AAS 24-023

META-REINFORCEMENT LEARNING FOR SPACECRAFT PROXIMITY OPERATIONS GUIDANCE AND CONTROL IN CISLUNAR SPACE

Giovanni Fereoli[*], Hanspeter Schaub[†] and Pierluigi Di Lizia[‡]

In order to address the challenges of future space exploration, new lightweight and model-free guidance algorithms are necessary to make the spacecraft completely autonomous. In recent years, autonomous spacecraft guidance has been a subject of intense research, and in the near future, this technology will be a great advantage for proximity operations in cislunar space. For instance, NASA's Artemis program plans to establish a lunar Gateway, and this type of autonomous maneuver, besides the nominal Rendezvous and Docking (RVD) ones, is also necessary for the assembly and maintenance procedures. In this context, a Meta-Reinforcement Learning (Meta-RL) algorithm is applied to address the real-time relative optimal guidance problem of a spacecraft in the cislunar environment. Non-Keplerian orbits have more complex dynamics, and classic control theory may be less flexible and more computationally expensive with respect to Machine Learning (ML) methods. Furthermore, Meta-RL is chosen for its peculiar and promising ability of "learning how to learn" through experience. It is an ML approach in which a model is trained on a variety of tasks in such a way that it becomes more efficient and effective at learning new ones. A stochastic optimal control problem is modeled in the Circular Restricted Three-Body Problem (CR3BP) framework as a discrete time-scale Markov Decision Process (MDP). The agent, an LSTM-based network, is then trained with a state-of-the-art actor-critic algorithm known as Proximal Policy Optimization (PPO). Additionally, operational constraints and stochastic effects are considered to assess policy safety and robustness. An MLP-based agent and an optimal control solution using pseudospectral methods are also evaluated for comparison purposes. The resulting tool is a closed-loop controller able to autonomously guide a spacecraft in the context of cislunar proximity operations. It is able to approximate the optimal control solution with a very general and not hand-crafted algorithmic framework, guaranteeing at the same time high robustness and computational efficiency.

INTRODUCTION

Rendezvous and Docking (RVD) maneuvers play a critical role in various space mission tasks, including crew transfer, cargo exchange, repairs, and structure assembly. For more than five decades, various countries have conducted missions of this nature[1] and various chaser spacecraft, with their origins in Gemini and Apollo, have successfully performed these tasks. Some contemporary examples, such as the Demonstration of Autonomous Rendezvous Technology (DART), the Automated Transfer Vehicle (ATV) of the European Space Agency (ESA), and the H-II Transfer Vehicle (HTV) from Japan, are now unmanned and fully automated. This progress, considerably more robust and secure than previous manual astronaut maneuvers, required the adoption of new autonomous algorithms for *Guidance, Navigation, and Control (GNC)*. In addition, these techniques have become indispensable for the most intricate spacecraft proximity operations, including on-orbit servicing, active space debris removal, and other missions. This kind of problem can be formulated as an *Optimal Control Problem (OCP)* and solved with active set or interior point techniques after

[*]Graduate Research Assistant, Ann and H.J. Smead Department of Aerospace Engineering Sciences, University of Colorado, Boulder, 431 UCB, Colorado Center for Astrodynamics Research, Boulder, CO, 80309. AAS Member, AIAA Member.

[†]Professor and Department Chair, Schaden Leadership Chair, Ann and H.J. Smead Department of Aerospace Engineering Sciences, University of Colorado, Boulder, 431 UCB, Colorado Center for Astrodynamics Research, Boulder, CO, 80309. AAS Fellow, AIAA Fellow.

[‡]Associate professor, Department of Aerospace Science and Technology, Politecnico di Milano, Via Privata Giuseppe La Masa 34, Milano, Italy, 20156. AAS Member, AIAA Member.

transcription.[2] However, these methods are open-loop, unable to manage unpredictable circumstances without a controller, and too computationally expensive to be executed on-board. Autonomous G&C algorithms require two fundamental ideas: *robustness*, which should be achieved by satisfying numerous requirements, such as closed-loop control, for mission safety, and *computational efficiency* for on-board implementation. Today, to acquire this capability, the high-level task is usually transformed into a pre-calculated reference trajectory (Guidance) and then tracked by a controller (Control). Several effective algorithms suitable for implementation on-board have been proposed, as documented in references.[3] However, additional criteria can be introduced. First, to handle more intricate dynamical environments where engineering modeling would be impractical, these algorithms should be *model-free*. Second, they should incorporate an *integrated* G&C law, eliminating the need to break down the high-level task into separate processes for reference computation and tracking. *Reinforcement Learning (RL)* possesses the aforementioned characteristics.[4] RL agents have the capability to undergo training by utilizing high-fidelity simulators. This training allows them to learn the optimal closed-loop policy, rather than being explicitly designed. When the policy is tested on-board, it only needs minimal computational resources. Additionally, RL can directly optimize the task-level objective and utilize domain randomization to handle model uncertainty. This allows for the identification of control responses that are more empirically reliable, as mentioned in Ref. 5.

Deep-Reinforcement Learning (Deep-RL) is currently demonstrating effectiveness in cutting-edge applications, including robotics,[6] automotive,[7] and aerospace.[8] Employing agents equipped with *Artificial Neural Networks (ANNs)*, Deep-RL is adept at addressing sequential decision-making problems, known as *Markov Decision Process (MDPs)*, through interactive engagement with the environment. Neural networks excel in accurately estimating functions, making them well suited, for example, to learn optimal closed-loop G&C policies by estimating the solution to the Hamilton-Jacobi-Bellman (HJB) equations.[9] The development of GNC autonomous systems presents a major challenge in terms of robustness to uncertain spacecraft models and environments. RL deep agents, such as Feedforward ANNs, are usually successful in learning within the training distributions, but often have difficulty extrapolating beyond them (low generalization capability). This can lead to an unstable guidance law when the spacecraft encounters states that are not part of the training distribution. Moreover, traditional RL requires a large amount of experience to even learn basic tasks (sample inefficiency). To address these two main weaknesses, recent advances in *Meta-Reinforcement Learning (Meta-RL)* have been made. This is a *Machine Learning (ML)* technique in which an agent is taught a range of tasks (that is, randomized environment parameters) instead of just one, allowing it to create a meta-policy that can quickly adjust to new and unseen tasks with minimal experience.[10] This transfer learning ability has been referred to as *"learn to learn"* by Ref. 11. There are numerous approaches to implementing these ideas in practice, such as Meta-Agnostic-Meta-Learning (MAML), Task-Agnostic Meta-Learning (TAML), REPTILE, Meta-SGD, and many more.[12] One of the most popular approaches, as proposed by Ref. 13, is to employ *Long Short-Term Memory (LSTM)* networks within a RL agent trained through gradient-based methods, referred to as *Meta-Recurrent Neural Networks (Meta-RNN)*. This approach has been studied in the aerospace field and has been shown to be more effective than traditional RL in managing uncertain environments, actuator failures, and Partially Observable MDPs (POMDPs).[14,15] The hidden states, acting as an internal memory, of LSTMs networks provide them with an internal dynamics that is continually updated by new observations along the trajectory. This contributes to better learning in high-uncertainty environments during the training phase and allows for an adaptive policy during the testing phase. The resulting G&C algorithm has a higher overall robustness due to these properties, which is especially important after deployment. Unlike *Multi-Layer Perceptrons (MLPs)*, LSTMs can modify their hidden state and adjust in real time to capture data that have not been modeled, such as dynamical disturbances or hardware failure, during spacecraft operations. Meta-RL has shown its effectiveness in a range of spacecraft G&C applications, including planetary landing,[14] under-actuated cubesat,[16] trajectory design,[15] rendezvous missions,[17] and asteroid proximity operations.[18] All of these studies employ a particular actor-critic gradient-based RL algorithm, *Proximal Policy Optimization (PPO)*, to optimize Meta-RL policies. PPO has been shown to be highly effective for continuous control applications and is currently the state-of-the-art method.[19]

This study will address, through the proposed Meta-RL solution, the challenge of autonomous relative *Guidance and Control (G&C)* in cislunar space. This is especially pertinent in the context of NASA's ARTEMIS project, which intends to establish a Lunar Gateway, a cislunar space station, in the next decade.[20]

Positioned at the *Southern 9:2 Resonant L_2 Near-Rectilinear Halo Orbit (NRHO)* of the Earth-Moon system, the station's construction and operations necessitate rendezvous and docking missions. *Autonomous Rendezvous, Proximity Operations and Docking (ARPOD)* capabilities in cislunar space have the potential to significantly improve the deployment and functionality of the Lunar Gateway. Nevertheless, the dynamics within non-Keplerian environments, characterized by high non-linearity and chaoticity, still demand a thorough examination today.[21] Traditional protocols employed in the Apollo/Shuttle programs and automated ISS operations of ATV/HTV, originally designed for robust gravitational fields, prove inadequate for the complexity of cislunar space. Consequently, to overcome these challenges, there is a pressing need for the design of innovative RVD algorithms and procedures. Recent research[21,22] has partially addressed this need by formalizing the various safety constraints that should be applied in this scenario (e.g., keep-out sphere, approach corridors, etc.). The researchers also analyzed the effectiveness of different equations of motion, highlighting that non-linear equations of motion are necessary for designing accurate G&C solutions. The works of Ref. 23, 24 introduced a combination of stable/unstable manifold exploitation with impulsive control for passive safety during far-range rendezvous. In addition, they used traditional G&C algorithms, such as PID, SDRE, etc., with continuous control for active safety in close-range rendezvous and docking procedures. Despite these contributions, there is still a notable scarcity of research on GNC algorithms specifically designed for relative multi-body dynamics.

CISLUNAR SPACE RELATIVE DYNAMICS

This section presents a summary of the equations of motion and the reference frame employed in this study to model the relative motion between a target and a chaser spacecraft in the cislunar space environment.

Relative Circular Restricted Three-Body Problem

The motion of a massless particle influenced by two massive bodies is described by the *Restricted Three-Body Problem (R3BP)*. This model considers the mass of the particle m_B and the masses of massive bodies m_1 and m_2, assuming $m_B \ll m_1,\ m_2$ and $m_1 > m_2$. The problem can be simplified by representing the equations in a rotating reference frame and assuming circular motion of the primary bodies, leading to the *Circular Restricted Three-Body Problem (CR3BP)*. This problem is studied in its normalized form. The CR3BP is an autonomous system of equations, where the only parameter defining the three-body system is the mass parameter μ. In the Earth-Moon case, this parameter is denoted as $\mu = 0.012150584269542$.

The equations of motion for the *Relative Circular Restricted Three-Body Problem (RCR3BP)* can be obtained by subtracting the absolute CR3BP equations of motion of the chaser and the target, resulting in $\delta\mathbf{x} = \mathbf{x}^C - \mathbf{x}^T$. These non-autonomous equations are expressed in the Relative Synodic Earth-Moon frame (Figure 1) and are as follows:

$$
\begin{aligned}
\delta\ddot{x} &= 2\delta\dot{y} + \delta x + (1-\mu)\left[\frac{x^T+\mu}{\|\mathbf{r_{1T}}\|^3} - \frac{x^T+\delta x+\mu}{\|\mathbf{r_{1T}}+\boldsymbol{\rho}\|^3}\right] + \mu\left[\frac{x^T-\mu-1}{\|\mathbf{r_{2T}}\|^3} - \frac{x^T+\delta x+\mu-1}{\|\mathbf{r_{2T}}+\boldsymbol{\rho}\|^3}\right] \\
\delta\ddot{y} &= -2\delta\dot{x} + \delta y + (1-\mu)\left[\frac{y^T}{\|\mathbf{r_{1T}}\|^3} - \frac{y^T+\delta y}{\|\mathbf{r_{1T}}+\boldsymbol{\rho}\|^3}\right] + \mu\left[\frac{y^T}{\|\mathbf{r_{2T}}\|^3} - \frac{y^T+\delta y}{\|\mathbf{r_{2T}}+\boldsymbol{\rho}\|^3}\right] \\
\delta\ddot{z} &= (1-\mu)\left[\frac{z^T}{\|\mathbf{r_{1T}}\|^3} - \frac{z^T+\delta z}{\|\mathbf{r_{1T}}+\boldsymbol{\rho}\|^3}\right] + \mu\left[\frac{z^T}{\|\mathbf{r_{2T}}\|^3} - \frac{z^T+\delta z}{\|\mathbf{r_{2T}}+\boldsymbol{\rho}\|^3}\right]
\end{aligned}
\tag{1}
$$

Given $\boldsymbol{\rho} = [\delta x,\ \delta y,\ \delta z]$, and with $\mathbf{r}_{1T}$ and $\mathbf{r}_{2T}$ defined as $\left[x^T+\mu,\ y^T,\ z^T\right]$ and $\left[x^T+\mu-1,\ y^T,\ z^T\right]$, respectively. In the literature, various types of relative equations of motion for Near-Rectilinear Halo Orbits (NRHOs) have been explored. The work of Ref. 21 delved into the study of Clohessy–Wiltshire (CW) and Linearized Relative Equations (LRE). Meanwhile, Ref. 25 proposed a set of Non-Linear Relative (NLR) equations in the Local-Vertical/Local-Horizon (LVLH) frame, necessitating the computation of the target's angular velocity and acceleration vectors during its orbit around the Moon. However, the RCR3BP equations stand out for their generalization and simplicity.

Relative Synodic Reference Frame

The *Circular Restricted Three-Body Problem (CR3BP)* can be described in a rotating reference frame centered on the barycenter of two massive bodies. This frame rotates with a constant angular velocity ω_s and has a right-handed coordinate system with the $\hat{\mathbf{x}}$-axis pointing from the larger body to the smaller one, the $\hat{\mathbf{z}}$-axis aligned with the system's angular momentum, and the $\hat{\mathbf{y}}$-axis completing the right-handed coordinate system. The relative motion between a target and a chaser spacecraft in the CR3BP can be expressed in this frame, with masses m_T and m_C. The assumed origin of the three axes, taking into account $\delta\mathbf{x}$, is now positioned at the opening of the target's docking port. This establishes $\mathcal{C}_T : \{\hat{\mathbf{x}},\ \hat{\mathbf{y}},\ \hat{\mathbf{z}}\}$, as in Figure 1.

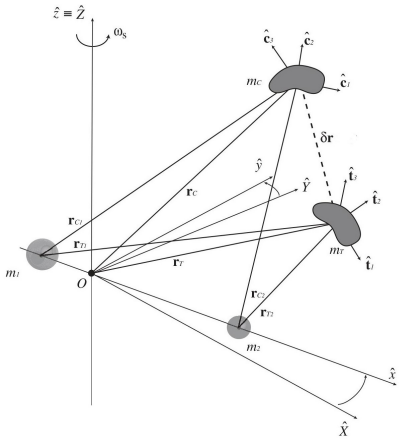


Figure 1: Relative synodic reference frame centered in the target body. Source: Ref. 26.

REINFORCEMENT LEARNING

This section introduces Reinforcement Learning (RL) with a focus on the Markov Decision Process (MDP) as the modeling framework. The discussion then highlights Proximal Policy Optimization (PPO) as a key gradient-based RL technique.

Markov Decision Process

Reinforcement Learning (RL) is a form of learning that associates actions with situations in order to maximize a reward signal. It is formulated based on the principles of dynamical systems, particularly utilizing *Markov Decision Processes (MDPs)* to model sequential decision-making. An *agent* interacts with an *environment* over discrete time steps ($t = 0, 1, 2, 3, ...$). The agent receives a state representation $S_t \in \mathcal{S}$, selects an action $A_t \in \mathcal{A}(s)$, and obtains a numerical reward $R_{t+1} \in \mathcal{R}$. This interaction creates a *trajectory* starting with: S_0, A_0, R_1, S_1, A_1, R_2, etc. (Figure 2).

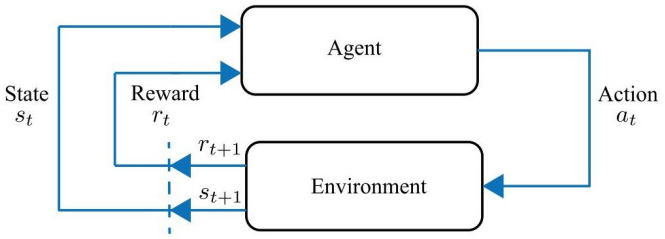


Figure 2: The agent-environment interaction at each discrete time step in a Markov Decision Process (MDP). Source: Ref. 9.

To employ this modeling framework, the foundational assumptions encompass the complete observability of the state, finite sets of states $\mathcal{S}$ and actions $\mathcal{A}$, and rewards $\mathcal{R}$, along with the *Markov Property*. The latter asserts that the probability of each possible value for S_t and R_t is only influenced by S_{t-1} and R_{t-1}, and not by any previous states and actions. The MDP environment is described by a discrete probability distribution $p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \to [0,\ 1]$, which determines the dynamics of the system as:

$$p\left(s',\ r \mid s,\ a\right) \doteq Pr\left(S_t = s',\ R_t = r \mid S_{t-1} = s,\ A_{t-1} = a\right) \tag{2}$$

For all $s' \in \mathcal{S}$, $s \in \mathcal{S}$, $r \in \mathcal{R}$ and $a \in \mathcal{A}\left(s\right)$.

The agent's objective is to maximize the long-term cumulative reward rather than the immediate rewards $R_t \in \mathbb{R}$. Typically, the interaction between an agent and its environment is divided into sequences known as *episodes*. At the conclusion of each episode, a specific state called the *terminal state* is reached, which signifies the completion or non-completion of the task. Following this, the system is restored to either a default initial state or a randomly chosen initial state from a standard distribution. In order to avoid excessive cumulative rewards and promote convergence, the agent usually employs the idea of discounting to determine which actions will maximize the total benefits it will receive in the future. Specifically, it selects action A_t to maximize the *discounted return*, represented as G_t in Eq. 3.

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = R_{t+1} + \gamma G_{t+1} = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k \tag{3}$$

Where $\gamma \in [0,\ 1]$ is the *discount rate*. The discount rate determines the current value of a future reward. A *policy* π can be formally expressed as a mapping of states to the likelihood of selecting each potential action; specifically, $\pi\left(a|s\right)$ is the probability that $A_t = a$ when $S_t = s$. The expected discounted return when starting in a state $s \in \mathcal{S}$ and following policy π is known as the *state-value function* v_π. This is defined as follows:

$$v_\pi\left(s\right) \doteq \mathbb{E}_\pi\left[G_t \mid S_t = s\right] \tag{4}$$

The expected value of a random variable is denoted as $\mathbb{E}_\pi\left[\cdot\right]$ when the agent follows the policy π. The expected discounted return when beginning in a state $s \in \mathcal{S}$ and taking an action $a \in \mathcal{A}\left(s\right)$ according to the policy π is denoted by the *action-value function* q_π:

$$q_\pi\left(s,\ a\right) \doteq \mathbb{E}_\pi\left[G_t \mid S_t = s,\ A_t = a\right] \tag{5}$$

The primary objective in reinforcement learning is to find a policy that maximizes the expected discounted return. A policy π is considered better than or equal to another policy π' if $v_\pi\left(s\right) \geq v_{\pi'}\left(s\right)$ for all states $s \in \mathcal{S}$. The most favorable policy is referred to as the *optimal policy* denoted by π_*. Its optimal state-value function v_* is defined as:

$$v_*\left(s\right) = \max_\pi\ v_\pi\left(s\right) \tag{6}$$

Proximal Policy Optimization

Policy Gradient Methods, including *Proximal Policy Optimization (PPO)*, are a type of approximated method solution that is commonly used to solve RL problems in large state spaces. These methods are preferred because they offer greater stability and the assurance of convergence, as demonstrated in Ref. 4. They achieve this by utilizing parameterized functions, such as *Artificial Neural Networks (ANNs)*, to approximate functions, which makes them well suited for dealing with high-dimensional, continuous-state, and

continuous-action spaces. Unlike traditional value-based methods, such as Q-learning, policy gradient methods calculate the policy directly without requiring a model. However, they are generally less data-efficient. To address this problem, an actor-critic framework, which is also utilized by PPO, can be utilized. This approach aids in reducing the variance in the estimated policy gradient, making it well-suited for online applications.

These methods acquire a parameterized policy* and make action selections without the need to compute a value function. The parameter vector is expressed as $\boldsymbol{\theta} \in \mathbb{R}^d$ and the policy as $\pi_{\boldsymbol{\theta}}\left(a|s\right)$. The algorithms for solving the policy parameters are based on the gradient of a scalar performance measure $J\left(\boldsymbol{\theta}\right)$. They are designed to maximize performance, so the update of the parameters at the training step k follows the gradient ascent of J:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha \widehat{\boldsymbol{\nabla} J\left(\boldsymbol{\theta}_k\right)} \tag{7}$$

Where $\widehat{\boldsymbol{\nabla} J\left(\boldsymbol{\theta}_k\right)}$ is an estimate of the gradient of the performance metric with respect to the current policy parameters $\boldsymbol{\theta}_k$, and the step size is determined by the *learning rate* α.

PPO is renowned as a state-of-the-art algorithm for continuous control in reinforcement learning.[19] It is a first-order approximation of the Thrust Region Policy Optimization (TRPO) method. As an actor-critic method, PPO introduces bias through a bootstrapping critic and uses a learned weight vector $\mathbf{w} \in \mathbb{R}^m$ to estimate the state-value function $\hat{v}_{\mathbf{w}}$. In PPO, a *clipped surrogate* objective function is used to produce a pessimistic and conservative approximation of policy performance. The corresponding figure of merit is displayed below:

$$\max_{\boldsymbol{\theta}} \quad \hat{\mathbb{E}}_{\pi}\left[\min\left(r_t\left(\boldsymbol{\theta}\right)\hat{A}_t,\ clip\left(r_t\left(\boldsymbol{\theta}\right),\ 1-\varepsilon,\ 1+\varepsilon\right)\hat{A}_t\right)\right] \tag{8}$$

The probability ratio r_t is expressed as:

$$r_t\left(\boldsymbol{\theta}\right) = \frac{\pi_{\theta}\left(A_t \mid S_t\right)}{\pi_{\theta,old}\left(A_t \mid S_t\right)} \tag{9}$$

The *Advantage Function*, denoted as $A_t\left(s,\ a\right) = q_{\pi}\left(S_t,\ A_t\right) - v_{\mathbf{w}}\left(S_t\right)$, serves as an indicator of the additional reward that the agent can take when returning a specific action from a given state. In practical PPO implementations, it is crucial to estimate both the advantage function $\hat{A}_t$ (dependent on the unknown environment) and the expectation operator $\hat{\mathbb{E}}_{\pi}$.

META-REINFORCEMENT LEARNING

As discussed in the introduction, *Meta-Reinforcement Learning (Meta-RL)*, embodied as the *Meta-Recurrent Network (Meta-RNN)* according to Ref. 13, integrates Long-Short Term Memory (LSTM)[27] into the RL agent. Subsequently, the agent undergoes training using gradient-based techniques such as PPO. This implementation aims to make the agent well suited for models with numerous free parameters, simultaneously enhancing the algorithmic framework's simplicity compared to other meta-learning algorithms. The Meta-RNN approach excels in handling highly non-stationary time-series and has demonstrated its applicability to learning and autonomous systems.

Training Algorithm

The PPO algorithm, which is a model-free method, has been previously discussed, and it has been noted that it requires the estimation of the advantage function in order to operate. To address this, the *Generalized*

*The policy $\pi_{\boldsymbol{\theta}}\left(a \mid s\right)$ is typically designed to be stochastic for exploration purposes. In the context of continuous action spaces, the policy can be formulated as a normal *Probability Density Function (PDF)*. As learning progresses, the focus shifts from exploration to exploitation, resulting in the standard deviation of the PDF converging to zero.

Advantage Estimate (GAE)[28] is used and can be adjusted using the coefficient λ to strike a balance between bias and variance. The GAE formula, as shown in Eq. 10, is truncated at the end of each episode ($t = T$) to accommodate LSTM-based networks.[29]

$$\hat{A}_t^{GAE} = \sum_{k=0}^{\infty} (\gamma\lambda)^k \, \delta_{t+k}^v \tag{10}$$

Referring to Ref. 30, the definition of the *Residual Temporal-Difference (Residual TD)* for the discounted value function can be expressed as $\delta_t^v = R_t + \gamma v_{\boldsymbol{\pi}}(S_{t+1}) - v_{\boldsymbol{\pi}}(S_t)$. In order to estimate $\hat{A}_t^{GAE}$, it is necessary to estimate the state-value function. When a non-linear function approximator is used to represent the state-value function, a commonly employed approach involves solving a non-linear regression problem by minimizing the *Mean Square Error (MSE)* in the following manner:

$$J^{MSE}(\mathbf{w}) = \hat{\mathbb{E}}_\pi \left[\left(v_{\mathbf{w}}(S_t) - \sum_{k=t}^{T} \gamma^{k-t} R_t \right)^2 \right] \tag{11}$$

Batch learning is a machine learning technique used to estimate expectancy operators. It works by generating a batch of trajectories (or roll-outs), with the number of trajectories controlled by the hyperparameter called *batch size*, before each training cycle. After processing the entire training dataset, the model is updated in a single iteration, adjusting all parameters at once. For a more detailed explanation of the algorithm's architecture, see Figure 3.

Gradient ascent is subsequently applied to both unrolled LSTM-based Actor and Critic networks, and *Backpropagation Through Time (BPTT)* is used to compute the gradients of the parameters $\boldsymbol{\theta}$ and $\boldsymbol{w}$. In each epoch k of the gradient-based optimizer, such as the *Adaptive Moment (ADAM) Estimation* method, the update equations (similar to Eq. 7) are implemented as follows:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \beta_\theta \widehat{\nabla J(\boldsymbol{\theta}_k)} \tag{12}$$

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \beta_w \widehat{\nabla J(\mathbf{w}_k)} \tag{13}$$

For each set of roll-out data, the gradient descent step is applied a certain number of times, called *epochs*, which is a hyperparameter. The algorithm will stop when the maximum number of predetermined learning steps is reached.

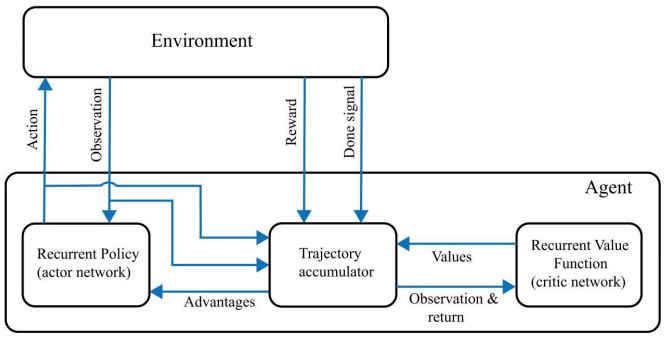


Figure 3: Meta-Reinforcement Learning (Meta-RL) training architecture for Recurrent Proximal Policy Optimization (Recurrent PPO) algorithm. Source: Ref. 4.

The weights in the LSTM networks are initialized using an *Orthogonal* approach. This is a critical step, as incorrect initialization can lead to issues such as all layers learning the same feature, or vanishing/exploding gradients. This method avoids ill-conditioned gradients by initializing ANNs through the multiplication of matrices with unitary eigenvalues. This involves creating a matrix from the weights obtained from a normal distribution and then orthogonalizing the rows.

The algorithm described above, due to the specific version of GAE implemented and the use of BPTT, produces a version of PPO that is suitable for use with LSTM networks, called *Recurrent Proximal Policy Optimization (Recurrent PPO)*.[16] The following is an illustration of the algorithm:

Algorithm 1 Recurrent PPO

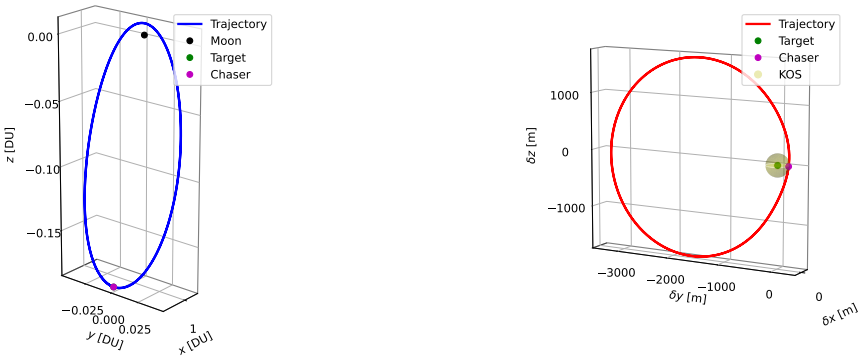
Initialization of neural network parameters $\boldsymbol{\theta}$ and $\mathbf{w}$
for step=1,2,..., learning steps **do**
  Reset environment
  for episode=1,2,..., batch size **do**
    Run Policy π_{old} in environment until *done*
    Compute advantage estimate $\hat{A}_t^{GAE}$
    Store trajectory into batch roll-out
  end for
  for iteration=1,2,..., epochs **do**
    Unroll agent LSTMs
    Optimize Actor J^{PPO} wrt $\boldsymbol{\theta}$ and Critic J^{MSE} wrt $\mathbf{w}$
    $\boldsymbol{\theta}_{old} \leftarrow \boldsymbol{\theta}$, $\mathbf{w}_{old} \leftarrow \mathbf{w}$
  end for
end for

The implementation of Recurrent PPO, and PPO in general, involves a number of complex optimizations that are not discussed in depth or even mentioned in Ref. 19. However, these optimizations have been found to have a significant effect on the efficacy of PPO.[31] To ensure robustness, consistency, and high flexibility, this work utilizes the *Stable-Baselines3 (SB3)*[†] library in *Python*, an open-source machine learning library created by the Facebook AI Research (FAIR) Laboratory.

PROBLEM FORMULATION

This project focuses on developing an autonomous G&C algorithm for the final approach and docking of a spacecraft in cislunar space. The emphasis is on continuous control and active collision avoidance. The chaser spacecraft initiates the maneuver from a holding point at the edge of the target Keep-Out-Sphere (KOS) with a radius of $200\ m$. It is also assumed that the chaser spacecraft is already situated at the apolune of the *Southern L_2 9:2 Resonant Near Rectilinear Halo Orbit (NRHO)*. The choice of this location for the final rendezvous stages is based on its favorable slower dynamics, ensuring passive safety and fuel efficiency.[21] Figure 4 illustrates this scenario, showing the initial absolute and relative conditions of the NRHO, together with the natural motion along one orbit. These specific initial conditions can also serve as a holding point until the chaser spacecraft receives the *GO Final Approach*. Utilizing this strategy ensures passive avoidance of collisions with the target, thanks to the relative periodic central manifold. Furthermore, in this study, the Lunar Gateway is assumed to be the target, and the chaser is the Orion spacecraft. As a result, the main characteristics of the Orion spacecraft, including $m = 21000\ kg$, $u_{max} = 29.3\ kN$, and $I_{sp} = 310\ s$, are incorporated.

[†] `https://stable-baselines3.readthedocs.io/en/master/index.html`



(a) Absolute synodic reference frame.

(b) Relative synodic reference frame.

Figure 4: Chaser and target spacecraft on the Southern 9:2 Resonant Near-Rectilinear Halo Orbit (NRHO) of the Earth-Moon system apolune at a relative distance of 200 meters, Keep-Out-Sphere (KOS) edge, during one orbital motion.

To ensure the safety of the approach and proper docking, the RVD maneuver must meet specific requirements. Therefore, the algorithm must take into account the following list of constraints:

- **Docking Port**: the final relative position and velocity *shall* be $\rho_f < 1\ m$ and $\dot{\rho}_f < 0.1\ m/s$, respectively;
- **Approach Corridor**: the chaser spacecraft *shall* remain within a truncated cone (Figure 5) with a half angle of $\beta = 20\ °$ for safety and vision-based navigation purposes. Assuming that the target docking port is located in its positive $\hat{\mathbf{t}}_2$ direction, the requirement for a truncated cone is expressed as follows:

$$\boldsymbol{\rho}^*(t) \cdot \hat{\mathbf{t}}_2 - \rho^*(t)\, cos(\beta) > 0 \quad \forall \delta y \geq 0, \forall t \tag{14}$$

$$\boldsymbol{\rho}^*(t) = \boldsymbol{\rho}(t) + \left[0,\ \frac{\rho_{f,max}}{\tan\beta},\ 0\right]^T \tag{15}$$

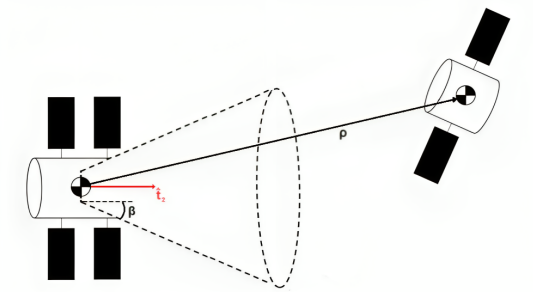


Figure 5: The approach corridor is represented by a truncated cone with a maximum semi-angle of β and a small frustum base designed to satisfy the docking port requirements.

Using a truncated cone, also known as a frustum,[20] provides a more realistic representation of the docking port.[1] Moreover, this model avoids singularities, which improves the convergence properties of the algorithm. The constraint involves an alternative relative position $\boldsymbol{\rho}^*$ from a point in the $-\hat{\mathbf{t}}_2$ direction. This vector is designed to meet the docking port requirements at the small frustum base, considering an angular distance from the approach axis of β. Additionally, since this requirement is defined in the target's body frame and is attitude-dependent, it is assumed that the target remains cooperative and aligned with the Earth-Moon Synodic frame, with $\hat{\mathbf{t}}_2 \equiv \hat{\mathbf{y}}$.

- **Maximum Time-Of-Flight**: the maneuver *shall* be accomplished within a restricted time of flight, therefore, $t < ToF_{max} = 100\ s$ must be satisfied;

- **Thrusters Performance**: the maximum thrust value specified in the data sheet *shall* be followed to formulate a feasible control action profile, ensuring that $u < u_{max} = 23.9\ kN$.

Optimal Control Problem Formulation

Taking into account the requirements mentioned above, it is simple to design an OCP: *"Find the control action profile that minimizes control effort and such that dynamics, initial conditions, maximum control action, maximum time of flight, approach corridor and final docking port requirements are respected"*. It can be expressed mathematically as follows:

$$\begin{aligned}
\min_{\mathbf{u}} \quad & J = \int_{t_0}^{t_f} ||\mathbf{u}\left(t\right)||dt \\
\text{s.t.} \quad & \dot{\mathbf{x}}\left(t\right) = \mathbf{f}\left(\mathbf{x}\left(t\right), \mathbf{u}\left(t\right)\right) \quad \forall t \\
& \mathbf{x}\left(t_0\right) = \mathbf{x}_0 \\
& u\left(t\right) < u_{max} \ \forall t \\
& t < ToF_{max} \\
& \boldsymbol{\rho^*}\left(t\right) \cdot \hat{\mathbf{t}}_2 - \rho^*\left(t\right) cos\left(\beta\right) > 0 \quad \forall \delta y \geq 0, \forall t \\
& \rho\left(t_f\right) - \rho_{f,max} < 0 \\
& \dot{\rho}\left(t_f\right) - \dot{\rho}_{f,max} < 0
\end{aligned} \tag{16}$$

The system's equations of motion, represented by $\mathbf{f}$, encompass the absolute target state, the relative chaser state, and its associated mass. In order to address this OCP using Meta-Reinforcement Learning (Meta-RL), it is essential to transform it into an MDP, as explained in the following subsection.

Markov Decision Process Formulation

The agent's goal in a MDP is: *"Find the control policy π^* that maximizes the discounted expected return of rewards received along a trajectory and such that the policy, the environment's dynamics, and the initial conditions are respected"*. The MDP is mathematically expressed as follows:

$$\begin{aligned}
\max_{\pi} \quad & J = \mathbb{E}_{\pi}\left[G_t \mid \mathbf{S}_t\right] \\
\text{s.t.} \quad & \mathbf{A}_t = \boldsymbol{\pi}\left(\mathbf{S}_t\right) \\
& \mathbf{S}_{t+1} = \phi\left(\mathbf{S}_t,\ \mathbf{A}_t\right) \\
& \mathbf{S}_0 \sim \mathcal{N}\left(\hat{\mathbf{S}}_0,\ \mathbf{C}\right)
\end{aligned} \tag{17}$$

The second constraint considers the dynamics of the MDP environment, resulting in the update of its state $\mathbf{S}_t$ at each time step t. To improve the robustness of the algorithm, the MDP, in contrast to the straightforward OCP formulation, can leverage domain randomization.[5] In this scenario, the dynamics of the spacecraft will be modified to include random processes and stochastic effects. Additionally, uncertainty will be introduced in the initial state by sampling it from a normal distribution, just as in the final restriction of Eq. 17. Moreover, it is crucial to note that operational constraints will be "translated" through the definition of the reward function. MDPs face challenges in handling constraints compared to OCPs. In MDPs, constraints should be represented as significant positive or negative rewards or as the conclusion of an episode.

State and Action Spaces The agent's neural networks utilize the *state space* as input to generate the action (Actor) or predict the state-value function (Critic). In spacecraft RVD problems, it is advantageous to incorporate both the absolute target and relative chaser states. The inclusion of the chaser's mass becomes necessary for optimizing fuel consumption in the reward function. To learn how to comply with time constraints, the remaining flight time $\Delta T_t = ToF_{max} - t$ is also taken into account. Moreover, observations include the action and reward from the previous time step, enabling LSTMs to update their dynamics.[13] Consequently, the state space $\mathbf{S} \in \mathbb{R}^{18}$ encompasses:

$$\mathbf{S}_t = \left[\mathbf{x}_t^T,\ \delta\mathbf{x}_t,\ m_t,\ \Delta T_t,\ \mathbf{u}_{t-1},\ R_{t-1}\right] \tag{18}$$

Normalizing the inputs of neural networks is essential to bring all input features to a comparable scale and improve numerical stability. Therefore, a simple *Min-Max normalization* is applied. The state is normalized to be within $\mathbf{S}_t^* \in [-1, +1]$ using the following method:

$$S_{t,i}^* = 2\left(\frac{S_{t,i} - S_{min,i}}{S_{max,i} - S_{min,i}}\right) - 1 \tag{19}$$

Once the problem is defined, the extremal components of $\mathbf{S}_t$ are identified. The actor network generates a set of all possible actions that the agent can perform in the environment, termed the *action space*. In this specific study, which focuses on a 3-degree-of-freedom (3DOF) spacecraft model, the action space is intended to depict thrust control. The suitable representation for this research case[32] includes an action space $\mathbf{A}_t \in \mathbb{R}^3$ and its unscaled control action $\mathbf{u}_t$, defined as:

$$\mathbf{A}_t = \left[\tilde{u}_{x,t},\ \tilde{u}_{y,t},\ \tilde{u}_{z,t}\right] \qquad\qquad \mathbf{u}_t = \sigma\tilde{\mathbf{u}}_t \tag{20}$$

The control action $\mathbf{u}_t$ serves as input to the spacecraft equations of motion. To maintain adherence to the maximum thrust value, the scaling coefficient is defined as $\sigma = u_{max}/\|\mathbf{1}_3\|$, since the output of the neural network is limited to the range of $[-1, 1]$.

Reward Function The reward signal's purpose is to convey the desired outcome to the agent, rather than specifying how it should be achieved. In scenarios where rewards are inherently sparse, such as the one described, RL tends to exhibit sub-par performance. To improve them, it is recommended to employ reward shaping techniques.[33] In such cases, one can make use of non-linear functions, such as logarithmic and exponential functions, leveraging their characteristic of increasing the first derivative near attractive or repulsive states. This contributes to a smoother appearance of the reward landscape. Additionally, squaring is used to enhance convergence performance.[34] The specific reward function employed is described below:

$$\begin{aligned} R_t = \alpha \log\left(\frac{\|\delta\mathbf{x}_t \oslash \delta\mathbf{x}_{max}\|}{\|\mathbf{1}_6\|}\right)^2 - \lambda\exp\left(\frac{acos\left(\hat{\boldsymbol{\rho}}_t^* \cdot \hat{\mathbf{t}}_2\right)}{\pi}\right)^2 - \gamma\exp\left(\frac{u_t}{u_{max}}\right)^2 \\ + \left(\rho_t < \rho_{f,max} \wedge \dot{\rho}_t < \dot{\rho}_{f,max} \Rightarrow\right)\zeta - \left(\boldsymbol{\rho}_t^* \cdot \hat{\mathbf{t}}_2 - \rho_t^* cos\left(\beta\right) < 0 \Rightarrow\right)\kappa \end{aligned} \tag{21}$$

The reward function consists of two main parts:

- The dense rewards provide ongoing feedback within each episode, featuring a primary bonus component for approaching the goal state $\delta\mathbf{x} = \mathbf{0}$, a penalty for the distance from the approach corridor as the second component, and a penalty for the control effort as the third. All inputs to these non-linear functions are normalized to be within the range $[0, 1]$. However, particular attention during reward engineering has been dedicated to the first dense component (the logarithmic one). The incorporation of the Hadamard operator, L_2-norm, and division by $\|\mathbf{1}_6\|$ aims to generate a scalar value that ensures

equal importance for each state component and offers no reward for the greatest distances and high velocities;

- Episodic rewards deliver feedback at the conclusion of a task, featuring a primary bonus component when the docking port requirements are satisfied and a secondary penalty component when a collision with the approach corridor occurs. These components also signal the completion of the episode to the environment.

The reward shaping logic, using logarithmic and exponential functions (with numerical values provided solely for demonstration), is summarized and illustrated in Figure 6.

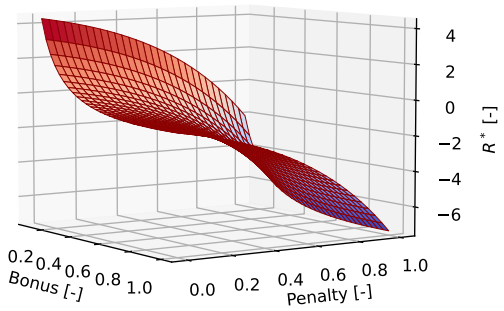


Figure 6: Graphic illustration of the reward shaping employing logarithmic and exponential functions. The bonus and penalty axes represent their input arguments.

Environment's Dynamics In the realm of an RVD problem, given its continuous nature, it is more appropriate to represent the transition function using a generative model, expressed as $\mathbf{S}_{t+1} = \phi\left(\mathbf{S}_t,\ \mathbf{A}_t\right)$. This model facilitates updates for both the absolute target and the relative chaser state by integrating the CR3BP and the RCR3BP. The equations of motion, denoted by $\mathbf{f}$, undergo discretization into discrete steps through integration, taking into account a constant control action. Additionally, the model includes updates for the residual flight time, as well as the action and reward from the preceding time step. At the beginning of each MDP episode, a random initial state $\mathbf{S}_0$ is selected. This randomness specifically influences the initial conditions of the relative chaser dynamics $\delta\mathbf{x}_0 = [\boldsymbol{\rho},\ \dot{\boldsymbol{\rho}}]$ and its initial mass m_0, derived from three as broad normal distributions as possible. Standard deviations are configured as follows: $\sigma_{\boldsymbol{\rho}_0} = 0.1\rho_0$, $\sigma_{\dot{\boldsymbol{\rho}}_0} = 0.5\ m/s$, and $\sigma_{m_0} = 100\ kg$. Furthermore, the environment sends out a *done* when the flight time exceeds its highest value (ToF_{max}).

Process noise is included in the chaser spacecraft dynamics to account for unmodeled accelerations (e.g. SRP, the gravity of other celestial bodies, thrust uncertainty, etc.). Hence, the equations of motion for the chaser, taking into account $\mathbf{h}$ as the non-autonomous RCR3BP, can be expressed as:

$$\delta\dot{\mathbf{x}}\left(t\right) = \mathbf{h}\left(\delta\mathbf{x}\left(t\right), \mathbf{u}\left(t\right), \mathbf{x}^T\left(t\right)\right) + \mathbf{w}\left(t\right) \qquad \mathbf{w}\left(t\right) \sim \mathcal{N}\left(\mathbf{0},\ diag\left(\mathbf{0}_3, \mathbf{10_3^{-8}}\right)\right) \tag{22}$$

$\mathcal{N}$ represents a *Gaussian White Noise (GWN)*, featuring a covariance that matches the non-dimensional magnitude ($\sigma = 10^{-4}$) of typical NRHO disturbances. The robustness of the algorithm is further enhanced by taking into account and modeling the possibility of a random failure in the control action. At the beginning of each episode, the environment randomly selects a control direction in which to reduce the magnitude of the thrust by 50% or have no failure. Each scenario has an equal probability of 25%, as illustrated in Table 1.

Table 1: Multinomial discrete distribution modeling a breakdown in 6DOF control.

Failure Type	$0.5 \cdot u_x$	$0.5 \cdot u_y$	$0.5 \cdot u_z$	No Failure
Probability	0.25	0.25	0.25	0.25

Artificial Neural Networks Architecture and Hyperparameters

The architecture of the model is inspired by *Stable-Baseline3 (SB3) Recurrent PPO*, featuring layers of LSTM followed by layers of MLP extractor, as shown in the Table 2. The use of a combination of LSTM networks and MLP extractor layers is a common approach in RL.[35] This is because LSTMs are particularly adept at capturing sequential dependencies, but may not be able to provide a suitable final representation. An MLP-layer at the end can be used to condense the variable-length sequence of LSTM outputs into a fixed-size representation, making it easier to make decisions or take actions. This combination of sequence modeling (handled by LSTMs) and representation learning (handled by MLPs) allows the model to effectively use sequential information in RL tasks. In this study, the size of the neural network is determined through experimentation to guarantee a high level of generalization and to accurately fit the optimal solution.

Table 2: Architecture of the Artificial Neural Network (ANN) implemented in the Actor (A) and Critic (C) networks of the agent.

Layer	Neurons (A/C)	Activation	Type
Hidden 1	256/256	Tanh	LSTM
Hidden 2	256/256	Tanh	LSTM
Hidden 3	64/64	Tanh	MLP
Hidden 4	64/64	Tanh	MLP

The hyperparameters used in this study are described in Table 3. These selected hyperparameters closely align with the default settings[9] and required only minor adjustments determined by trial and error. It is crucial to note that gradient clipping has become a highly significant hyperparameter for LSTM networks, given their susceptibility to catastrophic forgetting.[36]

Table 3: Hyperparameters for Actor's training using Proximal Policy Optimization (PPO) and Critic's training through Mean Squared Error (MSE) algorithms.

	Actor PPO	Critic MSE
Batch Size	64	64
Number Epochs	10	10
GAE Lambda (λ)	1	
Discount Factor (γ)	0.99	
Clip Parameter (ε)	0.1	
Learning Rate (α)	0.00005	0.00005
Entropy Coefficient	0.0001	
Clip Gradient	0.1	0.1

NUMERICAL RESULTS

Reinforcement learning encompasses two primary phases: training and testing. In the context of spacecraft G&C applications, ground-based high-fidelity simulators should be used for training, with testing of the policy occurring directly on-board during operations. The equations of motion are solved in the MDP environment with the *scipy.integrate* library in *Python*, using the non-stiff Adam predictor-corrector, called *LSODA*, method with relative and absolute tolerances of $2.22 \cdot 10^{-14}$. The integration time step to convert continuous dynamics to an MDP is set to $dt = 0.5\ s$. The device used for this work is a Laptop PC with an Intel(R) Core(TM) i7-8565U CPU 1.99 GHz and 16.0 GB of RAM.

Training

The algorithm is trained for 7 million steps, which corresponds to 40.5 days in the simulated environment. This training duration spans 3.2 days, and the resulting training curve, which illustrates the mean discounted cumulative reward of trajectory roll-outs in relation to the learning step, is shown in Figure 7. Furthermore, a standard PPO is also employed for training a simple fully connected MLP-policy. This serves as a benchmark

for Meta-RL performance. The MLP-policy employed in this comparison shares identical environmental setup, network width and depth, and hyperparameters. The results are presented within the same figure.

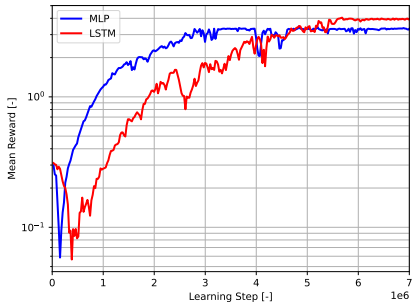


Figure 7: Mean discounted cumulative rewards of the trajectory roll-outs during the training phases.

Both learning curves exhibit a plateau, indicating the effectiveness of the training. The LSTM-agent, which has six times the number of parameters compared to the MLP-agent, clearly requires more training steps and wall-time. However, it consistently achieves higher cumulative rewards. Meta-RL employs a recurrent agent, which allows the policy to adapt and modify its actions within the same episode based on the given inputs. Consequently, during training, an MLP-policy acquires a generally effective strategy, while an LSTM-policy also learns to dynamically adjust itself to optimize performance for each task in a specific way.

Testing

The LSTM-policy, which has been trained, is currently being evaluated in the environment, taking into account the same uncertainty regarding the initial conditions of the MDP. The deployment results, depicted in Figures 8 and Figure 9, are derived from a single episode.

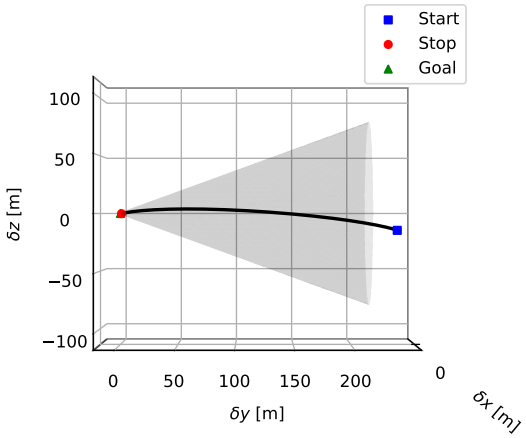
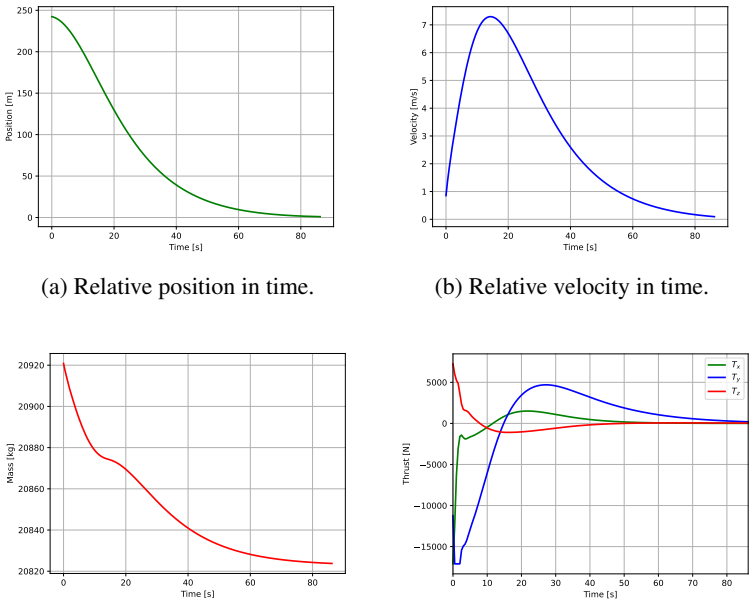


Figure 8: Trained LSTM-policy deployed in the environment and tested for a single episode. Trajectory inside the approach corridor starting from a randomized initial condition.

Figure 8 shows that the chaser spacecraft, originating from a random initial condition, effectively reaches the target ($\delta\mathbf{x}_f = \left[0.612\ m,\ 0.085\ m/s\right]$), fulfilling safety requirements. The confirmation of adherence to the maximum flight time constraint is visible in Figures 9a, 9b. Figure 9c specifically displays the mass profile over time and, according to Tsiolkovsky's equation, a $\Delta V = 13.113\ m/s$ is required. Furthermore, Figure 9d shows compliance with the restriction on maximum control action. It is essential to emphasize that the policy adeptly handles unforeseen thrust failures by learning to utilize half of the maximum thrust.



(a) Relative position in time.

(b) Relative velocity in time.

(c) Total mass in time.

(d) Control action in time.

Figure 9: Trained LSTM-policy deployed in the environment and tested for a single episode. Relative position, relative velocity, mass, and control action along the trajectory of Figure 8.

Testing is currently underway on the trained LSTM-policy and MLP-policy, generating 500 trajectories, each with a different initial state due to the randomness of the MDP. This Monte Carlo campaign aims to evaluate the robustness and computational efficiency of Meta-RL as an autonomous G&C algorithm for on-board deployment. Additionally, it involves a performance comparison with the non-recurrent policy. The trajectories are shown in Figure 10, and a summary of the results is presented in Table 4. To evaluate the fuel optimality of the Meta-RL approach in this study, a state-of-the-art OCP direct pseudospectral method[37] is utilized as a reference. The results of the Monte Carlo campaign with the OCP method, considering the same initial condition uncertainties as the MDP but not accounting for process noise and actuator malfunctions, are presented in the same table.

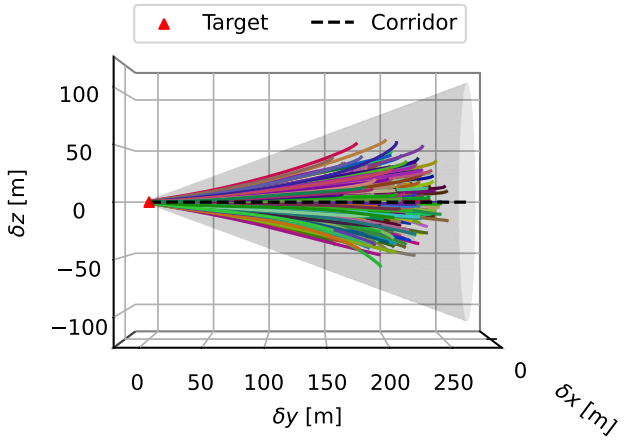


Figure 10: Trained LSTM-policy deployed in the environment and tested for a batch of episodes. Trajectories inside the approach corridor starting from a randomized initial condition.

Table 4: Monte Carlo performance of the trained LSTM-policy, trained MLP-policy tested in the environment, alongside the OCP pseudospectral solution.

	LSTM-policy: $\mu \pm \sigma$	**MLP-policy:** $\mu \pm \sigma$	**OCP:** $\mu \pm \sigma$
Success Rate: S_r $[\%]$	100	100	
Final Position ρ_f $[m]$	0.561 ± 0.096	0.719 ± 0.051	
Final Velocity $\dot{\rho}_f$ $[m/s]$	0.082 ± 0.004	0.088 ± 0.003	
Fuel Consumption ΔV $[m/s]$	11.981 ± 0.495	18.669 ± 0.537	11.153 ± 0.772
Time of Flight ToF $[s]$	82.571 ± 1.571	61.571 ± 1.274	76.619 ± 2.115
CPU-Time Step dt_{CPU} $[ms]$	2.334 ± 0.138		

The LSTM-policy and the MLP-policy demonstrate success in all situations, obtaining a $S_r = 100\%$, where S_r is the proportion of trajectories that meet all requirements. In spite of the numerous uncertainties and the potential for failure, the policies have effectively guided the spacecraft to the intended final state with minimal deviation in every instance. However, the MLP-policy has much higher fuel consumption, whereas the LSTM-policy is near-optimal. Recurrent layers generate an adaptive policy that can more effectively optimize the trajectory of each episode. Therefore, the LSTM policy showed robustness, achieving the highest possible success rate, near-optimality, and computational efficiency ($dt_{CPU} = 2.33\ ms$, as illustrated in Figure 13) in Monte Carlo simulations. This makes it well-suited for real-time on-board applications.

The trained LSTM-policy is now subjected to an additional testing phase, encompassing a set of 500 trajectories in an enhanced environment similar to the one described. The enhanced environment eliminates process noise and introduces into the chaser dynamics the following elements: the Sun's fourth-body gravitational effect, considering the Bicircular Restricted Four-Body Problem (BR4BP),[38] and the Solar Radiation Pressure (SRP), modeled with the cannonball method.[39] The objective is to show that the deployed policy can withstand unmodeled accelerations by introducing noise during training. The results are in Figure 11 and Table 5, which demonstrate that the LSTM-policy is effective in handling unmodeled accelerations.

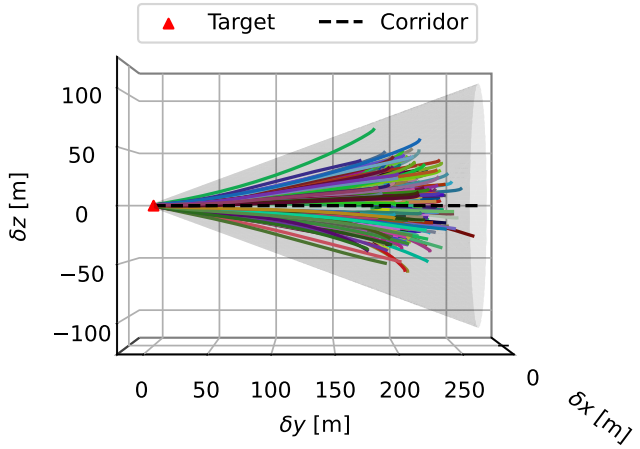


Figure 11: Trained LSTM-policy deployed in an environment with unmodeled accelerations, and tested for a batch of episodes. Trajectories inside the approach corridor starting from a randomized initial condition.

Table 5: Monte Carlo performance of the trained LSTM-policy tested with unmodeled accelerations.

Success Rate: $S_r = 100\%$	**Results MC:** $\mu \pm \sigma$
Final Position ρ_f $[m]$	0.497 ± 0.084
Final Velocity $\dot{\rho}_f$ $[m/s]$	0.094 ± 0.001
Fuel Consumption ΔV $[m/s]$	12.263 ± 0.524
Time of Flight ToF $[s]$	81.713 ± 0.955

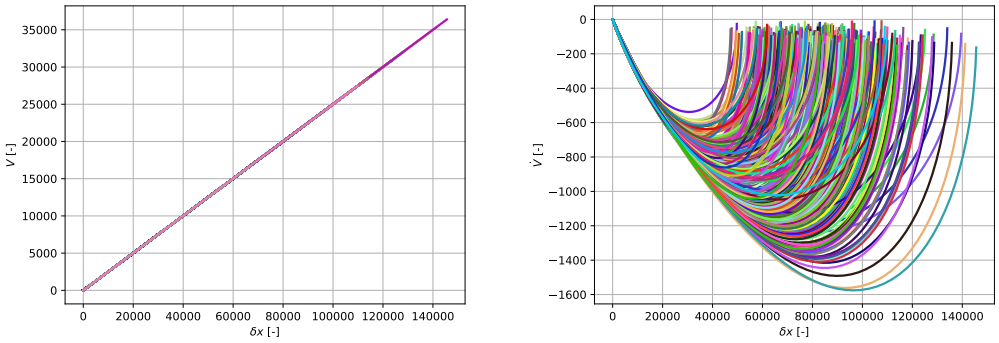
STABILITY ANALYSIS

A G&C algorithm should ensure the asymptotic stability (Def. 1) of the controlled system in order to achieve mission success, avoiding any erratic behavior and guaranteeing predictable vehicle performance, while also avoiding potential accidents or deviations from the intended trajectory. In the literature, there are a few analytical methods[30] that address the convergence of RL algorithms with function approximators. RL methods involve optimizing a criterion, inherently constraining states along the optimal trajectory, and stabilizing the system. However, in this section, the *Lyapunov's Direct Method*[40] is employed to highlight system stability from the standpoint of non-linear equations of motion. This approach, based on *Lyapunov's Second Stability Theorem* (Thm. 1), is advantageous as it enables the assessment of the stability of the controlled system without the need for linearization around an equilibrium solution.

In this setting, the RCR3BP is viewed as an autonomous dynamical system. This assumption holds provided that the target's absolute state $\mathbf{x}^T$ remains constant during the maneuver. This feasibility arises from the chaser's time of flight being considered negligible in comparison to the orbital target period. With these assumptions, a positive-definite quadratic, "energy-error-like", *Candidate Lyapunov Function* in $\mathbb{R}^6$ can be employed as follows:

$$V(\delta\mathbf{x}) = \delta\mathbf{x}^T\mathbf{P}\delta\mathbf{x} \tag{23}$$

The diagonal weight matrix considered is $\mathbf{P} = diag(\mathbf{0.5}_6)$, and the equilibrium point is $\delta\mathbf{x}^* = \mathbf{0}$, representing the desired outcome of the RVD maneuver. This equilibrium point would result in $\mathbf{h}(\delta\mathbf{x}^*,\ \mathbf{u}) = \mathbf{0}$ in the RCR3BP equations. The stability assessment utilizes a Monte Carlo technique by randomly selecting the chaser's initial conditions, as defined for the MDP. A set of 500 trajectories is simulated with the trained LSTM-policy deployed in the environment, and the numerical values of $V(\delta\mathbf{x})$ and $\dot{V}(\delta\mathbf{x})$ are determined along these trajectories. The results are shown in Figure 12, where the L_2-norm of the relative state vector serves as the abscissa in each graph. The developed LSTM-policy demonstrates asymptotic stability, as indicated by the identification of a Candidate Lyapunov Function that meets the criteria outlined in Lyapunov's Second Stability Theorem.



(a) Candidate Lyapunov Function.

(b) Candidate Lyapunov Function time-derivative.

Figure 12: Trained LSTM-policy deployed in the environment and tested for a batch of episodes. Candidate Lyapunov Function and its time-derivative with regard to the distance from the equilibrium point.

CONCLUSIONS

This work demonstrates the efficacy of Meta-RL in effectively handling the formulated MDP. More specifically, it is applied to address the final approach and docking scenario of a spacecraft within a Southern L_2 9:2 Resonant Near-Rectilinear Halo Orbit (NRHO) in the Earth-Moon system. Despite a great deal of uncertainty in initial conditions, process noise, and actuator malfunctions, all objectives have been achieved. The learning curve shows that the LSTM-agent training phase was successful, as it reached the expected plateau. During the testing phase, the LSTM-agent is evaluated through a Monte Carlo campaign. The results demonstrate

that the trained policy consistently meets all imposed requirements on the entire set of generated trajectories. As a result, the Meta-RL algorithm has proven its ability to effectively learn a G&C policy tailored for RVD maneuvers within the cislunar space. Furthermore, the reliability of the method has been confirmed through Monte Carlo simulations conducted not only in the training environment, but also in an enhanced setting that incorporates unmodeled dynamics not present during the learning phase. The computational efficiency for on-board execution of the LSTM-policy has been demonstrated, its fuel optimality has been verified through a comparison with a state-of-the-art OCP pseudospectral direct solution, and the asymptotical stability of the controlled system has been established from a nonlinear equations of motion perspective. Consequently, it can be considered a promising solution for autonomous G&C applications. For the sake of comparison, a fully MLP-agent has also been trained. It successfully learns the proposed MDP, reaching the expected plateau during training, and achieving the highest possible success rate in the Monte Carlo testing campaign. However, the LSTM-policy outperforms the MLP-policy by acquiring greater cumulative rewards during training and demonstrating significantly higher fuel efficiency during testing. The internal memory of the LSTM-policy enables it to generate an adaptive G&C policy better suited to the optimal solution of the problem. This highlights the evident success of recurrent policies over non-recurrent ones when addressing tasks characterized by significant parameter randomization.

ACKNOWLEDGMENT

The first author acknowledges Prof. John R. Martin, Dr. Adam Hermann, and Mark Stephenson from the University of Colorado Boulder for their valuable insights on reinforcement learning during the development of this research. The author also expresses special thanks to Prof. Hanspeter Schaub from the University of Colorado Boulder and Ann & H.J. Smead Department of Aerospace Engineering Sciences for providing the opportunity to conduct a visiting research period at the Autonomous Vehicle Systems (AVS) laboratory through the Europe-Colorado Program scholarship.

REFERENCES

[1] W. Fehse, *Automated Rendezvous and Docking of Spacecraft*. Cambridge University Press, 11 2003.
[2] Y. Xie, C. Chen, T. Liu, and M. Wang, *Guidance, Navigation, and Control for Spacecraft Rendezvous and Docking: Theory and Methods*. Springer Singapore, 2021.
[3] E. Capello, F. Dabbene, G. Guglieri, and E. Punta, ""Flyable" Guidance and Control Algorithms for Orbital Rendezvous Maneuver," *SICE Journal of Control, Measurement, and System Integration*, Vol. 11, 2018, pp. 14–024.
[4] M. Tipaldi, R. Iervolino, and P. R. Massenio, "Reinforcement learning in spacecraft control applications: Advances, prospects, and challenges," *Annual Reviews in Control*, Vol. 54, 1 2022, pp. 1–23.
[5] Y. Song, A. Romero, M. Müller, V. Koltun, and D. Scaramuzza, "Reaching the limit in autonomous racing: Optimal control versus reinforcement learning," *Science Robotics*, 2023.
[6] H. A. Pierson and M. S. Gashler, "Deep learning in robotics: a review of recent research," *Advanced Robotics*, Vol. 31, 8 2017, pp. 821–835.
[7] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. A. Sallab, S. Yogamani, and P. Perez, "Deep Reinforcement Learning for Autonomous Driving: A Survey," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 23, 6 2022, pp. 4909–4926.
[8] D. Izzo, M. Märtens, and B. Pan, "A survey on artificial intelligence trends in spacecraft guidance dynamics and control," *Astrodynamics*, Vol. 3, 12 2019, pp. 287–299.
[9] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction Second edition, in progress*. MIT Press, 2019.
[10] N. Schweighofer and K. Doya, "Meta-learning in Reinforcement Learning," *Neural Networks*, Vol. 16, No. 1, 2003, pp. 5–9.
[11] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick, "Learning to reinforcement learn," *ArXiv*, 11 2016.
[12] J. B. R. Vuorio, E. Z. Liu, Z. Xiong, L. Zintgraf, C. Finn, and S. Whiteson, "A Survey of Meta-Reinforcement Learning," *ArXiv*, 1 2023.
[13] S. Hochreiter, A. S. Younger, and P. R. Conwell, "Learning to learn using gradient descent," *Lecture Notes in Computer Science*, Vol. 2130, Springer Verlag, 2001, pp. 87–94.
[14] B. Gaudet, R. Linares, and R. Furfaro, "Deep reinforcement learning for six degree-of-freedom planetary landing," *Advances in Space Research*, Vol. 65, 4 2020, pp. 1723–1741.
[15] L. Federici and A. Zavoli, "Robust Design of Interplanetary Trajectories under Severe Uncertainty via Meta-Reinforcement Learning," *73rd International Astronautical Congress (IAC), Paris*, 2022.
[16] G. Calabrò, "Adaptive guidance via Meta-Reinforcement Learning: ARPOD for an under-actuated CubeSat," Master's thesis, Politecnico Di Milano, 2022.
[17] L. Federici, A. Scorsoglio, A. Zavoli, and R. Furfaro, "Meta-reinforcement learning for adaptive spacecraft guidance during finite-thrust rendezvous missions," *Acta Astronautica*, Vol. 201, 12 2022, pp. 129–141.

[18] B. Gaudet, R. Linares, and R. Furfaro, "Six Degree-of-Freedom Hovering over an Asteroid with Unknown Environmental Dynamics via Reinforcement Learning," *AIAA Scitech 2020 Forum*, American Institute of Aeronautics and Astronautics, 1 2020.

[19] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *ArXiv*, 7 2017.

[20] D. E. Lee, "Gateway Destination Orbit Model: A Continuous 15 Year NRHO Reference Trajectory," tech. rep., NASA Johnson Space Center, 2019.

[21] S. Lizy-Destrez, B. L. Bihan, A. Campolo, and S. Manglativi, "Safety Analysis for Near Rectilinear Orbit Close Approach Rendezvous in the Circular Restricted Three-Body Problem.," *68 th International Astronautical Congress (IAC), Adelaide*, 2017, pp. 25–29.

[22] F. Colombi, A. Colagrossi, and M. Lavagna, "Characterisation of 6DOF natural and controlled relative dynamics in cislunar space," *Acta Astronautica*, Vol. 196, 7 2022, pp. 369–379.

[23] G. Bucchioni and M. Innocenti, "Rendezvous in Cis-Lunar Space near Rectilinear Halo Orbit: Dynamics and Control Issues," *Aerospace*, Vol. 8, 2021.

[24] A. C. Lorenzo Bucci and M. Lavagna, "Rendezvous in Lunar Near Rectilinear Halo Orbits," *Advances in Astronautics Science and Technology*, Vol. 1, 9 2018, pp. 39–43.

[25] G. Franzini and M. Innocenti, "Relative motion dynamics in the restricted three-body problem," *Journal of Spacecraft and Rockets*, Vol. 56, 2019, pp. 1322–1337.

[26] A. Colagrossi, M. Lavagna, J. D. Biggs, and P. Masarati, *Absolute and Relative 6DOF Dynamics, Guidance and Control for Large Space Structures in Cislunar Environment*. PhD thesis, Politecnico di Milano, 2019.

[27] S. Hochreiter and J. U. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, Vol. 8, 1997.

[28] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-Dimensional Continuous Control Using Generalized Advantage Estimation," *ArXiv*, 6 2015.

[29] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous Methods for Deep Reinforcement Learning," *ArXiv*, 2 2016.

[30] R. S. Sutton, D. Mcallester, S. Singh, and Y. Mansour, "Policy Gradient Methods for Reinforcement Learning with Function Approximation," *ArXiv*, 2015.

[31] L. Engstrom, A. Ilyas, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph, and A. Madry, "Implementation Matters in Deep Policy Gradients: A Case Study on PPO and TRPO," *ArXiv*, 5 2020.

[32] S. Bonasera, C. J. Sullivan, N. Bosanac, J. W. Mcmahon, I. Elliott, and N. Ahmed, *Designing Impulsive Station-Keeping Maneuvers near a Sun-Earth L2 Halo Orbit via Reinforcement Learning*. PhD thesis, University of Colorado Boulder, 2021.

[33] A. Ng, D. Harada, and S. J. Russell, "Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping," *International Conference on Machine Learning*, 1999.

[34] S. Bonasera, *Incorporating Machine Learning into Trajectory Design Strategies in Multi-Body Systems*. PhD thesis, University of Colorado Boulder, 2022.

[35] H. Sak, A. Senior, and F. Beaufays, "Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition," *ArXiv*, 2 2014.

[36] S. Monika and G. Alexander, "A Study on Catastrophic Forgetting in Deep LSTM Networks," *Artificial Neural Networks and Machine Learning - ICANN 2019: Deep Learning*, 09 2019, pp. 714–728.

[37] I. M. Ross and M. Karpenko, "A review of pseudospectral optimal control: From theory to flight," *Annual Reviews in Control*, Vol. 36, No. 2, 2012, pp. 182–197.

[38] B. P. Mccarthy and K. C. Howell, "Quasi-Periodic Orbits in the Sun-Earth-Moon Bicircular Rrestricted Four-Body Problem," *31st AAS/AIAA Space Flight Mechanics Meeting*, 2021.

[39] P. W. Kenneally, "High Geometric Fidelity Solar Radiation Pressure Modeling via Graphics Processing Unit," Master's thesis, University of Colorado Boulder, 2016.

[40] H. Schaub and J. L. Junkins, *Analytical Mechanics of Space Systems*. Reston, VA: AIAA Education Series, 4th ed., 2018.

APPENDIX A: COMPUTATIONAL EFFICIENCY

In the course of the Monte Carlo simulation depicted in Figure 10, the computational efficiency of the policy has been assessed at each step. The policy evaluation had an average CPU-Time per step of $2.334\ ms$, which is equivalent to $18.672\ ms$ ($53.68\ Hz$) when taking into account the clock frequency of $250\ MHz$ on a LEON3FT[‡] on-board spacecraft processor in an average performance configuration. This value implies that the algorithm can be implemented in Orion's real flight software, which typically runs the GNC blocks[§] at a rate of $40\ Hz$. The method's computational efficiency is attributed to the fact that, once the agent has been trained on a high-fidelity simulator on the ground, the on-board policy evaluation during testing only requires a small number of matrix multiplications.

[‡] https://www.gaisler.com/index.php/products/boards/gr712rc-board?task=view&id=364

[§] https://ntrs.nasa.gov/api/citations/20190000011/downloads/20190000011.pdf

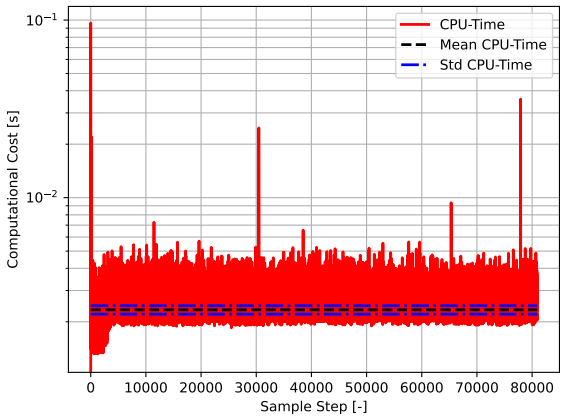


Figure 13: Trained LSTM-policy deployed in the environment and tested for a batch of episodes. CPU-Time for each policy evaluation step during testing.

APPENDIX B: LYAPUNOV'S STABILITY THEORY

Definition 1 (Lyapunov's Stability)**.** *Consider an autonomous non-linear dynamics system described by:*

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}), \ \mathbf{f}(\mathbf{x}^*) = 0 \tag{24}$$

Where $\mathbf{x}^*$ is an isolated equilibrium point. Then the equilibrium point is said to be:

1. ***Stable**: for $\forall \varepsilon > 0$, there $\exists \partial > 0$ such that if $\left\|\mathbf{x}(0) - \mathbf{x}^*\right\| < \partial$ then $\left\|\mathbf{x}(t) - \mathbf{x}^*\right\| < \varepsilon \ \forall t > 0$;*
2. ***Asymptotically Stable**: the equilibrium point $\mathbf{x}^*$ is stable and $\left\|\mathbf{x}(t) - \mathbf{x}^*\right\| \to 0$ as $t \to \infty$.*

The definitions describe a local stability in the vicinity of the equilibrium point; if the region ∂ is unbounded, the definitions describe a global stability.

Theorem 1 (Lyapunov's Second Stability Theorem)**.** *Consider an autonomous non-linear dynamics system described by:*

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}), \ \mathbf{f}(\mathbf{x}^*) = 0 \tag{25}$$

Where $\mathbf{x}^*$ is an isolated equilibrium point. If there exists in some finite neighborhood D of the equilibrium point $\mathbf{x}^*$, a scalar function $V(\mathbf{x})$ with a continuous first partial derivative with respect to $\mathbf{x}$ such that the following conditions hold:

$$\begin{aligned} &(i) \quad V(\mathbf{x}) > 0, \ \forall \mathbf{x} \neq \mathbf{x}^* \in D \wedge V(\mathbf{x}^*) = 0 \\ &(ii) \quad \dot{V}(\mathbf{x}) < 0, \ \forall \mathbf{x} \neq \mathbf{x}^* \in D \wedge \dot{V}(\mathbf{x}^*) \leq 0 \end{aligned} \tag{26}$$

*Then the system is said to be **asymptotically stable**. If D includes all possible states, the system is said to be **globally asymptotically stable**. If $\dot{V}(\mathbf{x}) \leq 0 \ \ \forall \mathbf{x} \in D$ then the system is said to be **stable**.*