



POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE



DLR

Institute of Space Systems

EXECUTIVE SUMMARY OF THE THESIS

Machine Learning-based Reentry Guidance for the ReFEx Mission: Analysis and Comparison of Reinforcement Learning and Genetic Programming

LAUREA MAGISTRALE IN SPACE ENGINEERING - INGEGNERIA SPAZIALE

Author: RICCARDO CADAMURO

Advisor: PROF. FRANCESCO TOPPUTO

Co-advisor: DR. FRANCESCO MARCHETTI

Academic year: 2024-2025

1. Introduction and Objectives

The reusability of launch vehicles is attracting increasing interest within the space industry, primarily due to its potential to significantly reduce mission costs. However, guiding a vehicle safely through atmospheric reentry from space remains a highly complex Guidance and Control (G&C) challenge, largely due to the uncertain and rapidly changing environmental conditions. As a result, the development of efficient and reliable G&C algorithms has become essential. Machine Learning (ML) techniques are gaining considerable attention, as they enable the offline training of models in simulated environments, which once deployed in real systems, can generate guidance commands in real-time with minimal computational overhead.

The objective of this thesis is to apply Reinforcement Learning (RL) and Genetic Programming (GP) to the design of the reentry guidance law of the REusability Flight Experiment (ReFEx) mission: a winged Reusable Launch Vehicle (RLV) designed according to the Vertical Take-off Horizontal Landing (VTHL) paradigm to follow a realistic trajectory for a general winged RLV. The ReFEx baseline guidance algorithm

computes the initial reference commands for angle of attack (α) and bank angle (μ) onboard prior to the start of the reentry phase [3]. Once reentry begins (i.e. a given dynamic pressure threshold is met), these reference commands are iteratively updated in real time by applying corrections $\delta\alpha$ and $\delta\mu$ until the End of Experiment (EoE) is reached [4]. This work aims to: 1) use RL and GP to train a model capable of generating online the guidance command update during the atmospheric reentry phase; 2) assess the performances of the trained ML models using the high-fidelity 6-DOF simulator, internally developed by DLR, used to certify the state-of-the-art algorithm designed for the real mission; and 3) produce a comparative study between RL and GP.

2. Reinforcement Learning

RL is a technique used to train an agent that learns to compute optimal actions through interaction with an environment, aiming to maximize a numerical reward signal. The problem is framed as a Markov Decision Process (MDP), where the decision maker (agent) observes the system's state as it evolves over time and selects

actions from a feasible set at each time step [6]. The function mapping states into feasible actions is called policy and it is learnt during the training process with the objective of maximizing the so-called cumulative discounted reward in Equation 1.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots = \sum_{i=0}^T \gamma^i R_{t+i+1} \quad (1)$$

The discount rate $\gamma \in [0, 1]$ controls the relative importance of future rewards ($\gamma \approx 1$) versus immediate ones ($\gamma \approx 0$). The training is performed over episodic tasks, where each episode corresponds to a full trajectory from Entry Interface (EI) to EoE. Once the episode is complete, the initial conditions are reset and a new one is begun.

This work uses the Proximal Policy Optimization (PPO) algorithm to train a Multilayer Perceptron (MLP). The PPO algorithm [5] is chosen because it is designed to deal with a continuous action space and to perform small and controlled policy update steps which strongly stabilize the training performances. It can learn the optimal behavior by interacting with the environment without having knowledge of its model (model free method) and it simultaneously trains two networks: a first one to compute the action and a second one to evaluate the action produced by the first (Actor-Critic method). On the other hand, the MLP has been chosen with respect to other more advanced Neural Network (NN) architectures because it computes the outputs only as function of the inputs without any internal loop or memory components, which is analogous to what is done by the GP, therefore ensuring a fair comparison. Since RL is used to train a NN, the technique is more properly defined as Deep Reinforcement Learning (DRL).

3. Genetic Programming

Genetic Programming (GP) is an Evolutionary Algorithm (EA) capable of autonomously evolving computer programs to achieve a user-defined objective. The high level goal is formulated in terms of a fitness function which is used to evaluate the performances of a program in solving a given problem. GP differs from other EAs because it represents individuals with a syntactic tree-based structure (Figure 1) in which internal nodes are labeled by elementary functions while

leaves contain constants, that can be evolved as parts of the program, or the actual function input variables.

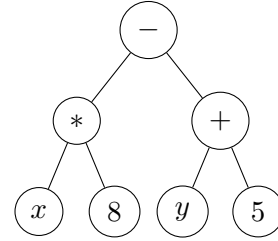


Figure 1: Depiction of a GP individual.

The GP evolutionary process starts with a randomly initialized population of individuals, who are first evaluated, then recombined and selected at each generation of the evolutionary process. The recombination is performed through the crossover and mutation operators, which consist respectively in swapping two subtrees from two parents individuals to generate two offspring individuals; and in randomly mutating a parent individual to generate an offspring one. The selection operator is applied to select the individuals to pass to the next generation. This process is repeated until the stopping criteria is met.

The specific GP formulation used in this work is the Inclusive Genetic Programming (IGP) [1], which has been designed to promote and preserve the genotypic diversity during the evolution allowing it to efficiently handle complex models. The IGP is well suited for G&C applications because it can account for constraints and produce multiple mathematical models simultaneously. The latter feature is of interest in the treated problem due to the guidance command being composed by the α and μ signals which can therefore be represented by two separated IGP trees. Moreover, the IGP has already been proved successful in a simplified version of the current problem [2].

4. Training Framework

RL and GP have been configured to ensure a fair comparison. In particular, they both interact with the same environment and they produce a model with the same Input/Output interface. On the other hand, the RL reward function and the GP fitness function are formulated according to the specifics of the two techniques.

4.1. Simulation Environment

The training simulator is a 3-DOF model (considering only translational dynamics), based on the version internally developed by DLR for the design of ReFEx. It features a spherical, rotating Earth model and uses a state vector defined in the geocentric reference frame, consisting of altitude (h), longitude (λ), latitude (ϕ), velocity (v), flight path angle (γ) and heading angle (χ). Flight constraints enforced during the mission are grouped into two categories: 1) constraints to ensure the physical integrity of the vehicle and 2) constraints to ensure its controllability. The vehicle is designed to satisfy the first set of constraints, namely limits on dynamic pressure, heat flux and load factor throughout the entire trajectory envelope, and thus, these are not actively enforced during trajectory updates. Controllability, on the other hand, is ensured by limiting the rate and acceleration of guidance commands ($\dot{\alpha}$, $\ddot{\alpha}$, $\dot{\mu}$, $\ddot{\mu}$) and by constraining the trajectory within a predefined flight corridor in the Mach- α domain to avoid regions where the vehicle becomes aerodynamically underactuated. Both the vehicle and environment models are formulated to incorporate their respective uncertainties during propagation, which are sampled from uniform or normal distributions depending on their nature. Environmental uncertainties include wind and atmospheric conditions, while vehicle-related uncertainties cover mass and aerodynamic coefficients. Initial conditions are defined based on the separation from the launcher scenario. A complete set of values of initial conditions and uncertainty parameters is called Uncertainty Case (UC). Additionally, navigation errors are modeled as autoregressive random processes, following the navigation performance model developed and validated for the ReFEx mission.

The training dataset consists of a set of UCs. As dataset size directly impacts training time, a sampling algorithm has been developed to select a representative subset from a larger pool, ensuring good coverage of the actual scenario. In the GP case, each individual needs to propagate all UCs in the training dataset, which remains fixed across individuals and generations. To maintain acceptable computational times, the GP training dataset includes 25 UCs. Conversely, the RL proceeds by randomly selecting,

at each training iteration, a subset of the training database and fully propagating each selected UC. Multiple episodes are evaluated per training epoch and the selected UCs may vary between iterations. Three different datasets have been tested for RL training: the same 25-cases set used for GP, an intermediate set with 60 cases, and a large set containing 500 cases. It is worth noting that, for RL, larger datasets provide a more accurate representation of real-world conditions during training, but they require more iterations to achieve proper generalization, since each UC will be seen less times by the agent.

4.2. Reinforcement Learning reward signal

The reward signal is composed of two terms: a dense reward, provided at each timestep to guide the agent toward the target, and a terminal reward, given when the vehicle reaches the EoE, to quantify the accuracy of the final state relative to the target.

The dense reward term aims to satisfy three drivers:

1. Keeping the agent in proximity of the reference trajectory computed during the planning phase. The proximity condition is modeled only with respect to the positional coordinates and the corresponding reward terms are $R_{trj,h}$, $R_{trj,\lambda}$ and $R_{trj,\phi}$.
2. Avoiding to fly in regions in which aerodynamic data are not reliable: aerodynamic data have been computed via CFD analysis only for flight conditions within a given corridor in the Mach- α domain. The term h_{corr} reduces the numeric value of the R_{trj} when the vehicle leaves the corridor, while a second term, R_{corr} , acts as a linear penalty to the final reward value.
3. Promoting a smooth guidance function: this is needed to stop the agent from producing a noisy guidance command which could potentially be very effective in the training environment, but that could produce unexpected and dangerous effects in the more realistic high-fidelity testing environment. This constraint is enforced through two penalty terms: a smoothness penalty (R_{smooth}), which is a negative term proportional to the time derivative of the guidance commands, and a saturation

penalty ($R_{saturation}$), which applies a fixed negative reward when a command reaches saturation.

Finally, the terminal reward ($R_{terminal}$ in Equation 2) reflects how close the vehicle is to the target at the EoE. It is designed to be positive and unbounded when the terminal accuracy is met and to decrease quadratically when the agent fails in reaching the target within the required tolerances.

$$R = h_{corr} (R_{trj,h} + R_{trj,\lambda} + R_{trj,\phi}) + R_{corr} + R_{smooth} + R_{saturation} + R_{terminal} \quad (2)$$

4.3. Genetic Programming fitness function

The GP fitness function is designed following the same drivers used in the RL case, with the main difference that it was not needed to enforce a smoothness constraint since the signal produced by the GP individuals is intrinsically smooth. It therefore accounts for 1) keeping the vehicle in proximity of the reference trajectory, particularly in the very last flight phase 2) avoid flying in regions in which aerodynamic data are not reliable and 3) terminating as close as possible to the target state. Since each GP individual is represented by the $\delta\alpha$ and $\delta\mu$ guidance laws, the fitness function needs to be specified for the performances of the individual across all the 25 uncertainty cases in the training dataset. For this reason, the GP fitness function has the following form: $F_{ind} = F_0 + g$, in which the term F_0 is proportional only to the success rate (S_R) of the individual (i.e. the number of runs across the training dataset that manage to reach the target within the prescribed tolerances), whereas the g term depends on the specific performances of each of the n_{tot} runs in the training dataset. The function is designed such that $S_{R,k} > S_{R,j} \rightarrow F_k < F_j$ (notice that, in the GP case, the fitness function shall be minimized).

5. Results

Each trained model has been tested on a 100 runs Monte Carlo (MC) campaign in 6DOF high-fidelity simulator. The best performing models have been also tested on a 500 run MC

campaign, obtained by adding 400 runs to the initial 100. The size of the MC campaign has been limited by the extensive cost of performing each high-fidelity simulation. The baseline guidance algorithm has been tested on the same scenario as well, with the results in Figure 2 which shows 500 trajectories. The outcome of each simulation can be classified as success if it reaches the EoE within the proper tolerance for each of the state vector components, failure if this condition is not met and instability if the vehicle becomes unstable during the flight. Unstable cases are worse than failed ones since in the first situation the vehicle is completely lost mid-flight while in the latter case it can still reach the EoE. In Table 1, Table 2 and Table 3, S_R , F_R and I_R refer respectively to success, failures and instability rates.

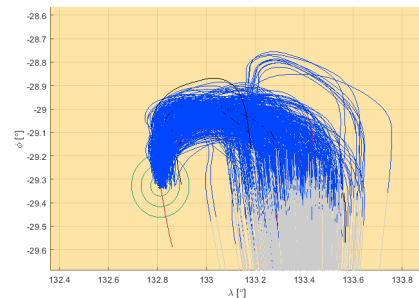


Figure 2: Baseline Trajectories, 500 runs, 6-DOF

Due to the GP evolution being driven by random processes, its outcome is stochastic, therefore to assess the actual performances of the algorithm, 15 full evolutions with the same setup have been performed. Table 1 contains the results of such campaign when the best individual of each evolution is evaluated in the high-fidelity simulator. Even if the median performances are below the

Experiment	S_R (%)	F_R (%)	I_R (%)
Baseline	97	2	1
EVOL 10 (worst)	84	13	3
GP - median	95	3	2
EVOL 13 (best S_R)	97	2	1
EVOL 4 (best I_R)	96	4	0

Table 1: GP results, 100 runs campaign, 6-DOF

baseline ones, the GP manages to obtain results comparable with the state-of-the-art algorithm proving its effectiveness.

Differently from the GP, the RL training does not rely on stochastic processes but it iteratively improves the same Neural Network (NN), therefore even if there exists some stochastic components (for instance, the initialization of the NN weights and biases), they never have a first order influence on the training direction. For this reason, stochastic effects have been considered negligible and not analyzed. Several RL configurations have been tested and the most relevant are reported in Table 2, where the UC suffix refers to the number of uncertainty cases in the training dataset and the Smooth or STD label refers to whether the reward signal contains the term promoting the policy’s smoothness (Smooth) or not (STD). The results in Table 2 prove the

Experiment	S_R (%)	F_R (%)	I_R (%)
Baseline	97	2	1
60UC STD	91	1	8
25UC Smooth	97	0	3
60UC Smooth	98	1	1
500UC Smooth	98	1	1

Table 2: RL results, 100 runs campaign, 6-DOF

importance of having a smooth guidance signal since even if in the 3-DOF training environment the **60UC STD** configuration could achieve a $S_R = 99.6\%$ whereas in the **60UC Smooth** case the S_R was 99% (tested on a 500 run MC campaign), when using the high-fidelity 6-DOF environment, the number of unstable runs in the first case is much higher than in the latter. The RL also can deliver performances equivalent to the baseline guidance and, when using the same dataset as GP (**25UC Smooth** configuration), it outperforms the GP median result but it is slightly worse than the GP best case.

5.1. Comparison

The best configurations have been tested in the 500 runs MC campaign with the results reported in Table 3 showing that the baseline algorithm and the RL best configuration achieve the same performances. The GP solutions have a lower accuracy than the baseline algorithm, even if capable of delivering the same instability rate. The trajectories corresponding to the best RL and GP experiment are shown respectively in Figure 3 and Figure 4. Figure 5 reports the probability density function associated to the absolute

Experiment	S_R (%)	F_R (%)	I_R (%)
Baseline	97.8	1	1.2
RL- 60UC Smooth	97.4	1.2	1.4
GP - EVOL 4	96.6	2.2	1.2
GP - EVOL 13	96.2	2.6	1.2

Table 3: Guidance Performances comparison, 500 runs, 6-DOF

errors in each state at the EoE, confirming the RL being closer to the baseline than the GP.

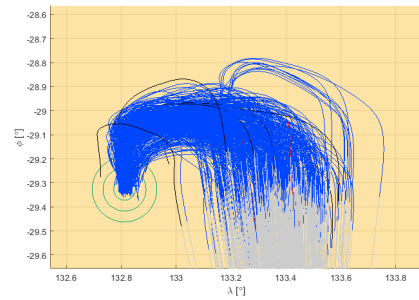


Figure 3: RL: 60UC Smooth, 500 runs, 6-DOF

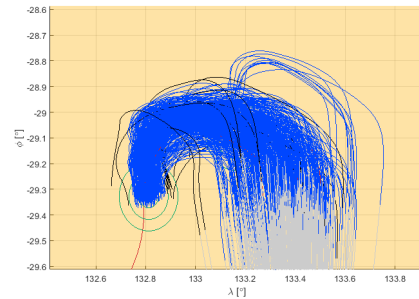


Figure 4: GP: RUN 13, 500 runs, 6-DOF

It is noteworthy that the baseline algorithm requires approximately 10 ms to generate the on-line guidance command, while the ML counterparts take only around 0.01 ms, demonstrating their low computational cost at runtime. However, training the models is significantly more demanding, requiring approximately 48 h for the RL implementation and 57 h for the GP-based one.

The generalization capability from the 3-DOF training environment to the 6-DOF testing environment was evaluated by computing the Jensen–Shannon (JS) divergence between the probability density functions of the terminal error norm at the EoE, obtained from MC simulations in both environments. The results indi-

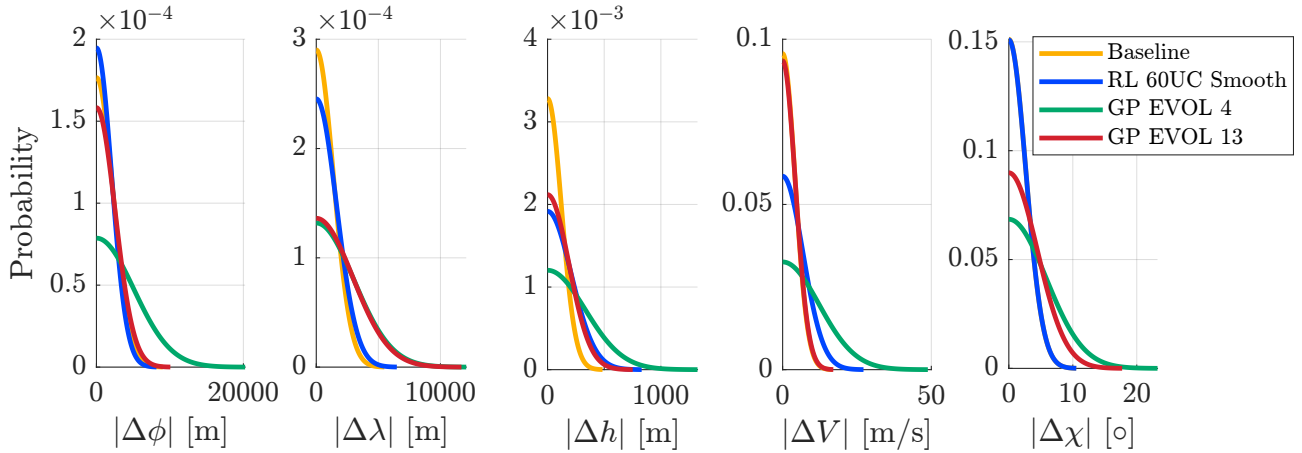


Figure 5: State errors distribution at EoE comparison, 500 runs, 6-DOF

cate that the RL model generalizes more effectively $median(\sqrt{JS})_{RL} = 0.0721$ compared to the GP: $median(\sqrt{JS})_{GP} = 0.1365$.

On the other hand, the GP allows to generate a relatively simple white box symbolic model, whose size and complexity can be limited at training level, whereas the RL returns a NN which is a completely black box function.

6. Conclusions and Future developments

Results show that both the DRL and the IGP algorithm are effective in solving the ReFEx online reentry guidance problem in the high-fidelity 6DOF simulator and providing performances comparable to the state-of-the-art ones. The DRL outperforms the IGP both in accuracy to the target and generalization capabilities, meaning that the 3-DOF performances are more similar to the 6-DOF ones than those of the IGP, due to its training methodology that allows to exploit larger training datasets keeping the computational time contained. Moreover, the MDP learning technique is very effective in this scenario. On the other hand, the GP can produce white box intrinsically smooth guidance laws that are well suited for critical control applications and that can still deliver performances comparable to the baseline guidance.

For this reason, among the possible future developments, the most interesting one consists in developing a hybrid ML technique that replaces the NN with a symbolic function (as done by the GP) in the RL training framework, aiming

to merge the advantages of both.

References

- [1] Francesco Marchetti and Edmondo Minisci. Inclusive genetic programming. In *Genetic Programming: 24th European Conference*, pages 51–65. Springer, 2021.
- [2] Francesco Marchetti, Jose Luis Redondo Gutierrez, and David Seelbinder. Genetic programming guidance for the reentry trajectory of the ReFEx vehicle. In *IAF Space Transportation Solutions and Innovations Symposium*, pages 362–375, Paris, France, 2024. International Astronautical Federation (IAF).
- [3] Jose Luis R Redondo Gutierrez, David Seelbinder, and Stephan Theil. Design of reflex guidance: Trajectory correction after ascent. In *AIAA SCITECH 2023 Forum*, page 1995, 2023.
- [4] Jose Luis R Redondo Gutierrez, David Seelbinder, and Stephan Theil. Design of reflex guidance: Periodic on-board trajectory updates. In *AIAA Scitech 2024 Forum*, page 0087, 2024.
- [5] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.
- [6] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.