17th International Conference on Space Operations, Dubai, United Arab Emirates, 6 - 10 March 2023.

SpaceOps-2023, ID 475

On-board Guidance and Control for low-thrust orbit transfers using Deep Reinforcement Learning

Luca Romanelli*, Matteo Stoisa†, Federica Paganelli Azza‡, Pietro De Marchi§ and Mattia Varile¶
AIKO S.r.l., Via Dei Mille 22, 10123, Torino, Italy

The increasing number of active satellites and the growing diversity of stakeholders in the space industry have emphasized the need for increased autonomy to improve performance and reduce human intervention. Autonomous Guidance and Control (G&C) systems are a crucial technology for achieving flexibility, robustness, and scalability in a wide range of applications, including low-thrust orbit transfer, orbit phasing, and rendezvous. Although autonomous G&C systems can be applied to both deep space missions and planetary environments, developing them effectively remains a challenge due to factors such as model uncertainties and environmental perturbations. This research explores the use of Deep Reinforcement Learning (DRL) algorithms for maneuvering a spacecraft to reach a target orbit while minimizing flight time and fuel consumption. The DRL-based approach is well suited to addressing the complex nonlinear dynamics of low-thrust G&C and can adapt to different initial conditions. The proposed solution is evaluated in a Low Earth Orbit scenario, and its performance is benchmarked against existing literature. This validation serves as a foundation for further exploration of more complex applications such as the management of satellite constellations and autonomous maneuvering during deep space missions.

I. Introduction

In recent years, there has been a growing interest in using low-thrust propulsion systems for orbit transfers in Low Earth Orbit (LEO). These systems offer several advantages over traditional high-thrust systems, such as lower fuel consumption and longer lifetime. However, the use of low-thrust propulsion also poses new challenges for autonomous G&C systems [1].

The state-of-the-art in G&C for low-thrust orbit transfers in LEO can be broadly divided into model-based and model-free methods [2–4]. Model-based approaches involve the use of mathematical models to represent the dynamics of the system and generate optimal control inputs. On the other hand, model-free approaches involve learning from experience and do not rely on explicit mathematical models.

These approaches involve formulating the problem as a mathematical optimization, where the goal is to find the control inputs that minimize a performance metric such as fuel consumption or time of flight (TOF). Different optimization algorithms are then used to find the optimal solutions, such as Sequential Quadratic Programming, Sliding Mode Control, Model Predictive Control, Genetic Algorithms, and Particle Swarm Optimization.

Another method that has gained attention in recent years is the use of Artificial Intelligence (AI) techniques, such as Reinforcement Learning (RL) [5]. These approaches have been used to address the challenges of G&C for low-thrust orbit transfers in LEO by learning from data and adapting to changing conditions. In particular, Deep Reinforcement Learning (DRL) has gained attention as a powerful alternative to classical algorithms for finding optimal controllers for nonlinear and possibly stochastic dynamics that are unknown or highly uncertain [6].

The Markov Decision Process (MDP) [7] is a valuable framework for modeling sequential decision problems to achieve long-term goals. Reinforcement Learning is a powerful technique for solving MDPs, in which the agent makes sequential decisions by continually interacting with the environment. The agent's ultimate goal is to discover an optimal

*Corresponding author, luca.romanelli@aikospace.com
†matteo@aikospace.com
‡federica@aikospace.com
§pietro@aikospace.com
¶mattia@aikospace.com

policy that maximizes the cumulative reward. In RL, the mapping between states and actions is typically stored in a tabular form, which can be impractical, mainly when the state space is ample, or the action space is continuous. Combined with some form of function approximator – such as Deep Neural Networks (DNN) – model-free Deep Reinforcement Learning (DRL) is capable of making intelligent sequential decisions in challenging environments. In this work, we propose a model-free DRL algorithm for a spacecraft's autonomous maneuvering during a LEO orbit transfer. This method is particularly suitable for tackling the complex nonlinear dynamics of low-thrust G&C and allows for dealing with external perturbations.

The paper is organized as follows. The second section presents the mathematical models adopted to describe the problem and provides its formulation from an RL perspective. The third section details the implemented method and applies it to reference maneuvers, comparing the obtained results with the ones available in the literature. Final remarks and future work are illustrated in the last section.

II. Problem statement

A. Orbit transfer problem

The low-thrust orbit transfer problem is defined as the identification of a smooth and continuous spacecraft trajectory that meets specific initial and final conditions for two different orbits. While computing these transfer trajectories, an emphasis is placed on optimizing resource usage to minimize fuel consumption and/or time of flight (TOF). The initial and target orbits are given in terms of their classical orbit elements (COEs), which are the semi-major axis a, the eccentricity e, the inclination i, the right ascension of the ascending node Ω, the argument of periapsis ω and the true anomaly θ. The transfer problem takes therefore as inputs the initial orbit $oe_0 = [a_0, e_0, i_0, \Omega_0, \omega_0, \theta_0]$, the target one $oe_T = [a_T, e_T, i_T, \Omega_T, \omega_T]$ and the electric propulsion system specifications. The true anomaly of the target orbit is not included in the description of the desired orbital state as this work investigates LEO orbit correction maneuvers and a target anomaly is not relevant to the description of the transfer. It is important to note that the vector specifying the target elements should be defined according to the maneuver to be executed, meaning that, in some cases, not all parameters are targeted, and a subset of them may remain free.

Since the problem is formulated as an optimal control one, the optimization output is the control trajectory that describes the thrusting profile in terms of magnitude and direction needed to reach the final orbit. In particular, the control action is modeled in terms of thrusting acceleration given in the Radial, Transverse, and Normal (RTN) reference frame, and it is expressed as:

$$\vec{f}_{thrust} = \frac{T}{m}\vec{u} = \frac{T_{max}\eta}{m}\begin{bmatrix} u_r \\ u_t \\ u_n \end{bmatrix} \tag{1}$$

where m is the mass of the body subject to the acceleration and T is the thrust value, equal to the maximum available thrust T_{max} multiplied by the thrust throttle $0 \leq \eta \leq 1$. The unit vector $\vec{u}$ represents the thrusting direction. The flow rate of the fuel mass is modeled according to [8] and included in the equations of motion used to describe the spacecraft and presented in the following subsection.

B. Spacecraft motion

Since a LEO scenario is considered, the motion of the spacecraft about the Earth could be described through its COEs. The perturbing acceleration f is given in the RTN frame and models both the propulsion system acceleration and the external perturbation effects. The spacecraft is therefore modeled by considering only its center of mass and assuming that its attitude is always aligned with the orientation prescribed by the control action. This simplification is considered valid in the specific scenario under study, where the control of thrust and attitude are decoupled as they operate at different frequencies.

The instantaneous rate of change of each orbital element due to an assigned acceleration vector in the RTN frame

can be expressed through the Gauss' Variational Equations (GVE) [9] as:

$$\begin{aligned}
\frac{da}{dt} &= \frac{2a^2}{h}\left(e\sin\theta f_r + \frac{p}{r}f_t\right)\\
\frac{de}{dt} &= \frac{1}{h}\{p\sin\theta f_r + [(p+r)\cos\theta + re]\, f_t\}\\
\frac{di}{dt} &= \frac{r\cos(\theta+\omega)}{h} f_n\\
\frac{d\Omega}{dt} &= \frac{r\sin(\theta+\omega)}{h\sin i} f_n\\
\frac{d\omega}{dt} &= \frac{1}{eh}\left[-p\cos\theta f_r + (p+r)\sin\theta f_t\right] - \frac{r\sin(\theta+\omega)\cos i}{h\sin i} f_n\\
\frac{d\theta}{dt} &= \frac{h}{r^2} + \frac{1}{eh}\left[p\cos\theta f_r - (p+r)\sin\theta f_t\right]
\end{aligned} \tag{2}$$

where t is the time, p is the osculating semilatus rectum, $h = \sqrt{\mu p}$ is the magnitude of the angular momentum, μ is the gravitational parameter of Earth, and r is the radius from Earth defined using the general equation of any conic section in polar coordinates [10]. It is important to note that the GVEs presented above may present singularities in the case of orbits that are close to being circular. Given that the primary focus of this study is on LEO, a modified approach is adopted to obtain a non-singular representation of the spacecraft's motion. Instead of using the COEs, circular orbital elements are utilized, and the GVEs are formulated specifically for near-circular orbits. This approach ensures a smooth and accurate description of the spacecraft's motion, avoiding any singularities that may arise from using the COEs for near-circular orbits, as already done in [11].

The perturbing acceleration f is expressed through its three components in the RTN frame (f_r, f_t, f_n). Its overall effect is composed of two main contributions, a first one that corresponds to the control action (f_{thrust}) and a second one used to model the non-spherical gravitational acceleration (f_g). The fidelity of the adopted model strongly depends on the accuracy of the Earth gravitation model and so on how the perturbing acceleration f_g is expressed. This means that models of increasing precision could be built by relying on a higher complexity in the description of f_g and by including the perturbation effects one at a time in the GVEs. This offers excellent advantages for learning-based approaches as it is possible to start from a simplified yet representative model of the LEO transfer and then move forward towards higher fidelity environments. For these reasons, this work employs a set of different models with increasing complexity to describe the spacecraft model, which differ in how the external perturbations are described. In particular, the following three representations are used to describe gravitational acceleration:

1) J_2 secular model, in which only the secular variations caused by the J_2 zonal harmonics on Ω and ω [9] are included in the GVEs.
2) J_2 complete model, where the overall effect due to J_2 is taken into account by including in the acceleration model the short period, long period, and secular variation of the Earth oblateness [12].
3) Generalized model that uses a spherical harmonic potential description based on Legendre polynomials [13] to include higher degrees than the only J_2.

Together with the three implemented spacecraft models, a validation environment is chosen to demonstrate the effectiveness of the proposed approach in a higher fidelity scenario. The Basilisk Astrodynamics Framework [14] has been selected as a suitable simulator to validate the performance of the proposed control strategy. To show how the implemented solution can generalize with respect to the simulation environment, the Basilisk simulation is set up to include also additional disturbances not considered in the models adopted during the learning phase.In particular, the Aerodynamic drag [9] is included in the model, while the gravitational field implements the GGM03S model [15].

An in-depth explanation of how the presented models are exploited to train a RL agent able to solve orbit correction maneuvers in the Basilisk environment is given in section III.

C. Reinforcement Learning

Following the well-known RL framework, the decision making process is formalized as finite MDP. The temporal evolution of the system is divided into discrete time steps, in which the state-action-reward loop is executed iteratively. At each step, the agent receives an encoded observation of the current orbital trajectory and selects an action accordingly. The actions chosen by the agent represent the continuous control of the electrical thruster, and the spacecraft dynamics are propagated for the time step while the chosen action is applied. Lastly, the agent receives the reward signal, which

expresses the "goodness" of the new state reached, and the cycle is repeated. The learning process occurs during the training phase, where the agent improves its decision-making by iterating through hundreds of thousands of maneuver attempts, referred to as episodes. Through this process, the agent learns to perform the maneuver and can act in the real world, assuming the training phase has been adequately designed. To formalize the orbital transfer problem as MDP, we have to define the duration of the step length t, the structure of the observation space s, the action space a and the logic that produces the reward $R \in \mathbb{R}$.

The step length t is a crucial hyperparameter representing the trade-off between high control frequency, which is very difficult to achieve optimally, and low frequency, which leads to coarse control. A 500-second time step is selected for our application as it allows for accurate control of maneuvers and thrusters while also being short enough to avoid excessive complications in the training phase.

Each observation s is composed of twelve continuous values. Six values identify the current orbital trajectory in the circular orbital representation mentioned above. In addition, there are the six differential values with respect to the previous step values. The Markov Property is assumed satisfied: past information is summarized in each current state encryption, i.e., each current observation has to be sufficient to choose the best action.

Actions represent the control to be imposed on the electric thruster equivalent to the force $\vec{f}_{thrust}$ as described in Section II. Each agent action a is composed of three continuous values in the range $[-1, 1]$. We interpret each of the three values as a vector lying on RTN directions; the direction of their vectorial sum $\vec{s}$ is the direction of $\vec{f}_{thrust}$, while the magnitude of $\vec{f}_{thrust}$ is calculated scaling the magnitude of $\vec{s}$ into the range $[0, T_{max}]$.

As mentioned, the reward signal R_t is in charge of informing the agent about the "goodness" of the state reached at step t. We can consider at a high level that RL agents succeed in achieving goals by maximizing the amount of reward received, so it is critical and nontrivial to define the reward function in a way that the desired behavior is translated properly into quantifiable numerical values. In the orbital transfer scenario previously described, success is achieved when the orbital parameters we want to control fall within specific tolerance ranges in the vicinity of the final target orbit. We define the reward function as the weighted sum of the differences between parameters to be controlled and corresponding target values. For the case where the target orbit is defined by a e and i, it is written as:

$$R_t = \omega_a \Delta_{a,t} + \omega_e \Delta_{e,t} + \omega_i \Delta_{i,t} \tag{3}$$

where $\Delta_{a,t}$ $\Delta_{e,t}$ $\Delta_{i,t}$ are the differences between the target keplerian value and the value at the current step t. ω_a ω_e ω_i are constant weights that are properly chosen experimentally. The selection of these weights is crucial as they must balance the contribution of deltas with varying magnitudes. Furthermore, a constant negative reward is assigned to optimize the time of flight. The reward contributions are adjusted so that the reward at each timestep is within the range [-1,1] to ensure a more stable learning process. If the agent fails to reach the target orbit within the maximum allowed number of steps, a negative value is assigned as a reward. A positive reward is added for successful completion.

III. Simulation scenario

The DRL agent's effectiveness is demonstrated by examining two specific scenarios. Simulation results show how the algorithm can attain consistent training and convergence toward a solution. In this context, two distinct types of orbital maneuvers are considered to evaluate the performance of the DRL agent:

1) An orbital correction maneuver in which the agent controls the values of the semi-major axis and inclination. This maneuver is chosen as an appropriate starting point, enabling simultaneous control of both in-plane and out-of-plane actions. This allows for testing the agent's ability to effectively allocate the available thrust to optimize both controls, providing insight into the agent's performance in managing the spacecraft's orbital trajectory.
2) An apogee raising maneuver, where adjustments are made to both the eccentricity and semi-major axis of the spacecraft's orbit. The primary challenge of this maneuver is that the control of the semi-major axis and eccentricity are often conflicting objectives that the agent must learn to balance to achieve successful results. This maneuver tests the agent's ability to effectively navigate trade-offs and make optimal decisions when faced with competing objectives.

The initial and target state of each maneuver is reported in Table 1 by listing the keplerian orbital parameters of the initial and target orbits.

A maneuver is successful if the targeted orbital parameters fall within a predefined tolerance range. These ranges are established based on the magnitude of orbital perturbations typically experienced in a LEO environment. The

performance of the RL agent is evaluated by comparing it to a widely known method that computes the instantaneous optimal thrust angles to obtain a desired change in the orbital elements [16]. The primary objective of this research is to assess the effectiveness of using an RL-based approach for onboard spacecraft applications. To this end, the benchmark trajectory for each maneuver is generated using navigation data from the Basilisk Astrodynamics framework. However, as they are inherently oscillatory, the osculating orbital elements are not appropriate to be used directly in the benchmark strategy. Therefore, a preprocessing step is required to convert the osculating elements to their mean values [10]. This is accomplished using the first-order J2 mapping implemented in Basilisk, which maps between mean and osculating elements [17].

All simulations are conducted using Python-based libraries. The environment is implemented using Openai-Gym ([18]), a widely used toolkit for reinforcement learning research, which offers compatibility with various RL frameworks. For the agent implementation and training, we relied on Stable-baselines3 ([19]), a Pytorch-based framework.

Table 1 LEO mission parameters.

Orbit parameters	**Orbital correction**		**Orbit raising**	
	Initial state	Target state	Initial state	Target state
a [km]	6895	6945	7050	6950
e [-]	0.01	0.01	0.02	0.01
i [°]	97.3	97.7	94	*Free*
Ω [°]	10	*Free*	10	*Free*
ω [°]	10	*Free*	10	*Free*

A. Agent setup

The DRL algorithm is chosen according to the characteristics of the problem to be solved and the simulation environment used to train the agent. Soft Actor-Critic (SAC) is a state-of-the-art deep reinforcement learning (DRL) algorithm widely used in continuous control problems. It is an actor-critic algorithm, where the 'actor' controls the agent's behavior, and the 'critic' evaluates the agent's performance, providing feedback to the actor to improve the policy. In this framework, the actor aims to maximize the expected reward while also maximizing entropy, which results in the agent succeeding at the task while behaving as randomly as possible to continuously explore the action space. In addition, the SAC algorithm is based on the off-policy approach, which uses a replay buffer to store the experiences, thus making it more sample efficient than on-policy algorithms that rely on the current policy to generate actions in the environment.

As previously stated, our implementation relies on the Stable-baselines3 framework. The algorithm's hyperparameters are manually tuned by directly testing the agent's performance on the environment. Additionally, specific procedures are implemented to save the best policy during the training phase. The network architecture is detailed in Table 2, which lists the number of neurons in each layer and the activation function used for each neuron. The same architecture is used for both the actor and the critic. The remaining hyperparameters values used in all simulations are listed in Table 3.

Layer	**Nodes**
Layer 1	400
Layer 2	300
Layer 3	128
Activation	ReLU

Table 2 Network configuration.

Hyperparameter	**Value**
batch size	1024
learning rate	7.3e-4
buffer size	1e6
discount factor	0.99

Table 3 SAC hyperparameters.

B. Incremental training strategy

Previous attempts to train an agent from scratch have revealed that it is challenging for the agent to accomplish multiple target states during a single training phase. Certain orbital parameters prove to be more complex to control than others. Additionally, training directly on the full dynamic model makes it difficult to address the oscillating elements resulting from external perturbation effects. Adopting the curriculum learning approach, we structured the

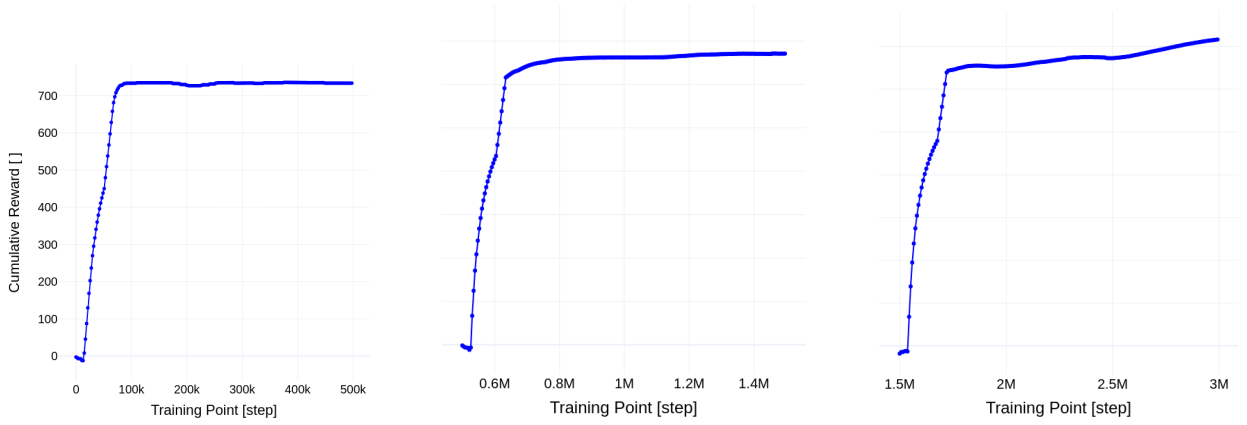
training procedure starting from a partial maneuver with a simplified model, then adding other target parameters and perturbations. This strategy has proven ideal, as well as in analogous applications in the literature [20], to tackle a complex scenario that can be gradually mastered, allowing convergence time and performance optimization.

As described in the previous section, the reward function employed is based on a weighted sum of distance metrics, and the fine-tuning of weights is performed in each training phase.

C. Simulation results

1. Orbital correction maneuver

The first maneuver is the orbit correction maneuver, which involves increasing both the inclination and semi-major axis while preserving the eccentricity value. The initial training phase is executed using the first model outlined in Section II, by incrementally incorporating the target values into the computation of the reward function and terminal conditions. In the second phase of training, the model is further fine-tuned by incorporating the effect of external perturbations. This is achieved by retraining the model using the complete GVE dynamic model and adjusting the weights of the reward function to optimize the control of the orbital parameters. This approach allows the model to learn how to reach the target position beforehand, and the curriculum step provides the agent with the capability to handle oscillations caused by external perturbations effectively. The agent's cumulative reward during training is shown in Fig. 1.



(a) Inclination change on GVE secular model.

(b) Combined change (sma-ecc-inc) on GVE secular model.

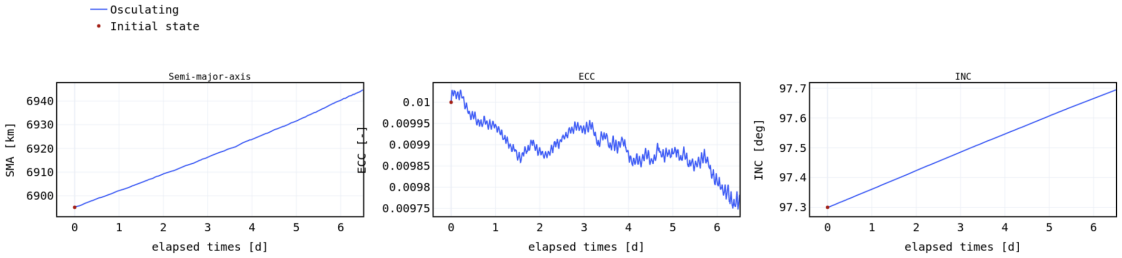
(c) Combined change (sma-ecc-inc) on GVE complete model.

Fig. 1 Mean cumulative reward achieved during the training phase.

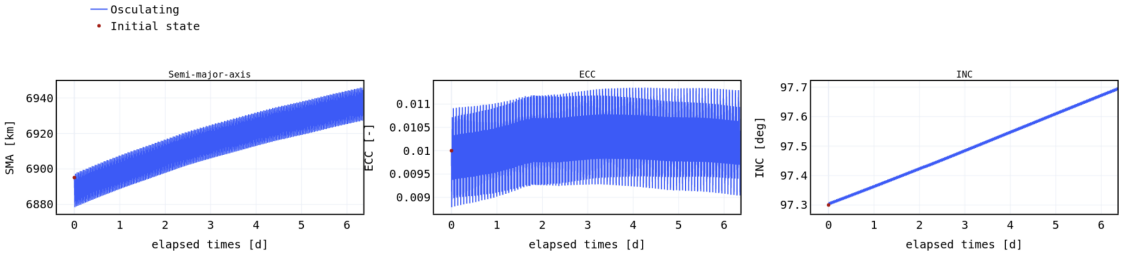
The transfer accomplished by the agent is depicted in Fig. 2, which illustrates the inference of the trained algorithm across both dynamic models (i.e. the J2 secular model and the complete one), displaying the key orbital parameters. Fig. 3, on the other hand, presents a comparison of the thrusting actions.

2. Apogee raising maneuver

In the apogee-raising maneuver, both the semi-major axis and eccentricity values are adjusted simultaneously. As the control of both values often poses conflicting objectives in relation to thrusting actions, the agent must learn to balance its actions for optimal results. Therefore, the training is executed by considering both target values simultaneously. To ensure optimal performance, the reward function is fine-tuned to normalize reward values in accordance with the new maneuver settings. It is important to note that for the apogee raising maneuver, the normal thrust action is locked and only the tangential and radial thrust actions are optimized by the RL agent. This simplification allows for faster convergence by reducing the number of state-action pairs explored during training. This simplification is valid as the maneuver's objective is to target only the semi-major axis and eccentricity, and the normal thrust action is not necessary for achieving success. As before, the model is initially trained on the secular GVE model before being retrained on the complete one. The transfer accomplished by the agent in both dynamic models is showcased in Fig. 4 Fig. 5.

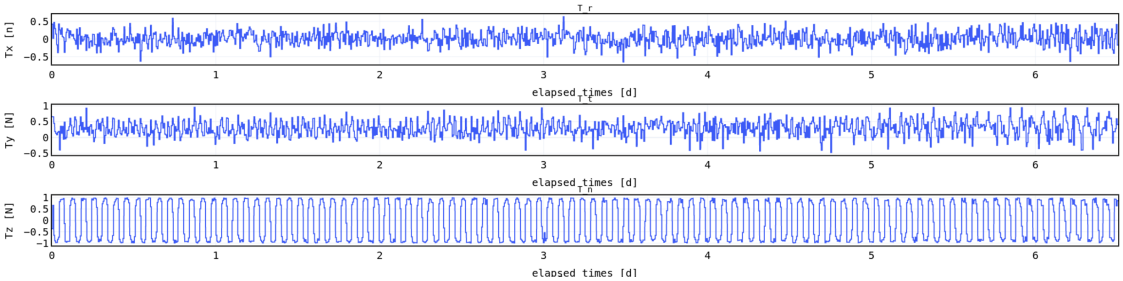


(a) Semi-major axis, eccentricity and inclination profiles with the GVE secular model.

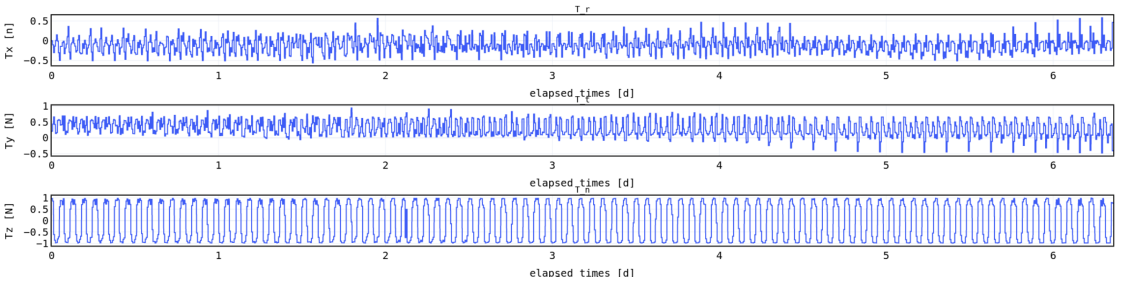


(b) Semi-major axis, eccentricity and inclination profiles with the GVE complete model.

Fig. 2 Trajectory of the RL agent during the orbital correction maneuver.

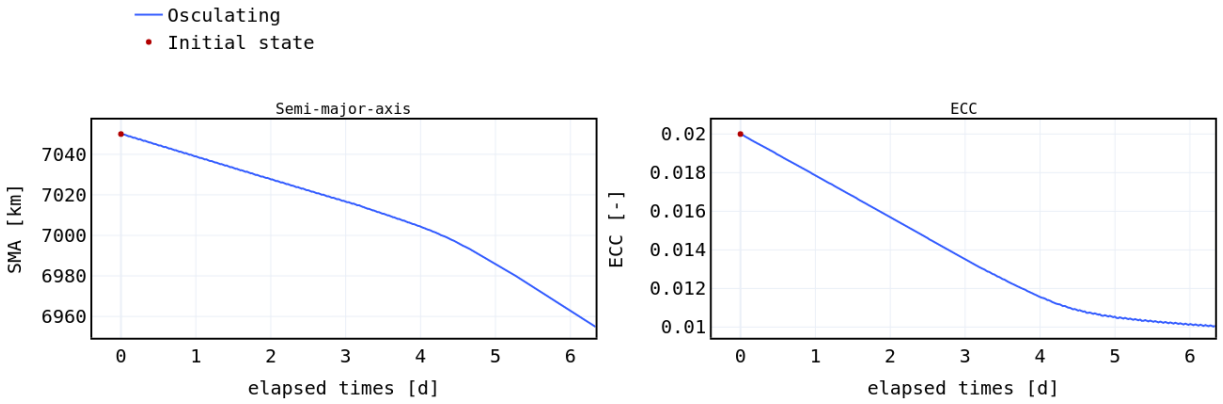


(a) Radial, tangential and normal thrust profiles with the GVE secular model.

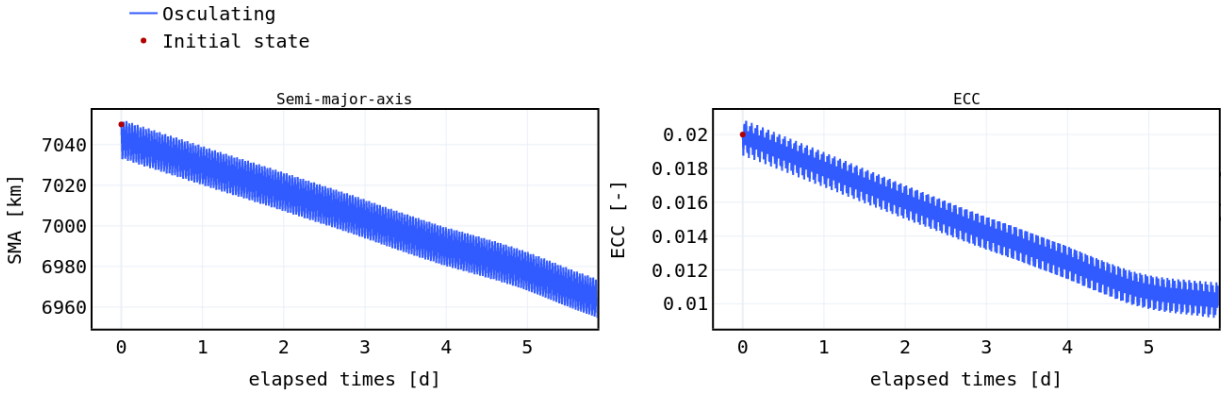


(b) Radial, tangential and normal thrust profiles with the GVE complete model.

Fig. 3 Control actions of the RL agent during the orbital correction maneuver.



(a) Semi-major axis and eccentricity profiles with the GVE secular model.



(b) Semi-major axis and eccentricity profiles with the GVE complete model.

Fig. 4 Trajectory of the RL agent during the apogee raising maneuver.

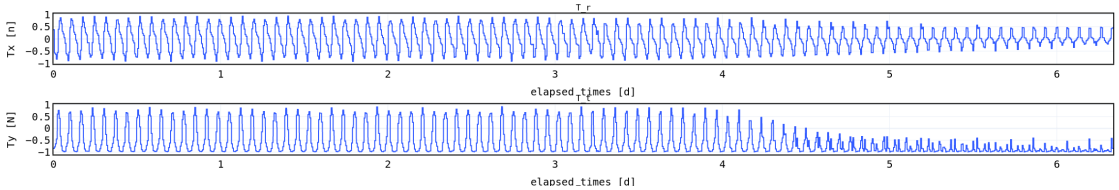
D. Validation on Basilisk framework

This section presents the results of applying the trained RL model in the Basilisk framework to validate the performance of the proposed control strategy and to evaluate its ability to generalize to different simulation environments. Simulation data are presented in Table 4, comparing the performance of the trained agent with the benchmark algorithm.

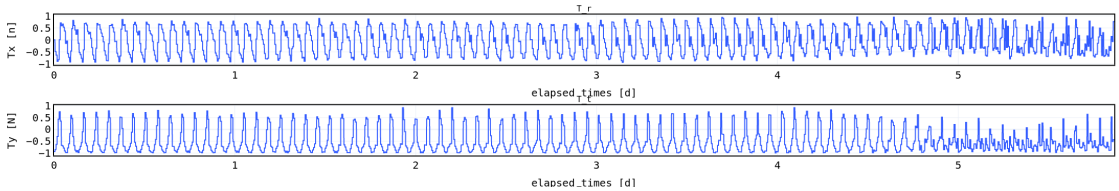
Algorithm	**Maneuver**	**TOF** [days]	**Fuel consumption** [g]
Benchmark	Orbit-correction	6.3	115
RL Agent (SAC)	Orbit-correction	6.12	109
Benchmark	Apogee-raising	5.65	103
RL Agent (SAC)	Apogee-raising	5.89	98

Table 4 Performance comparison in the Basilisk simulation environment.

The results of this study indicate that the performance of our RL agent is comparable to benchmark results. To ensure a fair comparison, we set the tolerances to match those of the benchmark. However, this is a limitation due to the agent's reliance on osculating elements as input. Despite this, the results demonstrate the agent's ability to successfully complete the maneuver, even when tolerances are reduced. It's worth noting that the criteria used to evaluate the success of the agent is based on whether the orbital parameters reach the target range within the specified success range.



(a) Radial and tangential thrust profiles with the GVE secular model.



(b) Radial and tangential thrust profiles with the GVE complete model.

Fig. 5 Control actions of the RL agent during the apogee raising maneuver.

IV. Conclusions and next steps

This paper presents an RL approach to autonomously maneuver a satellite during a low-thrust LEO orbit transfer, where the goal is to reach a target orbit starting from nominal conditions while considering dynamic models of increasing complexity. The problem is formalized as a Markov Decision Process (MDP) model, and a model-free deep RL algorithm is applied to solve two types of maneuvers. The effectiveness of RL agents in learning how to solve such optimization problems is demonstrated.

Preliminary assessments aimed at testing agent robustness starting in non-nominal initial conditions are currently under evaluation, early results are promising, but their systematic analysis represents a future goal. Other than this, future works could focus on improving the realism of the scenario by introducing random noise to observation values and action effectiveness, as well as generalizing initial and target conditions, both from a noise point of view and aiming at generalized maneuvers. While maintaining robustness when these improvements are addressed, another critical objective could be automating the tuning process of the algorithm's hyperparameters. Finally, we plan to explore the deployment of the model on embedded architecture, simulating a more realistic operational scenario.

References

[1] Morante, D., Sanjurjo Rivo, M., and Soler, M., "A survey on low-thrust trajectory optimization approaches," *Aerospace*, Vol. 8, No. 3, 2021, p. 88.

[2] Betts, J. T., *Practical methods for optimal control and estimation using nonlinear programming*, SIAM, 2010.

[3] Rao, A. V., "A survey of numerical methods for optimal control," *Advances in the Astronautical Sciences*, Vol. 135, No. 1, 2009, pp. 497–528.

[4] Topputo, F., and Zhang, C., "Survey of direct transcription for low-thrust space trajectory optimization with applications," *Abstract and Applied Analysis*, Vol. 2014, Hindawi, 2014.

[5] Izzo, D., and Öztürk, E., "Real-time guidance for low-thrust transfers using deep neural networks," *Journal of Guidance, Control, and Dynamics*, Vol. 44, No. 2, 2021, pp. 315–327.

[6] Kolosa, D. S., *A Reinforcement Learning Approach to Spacecraft Trajectory Optimization*, Western Michigan University, 2019.

[7] Sutton, R. S., and Barto, A. G., *Reinforcement Learning: An Introduction*, 2nd ed., The MIT Press, 2018. URL `http://incompleteideas.net/book/the-book-2nd.html`.

[8] Shannon, J. L., Ozimek, M. T., Atchison, J. A., and Hartzell, C. M., "Q-law aided direct trajectory optimization of many-revolution low-thrust transfers," *Journal of Spacecraft and Rockets*, Vol. 57, No. 4, 2020, pp. 672–682.

[9] Vallado, D. A., *Fundamentals of astrodynamics and applications*, Vol. 12, Springer Science & Business Media, 2001.

[10] Bate, R. R., Mueller, D. D., White, J. E., and Saylor, W. W., *Fundamentals of astrodynamics*, Courier Dover Publications, 2020.

[11] Larbi, M. B., and Stoll, E., "SPACECRAFT FORMATION CONTROL USING ANALYTICAL INTEGRATION OF GAUSS'VARIATIONAL EQUATIONS," *6th International Conference on Astrodynamics Tools and Techniques, ICATT*, 2016.

[12] Shen, H., Xue, S., and Li, D., "MPC-based Low-thrust Orbit Transfer Under J 2 Perturbation," *2020 39th Chinese Control Conference (CCC)*, IEEE, 2020, pp. 2467–2472.

[13] Vetter, J. R., "The evolution of Earth gravitational models used in astrodynamics." *Johns Hopkins APL Technical Digest*, Vol. 15, No. 4, 1994, pp. 319–335.

[14] Kenneally, P. W., Piggott, S., and Schaub, H., "Basilisk: A flexible, scalable and modular astrodynamics simulation framework," *Journal of aerospace information systems*, Vol. 17, No. 9, 2020, pp. 496–507.

[15] Tapley, B., Ries, J., Bettadpur, S., Chambers, D., Cheng, M., Condi, F., and Poole, S., "The GGM03 mean earth gravity model from GRACE," *AGU Fall Meeting Abstracts*, Vol. 2007, 2007, pp. G42A–03.

[16] Ruggiero, A., Pergola, P., Marcuccio, S., and Andrenucci, M., "Low-thrust maneuvers for the efficient correction of orbital elements," *32nd International Electric Propulsion Conference*, 2011, pp. 11–15.

[17] Schaub, H., and Junkins, J. L., *Analytical mechanics of space systems*, Aiaa, 2003.

[18] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W., "OpenAI Gym," , 2016. https://doi.org/10.48550/ARXIV.1606.01540, URL https://arxiv.org/abs/1606.01540.

[19] Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N., "Stable-Baselines3: Reliable Reinforcement Learning Implementations," *Journal of Machine Learning Research*, Vol. 22, No. 268, 2021, pp. 1–8. URL http://jmlr.org/papers/v22/20-1364.html.

[20] Song, Y., Lin, H., Kaufmann, E., Duerr, P., and Scaramuzza, D., "Autonomous Overtaking in Gran Turismo Sport Using Curriculum Reinforcement Learning," , 2021. https://doi.org/10.48550/ARXIV.2103.14666, URL https://arxiv.org/abs/2103.14666.