ESA GNC & ICATT Conference 2026
28 Sep. – 2 Oct. 2026, Sevilla, Spain
https://atpi.eventsair.com/gncicatt/

Genetic programming-based online landing guidance for a Vertical Takeoff Vertical Landing launch vehicle

Francesco Marchetti[1,*], **Jose Luis Redondo Gutierrez**[1], **and Marco Sagliano**[2]

[1]*German Aerospace Center (DLR), Institute of Space Systems, 28359, Bremen, Germany*
[2]*Università di Bologna, Department of Industrial Engineering (DIN), 47121, Forlì, Italy*
[*]*Corresponding author: francesco.marchetti@dlr.de*

ABSTRACT

This work investigates the application of Genetic Programming (GP) to the landing guidance problem of a Vertical Takeoff Vertical Landing (VTVL) rocket. Although several studies have explored the use of Machine Learning (ML) algorithms for GNC applications, most have focused on Reinforcement Learning or Neural Network-based algorithms, while little attention has been given to GP. GP offers significant potential for GNC applications, since, in contrast to other ML-based approaches, it produces white-box models that are human-readable. Moreover, unlike optimization-based approaches, GP generates a guidance model that can be used online at the onboard computer frequency with a negligible computational cost. The GP algorithm is trained and validated in a 3DoF simulation environment that includes attitude kinematics, atmospheric and aerodynamic perturbations. A Monte-Carlo campaign is conducted to assess robustness and overall performance, with a LQR controller used as benchmark. The results demonstrate the viability of the proposed method, which achieves performance comparable to to that of a traditional combined scheme based on the tracking of a reference optimal solution through an LQR control architecture.

1 Introduction

Vertical Take-Off Vertical Landing (VTVL) rockets are becoming an increasingly widespread topic in the academic literature, thanks to the success of companies such as Space X and Blue Origin. This also translates into growing efforts from industry and space agencies to catch up on their American competitors. Examples of such developments are the CALLISTO (Cooperative Action Leading to Launcher Innovation in Stage Tossback Operations) rocket [18], developed by the German Aerospace Centre (DLR), the French National Center for Space Studies (CNES) and the Japan Aerospace Exploration Agency (JAXA), the Zhuque-3 rocket developed by the Chinese startup Landscape [20] and the Maia rocket developed by the startup MaiaSpace, a spinoff of Ariane Group [21]. Very recently the China Aerospace Science and Technology Corporation (CASC) managed to successfully recover the first stage of the Long March 10B [19], making China the second country to recover an orbital booster. The Guidance, Navigation, and Control (GNC) subsystem is a key component to enable the safe operation of these vehicles. The GNC subsystem must ensure a correct estimate of the full state of the system, be capable of successfully determining a feasible solution for the vehicle towards the designated landing point, and ensure that the vehicle will follow such a trajectory in the presence of disturbances and modeling errors. Shrinking the focus on the guidance, recent studies mostly put their emphasis on the development of convex-optimization techniques for the landing-guidance generation during flight. Examples of this approach can be found in the study from Botelho et al. [5], who applied online convex-optimization and successive convexification to generate the online

© *Francesco Marchetti, Jose Luis Redondo Gutierrez and Marco Sagliano*

landing guidance for the booster recovery phase of the RETALT (Retro Propulsion Assisted Landing Technologies) vehicle; or in the work of Kamath et al. [6] where they introduced a sequential conic optimization algorithm to solve nonconvex optimization problem, specifically applied to the landing guidance problem of a multi-phase rocket with nonlinear dynamics and convex constraints. Sagliano et al. proposed a convex-based approach for atmospheric landing [4], [12], [15], as well as 6-DoF solutions for the entry phase [11], the landing phase [7], and, more recently, the combined entry-to-descent profile [14]. Apart from optimization-based approaches, also Machine Learning (ML) techniques are being considered for these applications, although their development remains limited to the academic domain. In contrast to optimization-based approaches, these techniques can handle nonlinear systems, thus avoiding the need of linearization or convexification techniques that might lead to modeling errors. Moreover, they are potentially easier to transfer between different problems, since the same algorithms can be applied with just some minor fine tuning, in contrast to optimization approaches, where the underlying optimal control problem must be formulated for each different scenario. Additionally, the ML model produced after training is extremely fast to execute compared to real-time optimization algorithms, making it more suitable for online execution. Examples of such approaches are in the work of Wang et al. [8] where they employed a classification network to account for the dispersion in the initial condition and a regression network to generate the guidance commands according to the output of the classification network; the work of Çelik et al. [9] where they developed a parametric guidance algorithm for the landing phase of a reusable rocket, using scientific ML and two types of enhanced Neural Network (NN); and the approach employed by Carradori et al. [17], where a Reinforcement Learning (RL) algorithm enhanced with the transformer architecture was used to solve the 6-DoF atmospheric landing guidance problem. In this work, Genetic Programming (GP) is used to autonomously generate a landing guidance feedback law for a VTVL rocket. GP is a ML technique pertaining to the Evolutionary Algorithms (EAs) family, whose distinguishing feature is the tree-shaped representation used to encode the individuals. Such tree-shaped encoding represents a mathematical model that can be applied to solve the defined task. Thanks to this feature, GP possesses the potential for interpretability, which makes it particularly suitable for Guidance and Control (G&C) applications, where attention is posed on the reliability of the employed algorithm. When ML techniques are employed, such reliability is evaluated through the interpretability and explainability of the employed algorithms and the obtained model. For this reason, GP possess an advantage in comparison to NN-based approaches, since the latter constitute black-box models, with a low or no degree of interpretability and explainability. The literature contains examples of the successful application of GP to the guidance generation task. The most recent example of which is represented by the generation of the guidance law for the Reusability Flight Experiment (ReFEx) reentry vehicle [16]. In this work, GP was used to produce the guidance commands (angle of attack and bank angle) to steer the vehicle towards the desired target during reentry. The produced models were then tested on the same high-fidelity 6DoF simulator used to develop the nominal optimization-based guidance approach. The obtained results show that GP can achieve performance comparable with those of the nominal approach and with a RL algorithm, motivating further research on the topic.

The remainder of this work is structured as follows: Section 2 provides an high-level description of the GP algorithm, while Section 3 provides a description of the tackled problem. Section 4 describes the guidance algorithm designed for this work, Section 5 introduces a benchmark for performance comparison and Section 6 details the obtained results. This manuscript is concluded by Section 7 which provides the drawn conclusions and future work directions.

2 Genetic Programming

GP is an EA introduced by Koza in 1992 [1]. By mimicking the principles of Darwinian evolution, it evolves computer programs capable of achieving a goal defined by the user. The goal is encoded inside a fitness function that is maximized by the algorithm. In contrast to other EAs, GP encodes the individuals as syntax tree, allowing standard mathematical operations, recursion, iteration, and conditional statements. For the purpose of this work, only standard mathematical functions are considered. Figure 1 depicts a GP individual, where the *sin*, *mul* and *add* functions are used, with a constant number of value 15 and two input variables x and y. Such an individual can be read as $sin(x) * (15 + y)$. The GP variant selected for this study is the Inclusive Genetic Programming (IGP) algorithm [3]. IGP was developed to work efficiently in nonlinear G&C problems, by employing niching strategies to maintain a high population diversity throughout the evolutionary process. Moreover, IGP can handle constraints and can evolve multiple GP models simultaneously, which is particularly important for G&C problems with

multiple control/guidance signals. The IGP evolutionary process is summarized in Algorithm 1. It is structured as a standard $\mu + \lambda$ Evolutionary Strategy (ES), but with the use of the Inclusive Crossover, Inclusive Mutation and Inclusive Tournament. For more information about these mechanisms and the IGP, please refer to [3].

Algorithm 1 IGP evolutionary process [16]

1: Randomly generate initial population P
2: Evaluate fitness of individuals in the initial population
3: Generation $\leftarrow 1$
4: **while** Termination criteria is not met **do**
5: Perform Inclusive Crossover and Inclusive Mutation to obtain λ offspring
6: Evaluate fitness of offspring
7: Select μ individuals with Inclusive Tournament selection, from a total population composed by μ parents and λ offspring
8: $P \leftarrow \mu$ selected individuals
9: Generation $\leftarrow$ Generation $+ 1$
10: **end while**

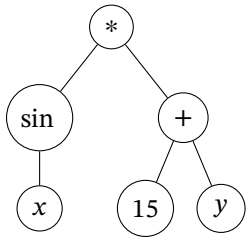

Figure 1 *Depiction of the GP tree.*

3 Problem Description

This paper aims to develop the powered landing guidance of a VTVL rocket. The problem is defined by the combination of the simulation environment (including vehicle and environment models as well as uncertainties) and the mission definition (including initial state and success criteria). Therefore, the problem tackled is to develop a guidance algorithm that can achieve the mission goals in the context of the simulation environment.

The simulation environment and the mission definition are inspired on [17], with some changes. They are further explained throughout this section.

3.1 Simulation Environment

This simulation environment was developed to train and test the GP algorithm. The vehicle is modeled, on a high level, after the CALLISTO rocket, as depicted in Figure 2. This model is further explained later in this section.

The environment contains models of atmosphere and gravity, which are equivalent to those defined in [17]. Therefore, these models are not further explained. The uncertainties considered are a subset of those considered in [17]. They are described later in this section, along with their effect in the mission goals in the absence of corrective action.

During simulation, the vehicle state is propagated using a Runge-Kutta 4 integration scheme with a $dt = 0.05$ s. The propagation is terminated when the vertical position either becomes negative or when starts increasing.



Figure 2 *CALLISTO rocket [15]*

Vehicle Model

The vehicle is modeled as a rigid body with variable mass. The concept for aerodynamic surfaces of the CALLISTO project is used, and in this work, they are modeled as always open at 0 deg deflection angle, as depicted in Figure 2. This is done to have a passive stabilization effect. The landing system is not modeled.

The equations of motion are defined in the Up-East-North (UEN) reference frame, which is a rotating frame centered in the landing pad center. The notation employed in the rest of this manuscript has the

following meaning: $\square^{UEN}$ is a quantity of the body frame B w.r.t. the UEN frame, expressed in the UEN frame (used for translational quantities), $\square_B^{UEN}$ is a quantity of the body frame B w.r.t. the UEN frame, expressed in the body frame B (used for rotational quantities). The aerodynamic data are taken from [17], as well as the rocket variable MCI model.

The modelled dynamics and kinematics of the vehicle are expressed as in Equation 1, where ω_E is the Earth's angular velocity, and $\mathbf{F}^{UEN}$ are the forces acting on the vehicle, hence $\mathbf{F}^{UEN} = \mathbf{F}^{UEN}_{gravity} + \mathbf{F}^{UEN}_{aero} + \mathbf{F}^{UEN}_{thrust}$. Rotational dynamics are not considered.

$$\begin{cases} \dot{\mathbf{r}}^{UEN} = \mathbf{v}^{UEN} \\ \dot{\mathbf{v}}^{UEN} = \mathbf{F}^{UEN}/m - 2\boldsymbol{\omega}_E \times \mathbf{v}^{UEN} - \boldsymbol{\omega}_E \times (\boldsymbol{\omega}_E \times \mathbf{r}^{UEN}) \\ \dot{\mathbf{q}}_B^{UEN}, \dot{\boldsymbol{\omega}}_B^{UEN} = f(\mathbf{q}_B^{UEN}, \boldsymbol{\omega}_B^{UEN}, \mathbf{q}_{B,command}^{UEN}) \\ \dot{m} = -\frac{T_{command}}{I_{sp} g_0} \end{cases} \tag{1}$$

The state vector is therefore given by $\mathbf{x} = [r_x, r_y, r_z, v_x, v_y, v_z, q_x, q_y, q_z, q_w, \omega_x, \omega_y, \omega_z, m]$ and the control vector is given by $\mathbf{u} = [T, \phi, \theta, \psi]$, where the Euler angles (ϕ, θ and ψ) are converted into the commanded quaternions $\mathbf{q}_{B,command}^{UEN}$.

The selection of the control vector, which commands an attitude, imposes the need of additional actions to guarantee that the limits on angular rates and accelerations are respected. The selection of the attitude as part of the control vector is linked to an observed benefit in GP convergence during training. In order to enforce this constraint, the angular acceleration is computed using a bounded law (i.e. with maximum angular rates and angular accelerations), driving the difference between commanded and estimated attitude to zero. This way, an angular acceleration is generated along the error axis using a bang-bang switching strategy. The remaining angular distance is compared with the braking distance required to stop the current angular velocity. If smaller than the remaining angular distance, maximum deceleration is generated; otherwise maximum acceleration toward the target. When the angular-rate limit is reached, acceleration along the slew axis is set to zero. The attitude kinematics are propagated using the same integration method as for the rest of the state variables. This strategy used to propagate the angular motion is aligned to the one used for previous works [13].

Uncertainties

Perturbations are applied on the atmospheric and aerodynamic model as described in [17]. In summary, the atmospheric density is uniformly sampled in a $\pm 10\%$ range from its nominal value, and the aerodynamic coefficients are uniformly sampled in a $\pm 15\%$ range from their nominal values.

Figures 3 and 4 show the effects of the applied perturbations. The shown trajectories and terminal states are obtained by flying 500 trajectories (indexes 501 to 1000) using the nominal guidance commands in Figure 5 and applying the perturbations described above. The computation of the nominal guidance is described in Section 4.1. As can be observed, the applied perturbations affect mostly the horizontal position and the vertical velocity, while the horizontal velocity and the attitude are not heavily influenced.

3.2 Mission

The chosen mission profile is the powered landing descent, starting from a altitude of 2km. The initial conditions are summarized in Table 1a. The final conditions are $\mathbf{r}_f = [0, 0, 0]$ m, $\mathbf{v}_f = [0, 0, 0]$ m/s, $\mathbf{q}_f = [0, 0, 0, 1]$ ($\phi = \theta = \psi = 0$ deg), $\boldsymbol{\omega}_f = [0, 0, 0]$ deg/s. Table 1b lists the success criteria used in this work. All the listed quantities refer to the final point of the trajectory, denoted by the subscript $\square_f$. In particular, $r_{x,f}$ is the vertical position, $r_{hor,f}$ is the 2-norm of the position in the y and z directions, v_x is the vertical velocity, $v_{hor,f}$ is the 2-norm of the velocity in the y and z directions, Ω is the 2-norm of the angular velocity in all three directions and Φ_f is the angle between the vehicle longitudinal axis and the vertical, computed as $\Phi_f = \arccos(1 - 2(q_{y,f}^2 + q_{z,f}^2))$. $v_{x,f}$ is allowed to be non zero, since, by design, the guidance stops very close to ground (below 2m), and it is assumed that the engine can be turned off when reaching the final position.

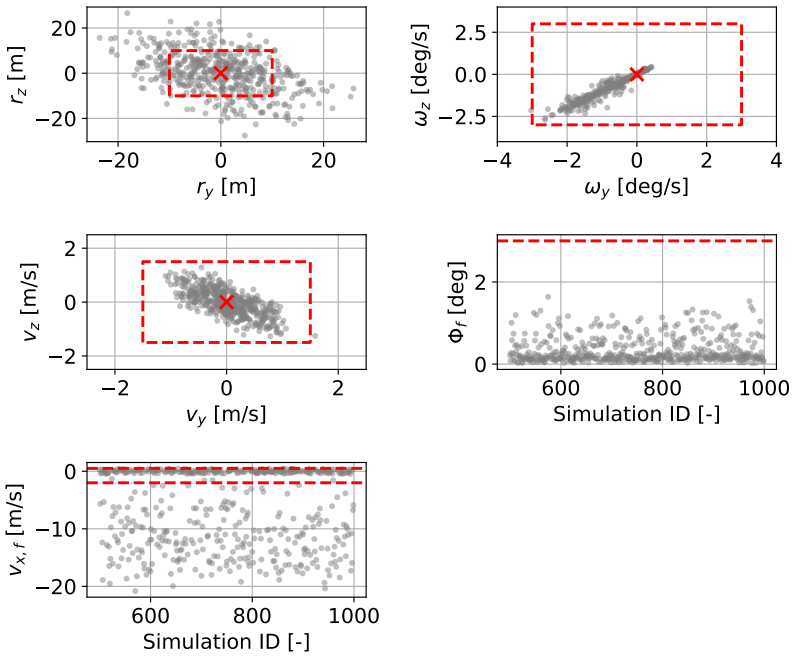

Figure 3 *Perturbed terminal states with open-loop reference guidance commands. Each gray dot represents a MC simulation. The red dashed line represents the tolerance for each quantity.*

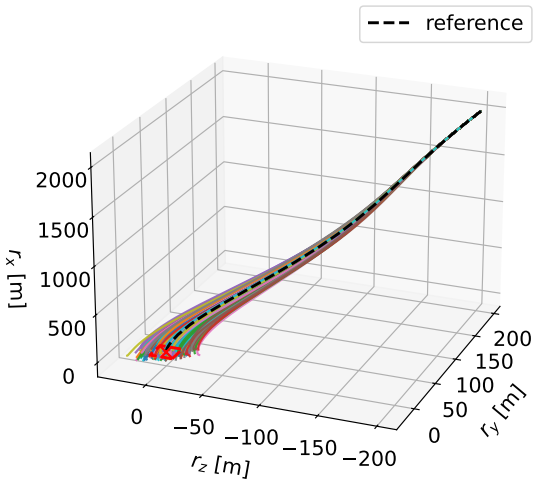

Figure 4 *Perturbed trajectory with reference open-loop guidance commands*

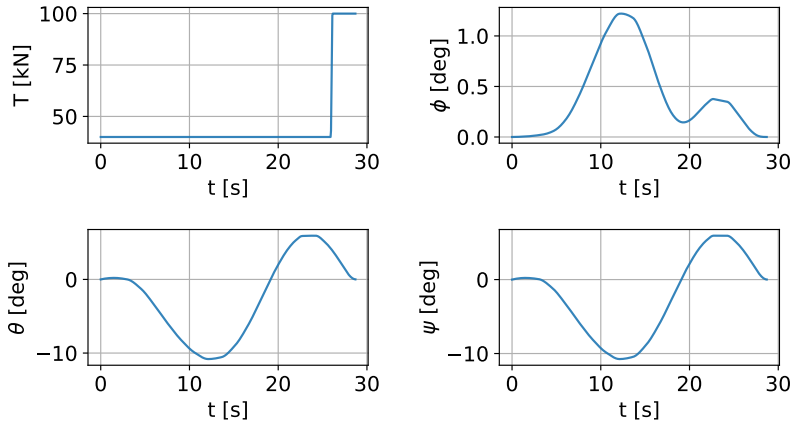

Figure 5 *Optimal open-loop controls*

(a) *Initial Conditions*

Parameter	Value
r	[2000, 200, -200] m
v	[-100, -10, 10] m/s
q	[0, 0, 0, 1] (scalar last)
$\boldsymbol{\omega}$	[0, 0, 0] deg/s
m	4000 kg

(b) *Success criteria*

Parameter	Condition	Value
$r_{x,f}$	$\leq$	3 m
$r_{hor,f}$	$\leq$	10 m
$v_{x,f}$	$\geq$	-2 m/s
$v_{x,f}$	$\leq$	0.5 m/s
$v_{hor,f}$	$\leq$	1.5 m/s
Ω	$\leq$	3 deg/s
Φ_f	$\leq$	3 deg

Table 1 *Summary of initial conditions and success criteria*

4 Guidance Algorithm

This sections introduces the process to generate the GP guidance. It starts by describing the structure of the guidance algorithm, which consists of the GP nonlinear trajectory tracking law that operates around a reference trajectory. Then, it describes the fitness function that the GP aims at optimizing, which is based on the success criteria previously explained. Finally it shows the GP settings used to obtain the guidance law as well as the training results.

4.1 Guidance Structure

The guidance commands are the thrust T, roll ϕ, pitch θ and yaw ψ angles. As mentioned above, the ϕ, θ and ψ angles are converted to quaternions.

The GP algorithm generates an additive correction to the optimal open-loop reference guidance commands. This optimal trajectory was generated performing an optimal control study, with the objective of landing the vehicle as close as possible to the landing pad center, with perfectly vertical attitude and zero vertical velocity. The resulting control profile is shown in Figure 5 and the unperturbed trajectory is shown in Figure 6.

The guidance algorithm modifies only the T, θ and ψ commands, while the ϕ reference command is used directly. This decision is taken since the roll command has negligible effect on the landing performance and having one less guidance command improves the GP convergence. The guidance commands issued to the plant are summarized in Equation 2

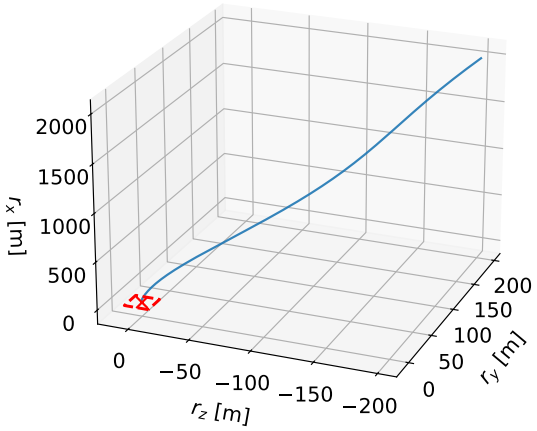

Figure 6 *Unperturbed optimal open-loop trajectory*

$$\begin{cases} T = T_{ref}(r_x) + T_{GP}(x) \\ \phi = \phi_{ref}(r_x) \\ \theta = \theta_{ref}(r_x) + \theta_{GP}(x) \\ \psi = \psi_{ref}(r_x) + \psi_{GP}(x) \end{cases} \tag{2}$$

The reference commands are obtained through PCHIP interpolation based on the altitude (r_x). The resulting guidance commands are then saturated as follows: 38 kN $\leq T \leq$ 105 kN, -40 deg $\leq \phi, \theta, \psi \leq$ 40 deg. The values used for the Thrust corresponds to $\pm$5% of the maximum and minimum values of the thrust output by the optimal control solver, while the values chosen for the angles were determined experimentally to have no instabilities. The additive corrections depend on a vector x, which is defined within the GP settings, in Section 4.3. This set of equations is evaluated at 20 Hz, which is therefore the guidance command frequency.

4.2 Fitness Function Design

The fitness function definition is a key step in the design of a GP algorithm, since this function defines the behavior of the algorithm and directly influences the performance and the convergence speed.

The fitness function must be designed accordingly to the considered definition of success. The definition of success adopted in this work is given in Section 3.2, and can be summarized as a soft landing within the desired range, with the vehicle vertically oriented. This success criteria is encoded in the fitness function by tracking two types of errors: the states' errors at the final position, which corresponds to the main driving force in the evolutionary process; and the states errors w.r.t. the reference trajectory during the full flight, to improve the convergence of the GP algorithm.

These two behaviors are encoded in the fitness function as follows. The fitness is formed by the tuple $F = [F_E, F_P]$, where F_E is a scalar term considering the tracking errors on the final position and during the trajectory, and F_P is a penalty term, used by the IGP algorithm to discard individuals with poor performance.

F_E is computed as

$$F_E = e_{final} + w_t e_{traj} \tag{3}$$

where e_{final} represents the errors on the final position, while e_{traj} represents the errors during the trajectory and w_t is a user defined weight to tune the importance of the trajectory tracking error w.r.t. to the error on the final position. e_{final} is computed as

$$e_{final} = \sqrt{\frac{\sum_{i=1}^{n_{states}} e_{f,i}^2}{n_{states}}} \tag{4}$$

where $e_{f,i}$ is the tracking error on the final position for the i-th state, computed as

$$e_{f,i} = |x_{target,i} - x_{f,i}|/\delta_i \tag{5}$$

where $x_{target,i}$ represents the desired value of the i-th state in the final position, $x_{f,i}$ is the value of the i-th state in the final position and δ_i is the tolerance allowed for the i-th state in the final position. The tolerance represents the range from the $x_{target,i}$ value, within which the x_i state is considered to have a successful value.

e_{traj} is computed as

$$e_{traj} = \sum_{i=1}^{n_{states}} \int_{E_0}^{E_f} e_{t,i}(E)dE \tag{6}$$

where E represents the sum of kinetic and potential energies of the system, computed as $E = \frac{1}{2}mv_x^2 + mgr_x$. $e_{t,i}$ is computed as $e_{t,i}(E) = (x_{ref,i}(E) - x_i(E))/\delta_i$ where $x_{ref,i}$ is the value of the i-th state taken from the reference open-loop trajectory (the one described in Section 3.2) at the energy value E, $x_i(E)$ is the value of i-th state in the flown trajectory at the energy E and δ_i is the aforementioned tolerance w.r.t. the target.

The penalty F_P is computed as

$$F_P = \sum_{i=1}^{n_{states,P}} max(0, (x_{f,i} - L_i)/s_i) \tag{7}$$

where L_i is a user defined threshold on the final value of the i-th state, above which the trajectory is considered severely unsuccessful. The threshold values L_i must be sufficiently big to ensure that only those trajectories that are too distant from the desired behavior are discarded. In fact, the penalty F_P is used by the IGP algorithm to first search for all individuals that can reach $F_P = 0$ and then, for those that minimizes F_E. More information on this mechanism can be found in [2].

To achieve robustness against perturbations, the GP algorithm must be trained on multiple combinations of perturbations, referred to as *training scenario* from here onward. Therefore, the fitness $F = [F_E, F_P]$ previously described, must be computed for multiple training scenarios and then their value must be aggregated in a single scalar. This is done through Equation 8, where $n_{training}$ is the number of chosen training scenarios, $n_{successes}$ is the number of successful trajectories over the $n_{training}$ evaluations and $F_{worst,40}$ represents the mean of the 40% $n_{training}$ fitnesses.

$$F_{ind} = (n_{training} - n_{successes})5 + F_{worst,40} \tag{8}$$

Only F_E (Equation 3) is considered in the computation of $F_{worst,40}$. For the penalty values, i.e. F_P computed as Equation 7, their values from each training scenarios are summed to obtain the penalty of the individual. This is performed as shown in Equation 9.

$$P_{ind} = \sum_{i=1}^{n_{training}} F_{P,i} \tag{9}$$

In summary, Equations 8 and 9 are used to compute the fitness and penalty of an individual in the GP evolutionary process. During the evolutionary process, the GP will aim at reaching $P_{ind} = 0$ and minimizing F_{ind}.

Regarding the number and characteristics of the training scenarios, as detailed in Algorithm 1 at line 6, the fitness evaluation happens every generation for every individual. Therefore, the number of training scenarios must be high enough to achieve robustness, but it must also be low enough to avoid prohibitive computational times. The training scenarios were selected using the approach detailed in [10], i.e. selected from a training dataset

generated by performing a Monte-Carlo simulation (using the simulation environment introduced in Section 3.1) with the reference open-loop guidance command, in an evenly spaced manner along the fitness value dimension.

4.3 Genetic Programming Settings

The IGP algorithm was set to receive as input all the translation (position and velocity) and the rotational (euler angles and angular velocities) states and their tracking errors w.r.t. the reference values. Therefore, the $\mathbf{x}$ vector with 24 values is obtained

$$\begin{aligned}\mathbf{x} = [&r_x, r_y, r_z, v_x, v_y, v_z, \phi, \theta, \psi, \omega_x, \omega_y, \omega_z, ... \\ &e_{r_x}, e_{r_y}, e_{r_z}, e_{v_x}, e_{v_y}, e_{v_z}, e_\phi, e_\theta, e_\psi, e_{\omega_x}, e_{\omega_y}, e_{\omega_z}]\end{aligned} \tag{10}$$

while, the output of the IGP algorithm is composed by three values $\mathbf{u}_{GP} = [T, \theta, \psi]$. The IGP algorithm was set with the parameters listed in Table 2. The mutation and crossover ranges represent the initial and final values of mutation and crossover rates. The final value is obtained by decreasing the mutation rate and increasing the crossover rate by the *Mutation/Crossover Probability Step* every generation until their final values is reached. The IGP $\mu+\lambda$ evolutionary strategy is employed, with $\mu = Population\ size$ and $\lambda = 1.3\mu$. The population is initialized by randomly generating 100 populations and then selecting the one with the highest entropy, as described in [3].

Parameter	Value
Population size	350
Generations number	300
Mutation probability range	[0.7, 0.25]
Crossover probability range	[0.3, 0.75]
Mutation/crossover probability step	0.01
Number of niches	10
Individual max size	60
Individual max depth	60
Ephemeral constants number	5
Ephemeral constants max value	1
Primitive set	+, -, ×, sin, cos, tanh $add3$, $plog$, $pexp$, $psqrt$

Table 2 *Summary of IGP Parameters.*

$add3$ in the primitive set in Table 2 represents a ternary addition, while $plog$, $pexp$ and $psqrt$ are protected versions of the natural logarithm, exponential and square-root functions, respectively, to avoid numerical singularities. They are defined in Equations 11 to 13.

$$plog(x) = \begin{cases}\log(x) & \text{if } x > 0 \\ 10^{-300} & \text{if } x \leq 0\end{cases} \tag{11}$$

$$psqrt(x) = \begin{cases}\sqrt{x} & \text{if } x > 0 \\ 0 & \text{if } x \leq 0\end{cases} \tag{12}$$

$$pexp(x) = \begin{cases}e^x & \text{if } -100 < x < 100 \\ e^{100} & \text{if } x \geq 100 \\ e^{-100} & \text{if } x \leq -100\end{cases} \tag{13}$$

To compute the fitness function, 25 training scenarios were selected from a Monte-Carlo conducted on simulation indexes 1 to 500. These training scenarios are computed using the simulation environment described in 3.1. Regarding the states considered to compute the fitness, those described in Section 3.2 were employed, i.e. $\mathbf{x}_{fitness} = [r_x, r_{hor}, v_x, v_{hor}, \Omega, \Phi]$, hence $n_{states} = 6$. The weight w_t in Equation 3 was set as $w_t = 3$. $\mathbf{x}_{target}$ and δ were set as $\mathbf{x}_{target} = [r_x, r_{hor}, v_x, v_{hor}, \Omega, \Phi] = [0, 0, 0, 0, 0, 0]$ (m, m, m/s, m/s, deg, deg) and $\delta = [r_x, r_{hor}, v_x, v_{hor}, \Omega, \Phi] = [3, 10, 2, 1.5, 3, 3]$ (m, m, m/s, m/s, deg, deg). To compute the penalty

in Equation 7, only the terminal vertical and horizontal position and the terminal vertical velocity were considered. The terminal velocity was checked against a positive and a negative limit, hence $n_{states,P} = 4$. L was set as $L = [r_x, r_{hor}, v_{x,up}, v_{x,down}] = [10, 30, 0.5, 10]$ (m, m, m/s, m/s) and $s = [r_x, r_{hor}, v_{x,up}, v_{x,down}] = [2000, 2000, 150, 150]$ (m, m, m/s, m/s).

4.4 GP Training Results

In order to assess the stochastic robustness of the GP evolutionary process, 10 independent evolutionary processes are performed and the best individual from each evolutionary process is tested on a Monte-Carlo campaign performed on simulation indexes 501 to 1000. The results are reported in Table 3. As can be observed, the majority of the performed evolutions lead to a 100% success rate, resulting in a median of 100% with a standard deviation of ±7.5%.

Run ID	1	2	3	4	5	6	7	8	9	10	median	std
Success	100%	100%	86.2%	100%	100%	77.4%	100%	92.8%	100%	100%	100%	7.5%

Table 3 *GP training results*

5 LQR Benchmark

As a baseline for comparison with the GP guidance, a Linear Quadratic Regulator (LQR) was designed to provide the guidance corrections as done by the GP in Equation 2. In particular, $\mathbf{u} = \mathbf{u}_{\text{ref}} + \mathbf{u}_{LQR}$, with

$$\mathbf{u}_{LQR} = -\mathbf{K}\,(\mathbf{x} - \mathbf{x}_{\text{ref}}). \tag{14}$$

As for the GP, the LQR acts on three channels, the thrust magnitude T and the commanded pitch θ and yaw ψ angles, while the roll command is held at its reference value.

The gain is obtained from a linearisation of the 3DoF descent dynamics about the vertical reference condition, expressed in the UEN frame. The regulator state is

$$\delta\mathbf{x} = \begin{bmatrix} \delta r_x & \delta r_y & \delta r_z & \delta v_x & \delta v_y & \delta v_z & \delta\theta & \delta\psi & \delta\omega_y & \delta\omega_z \end{bmatrix}^{\top}, \tag{15}$$

comprising the position and velocity errors, the pitch and yaw attitude errors θ and ψ, and the associated body rates ω_y and ω_z. The control vector is $\delta\mathbf{u} = [\,\delta T,\ \delta\theta_c,\ \delta\psi_c\,]^{\top}$. The linearised error dynamics are

$$\delta\dot{r}_x = \delta v_x, \quad \delta\dot{r}_y = \delta v_y, \quad \delta\dot{r}_z = \delta v_z, \tag{16}$$

$$\delta\dot{v}_x = \frac{\delta T}{m_{\text{ref}}}, \tag{17}$$

$$\delta\dot{v}_y = +\frac{T_{\text{ref}}}{m_{\text{ref}}}\,\delta\psi, \tag{18}$$

$$\delta\dot{v}_z = -\frac{T_{\text{ref}}}{m_{\text{ref}}}\,\delta\theta, \tag{19}$$

$$\delta\dot{\theta} = \delta\omega_y, \quad \delta\dot{\psi} = \delta\omega_z, \tag{20}$$

$$\delta\dot{\omega}_y = \omega_n^2\,(\delta\theta_c - \delta\theta) - 2\zeta\omega_n\,\delta\omega_y, \tag{21}$$

$$\delta\dot{\omega}_z = \omega_n^2\,(\delta\psi_c - \delta\psi) - 2\zeta\omega_n\,\delta\omega_z. \tag{22}$$

Equations 18 and 19 describe the tilting of the body-axial thrust vector by small attitude perturbations; the two coupling signs are specific to the simulator's quaternion rotation convention. Equations 21 and 22 model the time-optimal attitude controller as a critically-damped second-order tracker ($\zeta = 1$) of bandwidth $\omega_n = 3$ rad/s. Having the body rates ω_y and ω_z in the regulator state allows the cost to penalise them directly, which is required to

drive the terminal angular rate to zero. The reference mass m_{ref} and thrust T_{ref} were evaluated from the reference trajectory at mid-descent (1000 m altitude), and a single operating point was used.

The feedback gain $\mathbf{K} = \mathbf{R}^{-1}\mathbf{B}^{\top}\mathbf{P}$ follows from the stabilising solution $\mathbf{P}$ of the continuous-time algebraic Riccati equation with diagonal weights

$$\mathbf{Q} = \text{diag}\big(5{\times}10^{-2},\ 5{\times}10^{-2},\ 5{\times}10^{-2},\ 25,\ 1,\ 1,\ 20,\ 20,\ 50,\ 50\big) \tag{23}$$

$$\mathbf{R} = \text{diag}\big(10^{-7},\ 2,\ 2\big) \tag{24}$$

specified in (m, m/s, rad, rad/s for the state; N and rad for the inputs). The commanded corrections are saturated at $|\delta T| \leq 30$ kN and $|\delta\theta_c|,\ |\delta\psi_c| \leq 15°$.

6 Results

This section presents the performance of the GP algorithm, assessing it w.r.t. the LQR benchmark described in the previous section.

The test to evaluate algorithm performance is performed through a Monte-Carlo campaign, using the simulation environment described in 3.1. This Monte-Carlo analysis is performed with 500 runs with simulation indexes 501 to 1000, considering: 1) the reference optimal guidance commands with no corrections, 2) the reference guidance commands with LQR corrections, and 3) the reference guidance commands with GP corrections. All the results are closed-loop. The evaluation of both LQR and GP is conducted at the simulation frequency (i.e. 20Hz).

The results of the reference guidance commands with no corrections are depicted in Figures 4 and 3, resulting in a success rate of 17.4% (87/500 successful runs). The results using the LQR controller and the GP guidance are shown throughout this section.

The LQR controller is capable of obtaining a success rate of 100%. This can be observed in the final states depicted in Figure 8. In Figure 7, it can be observed that the euler angles guidance commands present a highly oscillatory behaviour that might lead to instabilities in more realistic simulation environments (e.g. 6DoF) or in reality. Nevertheless, it is acceptable as a benchmark with which to compare the GP performance.

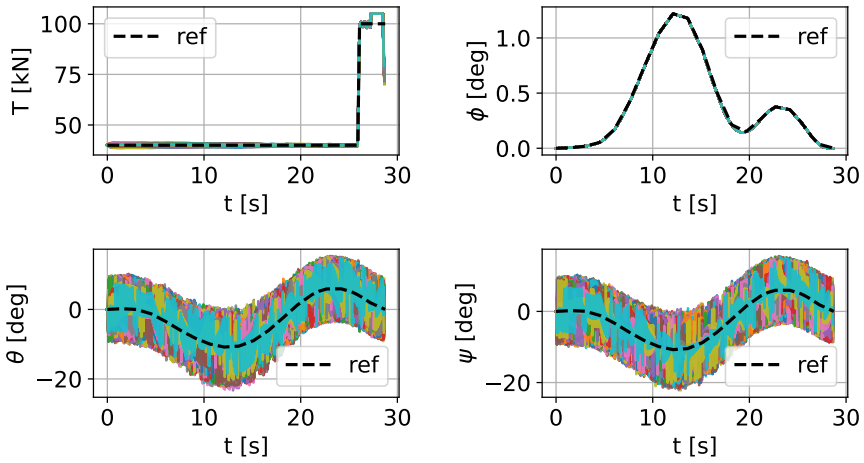

Figure 7 *LQR guidance commands.*

The evolution with the best performance in 4.4 is chosen for this comparison (Run ID 2). The obtained guidance commands and states at landing are depicted in Figures 9 and 10. As for the LQR controller, also the GP guidance law achieves a 100% success rate.

To better understand the performance of the three considered approaches, Figure 11 depicts the statistical results of the relevant states at landing, obtained by the three algorithms - reference commands with no correction,

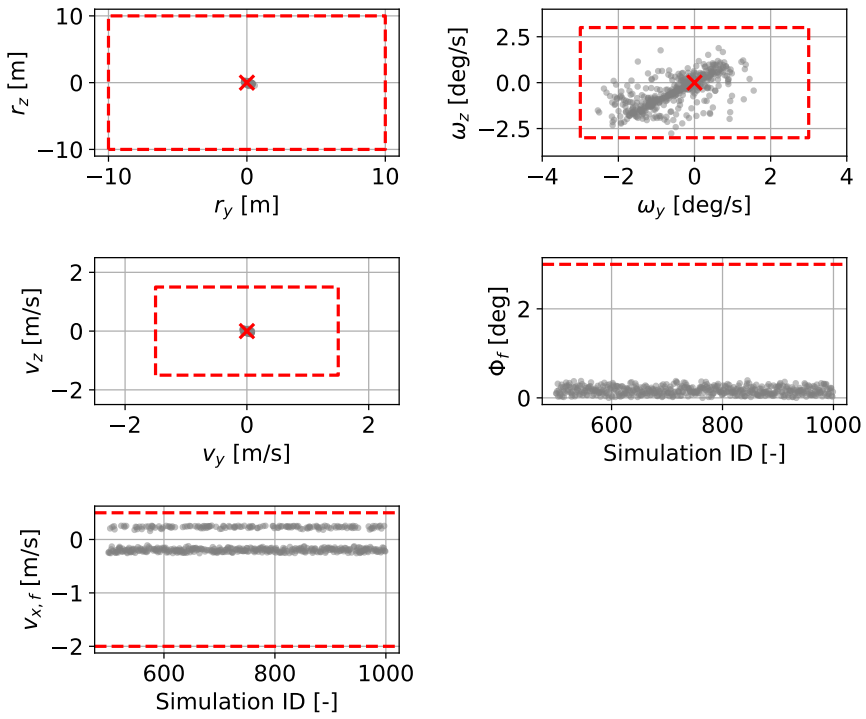

Figure 8 *Final states obtained with the LQR controller. Each gray dot represents a MC simulation. The red dashed line represents the tolerance for each quantity.*

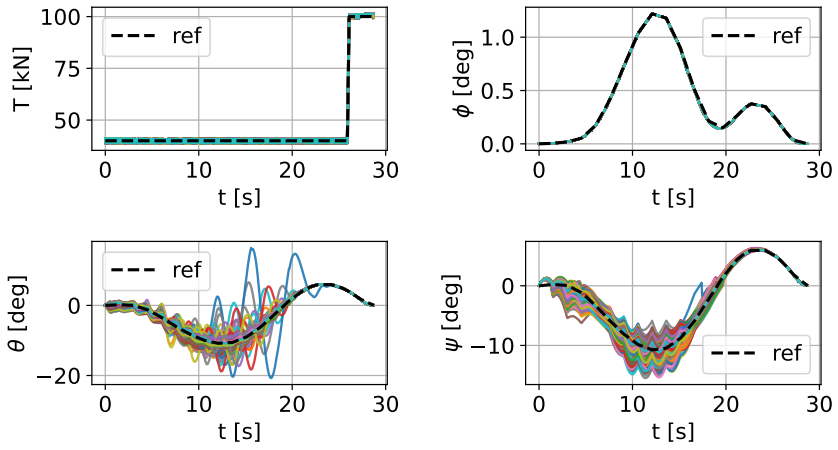

Figure 9 *Guidance commands produced with the GP individuals obtained at evolution #2*

reference commands with LQR correction and reference commands with GP correction - on the performed 500 MC runs. The boxplots used to analyze the results have the following meaning: the orange line inside the box represents the median of the dataset, the vertical rectangle covers the range from first (Q1) to the third (Q3) quartile, while the whiskers (vertical lines) spans from $Q1 - 1.5IQR$ to $Q3 + 1.5IQR$, with $IQR = Q3 - Q1$. The

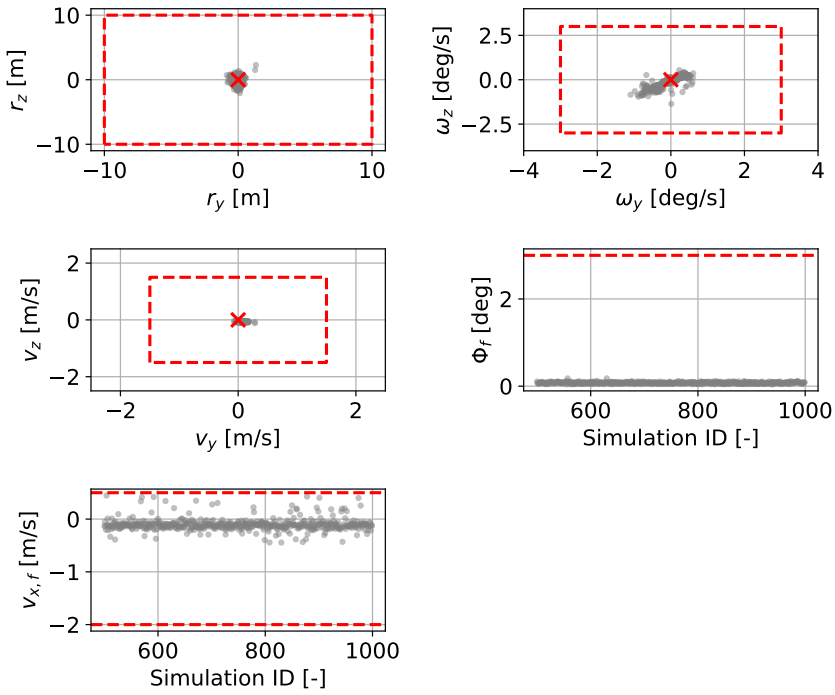

Figure 10 *Finals states obtained with the GP individuals obtained at evolution #2. Each gray dot represents a MC simulation. The red dashed line represents the tolerance for each quantity.*

circles represent the outliers of the dataset.

For the GP, the individuals obtained by evolution #2 are considered, since they led to the greatest precision at landing. As can be observed from Figure 11, the LQR and GP have comparable performance, on position and velocity the LQR is slightly more precise than GP, while on final angle and final angular rate, the GP outperforms the LQR. Moreover, they both outperform the reference commands with no corrections, except for the final angular rate where the LQR is the worst performing approach.

A further comparison between LQR and GP is provided in Figure 12 that displays the control efforts computed as the integral in time of the absolute command corrections (δT, $\delta\theta$, $\delta\psi$). This figure reveals a qualitative difference between the control strategies. The LQR predominantly compensates disturbances through attitude corrections, while the GP relies more on thrust modulation with substantially smaller attitude corrections. This behaviour is likely a consequence of the fundamentally different optimisation objectives. The LQR minimizes a quadratic cost around a single linearization point, making attitude changes an efficient means of generating lateral accelerations. The GP on the other hand can exploit the nonlinear dynamics and has learned that modulating thrust while remaining closer to the nominal attitude can achieve comparable terminal accuracy.

7 Conclusions

This work presented the application of GP to the powered landing guidance of a VTVL rocket. It represents the first attempt at solving the problem, hence it considers a simplified form: the rocket is modeled as a 3DoF model with attitude kinematics and perturbations are considered only on the aerodynamic and atmospheric models. To assess the effectiveness of the GP guidance commands, these are compared against the reference commands with no online correction and with the reference commands with an LQR correction. The experiment shows that the GP effectively enhances the reference guidance commands to compensate for disturbances, leading to a 100% success rate from an initial success rate of 17.4% (obtained with the reference commands with no corrections).

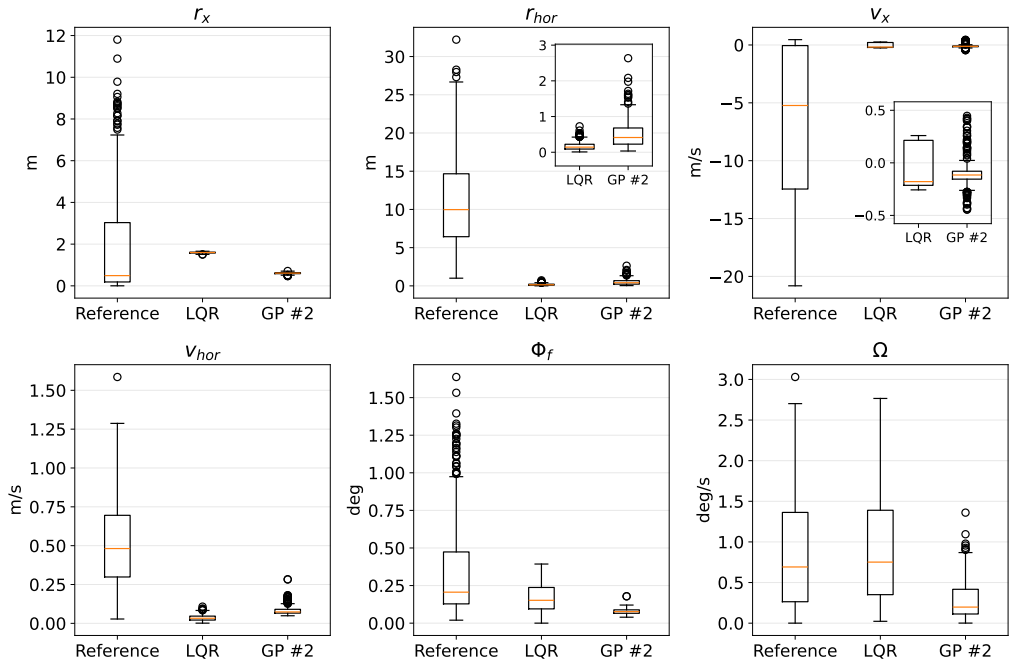

Figure 11 *Statistical results of the relevant states at landing. The insets in the r_{hor} and v_x plots show a magnification on the LQR and GP results, to make the results more intelligible.*

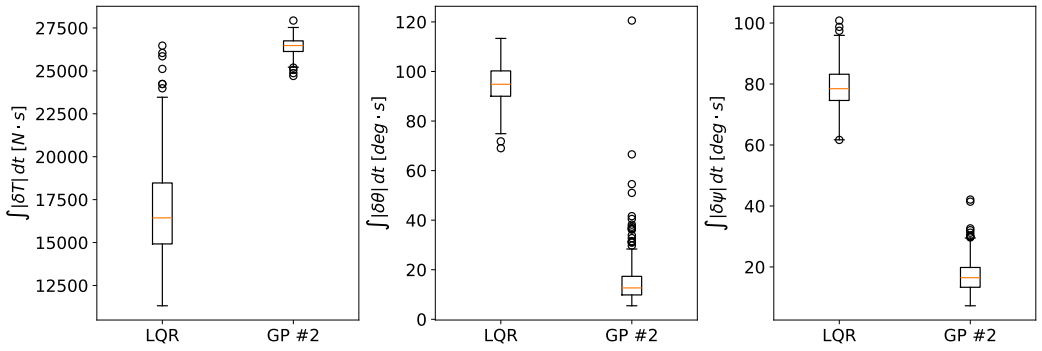

Figure 12 *Guidance commands control efforts.*

Moreover, it has comparable performance w.r.t. the LQR controller achieving the same success rate and smaller control efforts on the attitude commands. The advantages of the GP over simple techniques such as LQR are expected to become more apparent as the problem becomes more complex. In this regard, future efforts will be focused on tackling the full problem with the GP algorithm, i.e. 6DoF rocket model, with direct actuator commands and the full envelope of disturbances.

References

[1] J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*. MIT Press, 1992.

[2] F. Marchetti and E. Minisci, "A hybrid neural network-genetic programming intelligent control approach," in *Lecture Notes in Computer Science*, ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2020, pp. 240–254.

[3] F. Marchetti and E. Minisci, "Inclusive Genetic Programming," in *Genetic Programming*, T. Hu, N. Lourenço, and E. Medvet, Eds., Cham: Springer International Publishing, 2021, pp. 51–65, ISBN: 978-3-030-72812-0.

[4] M. Sagliano et al., "Onboard guidance for reusable rockets: Aerodynamic descent and powered landing," in *AIAA Scitech 2021 Forum*, VIRTUAL EVENT: American Institute of Aeronautics and Astronautics, Jan. 2021.

[5] A. Botelho, M. Martinez, C. Recupero, A. Fabrizi, and G. De Zaiacomo, "Design of the landing guidance for the retro-propulsive vertical landing of a reusable rocket stage," en, *CEAS Space J.*, vol. 14, no. 3, pp. 551–564, Jul. 2022.

[6] A. G. Kamath et al., "Real-time sequential conic optimization for multi-phase rocket landing guidance," en, *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 3118–3125, 2023.

[7] M. Sagliano, D. Seelbinder, S. Theil, and P. Lu, "Six-degree-of-freedom rocket landing optimization via augmented convex–concave decomposition," en, *J. Guid. Control Dyn.*, pp. 1–16, Sep. 2023.

[8] J. Wang, H. Ma, H. Li, and H. Chen, "Real-time guidance for powered landing of reusable rockets via deep learning," en, *Neural Comput. Appl.*, vol. 35, no. 9, pp. 6383–6404, Mar. 2023.

[9] U. Çelik and M. U. Demirezen, "Optimal reusable rocket landing guidance: A cutting-edge approach integrating scientific machine learning and enhanced neural networks," *IEEE Access*, vol. 12, pp. 16 805–16 829, 2024.

[10] F. Marchetti, J. L. Redondo Gutierrez, and D. Seelbinder, "Genetic programming guidance for the reentry trajectory of the ReFEx vehicle," in *IAF Space Transportation Solutions and Innovations Symposium*, Milan, Italy: International Astronautical Federation (IAF), 2024, pp. 362–375.

[11] M. Sagliano, P. Lu, B. Johnson, D. Seelbinder, and S. Theil, "Six-degrees-of-freedom aero-propulsive entry trajectory optimization," in *AIAA SCITECH 2024 Forum*, Orlando, FL: American Institute of Aeronautics and Astronautics, Jan. 2024.

[12] M. Sagliano et al., "Powered atmospheric landing guidance for reusable rockets: The CALLISTO studies," in *AIAA SCITECH 2024 Forum*, Orlando, FL: American Institute of Aeronautics and Astronautics, Jan. 2024.

[13] J. L. Redondo Gutierrez, D. Seelbinder, and S. Theil, "Reusability flight experiment guidance: Trajectory correction after ascent," *Aerospace*, vol. 12, no. 9, 2025, ISSN: 2226-4310. DOI: 10.3390/aerospace12090838 [Online]. Available: https://www.mdpi.com/2226-4310/12/9/838

[14] M. Sagliano, D. Seelbinder, S. Theil, B. J. Johnson, and P. Lu, "Six-degree-of-freedom entry to powered descent optimization for high-mass mars landings," in *AIAA SCITECH 2025 Forum*, Orlando, FL: American Institute of Aeronautics and Astronautics, Jan. 2025.

[15] M. Sagliano et al., "Operational atmospheric landing guidance for reusable rockets," in *AIAA SCITECH 2025 Forum*, Orlando, FL: American Institute of Aeronautics and Astronautics, Jan. 2025.

[16] R. Cadamuro, F. Marchetti, J. L. Redondo Gutierrez, and D. Seelbinder, "Validating online guidance for reusability flight experiment: Genetic programming and reinforcement learning," en, *J. Guid. Control Dyn.*, pp. 1–20, Aug. 2026.

[17] J. Carradori, M. Sagliano, and E. Mooij, "Transformer-based atmospheric rocket landing guidance using meta-reinforcement learning," en, *J. Guid. Control Dyn.*, vol. 49, no. 4, pp. 1083–1100, Apr. 2026.

[18] S. Krummen et al., "CALLISTO: On the design and development of a reusable first stage demonstrator," en, *CEAS Space J.*, Mar. 2026.

[19] https://spacenews.com/china-becomes-second-country-to-recover-orbital-booster-with-long-march-10b/, [Accessed 14-07-2026].

[20] *LandSpace's Zhuque-3 reusable rocket test a 'breakthrough' for China space race — scmp.com*, https://www.scmp.com/news/china/science/article/3278193/landspaces-zhuque-3-reusable-rocket-test-breakthrough-china-space-race, [Accessed 30-05-2026].

[21] *Maia - ArianeGroup — ariane.group*, https://ariane.group/en/space-transportation/maia/, [Accessed 30-05-2026].

© *Francesco Marchetti, Jose Luis Redondo Gutierrez and Marco Sagliano*