursuit in aerospace technology, aimed at reducing costs and environmental impact in space exploration. The primary focus centers on rocket landing control, involving the guidance of a nonlinear underactuated rocket with limited fuel in real-time. This challenging task prompts the application of reinforcement learning (RL), yet goal-oriented nature of the problem poses difficulties for standard RL algorithms due to the absence of intermediate reward signals. This paper, for the first time, significantly elevates the success rate of rocket landing control from 8% with a baseline controller to 97% on a high-fidelity rocket model using RL. Our approach, called Random Annealing Jump Start (RAJS), is tailored for real-world goal-oriented problems by leveraging prior feedback controllers as guide policy to facilitate environmental exploration and policy learning in RL. In each episode, the guide policy navigates the environment for the guide horizon, followed by the exploration policy taking charge to complete remaining steps. This jump-start strategy prunes exploration space, rendering the problem more tractable to RL algorithms. The guide horizon is sampled from a uniform distribution, with its upper bound annealing to zero based on performance metrics, mitigating distribution shift and mismatch issues in existing methods. Additional enhancements, including cascading jump start, refined reward and terminal condition, and action smoothness regulation, further improve policy performance and practical applicability. The proposed method is validated through extensive evaluation and Hardware-in-the-Loop testing, affirming the effectiveness, real-time feasibility, and smoothness of the proposed controller.**

I. Introduction

Rocket recycling is a pivotal pursuit in the field of aerospace technology, driven by its potential to significantly reduce costs and mitigate the environmental impact of space exploration. Central to this endeavor is the challenge of rocket landing control, a task that involves guiding a nonlinear underactuated rocket plant with limited fuel to land in real-time. The controller must overcome large random disturbances, and satisfy strict terminal state requirement at landing. Conventional control strategies, such as PID controllers, require tedious tuning to meet the accuracy prerequisites, while optimization methods face difficulties in meeting the real-time constraints of this task. Reinforcement learning (RL), employing offline training and online implementation (OTOI) model, presents a viable solution for addressing these complex challenges.

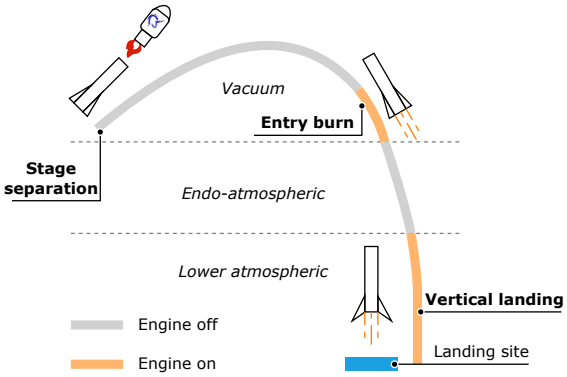


Fig. 1. Brief task demonstration of rocket landing control

RL offers a powerful paradigm to iteratively optimize policies for control problems by maximizing the expected cumulative rewards. It has shown remarkable success in various domains, including video games [1], board games [2], robotics [3], and autonomous driving [4]. In these problems, well-crafted dense reward signals are critical for RL agents to progressively improve their policies. However, rocket landing poses a unique challenge for RL. It is a *goal-oriented* problem, requiring the agent to reach a specific goal set in the state space, with intermediate reward signals not readily available. Standard RL algorithms, such as PPO [5] , SAC [6] and DSAC [7], fail in such scenarios due to their reliance on extensive exploration; the likelihood of randomly reaching the goal diminishes exponentially over time.

General goal-oriented tasks without any prior knowledge present intrinsic difficulties. Several algorithm categories, such as intrinsic reward methods [8], [9] and efficient exploration methods [10], [11] have been developed to optimize policies in such contexts. However, they still face significant challenges in scenarios with hard-to-reach goals and large continuous action spaces, requiring an exponential increase in sample complexity. In many cases, some prior knowledge of the target environment does exist, and leveraging this knowledge can accelerate learning and enhance performance. Reward shaping is a common method for integrating such knowledge. Although manually designed rewards are adaptable and straightforward to implement, they necessitate substantial effort to balance multiple factors. In OpenAI Five [12], for instance, over twenty shaped reward components are

This study is supported by Tsinghua University Initiative Scientific Research Program, and NSF China with U20A20334 and 52072213.

[1]College of Artificial Intelligence, Tsinghua University, Beijing, 100084, China. [2]School of Vehicle and Mobility, Tsinghua University, Beijing, 100084, China. [3]LandSpace Technology Corporation, Beijing, 100176, China. [*]All correspondences should be sent to S. E. Li with e-mail: lisb04@gmail.com

carefully weighted to achieve professional-level performance. Another approach, prevalent in robotic control tasks, involves using a reference trajectory with a quadratic tracking reward. For rocket landing control, designing feasible trajectories demands expert knowledge [13], and a single fixed trajectory becomes impractical under variable initial conditions and environmental disturbances. Alternatively, some methods utilize more specific prior knowledge, like offline datasets or prior policies, to aid exploration and learning. This class of methods typically target initialization, either initializing the policy before RL training [14], [15] or initializing the state before each episode [16], [17]. One simple and effective approach is jump start RL (JSRL) [18], which initialize each episode by applying a guide policy over the guide horizon before transitioning to the exploration RL policy for the remainder of the task. The effectiveness of the jump start technique hinges on the design of the guide horizon, with two variants, JSRL-Curriculum and JSRL-Random, proposed in JSRL paper.

In this paper, we build on the jump start framework, and introduce the Random Annealing Jump Start (RAJS) approach. RAJS is particularly suitable for real-world control problems like rocket landing, which often involve some form of prior feedback controllers based on PID or other classical control methods. For each episode, RAJS samples the guide horizon uniformly between zero and an upper bound, and anneals the upper bound to zero during training. This minimizes state distribution shifts, addressing the limitations of JSRL-Curriculum and broadening the applicability of RL algorithms from offline, off-policy to on-policy variants. Additionally, the final initial state distribution aligns with the underlying environment, countering the distribution mismatch issue seen in JSRL-Random. The stability afforded by RAJS simplifies the annealing schedule, allowing for either automatic scheduling based on training metric feedback or manual adjustment using a clamped ramp function. We also integrate several generally applicable practical techniques, including cascading jump start, refined reward and terminal condition design, and action smoothness regulation. These enhancements enable us to effectively tackle the rocket landing control problem, elevating the success rate from 8% with the baseline controller to an impressive 97%. Extensive evaluation reveals that the control maneuvers are both smooth and interpretable, and Hardware-in-the-Loop testing confirms the real-time feasibility of the control signals produced by our controller.

II. Preliminary

A. Problem Statement

Illustrated in Fig. 1, our study focuses on the critical phase of vertical landing within the lower atmospheric layer, which presents the highest demand for control precision in rocket recycling missions. In the context of controller operations, the fundamental objective of the rocket landing task is to guide the rocket from diverse initial states to land on the ground, while satisfying various terminal state constraints. The plant is a high-fidelity rocket model built in Simulink from LandSpace, containing calibrated subsystems for inertia, engine, actuator, aerodynamics and disturbance. The coordinate system used for the rocket landing problem has its origin on the ground. The y-axis points vertically upward, the x-axis points toward the north, and the z-axis points toward the east.

TABLE I
Main states, initial values and terminal constraints

State	Unit	Initial value & range	Terminal value & range
x	m	50 ± 500	0 ± 5
y	m	2000 ± 10	0
z	m	0 ± 500	0 ± 5
v_x	m/s	-10 ± 50	0 ± 1
v_y	m/s	-300 ± 50	$-1 \sim 0$
v_z	m/s	0 ± 50	0 ± 1
ϕ	$^\circ$	90 ± 0.5	90 ± 3
ψ	$^\circ$	0 ± 0.5	0 ± 3
γ	$^\circ$	0 ± 0.5	-
$\dot{\phi}$	$^\circ$/s	0 ± 0.1	0 ± 1.5
$\dot{\psi}$	$^\circ$/s	0 ± 0.1	0 ± 1.5
$\dot{\gamma}$	$^\circ$/s	0 ± 0.1	-
m	kg	50000 ± 500	≥ 40000

The task consists of two stages. On the first stage, all three engines of the rocket are available for deceleration and pose control. The task switches to the second stage when the vertical velocity v_y reaches a threshold of $v_{sw} = -60\,\mathrm{m/s}$, where the internal low-level control mechanism closes two of three engines, and continue the mission until landing. This internal switch does not affect the controller-rocket interaction directly, but would be reflected in the model dynamics.

The key states of the rocket plant include position x, y, z, velocity v_x, v_y, v_z, angular position ϕ, ψ, γ, angular velocity $\dot{\phi}$, $\dot{\psi}$, $\dot{\gamma}$, and total mass m, which dynamically reduces as fuel is consumed. Each simulation run allows these state variables to initialize within specified ranges, as shown in Table I. The constraints for the terminal state are defined to ensure a precise landing within a narrow vicinity of the target, along with parameters critical for maintaining landing stability, also delineated in Table I. The state observation includes 12 kinematic variables (encompassing position, velocity, angular position, and angular velocity) and the axial load. It is important to note that certain plant states, such as those related to engine response, remain hidden from the controller. The control inputs are modeled as three engine attitude signals and one engine thrust signal, normalized within -1 to 1. The simulation terminates upon the occurrence of one of the following events: either a successful or failed landing, fuel exhaustion, and vertical speed reversal.

Additional complexity is introduced in the form of wind disturbance, characterized by the wind speed that is uniformly sampled within a range of $0\,\mathrm{m/s}$ to $15\,\mathrm{m/s}$, and the wind direction from any direction within the horizontal plane. Crucially, these wind parameters remain unobservable to the controller, necessitating a control algorithm that possesses robustness to such external perturbations.

A baseline controller is integrated within the plant for

establishing a closed-loop simulation environment. This controller, based on several reference trajectories and PID control mechanisms, demonstrates the capability to maintain pose stability and manage the vertical descent under normal conditions. However, its limited adaptability in the face of disturbances results in frequent constraint violations and a modest overall success rate of 8%. Hence, although the baseline controller offers valuable reference for policy learning, the algorithm must refrain from mere imitation and instead employ it strategically.

B. Goal-oriented Markov decision process

Reinforcement Learning (RL) is based on the concept of a Markov Decision Process (MDP), where the optimal action solely depends on the current state [19]. To elaborate further, at each time step denoted as t, the agent selects an action $a_t \in \mathcal{A}$ in response to the current state $s_t \in \mathcal{S}$ and the policy $\pi : \mathcal{S} \to \mathcal{A}$. Subsequently, the environment transits to the next state according to its dynamics, represented as $s_{t+1} = f(s_t, a_t)$, and provides a scalar reward signal denoted as r_t. The value function $v^\pi : \mathcal{S} \to \mathbb{R}$ is defined as the expected cumulative sum of rewards generated by following policy π when initiated from an initial state s.

In the context of rocket landing control, we can abstractly model it as a goal-oriented MDP. The agent is required to reach the goal set $\mathcal{S}_{\text{goal}}$ from the initial set $\mathcal{S}_{\text{init}}$ while maximizing discounted return defined as:

$$\max_{\pi} \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T}\gamma^t r(s_t)\right],$$

where γ is the discount factor, and the goal-oriented reward function is expressed as:

$$r(s_t) = \begin{cases} 1 & s_t \in \mathcal{S}_{\text{goal}} \\ 0 & s_t \notin \mathcal{S}_{\text{goal}} \end{cases}.$$

An episode terminates when state s_t enters $\mathcal{S}_{\text{term}} = \mathcal{S}_{\text{goal}} \cup \mathcal{S}_{\text{fail}}$, where $\mathcal{S}_{\text{fail}}$ represents a collection of states disjoint with $\mathcal{S}_{\text{goal}}$ where continuation of the simulation is infeasible.

C. Proximal policy optimization

Proximal policy optimization (PPO) [5] is an on-policy RL algorithm rooted in the policy gradient method, featuring straightforward implementation, parallel sampling, and stability during training. PPO finds extensive application in addressing complex challenges, such as high-dimensional continuous control tasks, the development of advanced AI for professional-level gaming, and the recent advancements in reinforcement learning through human feedback for refining large language models. The core of PPO lies in its objective function, as expressed by:

$$\begin{aligned} J_\pi(\theta) = \frac{1}{|\mathcal{D}|}\sum_{x,u\in\mathcal{D}} \min\big(&\rho(\theta)A^{\pi_{\text{old}}}(x,u), \\ &\text{clip}\left(\rho(\theta), 1-\epsilon, 1+\epsilon\right)A^{\pi_{\text{old}}}(x,u)\big). \end{aligned} \tag{1}$$

Here, θ denotes the parameter of the policy network, $\rho(\theta) = \frac{\pi_\theta(u|x)}{\pi_{\text{old}}(u|x)}$ represents the importance sampling factor, and $A^{\pi_{\text{old}}}$ is the advantage function computed using generalized advantage estimation [20]. PPO also incorporates the learning of a state value network to estimate v_π, which serves as a baseline in generalized advantage estimation, contributing to variance reduction. While PPO excels in exploration capabilities, it encounters significant challenges when addressing goal-oriented environments independently, primarily due to the exponential sample complexity inherent in such scenarios. To overcome this obstacle effectively, it is necessary to combine PPO with our proposed RAJS method, thus facilitating the efficient attainment of satisfactory solutions.

III. Method

A. Jump start framework

The jump start framework comprises a fixed guide policy $\pi_g(a|s)$ and a learnable exploration policy $\pi_e(a|s)$. In each episode, the guide policy first navigates the environment for the guide horizon H steps, after which the exploration policy π_e takes over to complete the remaining steps. Selecting H close to the full task horizon reduces the exploration space of π_e, enhancing the likelihood of reaching the goal.

The effectiveness of the jump start framework hinges on the design of the guide horizon H. In essence, jump start modifies the initial state distribution d_{init} of the original MDP to an easier $\tilde{d}_{\text{init}}$ produced by the guide policy. $\tilde{d}_{\text{init}}$ depends solely on the guide horizon H, which can be a constant or a random variable. The gap between $\tilde{d}_{\text{init}}$ during training and d_{init} during practical evaluation can negatively impact policy performance, necessitating careful design of H. The distribution mismatch coefficient (Definition 1) is a useful metric to quantify such gap.

Definition 1 (Distribution mismatch coefficient [21])**.** The distribution mismatch coefficient D_∞:

$$D_\infty = \left\| \frac{d_\rho^\pi}{\mu} \right\|_\infty$$

quantifies the effect of policy gradient learning under a initial state distribution μ possibly deviated from the initial state distribution of interest ρ. d_ρ^π refer to the discounted stationary state distribution under policy π:

$$d_\rho^\pi = \mathbb{E}_{s_0\sim\rho}\left[(1-\gamma)\sum_{t=0}^{\infty}\gamma^t \text{Pr}^\pi(s_t = s|s_0)\right].$$

The JSRL paper introduces two variants: JSRL-Curriculum and JSRL-Random. JSRL-Curriculum dynamically adjusts H as a function of training progress using a pre-designed curriculum. In practical terms, a step function is employed:

$$H_k = \left(1 - \frac{k}{n}\right)\bar{H}, \quad k = 0, 1, \ldots, n,$$

where $\bar{H}$ is the initial guide horizon, n is the number of curriculum stages, and k is incremented based on the moving average of evaluated performance. The modified initial state distribution $\tilde{d}_{\text{init},k}$ associated with H_k ensures that $\tilde{d}_{\text{init},n}$ is equivalent to d_{init}, benefiting evaluation performance. However, the transition between stages introduces a notable flaw of distribution shift, adding unseen states to the space

and changing the initial state distribution at each stage switch. JSRL paper focuses on transitioning from offline RL to online RL, employing implicit Q Learning (IQL) [22] as the optimization algorithm in both stages. While IQL mitigates the distribution shift to some extent through its replay buffer, on-policy algorithms like PPO face significant challenges in policy updates due to this shift. Particularly in tasks such as rocket landing control, where initial state distributions $\tilde{d}_{\text{init}}$ for different guide horizons H can be completely disjoint, this would disrupt policy learning, rendering the current policy unlearned.

On the other hand, JSRL-Random samples the guide horizon H from a uniform distribution $U(0, \bar{H})$, where $\bar{H}$ is the horizon upper bound. Unlike JSRL-Curriculum, the initial state distribution induced by the random variable H remains fixed throughout training, essentially creating an alternative stationary MDP with an adjusted initial state distribution. This approach enhances stability during training compared to JSRL-Curriculum but introduces a different challenge of distribution mismatch. Although it is well-established that the optimal policy π^* for an MDP is independent of specific d_{init} under mild conditions, different d_{init} do affect the iteration complexity, particularly when function approximation is involved. As proven by [21], the suboptimality $V^*(s) - V^\pi(s)$ of policy gradient after T iterations is positively correlated to $D_\infty \epsilon_{\text{approx}}$. Here, ϵ_{approx} is the approximation error, representing the minimal possible error for the given parametric function to fit the target distribution. This correlation implies that, for a fixed positive ϵ_{approx} of a neural network function approximation, a larger distribution mismatch coefficient D_∞ contributes to slower convergence. In the context of solving goal-oriented problems with jump start, ρ corresponds to d_{init}, and μ corresponds to $\tilde{d}_{\text{init}}$ of the chosen jump start schedule. Within the support of d_{init}, JSRL-Random's $\tilde{d}_{\text{init}}$ is much smaller than d_{init}, leading to a larger D_∞. This explains the experimental observation that JSRL-Random's policy performance on the evaluation distribution is significantly inferior to optimality.

B. Random annealing jump start

Building upon the insights gained from the analysis presented earlier, and considering the effectiveness of the jump start framework in challenging goal-oriented environments when used with on-policy algorithms, we introduce Random Annealing Jump Start (RAJS). This approach addresses the limitations of distribution shift and distribution mismatch observed in prior works. RAJS achieves this by sampling the guide horizon from a uniform distribution, similar to JSRL-Random. However, a key distinction lies in initializing the upper bound of the uniform distribution with a large value and gradually annealing it to 0 during training, as expressed by the equation:

$$H \sim U(0, \bar{H}\beta(\cdot)), \tag{2}$$

where $\beta(\cdot)$ denotes an annealing factor transitioning from 1 to 0. RAJS effectively mitigates distribution mismatch, as the exploration policy directly engages with the underlying goal-oriented environment after the annealing of $\bar{H}$ to 0. Furthermore, RAJS significantly reduces distribution shift compared to JSRL-Curriculum. In the context of D_∞, assuming μ and ρ represent $\tilde{d}_{\text{init}}$ before and after a distribution shift, RAJS exhibits a substantial overlap between μ and ρ, resulting in a much smaller D_∞ in comparison to JSRL-Random.

Due to the minimal distribution shift during training, the tuning of $\beta(\cdot)$ can be simplified. We propose a schedule based on the moving average of training metrics. For goal-oriented tasks, the proportion of episodes terminating in the goal set (or success rate), denoted as P_{goal}, serves as a suitable metric. The update rule for $\beta(P_{\text{goal}})$ at the end of each training iteration is as follows:

$$\beta \leftarrow \max(\beta - \alpha \mathbb{I}(P_{\text{goal}} \geq P_{\text{thresh}}), 0), \tag{3}$$

where P_{thresh} a tunable performance threshold, $\mathbb{I}(\cdot)$ is the indicator function, and α is the update step size. Additionally, due to improved training stability, designing a ramp schedule $\beta(N)$ manually becomes trivial, with N representing the total number of environment interactions. The start and end steps of the ramp can be determined by solving the task once with $\beta \equiv 1$ and observing the success rate training curve.

RAJS relaxes JSRL's guide policy requirements, extending its applicability to on-policy RL algorithms, as outlined in Algorithm 1. In this paper, our focus is on PPO discussed in the preliminary section, leveraging its general applicability to address the complex challenge of rocket landing control.

Algorithm 1 Random annealing jump start w/ on-policy RL

Input: guide policy π_g, maximum guide horizon $\bar{H}$, metric threshold P_{thresh}, annealing step size α, training batch size B.
Initialize exploration-policy π_e, and other required function approximation, e.g., state value function V. Initialize annealing factor $\beta \leftarrow 1$, and moving mean metric $P \leftarrow 0$.
procedure ROLLOUT(π_g, π_e, $\bar{H}$, $\mathcal{D}$, P)
    Sample initial state $s_0 \sim d_{\text{init}}$, guide horizon $H \sim U(0, \bar{H})$.
    Rollout $\lfloor H \rfloor$ steps with guide policy π_g.
    Rollout until termination with exploration policy π_e, logging trajectory $\{(s, a, r), \ldots\}$ to $\mathcal{D}$.
    Update P with metric of current episode.
end procedure
repeat
    Initialize trajectory dataset $\mathcal{D} = \{\}$.
    Sample with ROLLOUT(π_g, π_e, $\bar{H}\beta$, $\mathcal{D}$, P), until $|\mathcal{D}| \geq B$.
    $\pi_e, V \leftarrow \text{TRAINPOLICY}(\pi_e, V, \mathcal{D})$.
    $\beta \leftarrow \max(\beta - \alpha \mathbb{I}(P \geq P_{\text{thresh}}), 0)$.
until $\beta = 0$ and convergence

C. Practical techniques

Several practical techniques are introduced to further enhance the resolution of goal-oriented tasks. While these techniques were initially developed in the context of rocket landing control, their underlying principles are broadly applicable to a diverse range of tasks.

1) Cascading jump start: In rocket landing control, the baseline controller, which serves as the guide policy, fails in 92% of cases. Consequently, the exploration policy may face a dilemma of hard exploration or unrecoverable failure, depending on the shorter or longer guide horizon length. Although RAJS significantly reduces the exploration space

for RL agents, the initial training phase remains challenging to explore. With respect to the PPO agent, it may suffer from premature entropy collapse, necessitating careful selection of the entropy regulation coefficient to maintain agent exploration. This complicates further performance tuning, as additional difficulty imposed on the agent (e.g., action smoothness requirement) can impede policy convergence.

In this scenario, an effective technique involves incorporating cascading jump start, where an agent is trained under the vanilla setting and subsequently used as the new guide policy π_g' in experiments with more demanding settings. Experiments demonstrate that this technique facilitates simplified exploration at the outset of training, with no significant impact on the final performance after convergence.

2) Reward design: The technical relevance of the outcome of a goal-oriented problem lies solely in the goal set, specifically the satisfaction of terminal state constraints associated with the set. As formulated in the preliminary section, the precisely equivalent terminal reward signal would be a binary signal indicating constraint satisfaction. However, the absence of a smooth reward gradient causes the policy's exploration to be purely driven by "luck", leading to a deterioration in training efficiency and performance.

In an effort to relax the reward, several rules must be followed to ensure that the new objective stays close to the original definition:

1) All intermediate steps should have zero reward. Non-zero intermediate rewards are prone to unexpected policy exploitation behavior and tend to significantly deviate the task from the original definition.
2) Terminal reward should be non-negative. Easily obtained negative rewards would drive the policy to extend intermediate steps and avoid reaching terminal states due to γ-discounting.

Adhering to these requirements, we can provide smooth rewards in $\mathcal{S}_{\text{prox}}$, where $\mathcal{S}_{\text{goal}} \subset \mathcal{S}_{\text{prox}} \subseteq \mathcal{S}_{\text{term}}$, to guide trajectories that terminate in the proximity of the goal set in the correct optimization direction. In rocket landing control, $\mathcal{S}_{\text{prox}}$ can be chosen as all landing states $\{s \mid y = 0\}$, while other conditions of failure, including fuel exhaustion and vertical speed reversal, still receives zero terminal reward. In $\mathcal{S}_{\text{prox}}$, we propose a logarithmic function:

$$r_{\text{prox}} = \max(b - \log(1 + p \max e), 0), \tag{4}$$

where p and b are the scale and bias terms controlling the effective range of terminal states to receive a positive reward. Each element of e corresponds to the normalized absolute terminal error of a constrained state:

$$e = \left| \frac{s_T - s_{\text{target}}}{s_{\text{range}}} \right|.$$

Compared to the typical quadratic reward form, this ensures a non-zero gradient even if the terminal state is far from the target, as well as a larger gradient close to the target to facilitate constraint satisfaction among noisy samples.

3) Terminal condition: In addition to the primary terminal condition, the application of early termination based on heuristics proves beneficial for credit assignment in the absence of intermediate rewards, leading to an acceleration in policy learning. In the rocket landing task, learning vertical speed control from scratch is time-consuming due to a long control horizon, difficulties in credit assignment, the coupling of y and v_y, and the tight bound of v_y when y reaches 0. To address this, kinematics rules are employed to provide coarse information about feasibility. An additional early termination condition is derived for situations where the task is inevitably bound to fail:

$$y_{\min} = \begin{cases} \frac{v_y^2 - v_{sw}^2}{2a_{\max,1}} + \frac{v_{sw}^2}{2a_{\max,2}} & v_y \le v_{sw}, \\ \frac{v_y^2}{2a_{\max,2}}, & v_{sw} < v_y < 0, \end{cases} \tag{5}$$

where $a_{\max,1}$ and $a_{\max,2}$ represent the approximate values of maximum deceleration in two control stages. The episode is terminated with zero terminal reward once $y \le y_{\min}$, signifying the impossibility of a proper landing even with maximum deceleration.

4) Action smoothing: In dealing with high-fidelity plants, it is common for them to account for actuator delay to a certain extent. However, modeling the response perfectly, especially the transient response under oscillating control signals, is impractical. Therefore, for practical actuators, achieving a smooth operating curve is desirable to minimize the disparity between simulation and reality. In this context, RL policies, supported by powerful neural networks, are typically less effective than classical control methods, which is attributed to the high nonlinearity and the absence of intrinsic motivation for smoothness. The challenge becomes more pronounced when addressing goal-oriented tasks, where the requirement of zero intermediate reward hinders direct smoothness regulation.

To address these challenges, two measures are implemented to improve action smoothness in goal-oriented tasks. Firstly, we redefine the action $\tilde{a}$ as the actuator increment, incorporating the original action a into the state observation $\tilde{s}$:

$$\begin{aligned} \tilde{s} &= \begin{bmatrix} s & a \end{bmatrix} \\ \tilde{a} &= \Delta a \\ a' &= \text{clip}(a + k\Delta a, -1, 1) \end{aligned} \tag{6}$$

where k is a scaling factor. Secondly, instead of relying on a regulatory reward, we intervene in the learning process at the loss level. This is achieved by adding a term to PPO's policy loss (1) alongside the advantage:

$$\tilde{J}_\pi(\theta) = J_\pi(\theta) + \epsilon \sum \|\tilde{a}\|^2, \tag{7}$$

where ϵ is a small positive coefficient. While these two measures significantly reduce oscillation, they also introduce increased learning difficulty. Through cascading jump start, these measures are only applied in the second training stage with a strong guide policy, thereby alleviating the associated difficulties.

IV. Experiment

A. Environment configuration

As detailed in the problem statement, the high-fidelity rocket plant and its baseline controller were modeled using Simulink. To enable interaction with the RL policy, we wrapped the system to comply with the standard RL environment interface, as is shown in Fig. 2. RL training demands a significant number of samples for convergence. However, Simulink's interpreted execution is not optimal for efficient and parallel environment interaction. To address this, we utilized Simulink Embedded Coder to generate C code, compiling it into an efficient native module. The use of GOPS Slxpy [23] facilitated automated glue code generation, producing a cross-platform binary with deterministic simulation and flexible parameter tunability. Notably, the control signal supplied to the plant can dynamically transition between external action and the baseline controller through a parameter, streamlining integration with RAJS.

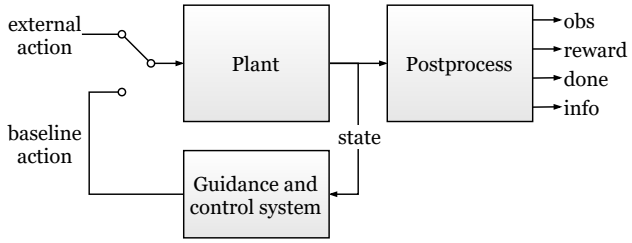


Fig. 2. Wrapped plant for RL training

In terms of reward formulation, all experiments, except for the trajectory tracking baseline mentioned later, share an intermediate reward of zero and a terminal reward defined as follows:

$$r_T = \begin{cases} r_{\text{prox}} & s_T \in \mathcal{S}_{\text{prox}}, \\ 0 & s_T \notin \mathcal{S}_{\text{prox}}. \end{cases}$$

Here, r_{prox} is defined according to (4), with $p = 0.1$ and $b = 3.5$. For the extra early termination condition, $a_{\max,1}$ and $a_{\max,2}$ are set to $40\,\mathrm{m/s^2}$ and $8\,\mathrm{m/s^2}$, respectively, based on preliminary testing with open-loop signals.

B. Benchmark experiment

In the experiments, we combine RAJS with the on-policy PPO algorithm, referred to as PPO-RAJS, and compare its performance against several baselines, all utilizing PPO as the underlying RL algorithm:

1) PPO: This baseline applies the PPO to solve the goal-oriented environment without modification.
2) PPO-Track: The environment definition is adjusted to track predefined trajectories of the baseline controller while incorporating a shaped tracking reward.
3) PPO-JSRL: This variant integrates PPO with JSRL-Random, where the baseline controller serves as the guide policy. We excluded JSRL-Curriculum from the benchmark as it is incompatible with on-policy RL.

We provide the hyperparameters of these algorithms in Table II. To mitigate the influence of randomness, we conduct

TABLE II
Hyperparameters

Algorithm	Parameter	Value
Shared	Learning rate	3×10^{-4}
	Network size	$(256, 256)$
	Network activation	tanh
	Discount factor γ	0.995
	GAE λ	0.97
	Train batch size	20 000
	Gradient steps	30
	Clip parameter ϵ	0.2
	Target KL divergence	0.01
	Entropy coefficient	0.007
PPO-RAJS	Maximum guide horizon $\bar{H}$	18
	Success rate threshold P_{thresh}	0.3
	Annealing step size α	$1/1500$

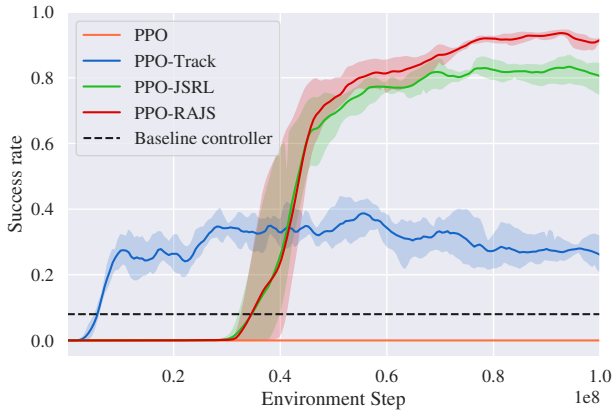
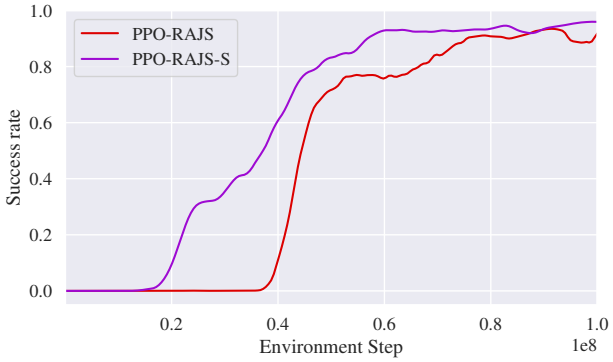


Fig. 3. Success rate for PPO-RAJS and baselines. The solid lines correspond to the mean and the shaded regions correspond to 95% confidence interval over three runs.

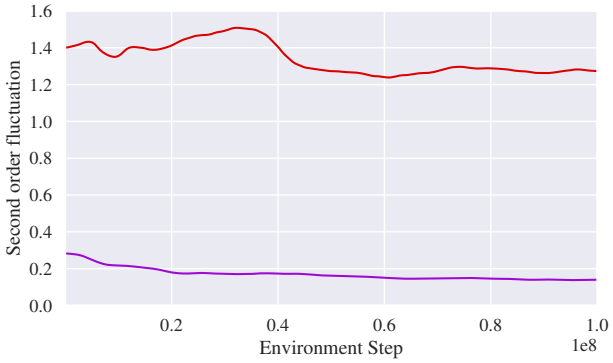
three experiments for each algorithm using different seeds. The combination of an efficient native environment module and PPO's capability for parallel sampling allows for rapid training, achieving 10^8 steps in 5 hours.

We evaluate the performance of these algorithms based on their success rate, defined as the fraction of trajectories that reach the goal set. The learning curves depicted in Fig. 3 illustrate the following results. Our algorithm PPO-RAJS achieves a high success rate and a small variance across different seeds at convergence. PPO fails to optimize the policy due to the sparse reward signals. PPO-Track has poor performance because of insufficient adaptiveness to different initial condition and disturbance. PPO-JSRL exhibits similar trend compared to PPO-RAJS during the initial rise, but cannot further improve performance for the rest of the run due to distribution mismatch issues.

The trained policy from PPO-RAJS is then employed as the new guide policy, following the principles of the cascading jump start technique, to learn the smoothness-focused PPO-RAJS-S policy. This policy incorporates equations (6) and (7) with a smoothness regulation coefficient $\epsilon = 0.01$ and an incremental action scaling factor $k = 0.4$ for all action components. Fig. 4a illustrates that PPO-RAJS-S achieves



(a) Success rate



(b) Action fluctuation (lower is better)

Fig. 4. Training curve comparison between PPO-RAJS and PPO-RAJS-Smooth.

faster learning and further improves the success rate, even with the additional challenge of action smoothness regulation. To quantify the effect of action smoothness, we introduce the second-order fluctuation F_2, defined as:

$$F_2 = \frac{1}{T-2}\sum_{t=2}^{T-1} |a_t + a_{t-2} - 2a_{t-1}|.$$

The training curve presented in Fig. 4b clearly indicates the advantage of incremental action from the outset, with further improvements in action smoothness observed during training through the simple regulation measure.

C. Evaluation result

We generate 10^6 distinct initial conditions to comprehensively evaluate the final performance of the PPO-RAJS-S policy across a vast initial state space. The statistical analysis presented in Table III aligns closely with the training curves. Notably, the component v_y exhibits the highest frequency of violations among all constraints, corroborating the earlier discussion on the challenge posed by controlling v_y effectively. Figure 5 illustrates that the majority of trajectories either achieve success or fail proximal to the goal set boundary. However, a small subset experiences pose instability due to aggressive pose control, leading to landing significantly distant from the target, highlighting a current limitation of our approach.

The enhanced smoothness of actions is depicted in Figure 6. While both PPO-RAJS and PPO-RAJS-S successfully

TABLE III
FINAL POLICY EVALUATION STATISTICS

Success rate	0.9739	
Landing rate	0.9953	
(Within non-landing trials)		
Vertical speed reversal	0.0043	
Fuel exhaustion	0.0003	
(Within landing trials)	Satisfaction rate	Error 99$^{\text{th}}$ percentile
x	0.9930	4.5849
z	0.9944	4.2008
v_x	0.9934	0.8599
v_y	0.9831	1.0869
v_z	0.9939	0.8338
ϕ	0.9949	1.9780
ψ	0.9957	1.3992
$\dot{\phi}$	0.9955	0.6678
$\dot{\psi}$	0.9955	0.6511

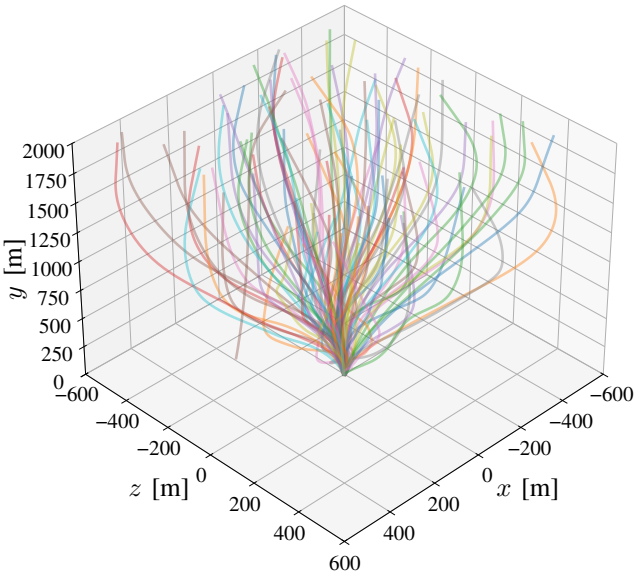


Fig. 5. A subset of 100 trajectories controlled by PPO-RAJS-S policy.

reach the goal set, the former displays notable fluctuations, which could potentially challenge practical actuator implementations. Conversely, actions produced by PPO-RAJS-S exhibit considerably smoother behavior.

Furthermore, we deploy the model on the ZU9E embedded platform and perform hardware-in-loop co-simulation with a rocket dynamics simulation engine. Results from the simulation demonstrate that policy inference aligns within the control interval of 10 ms, validating its real-time applicability.

V. CONCLUSIONS

This paper presents the random annealing jump start approach, which utilizes baseline controllers to empower RL algorithms in tackling complex real-world goal-oriented tasks. Given the safety-critical nature of rocket landing control, our future research will delve into integrating safe RL theory, such as neural barrier certificate [24], to manage state constraints more effectively. This integration holds promise

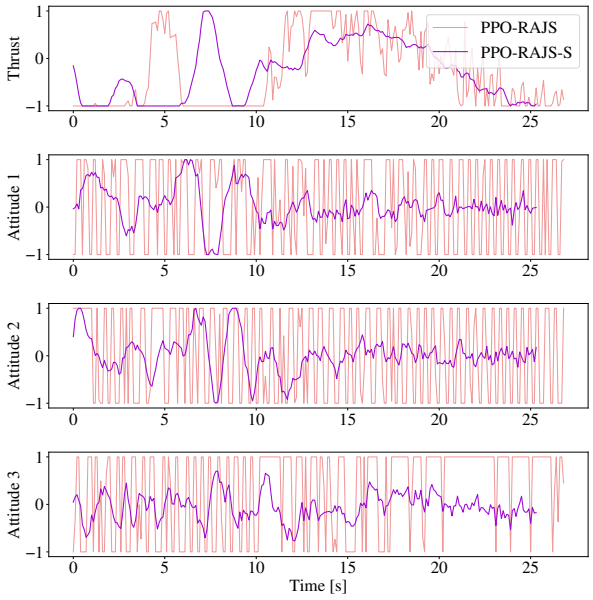


Fig. 6. Action sequence comparison between PPO-RAJS and PPO-RAJS-S, starting from a common initial state.

in addressing the current challenge of unguaranteed pose stability and in further augmenting task success rates.

ACKNOWLEDGMENT

We extend our gratitude to our colleagues at LandSpace for providing their rocket model, contributing innovative insights, and offering consistent support throughout the entirety of this research endeavor, without which this study would not have been possible. LandSpace Technology Co., Ltd. (LandSpace) was founded in 2015 and is a leading Chinese enterprise in the creation and operation of space transportation systems. It is dedicated to establishing a comprehensive industrial chain encompassing "research and development, manufacturing, testing, and launching," with a focus on medium and large Liquid Oxygen-Methane launch vehicles. LandSpace aims to create a technological hub in the space sector and provide highly cost-effective and reliable space transportation services to the global market.

REFERENCES

[1] L. Kaiser, M. Babaeizadeh, P. Milos, B. Osinski, R. H. Campbell, K. Czechowski, D. Erhan, C. Finn, P. Kozakowski, and S. Levine, "Model-based reinforcement learning for atari," in *International Conference on Learning Representations*, 2020, Conference Proceedings.

[2] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver, "Mastering atari, go, chess and shogi by planning with a learned model," *Nature*, vol. 588, no. 7839, pp. 604–609, 2020.

[3] M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, and A. Ray, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

[4] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

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

[6] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 1861–1870.

[7] J. Duan, Y. Guan, S. E. Li, Y. Ren, Q. Sun, and B. Cheng, "Distributional soft actor-critic: Off-policy reinforcement learning for addressing value estimation errors," *IEEE transactions on neural networks and learning systems*, vol. 33, no. 11, pp. 6584–6598, 2021.

[8] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu, "Reinforcement learning with unsupervised auxiliary tasks," in *International Conference on Learning Representations*, 2017.

[9] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *International conference on machine learning*. PMLR, 2017, pp. 2778–2787.

[10] A. Zanette, A. Lazaric, M. Kochenderfer, and E. Brunskill, "Learning near optimal policies with low inherent bellman error," in *International Conference on Machine Learning*. PMLR, 2020, pp. 10 978–10 989.

[11] T. Xie, N. Jiang, H. Wang, C. Xiong, and Y. Bai, "Policy finetuning: Bridging sample-efficient offline and online reinforcement learning," *Advances in neural information processing systems*, vol. 34, 2021.

[12] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse *et al.*, "Dota 2 with large scale deep reinforcement learning," *arXiv preprint arXiv:1912.06680*, 2019.

[13] K. S. G. Anglim, *Minimum-fuel optimal trajectory for reusable first-stage rocket landing using Particle Swarm Optimization*. California State University, Long Beach, 2016.

[14] A. Nair, A. Gupta, M. Dalal, and S. Levine, "Awac: Accelerating online reinforcement learning with offline datasets," *arXiv preprint arXiv:2006.09359*, 2020.

[15] Y. Lu, K. Hausman, Y. Chebotar, M. Yan, E. Jang, A. Herzog, T. Xiao, A. Irpan, M. Khansari, D. Kalashnikov *et al.*, "Aw-opt: Learning robotic skills with imitation andreinforcement at scale," in *Conference on Robot Learning*. PMLR, 2022, pp. 1078–1088.

[16] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 6292–6299.

[17] A. Agarwal, M. Henaff, S. Kakade, and W. Sun, "Pc-pg: Policy cover directed exploration for provable policy gradient learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 13 399–13 412, 2020.

[18] I. Uchendu, T. Xiao, Y. Lu, B. Zhu, M. Yan, J. Simon, M. Bennice, C. Fu, C. Ma, J. Jiao, S. Levine, and K. Hausman, "Jump-start reinforcement learning," in *Proceedings of the 40th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., vol. 202. PMLR, 23–29 Jul 2023, pp. 34 556–34 583.

[19] S. E. Li, *Reinforcement Learning for Sequential Decision and Optimal Control*. Springer Verlag, Singapore, 2023.

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

[21] A. Agarwal, S. M. Kakade, J. D. Lee, and G. Mahajan, "On the theory of policy gradient methods: Optimality, approximation, and distribution shift," *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 4431–4506, 2021.

[22] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit q-learning," in *International Conference on Learning Representations*, 2022.

[23] W. Wang, Y. Zhang, J. Gao, Y. Jiang, Y. Yang, Z. Zheng, W. Zou, J. Li, C. Zhang, W. Cao *et al.*, "Gops: A general optimal control problem solver for autonomous driving and industrial control applications," *Communications in Transportation Research*, vol. 3, p. 100096, 2023.

[24] Y. Yang, Y. Jiang, Y. Liu, J. Chen, and S. E. Li, "Model-free safe reinforcement learning through neural barrier certificate," *IEEE Robotics and Automation Letters*, vol. 8, no. 3, pp. 1295–1302, 2023.