Reinforcement Learning for Robust Missile Autopilot Design

Bernardo Gonçalves Ferreira Cortez
bernardo.g.f.cortez@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa, Portugal

January 2021

Abstract

Designing missiles' autopilot controllers has been a complex task, given the extensive flight envelope and the nonlinear flight dynamics. A solution that can excel both in nominal performance and in robustness to uncertainties is still to be found. While Control Theory often debouches into parameters' scheduling procedures, Reinforcement Learning has presented interesting results in ever more complex tasks, going from videogames to robotic tasks with continuous action domains. However, it still lacks clearer insights on how to find adequate reward functions and exploration strategies. To the best of our knowledge, this work is pioneer in proposing Reinforcement Learning as a framework for flight control. In fact, it aims at training a model-free agent that can control the longitudinal non-linear flight dynamics of a missile, achieving the target performance and robustness to uncertainties. To that end, under TRPO's methodology, the collected experience is augmented according to HER, stored in a replay buffer and sampled according to its significance. Not only does this work enhance the concept of prioritized experience replay into BPER, but it also reformulates HER, activating them both only when the training progress converges to suboptimal policies, in what is proposed as the SER methodology. Besides, the Reward Engineering process is carefully detailed. The results show that it is possible both to achieve the target performance and to improve the agent's robustness to uncertainties (with low damage on nominal performance) by further training it in non-nominal environments, therefore validating the proposed approach and encouraging future research in this field.
Keywords: Reinforcement Learning, TRPO, HER, flight control, missile autopilot.

1. Introduction

Over the last decades, designing the autopilot flight controller for a system such as a missile has been a complex task, given (i) the non-linear dynamics and (ii) the demanding performance and robustness requirements. This process is classically solved by tedious scheduling procedures, which often lack the ability to generalize across the whole flight envelope and different missile configurations.

Reinforcement Learning (RL) constitutes a promising approach to address the latter issues, given its ability of controlling systems from which it has no prior information. This motivation increases as the system to be controlled grows in complexity, for the plausible hypothesis that the RL agent could benefit from having a considerably wider range of possible (combinations of) actions. By learning the cross-coupling effects, the agent is expected to converge to the optimal policy faster.

The longitudinal dynamics (cf. section 3), however, constitutes a mere first step, necessary to the posterior possible expansion of the approach to the whole flight dynamics. This work is, hence, motivated by the will of finding a RL algorithm that can control the longitudinal flight dynamics of a Generic Surface-to-Air Missile (GSAM) with no prior information about it, being, thus, model-free.

2. Problem Formulation

One episode is defined as the trial of following a 5s-long a_z reference signal consisting of two consecutive steps whose amplitude and rise times are randomly generated except when deploying the agent for testing purposes [1]. With a sampling time of 1ms, an episode includes 5000 steps, from which 2400 are defined as transition periods (the 600 steps composing 0.6s after each of the four rise times), whilst the remaining 2600 are resting periods.

The algorithm must achieve the target performance established in terms of the following requirements:

1. Static error margin = 0.5%
2. Overshoot < 20%
3. Rise time < 0.6s
4. Settling time (5%) < 0.6s
5. Bounded actuation

6. Smooth actuation

Besides, the present work also aims at achieving and improving the robustness of the algorithm to conditions different from the training ones.

From the aforementioned requirements, the RL problem was formulated as the training of an agent that succeeds when the performance of an episode meets the levels defined in table 1 in terms of the maximum stationary tracking error $|e_z|_{\max,r}$, of the overshoot, of the actuation magnitude $|\eta|_{\max}$ and of the actuation noise levels both in resting ($\eta_{noise,r}$) and transition ($\eta_{noise,t}$) periods.

Requirement	Achieved Value	
$\|e_z\|_{\max,r}$	0.5	[g]
Overshoot	20	[%]
$\|\eta\|_{\max}$	15	[º]
$\eta_{noise,r}$	1	[rad]
$\eta_{noise,t}$	0.2	[rad]

Table 1: Performance Objectives

3. Model

The GSAM's flight dynamics to be controlled is modelled by a non-linear system decomposed into Translation (cf. equation (1)), Rotation (cf. equation (2)), Position (cf. equation (3)) and Attitude (cf. equation (4)) terms, as Peter et al. [2] described. Equations (1) to (4) follow Peter et al.'s [2] notation. The Longitudinal Approximation consists of the hypothesis that the longitudinal ($x0z$ plane of the B-frame [2]) can be considered separately from the remaining ones, not considering cross-coupling effects.

$$\dot{V}_G^E = \frac{1}{m} F_{T,G} - \omega^E \times V_G^E \tag{1}$$

$$\dot{\omega}^E = (J_G)^{-1} \left(M_{T,G} - \omega^E \times J_G \omega^E \right) \tag{2}$$

$$\dot{r}_G^E = M_T V_G^E \tag{3}$$

$$\left[\dot{\Phi}^E, \dot{\Theta}^E, \dot{\Psi}^E \right]^T = R.\omega^E \tag{4}$$

3.1. Actuator Dynamics

The system is actuated by the deflection η of the aerodynamic equivalent elevator, which is mapped from the GSAM's physical control surfaces'[1] deflections δ_i according to equation (5).

$$\eta = \frac{\delta_1 - \delta_2 - \delta_3 + \delta_4}{4} \tag{5}$$

The actuator system is, thus, the system that receives the desired (commanded) elevator deflection η_{com} and outputs the actual deflection, modelling the dynamic response of the physical fins with its deflection limit of 30. The latter is assumed to be a second order system with the following closed loop characteristics:

1. Natural frequency ω_n of 150 rad.s^{-1}
2. Damping factor λ of 0.7

[1] Four fins attached to the missile's tail.

4. Background

4.1. Topic Overview

RL has been the object of research after the foundations laid out by Sutton et al. [3], with applications going from the Atari 2600 games, to the MuJoCo games, or to robotic tasks (like grasping...) or other classic control problems (like the inverted pendulum).

TRPO [4] has become one of the commonly accepted benchmarks for its success achieved by the cautious trust region optimization and monotonic reward increase. Perpendicularly, Lillicrap et al. [5] proposed DDPG, a model-free off-policy algorithm, revolutionary not only for its ability of learning directly from pixels while maintaining the network architecture simple, but mainly because it was designed to cope with continuous domain action. Contrarily to TRPO, DDPG's off-policy nature implied a much higher sample efficiency, resulting in a faster training proccess. Both TRPO and DDPG have been the roots for much of the research work that followed.

On the one side, some authors valued more the benefits of an off-policy algorithm and took the inspiration in DDPG to develop TD3 [6], addressing DDPG's problem of over-estimation of the states' value. On the other side, others preferred the benefits of off-policy algorithm and proposed interesting improvements to the original TRPO, either by reducing its implementation complexity [7], by trying to decrease the variance of its estimates [8] or even by showing the benefits of its interconnection with replay buffers [9].

Apart from these, a new result began to arise: agents were ensuring stability at the expense of converging to suboptimal solutions. Once again, new algorithms were conceived in each family, on- and off-policy. Haarnoja and Tang proposed to express the optimal policy via a Boltzmann distribution in order to learn stochastic behaviors and to improve the exploration phase within the scope of an off-policy actor-critic architecture: Soft Q-learning [10]. Almost simultaneously, Schulman et al. published PPO [11], claiming to have simplified TRPO's implementation and increased its sample-efficiency by inserting an entropy bonus to increase exploration performance and avoid premature sub-optimal convergence. Furthermore, Haarnoja et al. developed SAC [12], in an attempt to recover the training stability without losing the entropy-encouraged exploration and Nachum et al. pro-

posed Smoothie [13], allying the trust region implementation of PPO with DDPG.

Finally, there has also been research done on merging both on-policy and off-policy algorithms, trying to profit from the upsides of both, like IPG [14], TPCL [15], Q-Prop [16] and PGQL [17].

4.2. Trust Region Policy Optimization

TRPO is an on-policy model-free RL algorithm that aims at maximizing the discounted sum of future rewards (cf. equation (6)) following an actor-critic architecture and a trust region search.

$$R(s_t) = \sum_{l=0}^{\infty} \gamma^l r(t+l) \tag{6}$$

Initially, Schulman et al. [4] proposed to update the policy estimator's parameters with the conjugate gradient algorithm followed by a line search. The trust region search would be ensured by a hard constraint on the Kullback-Leibler divergence D_{KL}.

Briefly after, Kangin et al. [9] proposed an enhancement, augmenting the training data by using replay buffers and GAE [8]. Also contrarily to the original proposal, Kangin et al. train the value estimator's parameters with the ADAM optimizer and the policy's with K-FAC [18]. The former was implemented within a regression between the output of the Value NN, $\hat{V}$, and its target, V', whilst the latter used equation (7) as the loss function, which has got a first term concerning the objective function being maximized (cf. equation (6)) and a second one penalizing differences between two consecutive policies outside the trust region, whose radius is the hyperparameter δ_{TR}.

$$L_P = -\mathbb{E}_{s_0,a_0,\dots}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t,a_t)\right] + \\ +\alpha \max(0, \mathbb{E}_{a\sim\pi_{\theta_{old}}}\left[D_{KL}(\pi_{\theta_{old}(a)}, \pi_{\theta_{new}}(a))\right] - \delta_{TR}) \tag{7}$$

For matters of simplicity of implementation, Schulman et al. [4] rewrite equation (7) in terms of collected experience (cf. equation (8)), as a function of two different stochastic policies, $\pi_{\theta_{new}}$ and $\pi_{\theta_{old}}$ and the GAE.

$$L_P = -\mathbb{E}_s \mathbb{E}_{a\sim\pi_{\theta_{old}}}\left[GAE_{\theta_{old}}(s)\frac{\pi_{\theta_{new}}(a)}{\pi_{\theta_{old}}(a)}\right] + \\ +\alpha \max(0, \mathbb{E}_{a\sim\pi_{\theta_{old}}}\left[D_{KL}(\pi_{\theta_{old}}(a), \pi_{\theta_{new}}(a))\right] - \delta_{TR}) \tag{8}$$

Both versions of TRPO use the same exploration strategy: the output of the Policy NN is used as the mean of a multivariate normal distribution whose covariance matrix is part of the policy trainable parameters.

4.3. Hindsight Experience Replay

The key idea of HER [19] is to store in the replay buffers not only the experience collected from interacting with the environment, but also about experiences that would be obtained, had the agent followed a different goal. HER was proposed as a complement of sparse rewards.

4.4. Prioritized Experience Replay

When working with replay buffers, randomly sampling experience can be outperformed by a more sophisticated heuristic. Schaul et al. [20] proposed Prioritized Experience Replay, sampling experience according to their significance, measured by the TD-error. Besides, the strict required performance (cf. table 1) causes the agent to seldom achieve success and the training dataset to be imbalanced. Thus, the agent simply overfits the "failure" class. Narasimhan et al. [21] have addressed this problem by forcing 25% of the training dataset to be sampled from the less represented class.

5. Application of RL to the Missile's Flight

5.1. Algorithm

As explained in section 1, the current problem required an on-policy model-free RL algorithm. Among them, not only is TRPO a current state-of-the-art algorithm (cf. section 4.1), but it also presents the attractiveness of the trust region search, avoiding sudden drops during the training progress, which is a very interesting feature to be explored by the industry, whose mindset is often aiming at robust results. TRPO was, therefore, the most suitable choice.

5.1.1 Modifications to original TRPO

The present implementation was inspired in the implementations proposed by Schulman et al. [4] and by Kangin et al. [9] (cf. section 4.2). There are several differences, though:

1. The reward function is given by equation (15), whose relative weights w_i can be found in [1].

$$f_1 = -w_1.|e_z| \tag{9}$$

$$f_2 = \begin{cases} 0 & \text{if } |\eta| < \eta_{max} \\ -w_2 & \text{otherwise} \end{cases} \tag{10}$$

$$\eta_{slope} = \frac{\Delta\eta}{t_s} \tag{11}$$

$$f_3 = -w_3.|\eta_{slope}| \tag{12}$$

$$condition = |e_z| < 3g \wedge |\eta| < 0.2 \wedge \wedge |e_u| < e_{u,\max} \tag{13}$$

$$f_4 = \begin{cases} w_4.\frac{e_{u,\max} - |e_u|}{e_{u,\max}} & \text{if } condition \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

$$r = \sum_{i=1}^{4} f_i \tag{15}$$

2. Both neural networks have got three hidden layers whose sizes - h_1, h_2 and h_3 - are related with the observations vector (their input size) and with the actions vector (output size of the Policy NN).

3. Observations are normalized (cf. equation (16)) so that the learning process can cope with the different domains of each feature.

$$obs_{norm} = \frac{obs - \mu_{online}}{\sigma_{online}} \tag{16}$$

In equation (16), μ_{online} and σ_{online} are the running mean value and the running variance of the set of observations collected along the whole training process, which are updated with information about the newly collected observations after each training episode.

4. The proposed exploration strategy is deeply rooted on Kangin et al.'s [9], meaning that, the new action η is sampled from a normal distribution (cf. equation (17)) whose mean is the output of the policy neural network and whose variance is obtained according to equation (18).

$$\eta \sim N(\mu_\eta, \sigma_\eta^2) \tag{17}$$

$$\sigma_\eta^2 = e^{\sigma_{\log}^2} \tag{18}$$

Although similar, this strategy differs from the original in $\sigma_{\log}^2$ (cf. equation (19)), which is directly influenced by the tracking error e_z through $\sigma_{\log,tune}^2$.

$$\sigma_{\log}^2 = \sigma_{\log,train}^2 + \sigma_{\log,tune}^2(e_z) \tag{19}$$

5. Equation (22) was used as the loss function of the policy parameters, modifying equation (8) in order to emphasize the need of reducing D_{KL} with a term that linearly penalizes D_{KL} and a quadratic term that aims at fine-tuning it so that it is closer to the trust region radius δ_{TR}, encouraging as big update steps as possible.

$$L_1 = \mathbb{E}_s \mathbb{E}_{\eta \sim \pi_{\theta_{old}}} \left[GAE_{\pi_{\theta_{old}}}(s) \frac{\pi_{\theta_{new}}(\eta)}{\pi_{\theta_{old}}(\eta)} \right] \tag{20}$$

$$L_2 = \mathbb{E}_{\eta \sim \pi_{\theta_{old}}} \left[D_{KL}(\pi_{\theta_{old}}(\eta), \pi_{\theta_{new}}(\eta)) \right] \tag{21}$$

$$L_P = -L_1 + \alpha \left(\max(0, L_2 - \delta_{TR}) \right)^2 + \beta L_2 \tag{22}$$

Having the previously mentioned exploration strategy, π_θ is a Gaussian distribution over the continuous action space (cf. equation (23), where μ_η and θ_η are defined in equation (17)).

$$\pi_\theta(\eta) = \frac{1}{\sigma_\eta \sqrt{2\pi}} e^{-\frac{(\eta - \mu_\eta)^2}{2\sigma_\eta^2}} \tag{23}$$

Hence, L_1 (cf. equation (20)) is given by equation (24), where n is the number of samples in the training batch, assuming that all samples in the training batch are independent and identically distributed[2].

$$L_1 = \left[\frac{1}{n} \sum_{i=1}^{n} GAE_{\pi_{\theta_{old}}} s_i \right] . \Pi_{i=1}^{n} \left[\frac{\pi_{\theta_{new}}(\eta_i)}{\pi_{\theta_{old}}(\eta_i)} \right] \tag{24}$$

Moreover, L_2 is given by equation (25).

$$L_2 = \frac{1}{n} \sum_{i=1}^{n} D_{KL}(\pi_{\theta_{old}}(\eta_i), \pi_{\theta_{new}}(\eta_i)) \tag{25}$$

6. ADAM was used as the optimizer of both NN for its wide cross-range success and acceptability as the default optimizer of most ML application.

[2] We can assume they are (i) independent because they are sampled from the replay buffer (stage 5 of algorithm 2, section 5.1.5), breaking the causality correlation that the temporal sequence could entail, and (ii) identically distributed because the exploration strategy is always the same and, therefore, the stochastic policy π_θ is always a Gaussian distribution over the action space (cf. equation (23)).

5.1.2 Hindsight Experience Replay

The present goal is not defined by achieving a certain final state, but, instead, a certain performance in the whole sequence of states that constitutes an episode. For this reason, choosing a different goal must mean, in this case, to follow a different reference signal. After collecting a full episode, those trajectories are replayed with new goals, which are sampled according to two different strategies. These strategies dictate the choice of the amplitudes of the two consecutive steps of each new reference signal.

The first strategy - *mean* strategy - consists of choosing the amplitudes of the steps of the command signal as the mean values of the measured acceleration during the first and second resting periods, respectively. Similarly, the second strategy - *final* strategy - consists of choosing them as the last values of the measured signal during each resting period. Apart from the step amplitudes, all the other original parameters [1] of the reference signal are kept.

5.1.3 Balanced Prioritized Experience Replay

Being l_i the priority level of the experience collected in step i (cf. equation (26), with $e_{u,\max} = 0.01$), N_j the number of steps with priority level j and ρ_j the proportion of steps with priority level j desired in the training datasets, BPER was implemented according to algorithm 1.

$$l_i = \begin{cases} 1 & \text{if } |e_z|_i < 0.5g \wedge |\eta|_i < \frac{|\eta|_{max}}{2} \wedge |e_u|_i < e_{u,\max} \\ 0 & \text{otherwise} \end{cases} \tag{26}$$

Notice that $P(i)$ and p_i follow the notation of Schaul et al. [20] (assuming $\alpha = 1$), in which $p_i = \frac{1}{rank(i)}$ matches the rank-based prioritization with $rank(i)$ meaning the ordinal position of step i when all steps in the replay buffers are ordered by the magnitude of their temporal differences.

Algorithm 1: Balanced Prioritized Experience Replay

if $N_1 < 0.25(N_0 + N_1)$ **then**
  $\rho_1 = 0.25$
  $s_j = \sum_i p_i, \forall_i (l_i = j)$ with $j \in \{0, 1\}$
else
  $\rho_1 = 0.5$
  $s_0 = s_1 = \sum_i p_i, \forall_i$
end
$\rho_0 = 1 - \rho_1$

$$P(i) = \begin{cases} \rho_1 \times \frac{p_i}{s_1} & \text{if } l_i = 1 \\ \rho_0 \times \frac{p_i}{s_0} & \text{otherwise} \end{cases}$$

As condensed in algorithm 1, when there is less than 25% of successful steps in the replay buffers, the successful and unsuccessful subsets of the replay buffers are sampled separately, with 25% coming from the successful subset. In a posterior phase of training, when there is already more than 25% of successful steps, both subsets are molten. In either cases, sampling is always done according to the temporal differences, i.e., a step with higher temporal difference has got a higher chance of being sampled.

5.1.4 Scheduled Experience Replay

Having HER (cf. section 5.1.2) and BPER (cf. section 5.1.3) dependent on a condition - the SER condition - is hereby defined as SER. The SER condition is exemplified in equation (27), where $\bar{e}_{z,past}$ stands for the mean tracking error of the previously collected episode .

$$\bar{e}_{z,past} \leq 2g \tag{27}$$

Contrarily to its original context (cf. section 4.3), the reward function is not sparse and was already able of achieving near-target performance without HER. The hypothesis, in this case, is that HER can be a complementary feature, by activating it only when the agent converges to suboptimal policies.

Moreover, without HER, BPER adds less benefit, since there is no special reason for the agent to believe that some part of the collected experience is more significant than other.

5.1.5 Algorithm Description

1. One batch B of trajectories T, is collected.
2. If the SER condition (cf. section 5.1.4) holds, B is augmented according to HER (cf. section 5.1.2).
3. The targets for the Value NN $V'(s_t)$ and the $GAE(s_t)$ are computed and added to the trajectories.
4. B is stored in the replay buffer, discarding the oldest batch: the replay buffer R contains data collected from the last policies and works as a FIFO queue.
5. If the SER condition (cf. section 5.1.4) holds, the training dataset is sampled from R according to BPER (cf. section 5.1.3). Otherwise, the entire information available in R is used as training dataset.
6. The value parameters and the policy parameters are updated.
7. The trust region parameters are updated.

Algorithm 2: Implemented TRPO with a Replay Buffer and SER

Initialization
while *training* **do**
1. Collect experience, sampling actions from policy π
2. Augment the collected experience with synthetic successful episodes (SER)
3. $(a_t, s_t, r_t) \leftarrow V'(s_t)$ and $GAE(s_t)$ **for all** (a_t, s_t, r_t) **in** T, **for all** T **in** B
4. Store the newly collected experience in the replay buffer
5. Sample the training sets from the replay buffer (SER)
6. Update all value and policy parameters
7. Update trust region

end

5.2. Methodology

As further detailed in [3], the established methodology (i) progressively increases the amplitude of the randomly generated command signal and (ii) intermediate testing of the agent's performance against a -10g/10g double step without exploration, in order , respectively, (i) to avoid overfitting and (ii) to decide whether or not to finish the training process.

5.3. Robustness Assessments

The formal mathematical guarantee of robustness of a RL agent composed of neural networks cannot be done in the same terms as the one of linear controllers. It was, hence, evaluated by deploying the nominal agent in non-nominal environments. Apart from testing its performance, the hypothesis is also that training this agent in the latter can improve its robustness. To do so, the train of the best found nominal agent was resumed in the presence of non-nominalities (cf. section 5.3.1). This train and its resulting best found agent are henceforward called *robustifying train* and *robustified agent*, respectively.

5.3.1 Robustifying Trains

Three different modifications were separately made in the provided model (cf. section 3), in order to obtain three different non-nominal environments, each of them modelling latency, estimation uncertainty in M and h (cf. equations (28) and (29)) and parametric uncertainty in the aerodynamic coefficients C_z and C_m (cf. equations (30) and (31)).

$$M = (1 + \Delta M) \times M_{nom} \tag{28}$$

$$h = (1 + \Delta h) \times h_{nom} \tag{29}$$

$$C_z = (1 + \Delta C_z) \times C_{z,nom} \tag{30}$$

$$C_m = (1 + \Delta C_m) \times C_{m,nom} \tag{31}$$

In the case of non-nominal environments including latency, the range of possible values is $[0, l_{max}] \cap \mathbb{N}^0$, whereas in the other cases, the uncertainty is assumed to be normally distributed, meaning that, following Peter's [2] line of thought, its domain is $[\mu - 3\sigma, \mu + 3\sigma]$[4].

Before each new episode of a robustifying train, the non-nominality new value (either latency or one of the uncertainties) was sampled from a uniform distribution over its domain and kept constant during the entire episode. In each case, four different values were tried for the bounds of the range of possibilities (either l_{max} or 3σ):

1. $l_{max} \in \{1, 3, 5, 10\}$ [ms]
2. $(3\sigma_{estimation}) \in \{1, 2, 3, 5\}$ [%]
3. $(3\sigma_{parametric}) \in \{5, 7, 10, 15\}$ [%]

Values whose robustying train had diverged after 2500 episodes were discarded. The remaining were run for a total of 5000 episodes (cf. section 6.4).

6. Results

6.1. Expected Results

Empirically, it has been seen that agents that struggle to control η's magnitude within its bounds (cf. table 1) are unable of achieving low error levels without increasing the level of noise, if they can do it at all. In other words, the only way of having good tracking results with unbounded actions is to fall into a bang-bang control-like situation. Therefore, the best found agent will have to start by learning to use only bounded and smooth action values, which will allow it to, then, start decreasing the tracking error.

Notice that, while the tracking error and the noise measures are expected to tend to 0, η's magnitude is not, since it would mean that the agent had given up actuating in the environment. Hence, it is expected that, at some point in training, the latter stabilizes.

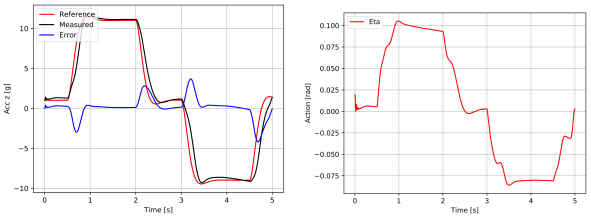
Moreover, the exploration strategy proposed in section 5.1.1 insert some variance in the output of the policy neural network, meaning that the noise measures are also not expected to reach exactly 0.

6.2. Best Found Agent

As figure 1 evidences, the agent is clearly able of controlling the measured acceleration and to track

[3] INCLUIR AUTOCITAÇÃO

[4] To be accurate, this interval covers only 99.73% of the possible values, but it is assumed to be the whole spectrum of possible values.



(a) Tracking performance (b) Action, η

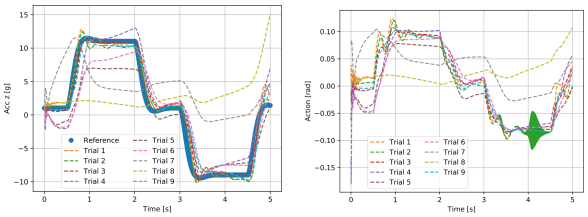
Figure 1: Test performance of the Best Found Agent

the reference signal it is fed with, satisfying all the performance requirements defined in table 1 (cf. table table 2).

Requirement	Achieved Value	
$\|e_z\|_{\max,r}$	0.4214	[g]
Overshoot	8.480	[%]
$\|\eta\|_{\max}$	0.1052	[rad]
$\eta_{noise,r}$	0.04222	[rad]
$\eta_{noise,t}$	0.005513	[rad]

Table 2: Performance achieved by the Best Found Nominal Agent

6.3. Reproducibility Assessments



(a) Tracking performance (b) Action, η

Figure 2: Nominal test performance of the Reproducibility Trials

Figure 2 shows the tests of the best agents obtained during each of the nine trains run to assess the reproducibility of the best found nominal agent (cf. section 6.2) and it is possible to see that none of them achieved the target performance, i.e., none meets all the performance criteria established in table 1. Although most of the trials can be considered far from the initial random policy, it is possible to verify that trials 7 and 8 present a very poor tracking performance and that trial 2's action signal is not smooth.

6.4. Robustness Assessments

6.4.1 Latency

As figure 3 shows, from the four different values of l_{max}, only 5ms converged after 2500 episodes,

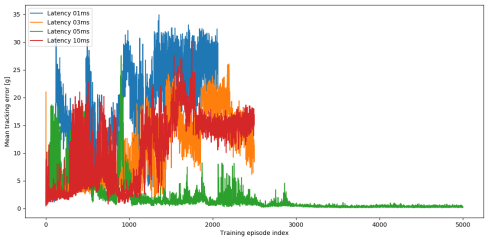
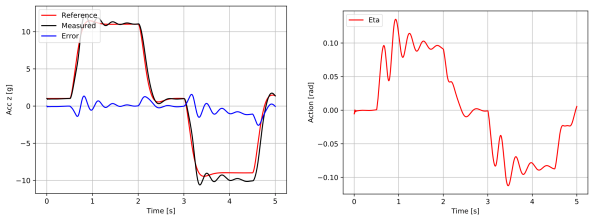


Figure 3: Mean tracking error of latency robustifying trains

meaning that it was the only robustifying train being run for a total of 5000 episodes, during which the best agent found was defined as the Latency Robustified Agent. The nominal performance (cf. figure 4) is damaged, having a less stable action signal and a poorer tracking performance.



(a) Tracking performance (b) Action, η

Figure 4: Nominal Performance of the Latency Robustified Agent

Furthermore, its performance in environments with latency (0ms to 40ms) did not provide any enhancement, as its success rate is low in all quantities of interest (cf. table 3).

Requirement	Success %
$\|e_z\|_{\max,r}$	0.00
Overshoot	25.00
$\eta_{\max}$	0.00
$\eta_{noise,r}$	7.50
$\eta_{noise,t}$	5.00

Table 3: Robustified Agent success rate in improving the robustness to Latency of the Nominal Agent

Since the Latency Robustified Agent was worse than the Nominal Agent in both nominal and non-nominal environments, it lost in both Performance and Robustness categories. Thus, it is possible to say, in general terms, that, concerning latency, the robustifying trains failed.

6.5. Estimation Uncertainty

As figure 5 shows, from the four different values of 3σ tried, only the 3% one converged after 2500

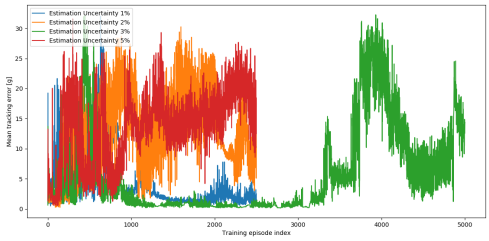
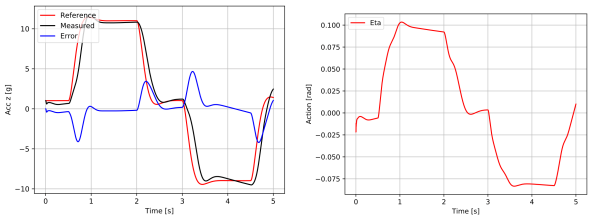


Figure 5: Mean tracking error of estimation uncertainty robustifying trains

episodes, meaning that it was the only robustifying train being run for a total of 5000 episodes, during which the best agent found was defined as the Estimation Uncertainty Robustified Agent. The nominal performance is improved, having a lower overshoot and a smoother action signal.



(a) Tracking performance (b) Action, η

Figure 6: Nominal Performance of the Estimation Uncertainty Robustified Agent

Furthermore, it enhanced the performance of the Nominal Agent in environments with estimation uncertainty (-10% to 10%), having achieved high success rates (cf. table 4).

Requirement	**Success %**
$\lvert e_z \rvert_{\max,r}$	60.38
Overshoot	66.39
$\eta_{\max}$	91.97
$\eta_{noise,r}$	77.51
$\eta_{noise,t}$	82.33

Table 4: Robustified Agent success rate in improving the robustness to Estimation Uncertainty of the Nominal Agent

Since the Estimation Uncertainty Robustified Agent was better than the Nominal Agent in both nominal and non-nominal environments, it won in both Performance and Robustness categories. Thus, it is possible to say, in general terms, that, concerning estimation uncertainty, the robustifying trains succeeded.

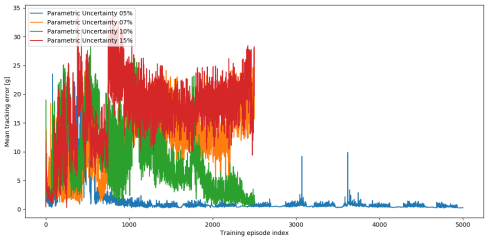
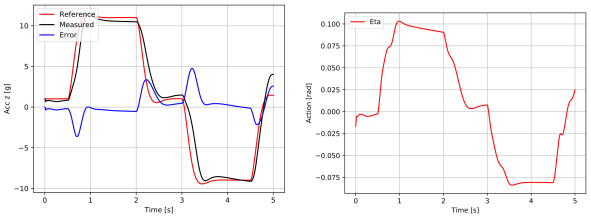


Figure 7: Mean tracking error of parametric uncertainty robustifying trains

6.6. Parametric Uncertainty

As figure 7 shows, from the four different values of 3σ tried, only the 5% one converged after 2500 episodes, meaning that it was the only robustifying train being run for a total of 5000 episodes, during which the best agent found was defined as the Parametric Uncertainty Robustified Agent. The nominal tracking performance is damaged, but the action signal is smoother (cf. figure 8).



(a) Tracking performance (b) Action, η

Figure 8: Nominal Performance of the Parametric Uncertainty Robustified Agent

Furthermore, it enhanced the performance of the Nominal Agent in environments with parametric uncertainty (-40% to 40%), having achieved high success rates (cf. table 5).

Requirement	**Success %**
$\lvert e_z \rvert_{\max,r}$	61.55
Overshoot	83.16
$\eta_{\max}$	98.28
$\eta_{noise,r}$	100.00
$\eta_{noise,t}$	99.31

Table 5: Robustified Agent success rate in improving the robustness to Parametric Uncertainty of the Nominal Agent

Since the Parametric Uncertainty Robustified Agent was better than the Nominal Agent in non-nominal environments, it won in the Robustness categories, remaining acceptably the same in terms of performance in nominal environments. Thus, it

is possible to say, in general terms, that, concerning parametric uncertainty, the robustifying trains succeeded.

7. Achievements

The proposed algorithm has been considered successful, since all the objectives established in section 1 were accomplished, confirming the motivations put forth in section 1. Three main achievements must be highlighted:

1. the nominal target performance (cf. section 6.2) achieved by the proposed algorithm with the non-linear model of the dynamic system (cf. section 3);

2. the ability of SER (cf. section 5.1.4) in boosting a previously suboptimally converged performance;

3. the very sound rates of success in overtaking the performance achieved by the best found nominal agent (cf. sections 6.5 and 6.6). RL has confirmed to be a promising learning framework for real life applications, where the concept of Robustying Trains can bridge the gap between training the agent in the nominal environment and deploying it in reality.

8. Future Work

The first direction of future work is to expand the current task to the control of the whole flight dynamics of the GSAM, instead of solely the longitudinal one. Such an expansion would require both (i) straightforward modifications in the code and in the training methodology and (ii) some conceptual challenges, concerning the expansion of the reward function and of the exploration strategy.

Secondly, it would be interesting to investigate how to tackle the main challenges faced during the design of the proposed algorithm, namely (i) to avoid the time-consuming reward engineering process, (ii) the definition of the exploration strategy and (iii) the reproducibility issue.

References

[1] Bernardo Cortez. Reinforcement Learning for Robust Missile Autopilot Design. Technical report, Instituto Superior Técnico, Lisboa, 12 2020.

[2] Florian Peter. *Nonlinear and Adaptive Missile Autopilot Design*. PhD thesis, Technischen Universitat München, Munich, 5 2018.

[3] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 2 edition, 2018.

[4] John Schulman, Sergey Levine, Philipp Moritz, Michael Jordan, and Pieter Abbeel. Trust region policy optimization. *32nd International Conference on Machine Learning, ICML 2015*, 3:1889–1897, 2015.

[5] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 2016.

[6] Scott Fujimoto, Herke Van Hoof, and David Meger. Addressing Function Approximation Error in Actor-Critic Methods. *35th International Conference on Machine Learning, ICML 2018*, 4:2587–2601, 2018.

[7] Yuhuai Wu, Elman Mansimov, Shun Liao, Roger Grosse, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation. *Advances in Neural Information Processing Systems*, 2017-Decem(Nips):5280–5289, 2017.

[8] John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, pages 1–14, 2016.

[9] Dmitry Kangin and Nicolas Pugeault. On-Policy Trust Region Policy Optimisation with Replay Buffers. pages 1–13, 2019.

[10] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. *34th International Conference on Machine Learning, ICML 2017*, 3:2171–2186, 2017.

[11] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. pages 1–12, 2017.

[12] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *35th International Conference on Machine Learning, ICML 2018*, 5:2976–2989, 2018.

[13] Ofir Nachum, Mohammad Norouzi, George Tucker, and Dale Schuurmans. Smoothed action value functions for learning Gaussian

policies. *35th International Conference on Machine Learning, ICML 2018*, 8:5941–5953, 2018.

[14] Shixiang Gu, Timothy Lillicrap, Zoubin Ghahramani, Richard E. Turner, Bernhard Schölkopf, and Sergey Levine. Interpolated policy gradient: Merging on-policy and off-policy gradient estimation for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 2017-Decem(Nips):3847–3856, 2017.

[15] Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. TruST-PCL: An off-policy trust region method for continuous control. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, pages 1–14, 2018.

[16] Shixiang Gu, Timothy Lillicrap, Zoubin Ghahramani, Richard E. Turner, and Sergey Levine. Q-PrOP: Sample-efficient policy gradient with an off-policy critic. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, pages 1–13, 2019.

[17] Brendan O'Donoghue, Rémi Munos, Koray Kavukcuoglu, and Volodymyr Mnih. Combining policy gradient and q-learning. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, pages 1–15, 2019.

[18] James Martens and Roger Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. *32nd International Conference on Machine Learning, ICML 2015*, 3:2398–2407, 2015.

[19] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight Experience Replay. In *Advances in Neural Information Processing Systems*, 2017.

[20] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, pages 1–21, 2016.

[21] Karthik Narasimhan, Tejas D Kulkarni, and Regina Barzilay. Language Understanding for Text-based Games using Deep Reinforcement Learning. Technical report, Massachusetts Institute of Technology, 2015.