Deep Reinforcement Learning based Integrated Guidance and Control for a Launcher Landing Problem

P. Rosa [(1)], J. P. Belfo [(1)], N. Somma [(1)], A. Botelho [(1)], G. Tofanelli [(1)], J. I. Bravo [(2)], R. Hinz [(2)], J. Belhadj [(3)], M. Casasco [(3)], S. Bennani [(3)]

[(1)] DEIMOS Engenharia SA, Av. Columbano Bordalo Pinheiro N75 9th floor A1, Lisbon, Portugal, paulo.rosa@deimos.com.pt

[(2)] DEIMOS Engineering and Systems SLU, Spain

[(3)] ESA/ESTEC, The Netherlands

ABSTRACT

In this work, a Deep Reinforcement Learning (Deep-RL) algorithm is used to train an actor-critic agent to simultaneously control the engine thrust magnitude and the two Thrust Vector Control (TVC) gimbal angles to land a Reusable Launch Vehicle (RLV) in 6-DoF (Degree-of-Freedom) simulation. An incremental design approach is presented, progressively augmenting the number of DoFs and introducing more complexity factors such as nonlinearity in models. Ultimately, the 6-DoF problem is addressed using a high-fidelity simulator that includes a nonlinear actuator model and a realistic vehicle aerodynamic model. Starting from an initial state at the end of the re-entry trajectory, the problem consists in precisely landing the RLV, while ensuring system requirements satisfaction, such as saturation and rate limits in the actuation, and aiming at fuel consumption optimality. The Deep Deterministic Policy Gradient (DDPG) algorithm was adopted as candidate strategy to allow the design of an integrated Guidance and Control (G&C) algorithm in continuous action and observation spaces. A 1000-shot Monte-Carlo campaign was performed, leading to a 97% success rate in terms of requirements satisfaction. A reachability analysis was also performed to further assess the robustness of the closed-loop system composed of the integrated Deep-RL guidance and control, trained for the 1-DoF scenario. The design, implementation, and validation of the Deep-RL integrated G&C was performed within the ESA-i4GNC framework, developed within the ESA-funded AI4GNC project, led by DEIMOS, and participated by the University of Lund, INESC-ID, and TASC. Taking into account the fidelity of the benchmark adopted and the results obtained, this approach is deemed to have a significant potential for further developments and ultimately space industry applications, beyond RLVs, that also require a high level of autonomy, such as In-Orbit Servicing (IOS) and Active Debris Removal (ADR).

1 INTRODUCTION

Future spacecraft (S/C) missions will require the ability to adapt to (at least partially) unknown conditions and have the ability to perform control reconfiguration. Some examples are In-Orbit Servicing (IOS) and Entry, Descent and precise Landing (EDL) missions. Among all the technical challenges that characterize EDL missions, a key one is that they may experience significant changes and uncertainty in aerodynamic coefficients, actuator characteristics and initial conditions. Therefore, the ability to autonomously adapt to variations in the dynamics and to re-plan the trajectory under the presence of changes to the descent conditions, can play a major role in ensuring mission success. The increasingly more powerful processing units (HW), together with enhanced and less computationally intensive Artificial Intelligence (AI) algorithms, make it possible to explore new AI applications. Given that results in this direction indicate the plausibility of implementing onboard spacecraft these algorithms, the use of AI within the avionics system is within reach and this is a remarkable target to be pursued. Nevertheless, a significant number of challenges still exist in order to reach the goal of fully exploiting AI for space applications. These challenges are also motivated by the gap between the performance of AI and the theoretical understanding and modelling of the behaviour of those algorithms. Moreover, due to the intrinsic nonlinear structure of the AI algorithms, the generalization of the learned features to different scenarios is not straightforward. In particular, while linear approaches guarantee that the results obtained at a given linearization point are satisfied within a region around that point (in the space of parameters), AI algorithms typically do not provide such qualities. As a consequence, Monte-Carlo (MC) simulations per se are not guaranteed to cover the whole space of potential configurations of parameters, as minor changes in the inputs of the AI can lead to completely different results in the output [1], while probability distribution functions are typically unknown for these problems. When it comes to the Space sector, this challenge is further amplified by the lack of existing large data sets that could be used for training and validation purposes. In applications such as EDL, this is exacerbated by the need to fly-right in the first flight, where no incremental validation is possible. In addition, there are also challenges coming from the software side, such as predictability, clear/understandable behaviour, memory and CPU usages, compatibility with real-time constraints, etc. Hence, validation limitations need to be considered at all levels, especially because Guidance, Navigation, and Control (GNC) has an impact on the S/C itself and possibly on others.

The work presented in this paper has been developed in the context of the ESA-funded project entitled AI4GNC, led by DEIMOS, and participated by the University of Lund, INESC-ID, and TASC. The overall objectives of AI4GNC were as follows:

- Implement ESA-i4GNC, an AI-based GNC E2E design & analysis (D&A) framework for layered architectures, designed and developed during the AI4GNC project
- Exploit recent advances in control and AI
- Perform trade-off analyses
- Evaluate the proposed AI-based GNC design and Verification & Validation (V&V) tool in a representative benchmark

The paper is organized as follows. Section 2 provides an overview of the ESA-i4GNC framework, developed within AI4GNC. In section 3, the problem addressed in this paper is formulated. In section 4, the approach used to tackle the problem is explained, focusing mostly on the final results obtained for the complete 3-Degree-of-Freedom (DoF) scenario. In section 5, the Monte-Carlo analysis is performed and the results are compared with more traditional methods. Finally, section 6 describes the reachability analysis performed for the 1-DoF scenario and section 7 summarizes the main conclusions.

2 THE ESA-i4GNC FRAMEWORK

The ESA-i4GNC framework (Figure 2-1) was developed in MATLAB/Simulink© using an object-oriented approach, such that general GNC architectures can be implemented in a systematic and modular manner. It supports several libraries and functionalities that further increase the flexibility of the tool. An interface between MATLAB and Python was developed (Figure 2-2), allowing the extension of MATLAB functionalities to Python libraries and toolboxes, that also enable the interface to other external frameworks, such as Julia. In order to thoroughly evaluate the AI-based algorithms, the RETALT-1 Reusable Launch Vehicle (RLV) – a two-stage to orbit launcher defined by the H2020 RETALT consortium, with its fully nonlinear flight dynamics simulator developed by Deimos Space – was selected as benchmark, equipped with Thrust Vector Control (TVC), a Reaction Control System (RCS), and aerodynamics fins. For comparison purposes, the baseline GNC system composed of a successive convexification guidance, a PID controller, and ideal navigation, was also developed and is provided as part of the ESA-i4GNC tool.

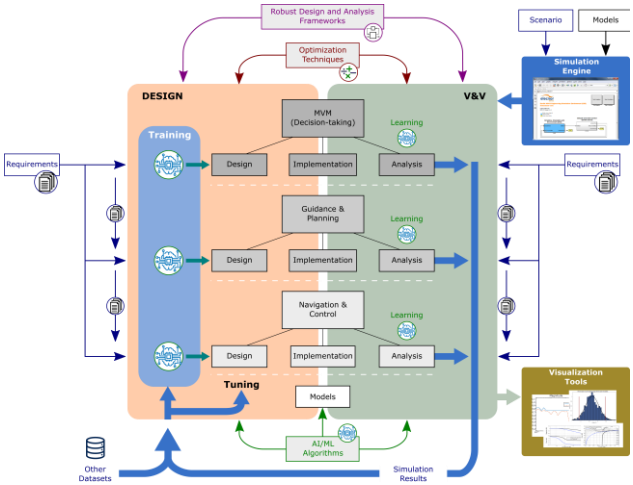

Figure 2-1: The ESA-i4GNC design and V&V framework

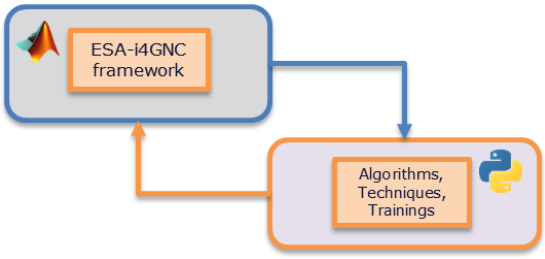

Figure 2-2: Inter-connection between MATLAB© and Python

Eight case studies were addressed for which specific methods were discussed, implemented, and tested:
(1) Reinforcement Learning (RL) adaptive control to regulate the attitude of the RLV;
(2) cascade control to regulate the position around the reference values provided by guidance;
(3) pipeline for state estimation in a navigation system with Neural Networks (NNs);
(4) compressed sparse regression for online system;
(5) learning-based model predictive control for attitude control;
(6) Integral Quadratic Constraints (IQC) formalism for NN-based attitude control verification;
(7) learning-based Kalman filtering for attitude estimation;
(8) integrated Guidance & Control Deep-RL approach for trajectory.

Case studies 1 to 3 were led by INESC-ID, case study 4 by TASC, case studies 5 to 7 by the University of Lund, and case study 8 by DEIMOS.

3 PROBLEM FORMULATION

The study conducted is concerned with the use of a Deep Reinforcement Learning (Deep-RL) technique to address the Guidance and Control (G&C) problem for the landing phase of a Reusable Launch Vehicle (RLV). The goal is set to investigate possible advantages and/or complementarities

of this AI method with respect to the classical G&C approaches; in particular, the onboard solution execution and the possibility of exploiting the nonlinear dynamics of the launcher dynamics during the landing phase. The main objective is to use this RL algorithm to fully replace the integrated Guidance and Control subsystem (assuming ideal navigation), i.e., to generate a policy that is able to map the sensor measurements directly to the action commands. The ultimate agent considered for this study case is the Deep Deterministic Policy Gradient (DDPG) which belongs to the Actor-Critic family of RL algorithms and allows to work with continuous action and observation spaces, necessary for the 3-DoF landing scenario. The Deep RL technique adopted is based on the training of an agent composed of two types of NNs:

- Actor NN: that provides the control commands (thrust magnitude and two gimbal angles of the TVC actuation), according to the current observation state;
- Critic NN: that estimates the Q value, given the action and observed state.

Furthermore, within the DDPG algorithm implementation, two additional sets of NNs are defined: a set of actor-critic online networks and a set of actor-critic target networks. The online networks are the ones trained at each training step, and their training depend on the target network, [2]. In each training step, a random batch of transitions (a group of data composed by a state, the action applied, the reward received, and the next state) is sampled from a replay buffer which stores batches of data collected during the exploration phase. Each transition is used to compute the expected Q value based on the critic target network, which is then used to train the critic online network. The critic online network is used to teach the actor online network using the sampled policy gradient. Afterwards, the target networks are updated according to an update rate.

Within ESA-i4GNC, this algorithm has been implemented by exploiting the RL Coach toolbox. This open-source tool uses an object-oriented programming approach, in Python language, that allows to build up the algorithm in a modular way, by defining an agent and an environment. For this case study, the selected agent is the DDPG algorithm as mentioned above, while the environment in which it acts is the benchmark simulator for the landing phase. In this case, the benchmark corresponds to the RETALT-1 RLV landing problem [6], for which aerodynamics, wind and actuator models are available, between others, yielding a high-fidelity simulator.

The basic principle of RL is the following: the agent learns through the interaction with the environment, i.e., through the feedback (positive/negative reward) it receives from the action taken. Since the environment has been developed in MATLAB/Simulink through the ESA-i4GNC framework, and the RL Coach agent runs in Python, a two-way communication (Figure 2-2) between the two frameworks was needed. The communication link implemented uses a TCP/IP connection, making it extremely fast, which is of paramount importance for the training process.

4 DEEP-RL INTEGRATED GUIDANCE AND CONTROL APPROACH

The main challenge in of solving an RLV landing problem by using Deep-RL is the definition of a suitable reward function that works well with a highly sparse reward setting, as presented in the sequel. Moreover, the tuning of a large number of hyperparameters involved during the training increases significantly the complexity of the problem.

This topic has deserved some attention during the last years, but, due to the large sparsity and high complexity of the AI-based RLV landing problem, the state-of-the-art is still in early stages in this field. Hence, the design was performed following an iterative/incremental approach: the work started with the analysis of a simplified 1D problem (vertical landing), in which the goal was set to control just the thrust magnitude; then, the problem has been extended to 2D vertical landing but starting with no horizontal error, subsequently moving to the 2D landing with horizontal displacement, first

without, and then with, TVC rate limiters (which revealed themselves as particularly challenging), and finally the full 3D (6 DoF) landing scenario, without and with TVC rate limiters.

Due to space constraints, only the final complete 3D scenario with rate limiters is reported in this paper. Nevertheless, the main lessons learnt and the most relevant strategies that led to the ultimate results during the incremental approach are reported. For comparison purposes, a baseline guidance algorithm was developed and implemented. The selected guidance algorithm is based on the so-called Successive Convexification (SCVX), which is one of the most well-established techniques to solve non-convex optimal control problems with nonlinear dynamics and non-convex state and constraints. The algorithm computes the solution of the original non-convex problem by iteratively solving convex optimization sub-problems, obtained by iteratively linearizing the dynamics and constraint from the previous iteration. This guidance algorithm, together with a simplified PID controller, allow to generate reference trajectories, starting at given initial conditions.

Typically, during the RL training, the initial condition of each episode is set to be always the nominal initial condition with a certain dispersion being added. In situations where the number of episode steps required to go from an initial dispersed condition to the end of the episode is very large, the required training steps may also increase. In other words, since the RL agent learns backwards (i.e., from positive/negative final steps to initial steps of the episode), if the length of an episode is large, the number of episodes required for the agent to learn would also be large. To address this problem, the initial conditions of each episode have been randomized along a reference trajectory (obtained from the SCVX guidance), and then further dispersed around the selected point of the trajectory. An illustration (in 2D) of this operation is provided in Figure 4-1. It is also remarked that this approach of initialization of each episode allows to obtain a higher number of positive landings, which can also improve the agent's learning rate.

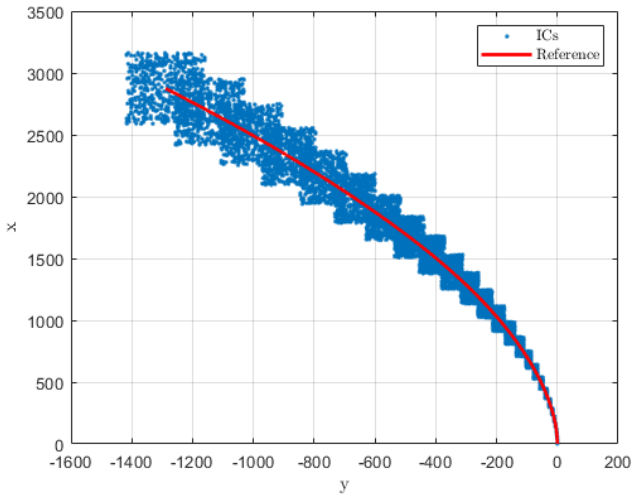

Figure 4-1: Randomization and dispersion of the initial conditions

The nominal initial conditions for the 3D landing phase used for the training are reported in Table 4-1.

Table 4-1: Initial conditions for the 3D landing phase

Mass [kg]	Position [m]	Velocity [m/s]	Attitude [q]	Angular velocity [rad/s]
80345	[2874, -1288, -82.2]	[-189.9, 151.29, 5.57]	[0.943; 0.006; 0.018 -0.329]	[0, 0, 0]

For this scenario, the yaw, δ_y, and pitch, δ_p, gimbal angles and the thrust magnitude, T, needs to be controlled and, therefore, the action space (i.e., the output of the actor NN) is 3 dimensional. As a first approach to the problem, the computed actuation values have been used for the training of the NN and an ideal actuation model was used for the benchmark during the training. Therefore, the NN-based G&C receives the observation state and output the actual actuation commands. The observation state for this case (input of the actor NN) assumed the form expressed in (1):

$$obs_{space} = \left[x, y, z, (v_x - v_{t_x}), \left(v_y - v_{t_y}\right), (v_z - v_{t_z}), \psi, \vartheta, \omega_z, \omega_y\right] \tag{1}$$

The reward function developed and tested for this case study is reported in (2), following the same rationale as in [5]:

$$\begin{aligned} R = &-|v - v_t| - 1e^{-6} \cdot Thrust \\ &+200 \cdot (x < 0 \;\&\&|r| < 5 \;\&\&\; v_x < 0 \;\&\&\; |v| < 4 \;\&\&\; |\psi| < 0.2 \;\&\&\; |\vartheta| < 0.2) \\ &+500 \cdot (x < 0 \;\&\&|r| < 3 \;\&\&\; v_x < 0 \;\&\&\; |v| < 2 \;\&\&\; |\psi| < 0.1 \;\&\&\; |\vartheta| < 0.1) \\ &-200 \cdot (\,|r| > |r_0|) - 200 \cdot (|\psi| > \pi/2) - 200 \cdot (|\vartheta| > \pi/2) \end{aligned} \tag{2}$$

where ψ, ϑ are yaw and pitch angles (in radians), respectively, and v_t is the target velocity profile, which is defined as:

$$v_t = -|v_0| \cdot \left(1 - e^{-\frac{t_{go}}{\tau}}\right), \qquad t_{go} = \frac{|r|}{|v|} \tag{3}$$

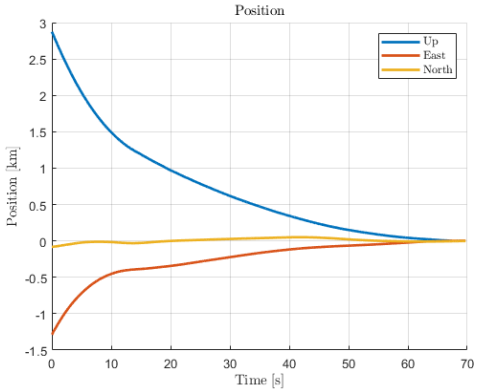

Figure 4-2: Position profile for the 3D landing scenario without rate limiters

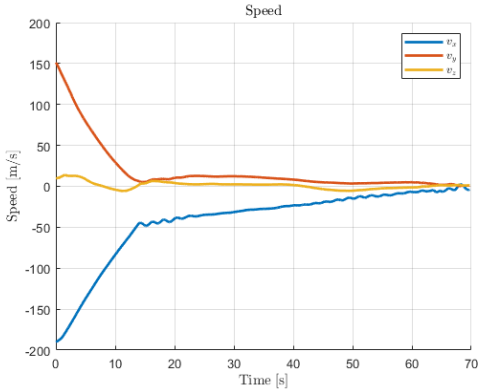

Figure 4-3: Velocity profile for the 3D landing scenario without rate limiters

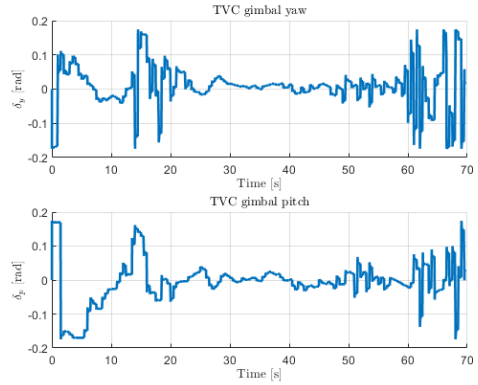

Figure 4-4: Gimbal angles profiles for the 3D landing scenario without rate limiters

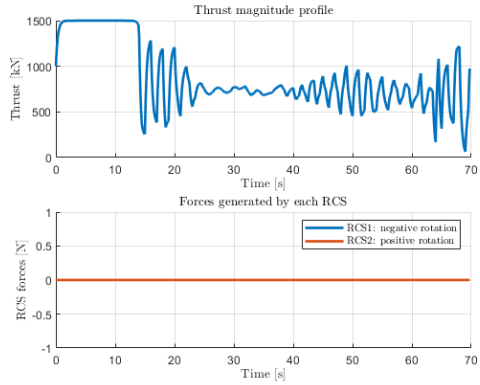

Figure 4-5: TVC and RCS thrust profile for the 3D landing scenario without rate limiters

Using this approach, although the NN is capable of landing the launcher (cf. Figure 4-2 to Figure 4-5) the actuation signals are change abruptly, as depicted in Figure 4-4. In practice, a TVC would not be able to adequately track such commands. This is modeled by introducing rate limiters in the actuator model. This additional term strongly affects the training since, for a given observation state, the action provided by the agent might not correspond to the actual action applied executed by the TVC, due to possible rate saturations, thus affecting negatively the learning capability of the agent. Therefore, in order to tackle this issue, a new solution has been proposed: the action rates, instead of the action themselves, are commanded. This conceptually changed the architecture of the actor NN, which now receives the observation state and outputs the actions variation/rates, as illustrated in Figure 4-6. This approach also exploits the possibility to directly limit the rate during the training, in addition to the gimbal angles saturation.

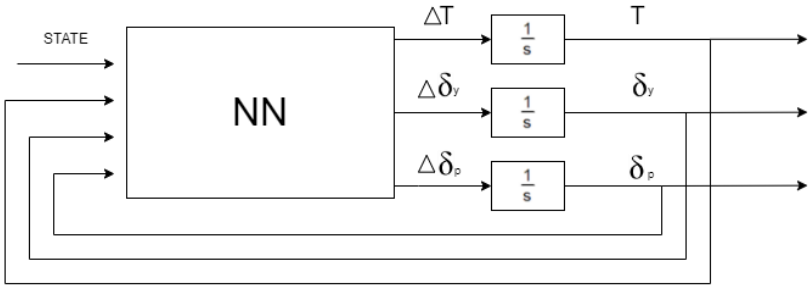

Figure 4-6: NN architecture considering the rate limiters in the actuator model. The NN commands the control rates, which are then integrated to generate the actual commands.

Nevertheless, in order to learn properly the next rate to output, the current value of the actions/commands is passed to the NN during the training, since the action to be applied depends on the initial value of the commands at the same time step. Hence, the new observation space for the complete scenario is reported in (4):

$$obs_{space} = \left[x, y, z, \left(v_x - v_{t_x}\right), \left(v_y - v_{t_y}\right), \left(v_z - v_{t_z}\right), \psi, \vartheta, \omega_z, \omega_y, \frac{T_c}{1e^4}, \delta_{y_c} \cdot 1e^3, \delta_{p_c} \cdot 1e^3\right], \quad (4)$$

where the subscript ‘c’ stands for ‘current’.

The initial velocity v_0 in (2) has been set to 200 m/s, while the value of τ changes along with the landing; in particular, $\tau = 20$ above 15 meters and $\tau = 100$ below. This allowed to have a higher target velocity at the beginning of the trajectory and a lower one at the end, which turned into a smoother landing, with a lower final velocity. It is also remarked that the selection of the values for these parameters (together with the hyperparameters of the NN that will be presented next) are the results of an extensive trade-off analysis performed with also the 1D and 2D scenarios described previously.

It is remarked that the scaling value of the thrust and gimbal signals in the observation space of (4) has a great impact on the results. After a detailed analysis, it was possible to determine that the scaling values presented in the observation space presented above allow the agent to properly learn and improve the Q-value during training. Indeed, if the observed data is not properly scaled, then the model to be trained tends to provide higher weights to greater values and small weights to smaller values. In other words, the trained model will give more importance to the inputs that have a higher scale than to the other inputs, creating an imbalance between inputs.

As described in Figure 4-6, the current command values have been obtained by integrating and feeding back the rates at the previous time step. The initial conditions for the integrators have been set to 0 for the gimbal and 1000 kN for the thrust. It is worth to mention that an upper and lower saturation limit has been added to the integrator according to the saturation values of the control

command, so that the integrators do not keep integrating above the maximum values. Another important remark is that the initial condition of the thrust integrator and also of the thrust actuator model was modified during training, depending on the thrust signal from the trajectory used for the randomization of the initial conditions. Otherwise, when an episode starts with a small altitude, the initial thrust of 1000 kN prevents a successful landing, leading to worse results in terms of learning. By using the initial thrust from the trajectory, as well as the initial condition, the learning process converges faster.

For this new approach, the expression of the reward function changed accordingly. Since the agent no longer generates the actions but instead the rates, the penalization term on the thrust has been removed from the reward. However, penalization terms have been added when the rates provided by the agent lead saturations of the control commands. The final expression for the reward function (R) used is as follows:

$$
\begin{aligned}
R = &-|v - v_t| - 5e^{-6} \cdot (\Delta T > 0)|\Delta T| \\
&+200 \cdot (x < 0 \;\&\&|r| < 5 \;\&\&\; v_x < 0 \;\&\&\; |v| < 5 \;\&\&\; |\psi| < 0.2 \;\&\&\; |\vartheta| < 0.2) \\
&+500 \cdot (x < 0 \;\&\&|r| < 3 \;\&\&\; v_x < 0 \;\&\&\; |v| < 3 \;\&\&\; |\psi| < 0.1 \;\&\&\; |\vartheta| < 0.1) \\
&-100 \cdot (\,|r| > |r_0|) - 100 \cdot \left(|\psi| > \frac{\pi}{2}\right) - 100 \cdot \left(|\vartheta| > \frac{\pi}{2}\right) \\
&-0.1 \cdot \left((T_c + \Delta t \cdot \Delta T) > T_{max} \;\|\; (T_c + \Delta t \cdot \Delta T) < 0\right) \\
&+0.01 \cdot \left((T_c + \Delta t \cdot \Delta T) \leq T_{max} \;\&\&\; (T_c + \Delta t \cdot \Delta T) \geq 0\right) \\
&-0.1 \cdot \left(\left(\delta_{y_c} + \Delta t \cdot \Delta\delta_y\right) > \delta_{y_{max}} \;\|\; \left(\delta_{y_c} + \Delta t \cdot \Delta\delta_y\right) < -\delta_{y_{max}}\right) \\
&+0.01 \cdot \left(\left(\delta_{y_c} + \Delta t \cdot \Delta\delta_y\right) \leq \delta_{y_{max}} \;\&\&\; \left(\delta_{y_c} + \Delta t \cdot \Delta\delta_y\right) \geq -\delta_{y_{max}}\right) \\
&-0.1 \cdot \left(\left(\delta_{p_c} + \Delta t \cdot \Delta\delta_p\right) > \delta_{p_{max}} \;\|\; \left(\delta_{p_c} + \Delta t \cdot \Delta\delta_p\right) < -\delta_{p_{max}}\right) \\
&+0.01 \cdot \left(\left(\delta_{p_c} + \Delta t \cdot \Delta\delta_p\right) \leq \delta_{p_{max}} \;\&\&\; \left(\delta_{p_c} + \Delta t \cdot \Delta\delta_p\right) \geq -\delta_{p_{max}}\right)
\end{aligned}
\tag{5}
$$

with $T_{max} = 1.4e6\ N,\ \delta_{y_{max}} = \delta_{p_{max}} = 0.15\ rad.$

The final selection of the hyperparameters used for the training stage are summarized in Table 4-2, where $C_{w\ and}$ τ are the number of steps and rate between copying the weights from the online networks to the target networks, while γ is the discount factor used by the DDPG algorithm. As part of the setup of the analysis, the architectures used for the actor and critic NNs are presented in Table **4-3**.

Table 4-2: Hyperparameters setting

Heat-up steps	**Training steps**	**Steps between evaluations**	**Actor/Critic Learning rate**	**Actor/Critic Batch size**	**γ**	**τ/C_w**
3000	900000+200000	10	0.0001/0.001	256/256	0.9	0.001/1

Table 4-3: Architecture of the Actor and Critic NNs

	Actor NN		**Critic NN**	
Layer	**# units**	**Activation function**	**# units**	**Activation function**
Input	12	ReLu	15	ReLu
Hidden	256	ReLu	384	ReLu
Hidden	512	ReLu	512	ReLu
Output	3	Tanh	1	Tanh

4.1 Results

With the presented framework, the NN-based Guidance and Control led to the results for the final 3D landing scenario depicted in Figure 4-7 to Figure 4-14. The maximum gimbal angle considered during training is 0.15 rad, while for execution this value was increased to 0.17 rad. This demonstrated the ability of the NN to extrapolate its behavior to a higher range of gimbal values. This extrapolation not only improves the landing accuracy in terms of position and velocity, but also the overall robustness of the NN, as it will be discussed in the next section, based on the Monte-Carlo analysis. In addition, although the NN has been trained without wind in the environment, it has been tested considering wind disturbances.

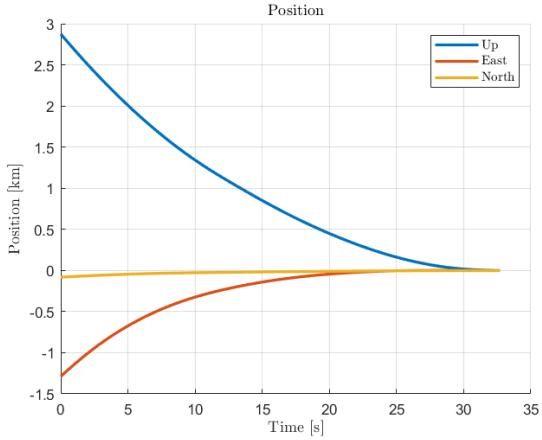

Figure 4-7: Position profile for the 3D landing scenario

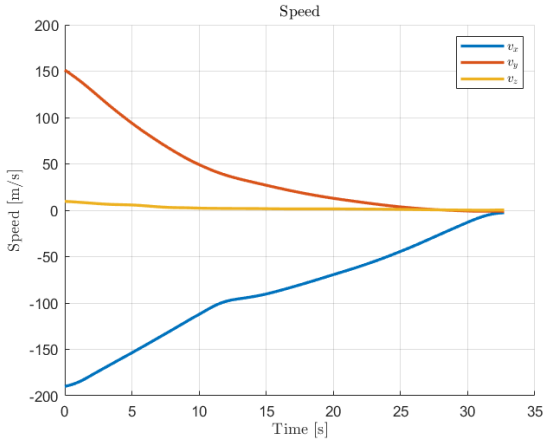

Figure 4-8: Velocity profile for the 3D landing scenario

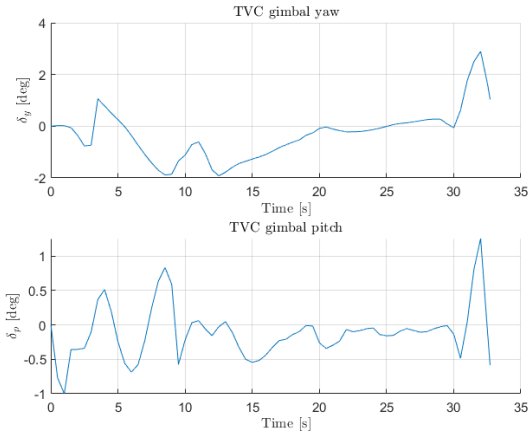

Figure 4-9: Gimbal angles profiles for the 3D landing scenario

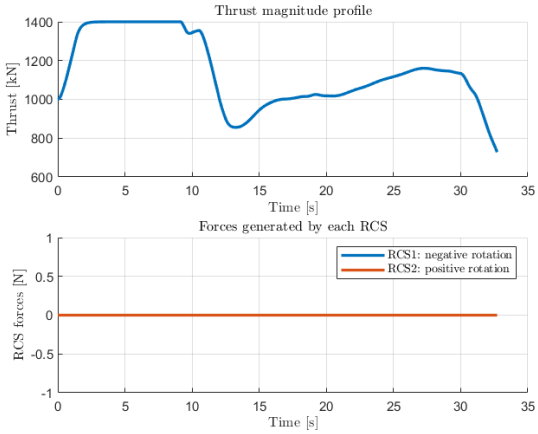

Figure 4-10: TVC and RCS thrust profile for the 3D landing scenario

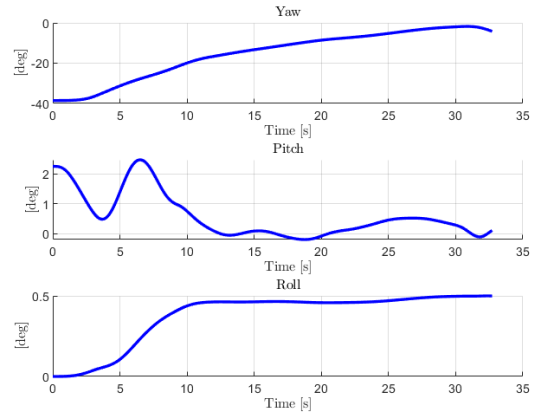

Figure 4-11: Attitude angles profiles for the 3D landing scenario

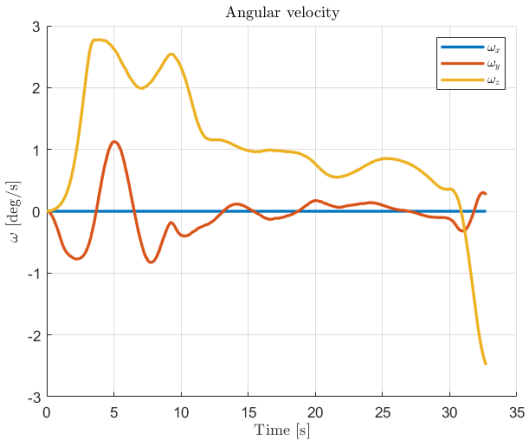

Figure 4-12: Angular velocity profiles for the 3D landing scenario

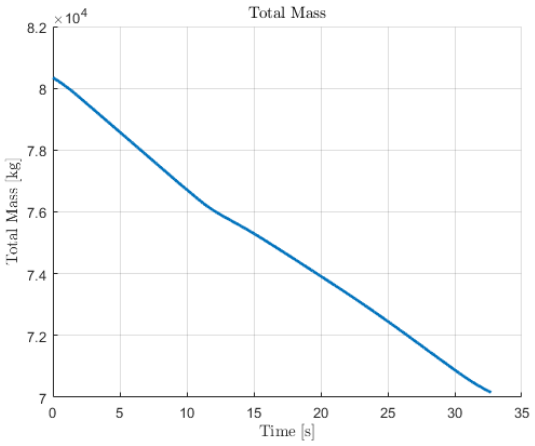

Figure 4-13: Mass profile for the 3D landing scenario

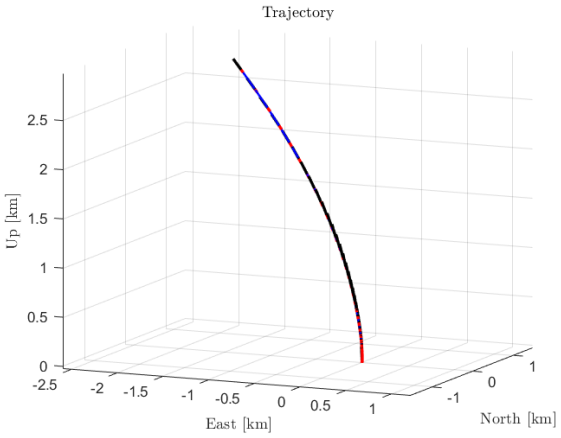

Figure 4-14: 3D landing trajectory

Table 4-4 confirms the validity of the obtained results in terms of final landing conditions.

Table 4-4: Position, velocity and attitude at touchdown for the 3D complete scenario

Coordinate	Final position [m]	Final velocity [m/s]	Attitude angles [deg]	Angular velocity [deg/s]
X / roll	0	-2.663	0.4993	0
Y / pitch	-1.813	-1.162	0.109	0.2673
Z / yaw	-1.105	0.2874	-4.205	-2.48

5 MONTE-CARLO ANALYSIS

This section presents the Monte-Carlo results obtained using the NN described above, considering wind and the following parameter dispersions:

- Propellant mass: uniform distribution with standard deviation of 500kg;
- Initial position: normal distribution, with mean equal to the nominal position, and standard deviation of 5% of the initial position, 3-sigma;
- Initial velocity: normal distribution, with mean equal to the nominal velocity, and standard deviation of 3% of the initial velocity, 3-sigma;
- Seed of the random generator for the wind model: any mean and standard deviation.

The landing requirements are defined in terms of landing accuracy with the following values:

- Horizontal position at touchdown: 10 m
- Horizontal velocity at touchdown: 3 m/s
- Vertical velocity at touchdown: 10 m/s
- Incidence angle at touchdown: 5 deg
- Rotational velocities at touchdown: 3 deg/s

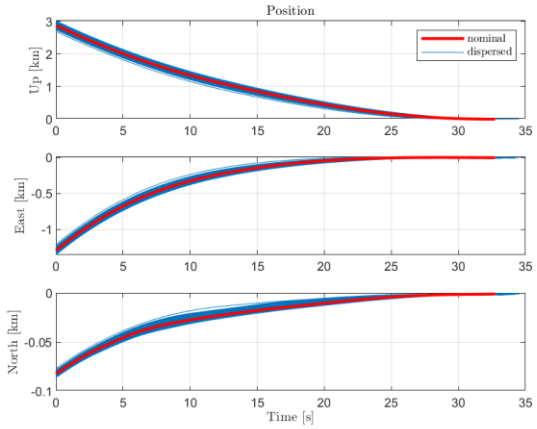

Figure 5-1: Position profiles for the 3D landing scenario for 1000 MC shots

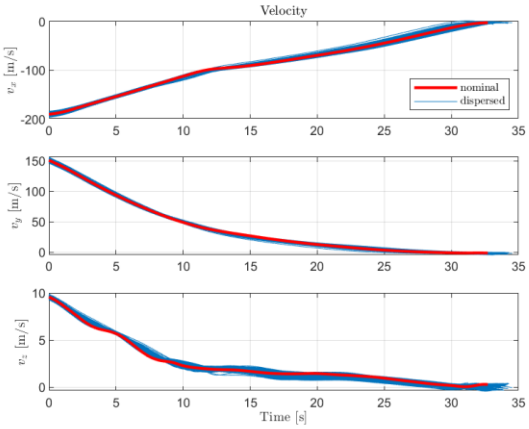

Figure 5-2: Velocity profile for the 3D landing scenario 1000 MC shots

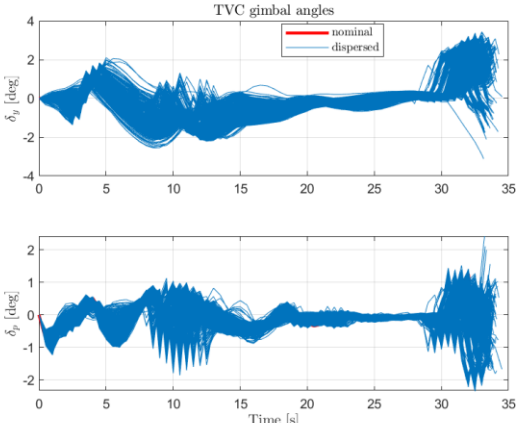

Figure 5-3: Gimbal angles profiles for the 3D landing scenario 1000 MC shots

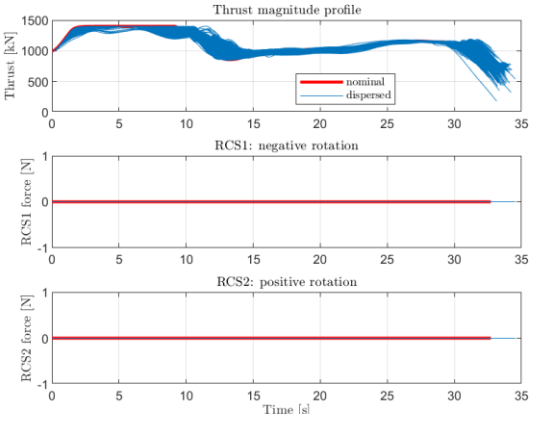

Figure 5-4: TVC and RCS thrust profile for the 3D landing scenario 1000 MC shots

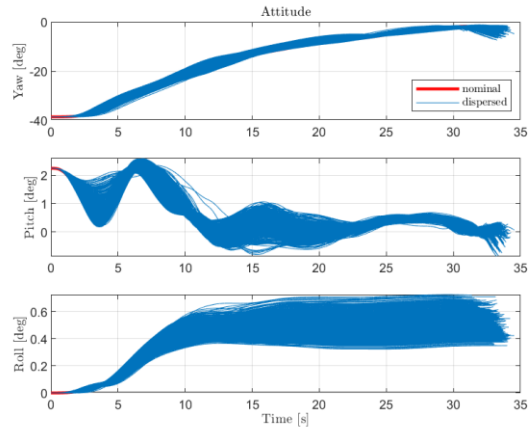

Figure 5-5: Attitude angles profiles for the 3D landing scenario 1000 MC shots

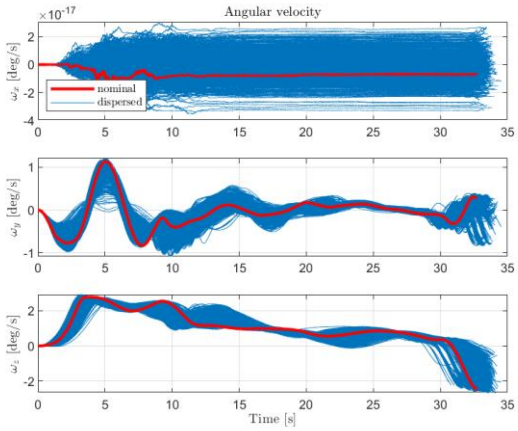

Figure 5-6: Angular velocity profiles for the 3D landing scenario 1000 MC shots

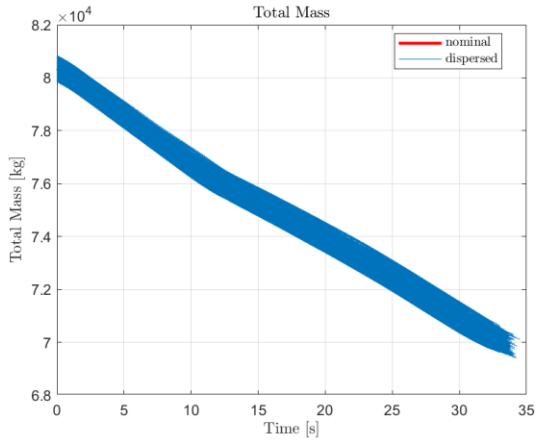

Figure 5-7: Mass profile for the 3D landing scenario 1000 MC shots

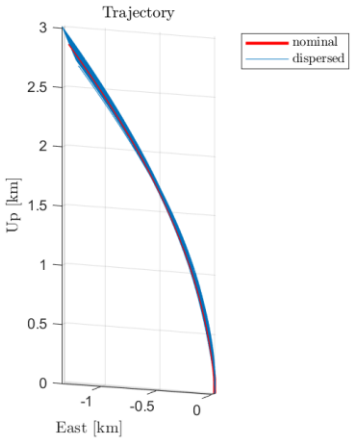

Figure 5-8: 3D landing trajectory 1000 MC shots

Figure 5-1 to Figure 5-8 illustrate the results from 1000 MC runs with the nominal dispersions described above, including wind. For this set of dispersions and disturbance, the success rate, i.e., the percentage of MC runs that satisfied the requirements, is 97%, meaning that only 30 runs out of 1000 did not satisfy all the requirements. By observing the failed runs, it is possible to confirm that all of the 30 failed runs complied with all of the requirements, except the horizontal velocity requirement of 3 m/s. In Figure 5-9, the norm of the horizontal velocity and the corresponding requirement, for the 1000 runs, is depicted, including the runs that successfully satisfied that requirement and the ones that did not. The error for the failed runs is always below 1.5 m/s, which is considerably small when compared to the dispersed initial conditions.

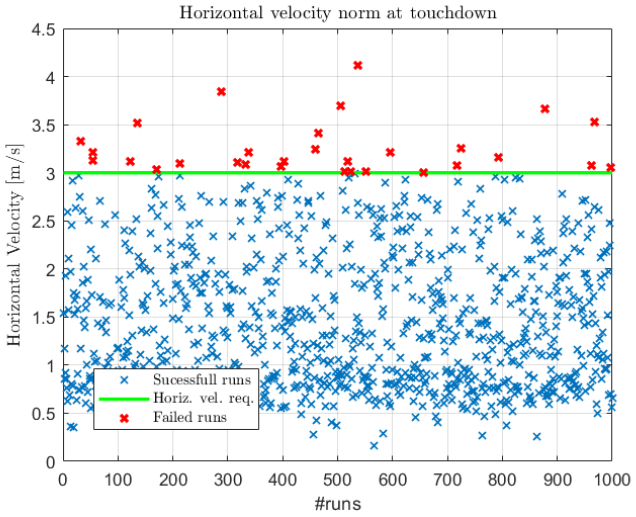

Figure 5-9: Horizontal velocity norm at touchdown and horizontal velocity requirement for the 1000 runs, including the successful and failed runs.

It is also remarked that the same NN led to a success rate of 100% in a 50-shot MC simulation for the scenario without wind. Since the NN was not trained with any wind disturbance, and given the success rate and the analysis of the failed runs reported here, one may reach the conclusion that the NN can deal with unknown disturbances relatively well and that further robustification of the NN is potentially achievable by considering with disturbance during the training process.

Furthermore, the MC results for the NN have been also compared against the ones obtained with the SCVX algorithm in terms of fuel consumption and landing accuracy. The SCVX algorithm used in this comparison is re-executed in flight at each 5 seconds. The results, presented in Table 5-1, show that, although the results are rather close, SCVX still provides the cheapest trajectory in terms of fuel consumption, but not in terms of landing accuracy, where Deep-RL yielded better results.

Another important remark is that, comparing the SCVX implementation, as well as the different selections of hyperparameters of the Deep-RL G&C, it is seen that improvements in fuel consumption generate larger errors in terms of landing position and velocity accuracies, and vice-versa. Hence, this can indicate that we are very close to a Pareto boundary of optimality, which provides a good indication in terms of reliability of the methods and algorithms developed.

Table 5-1: Comparison between the MC results of the Deep-RL Guidance and Control approach and the SCVX Guidance algorithm. The lowest fuel consumption/higher accuracy is highlighted in green

		Mean	**Standard Deviation**
Neural Network, MC simulation	Fuel consumption [kg]	10304.019	158.48
	Position accuracy [m]	3.957	1.4181
	Velocity accuracy [m/s]	2.89694	0.44976
SCVX, executed each 5s, MC simulation	Fuel consumption [kg]	9763.39	69.13
	Position accuracy [m]	8.7947	0.40740
	Velocity accuracy [m/s]	1.54723	0.36812

6 REACHABILITY ANALYSIS

The reachability analysis described next used the algorithms and tools from [3], [4], applied to the problem at hand, considering the NN trained for the vertical 1DoF scenario. The NN receives as input the altitude and the difference between the actual velocity and a target velocity (function of the state) and output the thrust magnitude.

In the context of the analysis, a smaller NN was trained, augmenting the main NN, so that the observation state becomes only the altitude and actual velocity. Moreover, in order to tackle the lack of support for 'tanh' activation function by the tool, another (smaller) NN was adopted to approximate the 'tanh' function present in the last layer of the original NN, with only 'relu' and 'linear' layers.

After having the NN and the launcher (linear) discrete-time dynamics defined, the closed-loop propagation has been performed considering the Greedy Sim Guided partitioner and the CROWN propagator. Setting an initial range for the initial conditions, the results of the simulation are reported in Figure 6-1, where the boxes correspond to the set-valued state propagation by using the RLV dynamics and the Deep-RL NN.

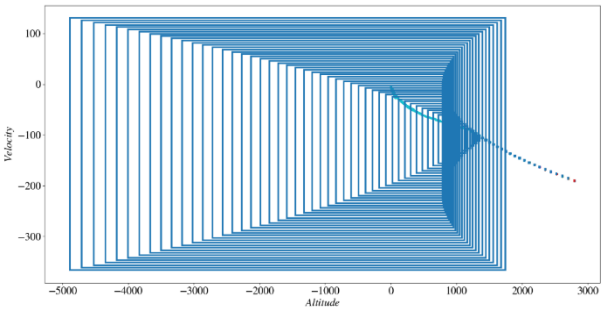

Figure 6-1 - Preliminary results with the V&V robustness toolbox

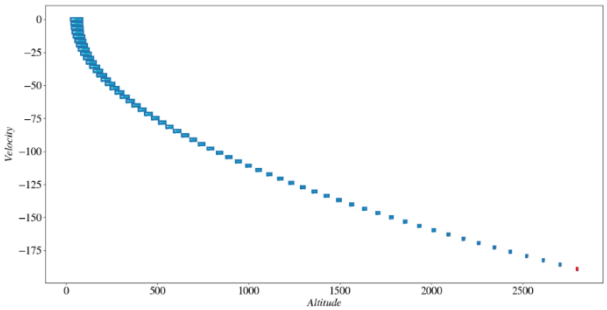

Figure 6-2 - Validation results obtained with the smaller NN that mimics the larger NN.

As observed in the figure, the cyan dots (equivalent to MC runs) follow a trajectory that lands at 0 m altitude with relatively small vertical velocity, indicating that the NN is able to correctly make the RLV land, at least for those cases. However, initially the blue (reachability) boxes tightly surround the cloud of blue dots until a specific step, where they start to increase in area, meaning that the estimated reachable set is also increasing.

In order to tackle this issue, a so-called 'teacher-student' approach has been adopted. This consists in the use of a smaller NN, i.e., with less layers, as described next. The larger NN, composed of the concatenation of the NNs described above, is used together with a reference trajectory of a successful landing to generate a dataset, where the inputs are the state and the outputs correspond to the thrust command. This dataset is then used to train the considerably smaller NN, with a supervised learning approach. After the training is complete, the smaller NN is then the one used in the validation tool, yielding the results presented in Figure 6-2. It is now apparent that the boxes actually validate this new and smaller NN. By focusing on the final part of the propagation, is possible to see that, while the NN is not able to exactly land the RLV at zero altitude, it does ensure that the RLV reaches a sufficiently low altitude such that, afterwards, another simpler (linear) controller technique can be employed for the remaining part of the landing.

7 CONCLUSIONS

The results showed in this paper represent an interesting step ahead towards the implication of AI techniques within the GNC field. The complexity and sparsity of the scenario makes the use of NN-based methods highly challenging. The approach adopted in this work, which led to the implementation of the NN-based Deep-RL G&C for the RETALT-1 launcher in the complete 3D (6DoF) landing scenario is to gradually increase the complexity of the problem. Starting from the simple 1D scenario, in which only the thrust magnitude was controlled, the study moved through the 2D vertical, 2D without rate limiters, 3D without rate limiters. This approach allowed to better handle the tuning of the NN hyperparameters and the reward function shaping. Finally, the Deep-RL technique presented, applied to a realistic RLV scenario, yielded remarkable results, not only for the nominal landing, but also on the extensive MC campaigns, passing all of the V&V tests with a significantly high level of confidence, with the additional advantage of being onboard implementable. This was further confirmed by the fact that the results obtained were comparable to the ones from more classical approaches such as SCVX.

Regarding the NN validation, in the context of this work, the reachability analysis has been performed only for the 1D scenario NN, showing remarkable outcomes about the robustness of the NN-based control. However, the room for improvements is still significant and further work towards the complete 3D scenario are required but promising.

Finally, the whole implementation and verification of the Deep-RL G&C for RETALT-1 was performed within the ESA-i4GNC framework, which allows for the design and testing of AI-based GNC solutions, providing a fast track to maturation of novel GNC techniques.

8 ACKNOWLEDGEMENTS

The results presented in this paper have been achieved under funding by the ESA TDE programme with contract ESA contract No. 4000134108/21/NL/CRS. The view expressed in this paper can in no way be taken to reflect the official opinion of the European Space Agency. The authors would also like to acknowledge the remaining contributors to the AI4GNC project, namely: J. M. Lemos, F. Parente, B. Costa, J. Igreja (INESC-ID), A. Marcos, D. Navarro-Tapia (TASC), and A. Rantzer, A. Cervin, V. Renganathan, J. Gronqvist (University of Lund).

9 REFERENCES

[1] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. and Fergus, R., *Intriguing properties of neural networks*, arXiv preprint arXiv:1312.6199., 2013.

[2] Lillicrap, Timothy P., Hunt, Jonathan J., Pritzel, Alexander, Nicolas, Hess, Erez, Tom, Tassa, Yuval, Silver, David & Wierstra, Daan (2019). *Continuous Control with Deep Reinforcement Learning*. ICLR, https://arxiv.org/abs/1509.02971.

[3] Everett M, Habibi G, How JP. Robustness Analysis of Neural Networks via Efficient Partitioning with Applications in Control Systems. Proc Am Control Conf. 2021; doi:10.23919/ACC50511.2021.9483033

[4] Everett M, Habibi G, How JP. Efficient Reachability Analysis of Closed-Loop Systems with Neural Network Controllers. 2021:4384-4390. doi:10.1109/icra48506.2021.9561348

[5] B. Gaudet, R. Linares, R. Furfaro (2018), Deep Reinforcement Learning for Six Degree-Of-Freedom Planetary Powered Descent and Landing, ArXiv abs/1810.08719.

[6] G. D. Zaiacomo, G. Medici, A. Princi, P. Ghignoni, A. Botelho, M. M. Arlandis, C. Recupero, A.Fabrizi, V. Fernandez, and G. Guidotti, "RETALT: Development of key flight dynamics and GNC technologies for reusable launchers," in Proceedings of the International Astronautical Congress (IAC), September 2022