Towards the Integration of Surrogate Models in the Robust Optimization of Hybrid Rocket Engines

Towards the Integration of Surrogate Models in the Robust Optimization of Hybrid Rocket Engines

F. Masseni*
Politecnico di Torino, Turin, 10129, Italy

Hybrid rocket engines combine the appealing features of solid and liquid propulsion systems, offering advantages such as cost-effectiveness, safety, and eco-friendliness. Their potential role in sustainable space transportation aligns with the goals of the NewSpace Economy, particularly in light of environmental concerns surrounding traditional propulsion systems. While hydrazine-based liquid rocket engines face potential bans in Europe, and solid rocket motors contribute significantly to pollution, hybrids using liquid oxygen and paraffin-based wax are regarded as sufficiently green. Additionally, advances in liquefying fuels have led to increased global interest, as demonstrated by numerous recent development programs. Design optimization for hybrid rocket engines is inherently multidisciplinary, with propulsion system and trajectory optimization closely coupled. While deterministic methods have been widely used, real-world uncertainties, especially in fuel regression rates, can significantly jeopardize system performance and reliability. Robust optimization approaches mitigate these effects, but they are computationally expensive. This paper presents a preliminary investigation into the capabilities of surrogate models in these scenarios. A binary classifier is coupled with a neural network to predict system performance in the presence of uncertainties. The design optimization of a hybrid-powered upper stage is selected as a case study to test the proposed method's effectiveness.

Nomenclature

A_b	=	burning surface area, m^2
A_p	=	port area, m^2
A_{th}	=	nozzle throat area, m^2
a	=	regression constant, $m^{1+2n}\ kg^{-n}\ s^{n-1}$
$\boldsymbol{b}$	=	design parameters vector
$\boldsymbol{b}_L$	=	lower bound vector
$\boldsymbol{b}_U$	=	upper bound vector
C_F	=	thrust coefficient
c	=	particles acceleration in PSO
c^*	=	characteristic velocity, m/s
$\boldsymbol{D}$	=	drag vector, N
D	=	rocket outer diameter, m
E	=	nozzle expansion area ratio
$\boldsymbol{F}$	=	thrust vector, N
G	=	gravitational constant, Nm^2/kg^2
$\boldsymbol{g}$	=	gravity acceleration, m/s^2
$g_j\ (\boldsymbol{b})$	=	j-th inequality constraint
h	=	altitude, km
h^*	=	target altitude, km
J	=	initial throat area to port area ratio
L	=	overall engine length, m
L_b	=	fuel grain length, m

*Fixed-term Tenure-track Assistant Professor, Department of Mechanical and Aerospace Engineering, Corso Duca degli Abruzzi 24, 10129 Turin, Italy, AIAA Member.

m = rocket mass, mass kg
$M_\oplus$ = Earth mass, kg
μ = payload mass, kg
N = number
n = mass-flux exponent
P = burning perimeter, m
$\boldsymbol{p}$ = uncertain variables vector
p = pressure, bar
R_g = grain outer radius, m
R_i = grain initial inner radius, m
R_{th} = throat radius, m
$\boldsymbol{r}$ = position vector, m
s = eroded distance, thickness, m
T = temperature, K
t = time, s
u_r = velocity component in the radial direction, km/s
V = volume, m^3
v_e = velocity component in the eastward direction, km/s
$\boldsymbol{v}$ = velocity vector, m/s
w = web thickness, m, inertia weight in PSO
w_n = velocity component in the northward direction, km/s
y = burning distance, m
Z = hydraulic resistance, 1/(kg m)
$\boldsymbol{z}^p$ = noise vector of $\boldsymbol{p}$
α = mixture ratio
γ = specific heat ratio
η = efficiency
ρ = density, kg/m^3
Φ = objective function, kg
ϕ = latitude, deg

Superscripts

$\dot{}$ = time derivative

Subscripts

0 = ambient
1 = combustion chamber at head-end
a = auxiliary gas
avg = average
BD = beginning of blow-down phase
F = thrust coefficient
c = combustion chamber
$case$ = case
cc = combustion chamber
dry = dry
$*$ = characteristic velocity
e = nozzle exit
F = fuel
g = pressurizing gas
He = Helium
i = initial value
id = ideal
max = maximum
min = minimum
nz = nozzle
O = oxidizer

p	=	overall propellant (oxidizer + fuel)
ref	=	reference
t	=	oxidizer propellant tank
ull	=	ullage
ves	=	vessel
$\oplus$	=	standard astronomical symbol for planet Earth

I. Introduction

Hybrid Rocket Engines (HREs) are emerging as promising propulsion systems, offering performance comparable to that of storable or semi-cryo liquid rocket engines, with an average specific impulse of around 300 seconds. HREs combine the advantages of both Solid Rocket Motors (SRMs) and Liquid Rocket Engines (LREs), including cost-effectiveness, safety, and throttleability. Furthermore, HREs represent a viable solution to the growing demand for sustainable space transportation within the NewSpace Economy. The European Commission is expected to ban hydrazine and its compounds, widely used in liquid rocket propellants, by classifying them under REACH (Registration, Evaluation, Authorization, and Restriction of Chemicals). This move could negatively impact the European space industry by approximately €2B/year[1]. Conversely, SRMs have a significant environmental impact, primarily due to the use of chloride compounds and energetic additives (e.g., aluminum powder) in their propellant grains. In contrast, HREs are much more eco-friendly compared to both hydrazine-based LREs and SRMs. The most promising propellant combination for large-scale hybrid rockets is liquid oxygen and paraffin-based wax, which produces primarily carbon dioxide, carbon monoxide, and water vapor, minimizing harmful emissions[2].

Despite these advantages, the widespread adoption of hybrid propulsion systems has been hindered by several significant challenges. Notably, low regression rates[3] and combustion instabilities have been the primary obstacles to overcome. In recent decades, however, research into liquefying fuels, such as paraffin-based waxes, and their unique entrainment phenomenon has helped address the low regression rate issue. Additionally, the development of accurate numerical models has enabled a better understanding and analysis of combustion instabilities[4].

The global commitment to hybrid propulsion is focused on closing the performance gap with legacy propulsion systems, as evidenced by the growing research interest[5–7] and the numerous active programs dedicated to HREs development over the past five years. These include Nammo's Nucleus sounding rocket and Nucleus XL launcher[8], as well as EU-funded projects such as ALTAIR, SMILE, HyTEX, HyTEC, and ENVOL under Horizon 2020[9–11]. Other notable developments include Hympulse's SL1 three-stage launcher[12], Gilmour Space's Eris Block 1 launcher, Reaction Dynamics' Aurora launcher, and several others[13].

The time and budget required for developing hybrid engines can be minimized through a well-executed conceptual design. HRE design optimization is inherently multidisciplinary, as propulsion system requirements and trajectory analysis are closely interdependent. Hybrid engines, being single-lever control systems, typically exhibit a changing thrust profile and shifting mixture ratios. For nearly 30 years, the development of design optimization techniques for HREs has been a core research area at the Polytechnic University of Turin[14]. This expertise covers a wide range of applications, with a focus on multi-stage small satellite launchers and upper stages. The complexity of the optimization process is reduced when design variables and model parameters are treated as deterministic. In previous works, the coupled optimization of launch vehicle trajectory and hybrid rocket engine design was performed using an in-house procedure from a strictly deterministic point of view, i.e., all model parameters were assumed as deterministic quantities and all design variables can be freely set and adjusted by the designers[15, 16].

However, uncertainties in design parameters cannot be ignored in real-world scenarios, particularly for HREs. One of the most critical parameters is the fuel grain regression rate, which unfortunately is well known to be affected by uncertainties. Even small fluctuations in this rate can significantly impact vehicle performance or lead to mission failure. Thus, it is crucial to factor in uncertainties in the early design phases, switching to an uncertainty-based design optimization, also known as robust design approaches. The main objective of these methods is to minimize the negative impact of small, unavoidable variations in design parameters on system performance, without eliminating the sources of uncertainty or variation[17]. In recent years, the author has explored robust optimization for hybrid-powered upper stages and small launchers, focusing on regression rate uncertainties[18, 19]. A broader range of uncertain parameters has also been considered, identified through a global sensitivity analysis based on Morris' method[20]. The results demonstrated that robust solutions are achievable, with a slight performance trade-off (e.g., reduced payload mass). However, the computational cost of robust optimization is highly sensitive to the number and selection of uncertain parameters, requiring up to fifty times more effort than deterministic optimization runs, with the in-house method

relying on orthogonal arrays.

To reduce the computational effort involved in robust-based optimization, a surrogate model can be employed. This involves selecting a set of sampling points using a design of experiments technique, aiming to extract the maximum possible information about the system's response with a relatively small sample size using an high fidelity approach. The surrogate model is then constructed to minimize discrepancies with the original model's response, and optimization is performed employing the surrogate. Common surrogate models used in aerospace engineering include kriging, co-kriging, radial basis function models, moving least squares, and neural networks. To obtain a robust solution, both the original model and the surrogate must account for uncertainties in the design parameters and evaluate their effects on system performance. Ultimately, achieving satisfactory performance requires a careful selection of database size, sampling method, surrogate architecture and optimization strategies.

In this paper, the author proposes a preliminary approach toward integrating a surrogate model into uncertainty-based HREs design optimization process. The proposed surrogate model consists of three main components: a deterministic tool for initial design evaluations, a binary classifier to filter feasible and robust solutions, and a neural network to predict the robust-based performance index without requiring additional calls to the full numerical model. The training database is built by means of a robust-based evaluation tool, which evaluates the performance of the candidate solutions when uncertainty comes into play. The optimization of the hybrid engine for an upper stage is here selected as a test case due to the extensive research by the author on this topic.

II. Numerical Models

A. Engine

The test case here considered is an hybrid-powered upper stage, suitable for the replacement of Vega launcher third and fourth stages. The initial mass of the stage is $m_i = 14522$ kg[21]. The propellants are Liquid OXygen (LOX) and paraffin-based wax. LOX/wax is a promising combination for HREs due to its good specific impulse and high regression rate[3], which enables the use of a single port grain design and avoids the addition of aluminum particles in the fuel grain. A partially regulated feed system is used: initially, the hybrid engine works at a constant tank pressure, followed by a Blow-Down (BD) phase.

The engine modeling is based on a set of proper assumptions:

1) $\eta_* = 0.96$, correction factor to account for c^* inefficiencies [22];
2) $\eta_F = 0.98$, accounting for losses in the evaluation of the thrust coefficient C_F [22];
3) frozen equilibrium expansion, i.e., γ = constant throughout the nozzle;
4) isentropic expansion in the nozzle [22];
5) single circular port grain design, uniform regression rate along port axis, and no contribution of grain lateral ends to overall combustion;
6) regression rate correlation coefficient in nominal conditions $a = 9.1 \cdot 10^{-5}\,\mathrm{m}^{2n+1}\mathrm{s}^{n-1}\mathrm{kg}^{-n}$ and exponent $n = 0.69$[3] (SI units);
7) third-order polynomial fittings of $c^*_{id}(\alpha)$ and $\gamma(\alpha)$ (computed by NASA CEA with $p_c = 10$ bar = constant) [23];
8) hydraulic resistance Z = constant and incompressible turbulent flow in the tank-chamber flow path;
9) $(p_c)_i = 0.4(p_d)_i$ to avoid the coupling between the engine and the feed system during operation;
10) $J = 0.5$ at engines ignition to limit pressure losses in the combustion chamber;
11) Bartz's method to model throat erosion[24], reference erosion rate $\dot{s}_{ref} = 0.1$mm/s[25];
12) initial ullage volume V_{ull} equal to 3% of the overall oxidizer volume[26];
13) initial auxiliary Helium vessel pressure $(p_a)_i = 250$ bar;
14) oxidizer tank pressure $p_t = 25$ bar = constant during the regulated operation;

Given this set of assumptions, the HRE design is determined by six input design parameters: grain outer radius R_g, web thickness w, grain length L_b, the overall oxidizer mass exhausted $(m_O)_f$, the oxidizer mass exhausted during the BD operation $(m_O)_{BD}$, and the initial nozzle expansion area ratio E_i. The input parameters are collected in:

$$\boldsymbol{b} = \left[R_g, w, L_b, (m_O)_f, (m_O)_{BD}, E_i\right] \tag{1}$$

The fuel grain geometry is given by R_g, w, and L_b. The grain inner radius is then $R_i = R_g - w$ and the burning perimeter P and the port area A_p can be evaluated for any given burning distance y ($0 \le y \le w$) as:

$$P = 2\pi\,(R_i + y) \tag{2}$$

$$A_p = \pi \left(R_i + y\right)^2 \tag{3}$$

An approximate relation accounts for pressure losses inside the combustion chamber [27]:

$$p_1 = \left[1 + 0.2 \left(\frac{A_{th}}{A_p}\right)^2\right] p_c \tag{4}$$

where p_c is the chamber nozzle-stagnation pressure, p_1 is the chamber head-end pressure, and A_{th} is the throat area. Oxidizer mass flow rate $\dot{m}_O$ is given by:

$$\dot{m}_O = \sqrt{(p_t - p_1)/Z} \tag{5}$$

where p_t is the oxidizer tank pressure. Grain geometry and $\dot{m}_O$ determine the fuel regression rate $\dot{y}$:

$$\dot{y} = a \left(\dot{m}_O / A_p\right)^n \tag{6}$$

Fuel mass flow rate $\dot{m}_F$ is then:

$$\dot{m}_F = \rho_F \dot{y} A_b = \rho_F \dot{y} L_b P \tag{7}$$

where ρ_F is the fuel grain density (940 kg/m^3 for paraffin-based wax), and A_b is the burning area. The mixture ratio α can be computed as:

$$\alpha = \frac{\dot{m}_O}{\dot{m}_F} \tag{8}$$

Characteristic velocity $c^*(\alpha) = \eta_* c^*_{id}(\alpha)$ is known, and thus p_c:

$$p_c = \frac{(\dot{m}_O + \dot{m}_F) c^*}{A_{th}} \tag{9}$$

The oxidizer tank is pressurized by a Helium flow from an auxiliary vessel of volume V_a, and given initial pressure p_a. V_a is one of the engine design parameter, but it is conveniently replaced by the exhausted oxidizer mass at the beginning of the BD phase $(m_O)_{BD}$. During the BD phase p_t is calculated assuming an isentropic expansion of the Helium flown in the oxidizer tank:

$$p_t = (p_t)_i \left[\frac{(V_g)_{BD}}{V_g}\right]^{\gamma_g} \tag{10}$$

where the Helium volume in the tank $V_g = \left(V_g\right)_i + m_O/\rho_O$ depends on the oxidizer mass m_O that has been already exhausted, $\left(V_g\right)_{BD} = \left(V_g\right)_i + (m_O)_{BD}/\rho_O$ and γ_g is Helium's specific heat ratio.

The thrust coefficient C_F is calculated as:

$$C_F = \eta_F \left\{ \sqrt{\frac{2\gamma^2}{\gamma - 1} \left(\frac{2}{\gamma + 1}\right)^{\frac{\gamma+1}{\gamma-1}} \left[1 - \left(\frac{p_e}{p_c}\right)^{\frac{\gamma-1}{\gamma}}\right]} + E \frac{p_e}{p_c} \right\} - E \frac{p_0}{p_c} \tag{11}$$

The term related to the atmospheric pressure p_0/p_c is always small since the upper stage always flies at high altitude. Propellant mass flow rate at ignition (i.e. at t=0) $(\dot{m}_p)_i$ can be found as:

$$(\dot{m}_p)_i = (1 + \alpha_i)(\dot{m}_F)_i = \frac{1 + \alpha_i}{\alpha_i} (\dot{m}_O)_i \tag{12}$$

Initial throat area $(A_{th})_i$ and initial port area $\left(A_p\right)_i$ are then determined as:

$$(A_{th})_i = \frac{(\dot{m}_p)_i}{(p_c)_i c^*_i} \;; \qquad (A_p)_i = \frac{(A_{th})_i}{J} \tag{13}$$

Bartz's method models the dependence of the rate of throat erosion $\dot{s}$ on throat radius R_{th} and chamber pressure p_c:

$$\dot{s} = \dot{s}_{ref} \left(\frac{p_c}{p_{c,ref}}\right)^{0.8} \left(\frac{R_{th,ref}}{R_{th}}\right)^{0.2} \tag{14}$$

R_{th} and E throughout mission are computed by integrating Eq.14. The erosion along the nozzle is neglected and the eroded mass isn't considered, either for thrust augmentation or for rocket mass reduction.

It is now useful to introduce the upper and lower boundary vector for the design variables $\boldsymbol{b}_U$ and $\boldsymbol{b}_L$, shown in Tab. 1, given by mission requirements and user experience on the topic. It is worth noting that these boundaries must be imposed both in the building process of the surrogate model (e.g., in data generation for training) and in the optimization algorithm.

Table 1 Design parameters ranges.

	R_g	w	L_b	$(m_O)_f$	$(m_O)_{BD}$	E_i
	m	m	m	kg	kg	-
$\boldsymbol{b}_L$	0.57	0.25	4.30	7129	3195	15
$\boldsymbol{b}_U$	0.60	0.32	4.50	7697	3631	20

B. Trajectory

Once the launcher design is given by $\boldsymbol{b}$ and the initial relevant masses are computed by means of a proper set of simplifications and assumptions[28], an in-house indirect method based on the optimal control theory optimizes the trajectory, aiming at the maximization of the insertion altitude h. A point mass rocket is considered and the state equations provide the derivative of position $\boldsymbol{r}$ (radius, latitude, and longitude), velocity $\boldsymbol{v}$ (radial, eastward, and northward components), and rocket mass m[29]. In a vectorial notation one has:

$$\frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}t} = \boldsymbol{v} \qquad \frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}t} = \boldsymbol{g} + \frac{\boldsymbol{F} - \boldsymbol{D}}{m} \qquad \frac{\mathrm{d}M}{\mathrm{d}t} = -\frac{|\boldsymbol{F}|}{c^* C_F} \tag{15}$$

An inverse-square gravity field is assumed:

$$\boldsymbol{g} = -\frac{GM_\oplus}{||\boldsymbol{r}^3||}\boldsymbol{r} \tag{16}$$

where G is the gravitational constant and $M_\oplus$ is Earth mass.

Density and pressure are interpolated from standard atmosphere tables as a function of the rocket altitude during the ascent. The equations of motion are written in a non-dimensional form to improve the integration numerical accuracy. The details about the indirect optimization procedure can be found in Ref. [30].

The conditions at the ignition of the third stage, consistent with a launch from Kourou, are assigned: altitude $h = 86.88$ km, latitude $\phi = 9.11°$, velocity components in the radial, eastward and northward directions $u_r = 0.142$ km/s, $v_e = 0.230$ km/s, $w_n = 4.146$ km/s, respectively. The target final orbit is specified by assigning altitude, eccentricity and inclination (700 km, zero and 90 deg. respectively). The longitude of the ascending node is left free.

C. Robustness

Once the launcher design (i.e., the payload mass μ) and the insertion altitude h are known, the performance index $\Phi(\boldsymbol{b})$ can be computed as:

$$\Phi(\boldsymbol{b}) = \mu - k \cdot \max(0, \Delta_h) \text{ where } \Delta_h = h^* - h \tag{17}$$

where an ϵ-constraint approach is used to maximize μ and force altitude violation Δ_h to zero ($k = 20$ kg/km, $h^* = 700$ km).

From a deterministic point of view, i.e., when all model parameters assume their nominal values, a single run of the proposed numerical model for engine and trajectory is enough to compute Φ. On the contrary, multiple calls are required when uncertainties in one (or more) of the model parameters are taken into account[20]. In this work, only regression rate uncertainties are considered. Three levels are assigned to correlation coefficient a and exponent n: $a_i \cdot 10^5 = 9, 9.1, 9.2\mathrm{m}^{2n+1}\mathrm{s}^{n-1}\mathrm{kg}^{-n}$ for $i = 1, 2, 3$, and $n_j = 0.68, 0.69, 0.7$ for $j = 1, 2, 3$, respectively.

In the literature[31] the robust optimization problem is usually formally cast as:

$$\begin{array}{ll} \text{find} & \boldsymbol{b} \in \mathbb{R}^n \\ \text{to maximize} & \Phi_{avg}(\boldsymbol{b}, \boldsymbol{p}) \\ \text{subject to} & g_j(\boldsymbol{b}, \boldsymbol{p} + \boldsymbol{z}^p) \leqslant 0, j = 1, ..., r \\ \text{and to} & \boldsymbol{b}_L \leqslant \boldsymbol{b} \leqslant \boldsymbol{b}_U \end{array} \tag{18}$$

where $\boldsymbol{p}$ is the uncertain parameters vector, z^p indicates the noise vector of $\boldsymbol{p}$, g_j is the j-th inequality constraint, $\boldsymbol{b}$ is the design parameters vector (see Eq. 1), $\boldsymbol{b}_L$ and $\boldsymbol{b}_U$ are the lower and upper boundary of the design parameters. The uncertain parameters vector is:

$$\boldsymbol{p} = [a, n] \tag{19}$$

The insertion altitude h_{ij} must be evaluated for each one of the nine possible couples a_i, n_j, when a robust-based solution is sought. Thus, unlike the evaluation of Φ which is quite cheap (just one call to the numerical model is required), nine execution of the trajectory optimization sub-model must be performed to compute Φ_{avg}. Fortunately, $\mu = \mu(\boldsymbol{b})$ is unaffected by uncertainty. In analogy to the definition of $\Phi(\boldsymbol{b})$, an average altitude violation Δ_{avg} is computed as:

$$\Delta_{avg} = \sum_{ij} p_i p_j \max_{ij}(0, h^* - h_{ij}) \tag{20}$$

where a binomial distribution is assumed ($p_1 = p_3 = 0.25$ and $p_2 = 0.5$) and the robust-based merit function is defined as:

$$\Phi_{avg}(\boldsymbol{b}, \boldsymbol{p}) = \mu - k \max(0, \Delta_{avg}) \tag{21}$$

It is worth noting that in nominal conditions (i.e., when a and n assume their reference values) the amount of oxidizer and fuel boarded in the upper stage are such that they end at the same moment (at engine burnout) and the maximum altitude is reached for that given payload mass μ, which is fixed by the launcher design (i.e., by $\boldsymbol{b}$). On the contrary, when uncertainties are introduced, the actual regression rate may deviate from the nominal one, leading to variations in propellant consumption. On one hand, if the regression rate exceeds the nominal value, all the fuel may burn while unburned oxidizer remains in the tank. On the other hand, if the regression rate is lower than nominal, the oxidizer may be fully consumed while unburned fuel remains in the combustion chamber. This surplus in oxidizer or fuel mass required to reach (at least) target orbit in off-nominal condition results in a payload penalty when robustness in the solution is sought. Obviously, the robust-based optimal designs can reach orbits higher than the target when nominal regression rate occurs. In this context, it is assumed that, if the upper stage is able to insert the payload at least in the target orbit for every uncertain parameters combination, it is also able to reach exactly that orbit for every uncertain parameters combination.

In this paper, just two uncertain parameters are taken into account to reduce the complexity of the robust-based model. This choice allows for more agile parametric studies in the building process of the surrogate model and computationally affordable optimization runs. The readers must keep in mind that the main focus of this paper is to explore the feasibility of surrogate approaches to robust-based optimization. The author analyzed more complex robust-based models in Ref. [20]. The applicability of surrogate models in those scenarios will be investigated in future works.

III. Surrogate Model

The structure of the surrogate model proposed in this work is shown in Fig. 1. The main components are:

- Deterministic Evaluation Tool (DET). This tool provides the deterministic solutions (i.e., payload mass μ and insertion altitude h_{DET}) given the input parameters $\boldsymbol{b}$ when uncertain parameters assume their nominal values. The computational cost of a call to the DET is equal to one: the numerical model (described in Sec.II.A) must perform engine design evaluation and trajectory optimization just once to compute $\Phi(\boldsymbol{b}$. After the deterministic evaluation, the surrogate model discards all candidate solutions such that $h_{DET} \leq 500$ km. Two are the rationales behind this filtering process: first, a candidate solution unable to have an insertion altitude close to the target one (i.e., 700 km) even in nominal conditions is unlikely to be a robust solution, and second, the altitude filter ensures that each candidate solution evaluated by the classifier and the neural network falls inside their training ranges;
- Binary Classifier. The classifier receives the six input parameters in $\boldsymbol{b}$ by the DET. It is trained to identify whether or not the corresponding candidate solution will be a feasible robust solution, i.e., a robust solution in which all off-nominal insertion altitudes $h_{i,j}$ are greater than zero. On one hand, feasible solutions are passed to the neural network to predict the robust-based performance index Φ_{avg}. On the other hand, candidate solutions that represent rocket crashes or very low insertion altitudes for one or more off-nominal conditions are here discarded, due to their unsatisfactory performance from a robust point of view. The classifier output is 1 (i.e., the candidate solution is expected to be feasible) when its confidence in the prediction is greater than or equal to 50%, otherwise, the output is 0, and the candidate solution is discarded;
- Neural Network. Ten are the inputs of the neural network: the six input design parameters of vector $\boldsymbol{b}$, the two uncertain parameters of vector $\boldsymbol{p}$ (i.e., a and n), along with the two outputs of the DET (μ and h_{DET}). The neural network has a single output: the expected insertion altitude for the candidate solution when an off-nominal condition occurs $(h_{i,j})_{pred}$, as specified by the values of the uncertain parameters a_i and n_j. Thus, one can obtain a prediction of the robust-based solution $\Phi_{avg}(\boldsymbol{b}, \boldsymbol{p})$ by calling the neural network for each one of the nine off-nominal a_i, n_j combination without further calls to the full numerical model.

Table 2 Binary classifier features and settings.

Input features	6
Database size	2500
Validation set	25%
Test set	20%
Hidden layer	3
Neurons	256 - 256 - 64
Dropout	0.40
L2 regularization	0.01

In order to build the training database, a RoBust evaluation Tool (RBT) is used. This tool provides the robust-based solutions (i.e., the merit functions Φ_{avg}) as described in Sec.II.A. The computational cost of a call to the RBT is nine times a call to the DET: the full numerical model must perform nine trajectory optimizations to evaluate Φ_{avg}, one for each uncertain parameters combination which may occur in off-nominal conditions. For this reason, the surrogate model makes use of the RBT just a few times (ideally, no times at all) to check the most promising solutions found at the end of the optimization runs.

In the following, a description of the main features of the used classifier and neural network will be provided. The building of these prediction tools has a relevant cost in terms of calls to the full numerical model that must be taken into account, as will be discussed.

A. Binary Classifier

The proposed surrogate model takes advantage of a binary classifier to discard candidate solutions that are expected to result in zero-altitude insertion orbit for at least one off-nominal condition. The classifier used in this paper has been developed in Python using the TensorFlow environment[32]. The training database is generated by performing a Latin Hypercube Sampling (LHS) of the design space. The RBT is used in this process to compute the required $h_{i,j}$ values. The features are the six design input parameters of $\boldsymbol{b}$ and the corresponding label is 1 when all the off-nominal conditions result in non-zero insertion altitudes, 0 otherwise. A brief parametric study is performed to tune up the classifier. Three database sizes are initially considered: small-sized database of 1000 points, medium-sized of 2500 points and large-sized of 4000 points. The corresponding computational cost is equal to 9000, 22500 and 36000 full numerical model calls, respectively. The model architecture is adjusted in terms of the number of layers and neurons accordingly to the size of the database to have good performance. Unfortunately, the small database is unable to provide the required prediction accuracy, whereas the large one did not significantly improve the performance of the classifier with respect to the medium-sized one. Thus, the 2500-point database is chosen for subsequent tuning tests. Fixed the database size, a tuning of the model architecture is carried out. ReLu (REctified Linear Unit) activation function is used in all the hidden layers, whereas sigmoid activation function is used in the output layer. Both two and three hidden layers are considered, with a number of neurons per layer ranging from 16 to 256. To prevent the overfitting phenomenon and to improve the generalization of the model, dropout and L2 regularization are implemented in the script. Tab.2 reports the main features of the tuned classifier used in the surrogate model during the optimization.

The use of the binary classifier in the proposed surrogate model requires to have both an high accuracy and to reduce the number of false positives. In fact, false negatives are conservative from an optimization point of view: the surrogate model discards an actual good candidate solution on the basis of a wrong classifiers output resulting in an underestimation of its performance, with respect to actual ones. On the other hand, a false positive is able to jeopardize the whole optimization run: the surrogate model is tricked into considering as feasible a candidate solution which is actually unfeasible on the basis of a wrong classifier's output. The candidate solution is then incorrectly passed to the neural network to evaluate the robust-based solutions. The neural network, in turn, is unlikely to detect zero-altitude robust-based cases due to the limitations of its training dataset, as will be discussed in the following sections, resulting in a performance overestimation for the candidate solution. For this reason, a Weighted Binary CrossEntropy (WBCE) loss function is adopted:

$$WBCE(y_{true}, y_{pred}) = -\left[y_{true}\log(y_{pred}) + w_{FP}(1 - y_{true})\log(1 - y_{pred})\right] \tag{22}$$

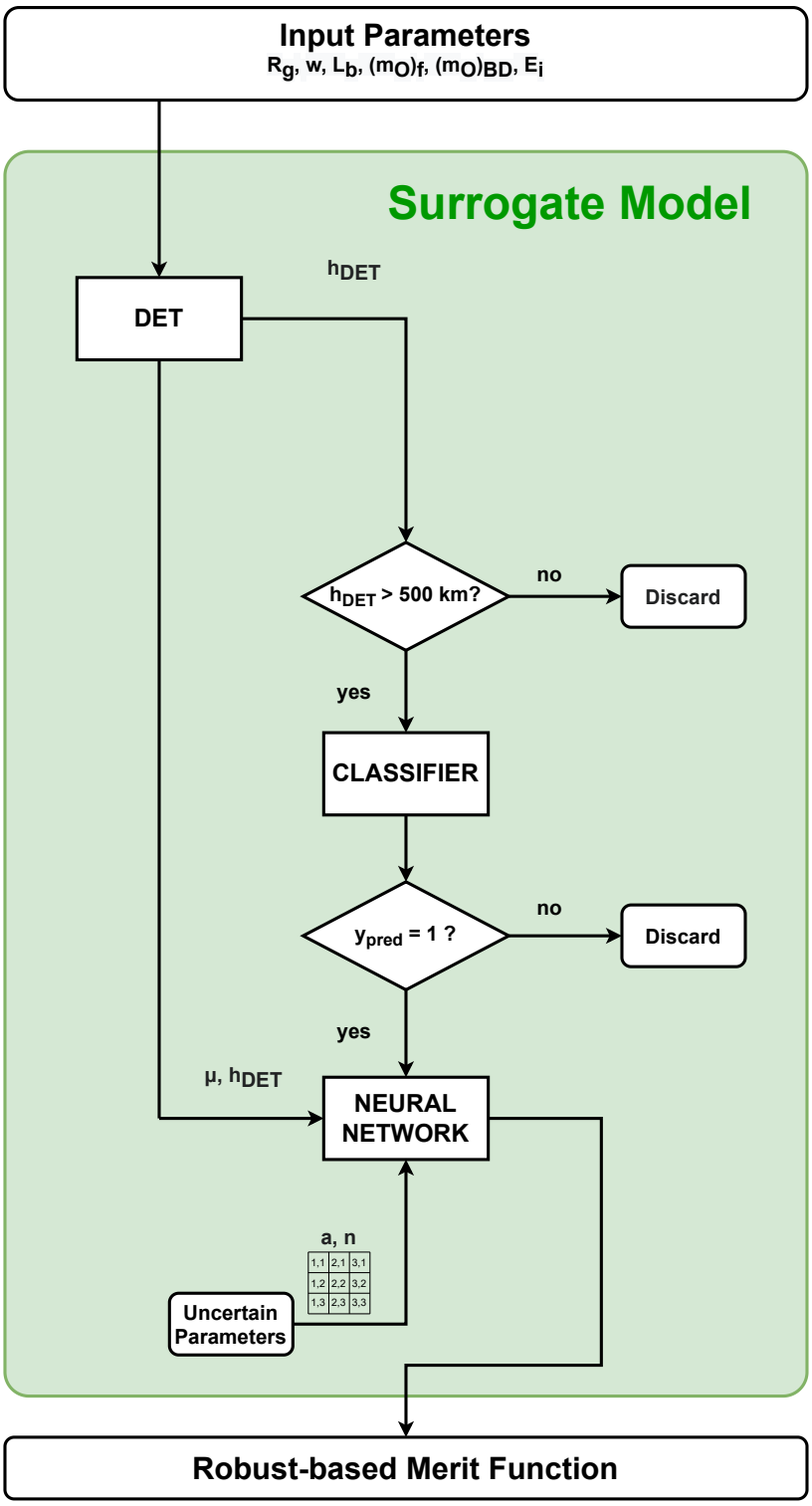


Fig. 1 Flow chart of the surrogate model.

where y_{true} is the true label, y_{pred} is the predicted label confidence, and w_{FP} is the false positive weight. A value of $w_{FP} = 5$ is chosen to penalize false positives and mitigate the surrogate's tendency to overestimate robust-based performance.

Table 3 reports the performance metrics for the tuned binary classifier computed as follows on the testing database (TN = true negatives, FN = false negatives, TP = true positives, and FP = false negatives):

$$Precision = \frac{TP}{TP + FP} \tag{23}$$

$$Recall = \frac{TP}{TP + FN} \tag{24}$$

$$Specificity = \frac{TN}{TN + FP} \tag{25}$$

The high value of recall means that the trained classifier is expected to be able to detect at least 90% of all positive instances (i.e., feasible robust-based solutions), while the high value of recall means that only about 0.5% of the detected positive instances are false positives. In the end, the high value of specificity means that the classifier can correctly identify true negatives while avoiding false positives. Various configurations of the classifier are explored during the tuning process, but it is not possible to improve both precision and recall simultaneously.

Table 3 Tuned binary classifier performance metrics.

Recall	0.900
Precision	0.995
Specificity	0.986

Figure 2 reports the prediction confidence distribution for the testing dataset. One can notice that more than 80% of the confidences are minor to 2% or major to 98%, meaning that the classifier can predict with quite high accuracy whether or not a candidate solution is feasible from a robust-based point of view. In the proposed surrogate model, the acceptance threshold for the solutions is set to the most typical value of confidence, which is 0.5. In other words, $y_{pred} = 1$ if its confidence is major or equal to 0.5, and $y_{pred} = 0$ otherwise. Higher thresholds (e.g., 0.7) are tested to reduce the occurrence of false positives, but they negatively impact the classifier's performance metrics. In the end, the candidate solutions whose predicted label is $y_{pred} = 1$ (i.e., feasible) are passed to the neural network to evaluate the insertion altitude in off-nominal conditions.

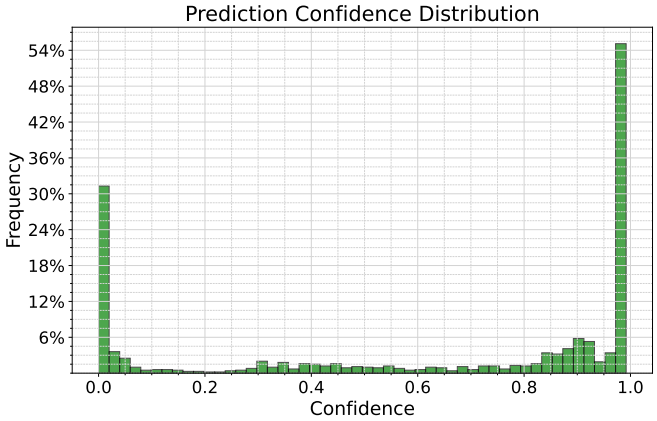


Fig. 2 Prediction confidence distribution.

B. Neural Network

In the proposed surrogate model, the neural network is devoted to the computation of the altitude reached by the launcher when the uncertain parameters, i.e., the regression rate correlation coefficient a and exponent n, assume off-nominal values. In the conventional robust-based optimization approach (i.e., without the surrogate model), these

Table 4 Neural network features and settings. The dropout is used only before the output layer.

Input features	10
Database size	20700
Validation set	20%
Test set	20%
Hidden layer	3
Neurons	256 - 128 - 64
Dropout	0.50
L2 regularization	0.01

altitude evaluations are performed by the RBT, increasing the number of full numerical model calls and the computational cost of the simulation.

The proposed neural network has been developed in Python using the TensorFlow environment. The input parameters of the network are ten: the six design input parameters (R_g, w, L_b, $(m_O)_f$, $(m_O)_{BD}$, and E_i), the two uncertain parameters (a, and n), and the output of the DET, namely the payload mass μ and the insertion altitude in nominal condition h_{DET}. The output parameter is the prediction for $h_{i,J}$, here called h_{pred}. The available database is the same already used for the binary classifier and built using the RBT. The number of different samples is now $2500 \cdot 9 = 22500$, because for each design input parameters combinations one has nine different insertion altitudes $h_{i,j}$, one for each uncertain parameters couple. Preliminary neural network training runs showed that the presence of zero-altitude data points in the database heavily worsened the prediction performance of the net for the non zero-altitude points. For this reason, the binary classifier is introduced in the surrogate model to filter these solutions (see Sec. III.A) prior to the call to the neural network. Thus, the actual available database for the training of the net is roughly 10% smaller (about 20000 samples).

Exponential Linear Unit (ELU) activation function is used in all the network layers. Both two and three hidden layer are considered, with a number of neurons per layer ranging from 16 to 256. In order to prevent the overfitting phenomenon and to improve the generalization of the model, dropout and L2 regularization are implemented in the script. Tab.4 reports the main features of the tuned neural network used in the surrogate model during the optimization. Due to the structure of the surrogate model and the formulation of Φ_{avg}, underestimation and overestimation of h_{pred} have different effects on the optimization outputs, as explained by the following examples.

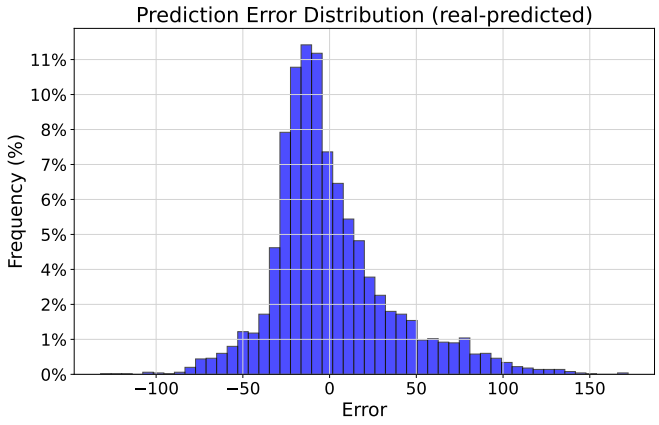
In the case of underestimation, the predicted altitude h_{pred} is lower than the real one h_{real}, and the prediction error $h_{real} - h_{pred}$ is positive. On one hand, if $h_{pred} \geq h^* = 700$ km (that is the target altitude), then $h_{real} > h_{pred} \geq h^*$ and their corresponding contributions to the robust-based merit function are both zero, because the altitude violation with respect to the target altitude is null. In this case, the surrogate model mimics the behavior of the numerical model. On the other hand, if $h_{pred} < 700$ km and $h_{real} \geq 700$ km, their contributions to the robust-based merit function are different, but the predicted one is conservative being negative (if $h_{pred} < 700$ km, altitude violation is positive and penalizes the optimization merit function). Thus, the surrogate prediction is always conservative if the prediction error is positive, i.e., $h_{pred} \leq h_{real}$;

In the case of overestimation, the predicted altitude h_{pred} is higher than the real one h_{real}, and the prediction error $h_{real} - h_{pred}$ is negative. On one hand, if $h_{pred} < h^* = 700$ km, then $h_{real} < h_{pred}$ and their corresponding contributions to the robust-based merit function are different but both negative. The high value of $k = 20$ in the optimization merit function is able to penalize enough these candidate solutions even if $h_{pred} > h_{real}$. On the other hand, if $h_{pred} \geq h^* = 700$ km but $h_{real} < h^* = 700$ km, the predicted contribution to the robust-based merit function is null (because altitude violation is zero), whereas the actual contribution is negative (because altitude violation is positive). In this latter case, the optimization algorithm is tricked into prefer a candidate solution which is robust according to the surrogate model but in reality should be penalized, because at least one insertion altitude actually does not match the mission target. Thus, the surrogate prediction may be not conservative if the prediction error is negative, i.e., $h_{pred} > h_{real}$, and the optimization results may be compromised;

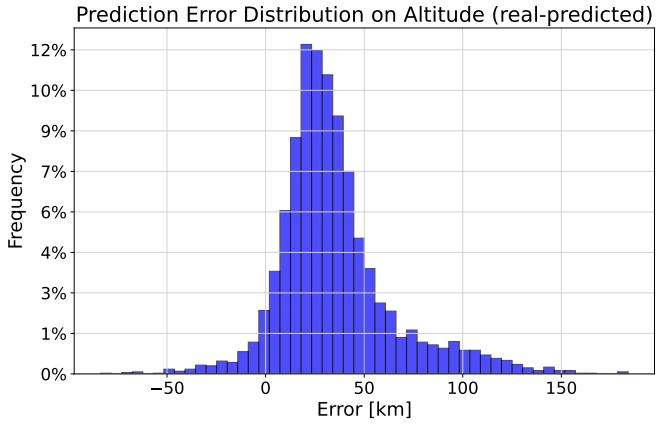
For this reason, a custom Asymmetric Loss Function (ALF) is adopted:

$$ALF = \frac{1}{N}\sum_{i=1}^{N}\begin{cases} w_{os}\cdot(h_{pred}-h_{real})^2 & \text{if } h_{pred} > h_{real} \\ (h_{pred}-h_{real})^2 & \text{otherwise} \end{cases} + w_{me}\cdot\max_{i=1}^{N}|h_{pred}-h_{real}| \tag{26}$$

where N is the number of available data points. The weight w_{os} is used to penalize the overestimation during the training of the neural network and force the prediction error distribution towards positive values. In addition, a second weight w_{me} is introduced in the function definition to narrow the error distribution penalizing excessive underestimation in the altitude predictions, otherwise the surrogate model may be too conservative, with respect to the numerical model, resulting in sub-optimal robust-based solutions. A parametric study is carried out to balance the weights in the ALF. Figure 3 shows the prediction error distributions obtained with a symmetrical loss function and using the ALF with $w_{os} = 3$ and $w_{me} = 0.15$. These latter are the values then used in the optimization process.



(a) Error distribution, symmetrical loss function: $w_{os} = 1$ and $w_{me} = 0$.



(b) Error distribution, ALF: $w_{os} = 3$ and $w_{me} = 0.15$.

Fig. 3 ALF weights comparison. Negative prediction errors are more common than positive ones in (a) when the symmetrical loss function is used in the neural network training. On the contrary, the number of negative errors (i.e., performance overestimation) is reduced in (b) thanks to the use of the ALF.

IV. Optimization

The optimization is carried out by means of a Particle Swarm Optimization (PSO) algorithm developed in Python to easily manage the interface with the classifier and the neural network. The PSO algorithm is based on an in-house fortran code finely tuned by PoliTO's Aerospace Propulsion Group[20, 33]. The algorithm embeds Trelea's improvements to PSO[34] aiming at an optimal tradeoff of exploration and exploitation of the design space. The main features of the algorithm are here summarized.

An high initial inertia weight $w_{max} = 0.95$ is used to favor a fast exploration of the whole design space, whereas a

lower final inertia weight $w_{min} = 0.5$, allows for a fine exploitation of the most promising portions of the design space once nice solutions are found out. During the optimization runs, the inertia weight at the i-th iteration w^i is adjusted as function of the maximum number of iterations N_{max}, as:

$$w^i = w_{max} - (i/N_{max})(w_{max} - w_{min}) \tag{27}$$

Along with inertia weight, the PSO algorithm employs three acceleration components to update particles' velocity and position, namely cognitive acceleration c_1, social acceleration c_2, and group acceleration c_3. c_1 is the single particle's self-confidence, which attracts the k-th particle to its own known best position $p_{k,best}$. c_2 is the social attraction, which attracts the particles to the global best position found out by the whole swarm g_{best}. c_3 is the group acceleration, which attracts the single particles towards the center of mass of the swarm's individual best positions, here called p_{avg}, helping the balance of exploration and exploitation. The proposed algorithm embeds variables accelerations c_1, c_2, and c_3, which are updated iteration by iteration analogously to the inertia weight w^i, as:

$$c^i = c_{max} - (i/N_{max})(c_{max} - c_{min}) \tag{28}$$

At the beginning of the optimization run, exploration is favored by $c_{1,max} = c_{2,max} = 2.5$, and $c_{3,max} = 1$. At the end of the optimization, exploitation is sought, thus $c_{1,min} = c_{2,min} = c_{3,min} = 0.5$.

The velocity of the k-th particle v_k is updated throughout the optimization runs as:

$$v_k = w^i \cdot v_k + c_1^i \cdot r_1 \cdot (p_{k,best} - p_k) + c_2^i \cdot r_2 \cdot (g_{best} - p_k) + c_3^i \cdot r_3 \cdot (p_{avg} - p_k) \tag{29}$$

where p_k is the k-th particle position, w^i, c_1^i, c_2^i, and c_3^i are the values at the i-th iteration and are updated at each iteration until $i = N_{max}$ is reached. r_1, r_2, and r_3 are randomly generated numbers in the range [0,1] which are useful to introduce stochasticity in the algorithm at each iteration.

In addition, before updating the particles positions, the velocities are clipped to $v_{max} = 0.25\,(\boldsymbol{b}_U - \boldsymbol{b}_L)$, where $\boldsymbol{b}_U$ and $\boldsymbol{b}_L$ are the upper and the lower bound for the optimization parameters, respectively. It is worth noting that the optimization bounds must be the same used to build the training database for the surrogate model, otherwise, the predictions may be unreliable during optimization runs. The introduction of v_{max} is useful to maintain the right balance between exploration (searching broadly) and exploitation (refining near the good solutions). A graphical representation of the PSO algorithm is provided in Fig.4.

V. Results and Conclusions

In this Section, the optimization results and conclusions will be presented. In order to have a reference, first the optimization is carried out using the RBT to evaluate the merit function inside the PSO algorithm. The optimized robust-based solution is called ROB in tables and figures for short. Figure 5 shows the values of Φ_{avg} throughout the reference optimization run for the global best solution (solid line) and the best solution found out at each iteration (dotted line). It is worth noting that at the beginning of the run (first fifty iterations) the global best is rapidly improving, whereas the best in each iteration is not monotonically increasing, meaning that the setup of accelerations and inertia favors a broader swarm and the exploration of the design space. Once promising solutions are found (Φ_{avg} about 2025 kg), the global best and the best in each iteration become close to each other, but the improvement rate is consistent until the end of the run, meaning that now exploitation is sought. In this reference optimization run, 20 particles are used and the maximum number of model calls is fixed to 60000, corresponding to about 24 hours of execution on a 20-core parallel computing machine (20x Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz). The global best solution stabilizes on a value of about $\Phi_{best} = 2053$ kg when the full numerical model is used.

Figure 6 shows a typical optimization run results employing the surrogate model to evaluate Φ_{avg}. Here, 20 particles are used and the maximum number of numerical model calls is fixed to 6000, that is 1/10 of the reference case. It is worth noting that the number of iterations is similar to the reference run with the RBT because here the computational cost is 1/9 than before: the numerical model is called just one time by the DET for each particle in each iteration when the surrogate model is used, whereas the RBT calls the numerical model nine times (one for each uncertain parameters combination) for each particle at each iteration. In addition, the results show that the global best solutions found out with the surrogate model, here called Φ_{sur}, stabilizes after 3000 model calls on a value between 2035 and 2045 kg (depending on the specific optimization run considered), which corresponds to Φ_{avg} between 2030 and 2040 kg when checked *a posteriori* with the RBT. Thus, the surrogate model is able to effectively predict the optimized payload

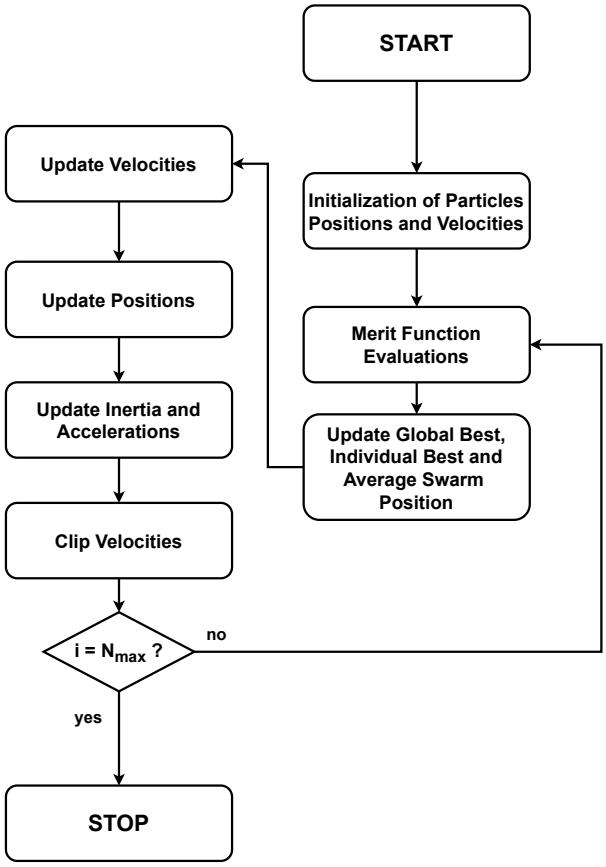


Fig. 4 PSO algorithm flow chart.

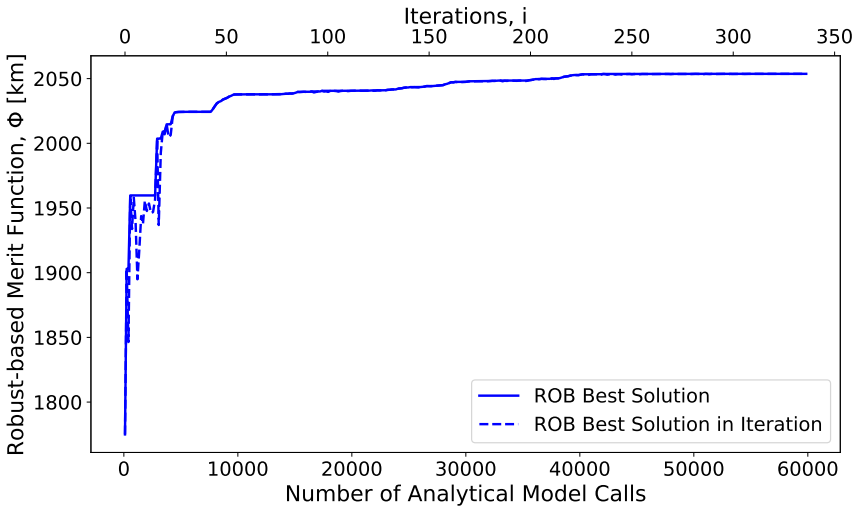


Fig. 5 Reference optimization run results obtained with the RBT.

mass with an error margin within 1%. Tables 5-7 report design summary, mass budget, performance, and other useful parameters for the best solutions found out.

The cost of building the training database must be considered for a fair assessment of the overall computational

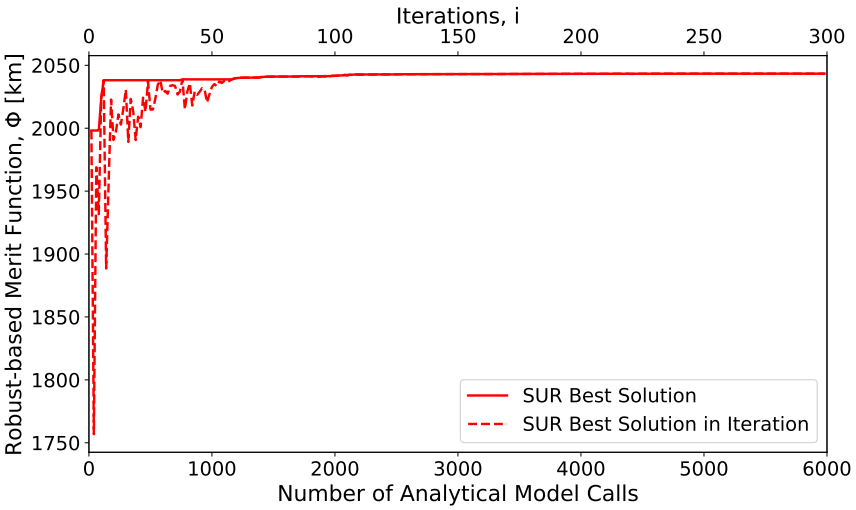


Fig. 6 Optimization run results obtained with the surrogate model.

Table 5 Design summary: ROB vs. surrogate best solution.

Design	R_g	w	L_b	$(m_O)_f$	$(m_O)_{BD}$	E_i
	m	m	m	kg	kg	-
ROB	0.592	0.298	4.350	7425	3256	15.93
Surrogate	0.591	0.298	4.396	7403	3297	16.22

Table 6 Mass budget and performance summary: ROB vs. surrogate best solution.

Design	μ	m_P	m_O	m_F	m_{dry}	Φ_{avg}	h_{avg}	Δ_{avg}
	kg	kg	kg	kg	kg	kg	km	km
ROB	2054	10812	7425	3387	1657	2053.7	890	0.0
Surrogate	2043	10816	7403	3413	1663	2036.2	909	0.5

Table 7 Other parameters: ROB vs. surrogate best solution.

Design	F_i	R_{th}	R_e	L	D	L/D	m_{cc}	m_t	m_{nz}	m_{case}	$m_{He}+m_{ves}$
	kN	m	m	m	m	-	kg	kg	kg	kg	kg
ROB	23.6	0.208	0.831	9.720	1.914	5.08	160	300	321	164	177
Surrogate	23.4	0.207	0.835	9.774	1.914	5.11	161	299	324	164	179

expense when employing a surrogate model. In this study, 2500 data points are used to train both the binary classifier and the neural network. This corresponds to 22500 numerical model calls, which, combined with the 3000 calls required for optimization, results in a total cost of 25500 model calls, comparable to the cost of optimization using the RBT. However, this represents a worst-case scenario, as training data may already be available from prior analyses on similar topics. In such cases, the surrogate model could significantly reduce the computational cost of optimization, albeit at the expense of a 1% payload mass reduction.

It is worth noting that the surrogate model is not able to correctly identify the ROB solution as a good candidate solution. In particular, the binary classifier output is 0 (i.e., unfeasible solution) and the neural network altitudes range from 400 km to 770 km, with just three $h_{i,j} \geq h^* = 700$ km. A possible reason behind this behavior seems to be related to a bad sampling of the solution space during the building of the database. Figure 7 reports examples of the design space (web thickness w vs. grain length L_b, on the left) and of the solution space ($h_{1,2}$ vs. $h_{3,3}$, on the right). The black

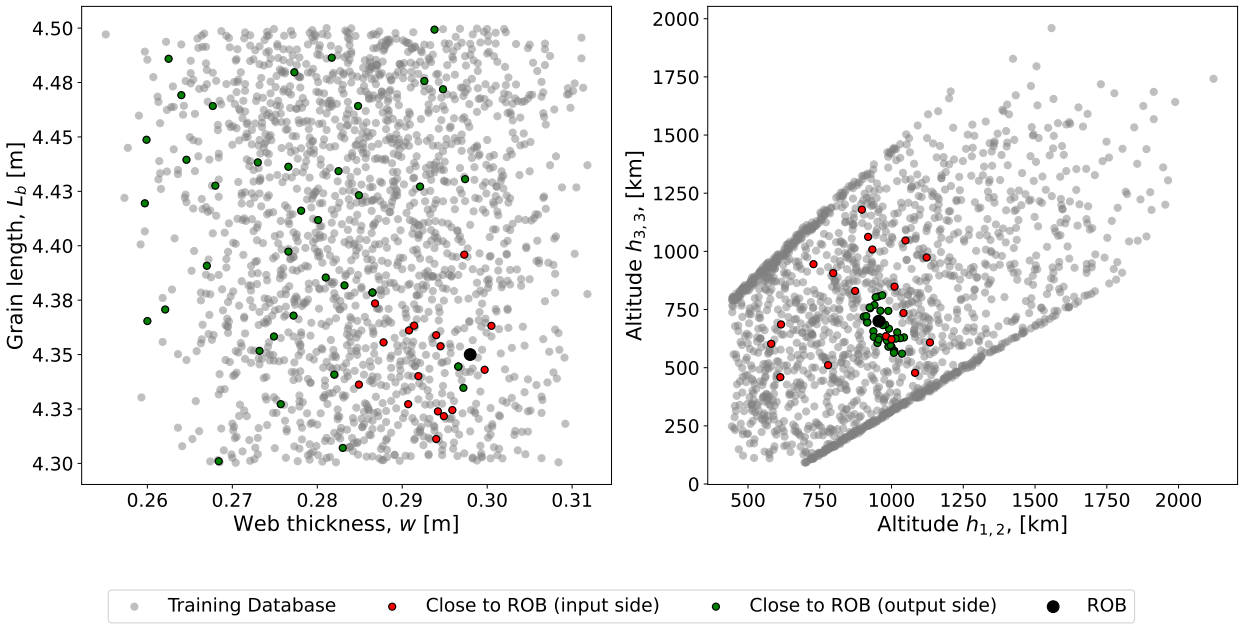


Fig. 7 Design and solution space sampling.

dot represents the ROB optimal design and its corresponding solution. The red dots in the plots indicate the data points closest to ROB in the design space, along with their corresponding solutions in the solution space. The green dots represent data points whose solutions are the closest to ROB solution in the solution space, along with their associated designs in the design space. After a scaling of the data, Euclidean distances and Wasserstein distances are used to identify the nearest data points in the design and solution spaces, respectively. These plots illustrate how the surrogate model attempts to predict the ROB solution by gathering information from the nearest points in the design space (red dots), whose corresponding solutions are, on the contrary, quite distant from the ROB solution in the solution space. On the other hand, the solutions close to ROB are associated with input combinations in the design space that are more dispersed and far away from the ROB design. In addition, this behavior suggests that the robust-based merit function may be discontinuous near the optimum. For these reasons, future developments of the proposed approach will focus on improving the sampling of the solution space, aiming to close the gap between actual robust performance and surrogate predictions.

Acknowledgments

I acknowledge ISCRA for awarding this project access to the LEONARDO supercomputer, owned by the EuroHPC Joint Undertaking, hosted by CINECA (Italy). The text was improved for readability using ChatGPT, an AI language model developed by OpenAI (OpenAI, 2024).

References

[1] "Hydrazine ban could cost Europe's space industry billions, SpaceNews," , 2017. URL `https://spacenews.com/hydrazine-ban-could-cost-europes-space-industry-billions/`, [Online; accessed 3-May-2024].

[2] Casalino, L., Ferrero, A., Masseni, F., and Pastrone, D., "Emission-Driven Hybrid Rocket Engine Optimization for Small Launchers," *Aerospace*, Vol. 9, No. 12, 2022, p. 807.

[3] Karabeyoglu, M., Altman, D., and Cantwell, B. J., "Combustion of Liquefying Hybrid Propellants: Part 1, General Theory," *J. Propuls. Power*, Vol. 18, No. 3, 2002, pp. 610–620. https://doi.org/10.2514/2.5975.

[4] Casalino, L., Ferrero, A., Folcarelli, L., Masseni, F., Muscará, L., Pastrone, D., Luisa Frezzotti, M., Cretella, A., Carmine Pellegrini, R., and Cavallini, E., "Multiphysics Modeling for Combustion Instability in Paraffin-Fueled Hybrid Rocket Engines," *Journal of Spacecraft and Rockets*, 2024, pp. 1–17.

[5] Zavoli, A., Maria Zolla, P., Federici, L., Tindaro Migliorino, M., and Bianchi, D., "Surrogate neural network for rapid flight performance evaluation of hybrid rocket engines," *Journal of Spacecraft and Rockets*, Vol. 59, No. 6, 2022, pp. 2003–2016.

[6] Messinger, T. L., Corbiell, M. S., and Johansen, C. T., "Optimization and parametric studies of two-stage-to-orbit liquid oxygen/paraffin hybrid rocket vehicles," *Aerospace Science and Technology*, Vol. 140, 2023, p. 108495.

[7] Zolla, P., Zavoli, A., Migliorino, M. T., and Bianchi, D., "Integrated Optimization of a Three-Stage Clustered Hybrid Rocket Launcher using Neural Networks," *AIAA SCITECH 2024 Forum*, 2024, p. 1184.

[8] Faenza, M., Boiron, A. J., Haemmerli, B., and Verberne, C. J., "The Nammo Nucleus Launch: Norwegian Hybrid Sounding Rocket over 100km," *AIAA Propulsion and Energy 2019 Forum*, 2019, p. 4049. https://doi.org/https://doi.org/10.2514/6.2019-4049.

[9] Bérend, N., Gauvrit-Ledogar, J., Perrel, F., Oswald, J., Diez, E., Dupont, C., Karl, C., Romano, D. G., Bastien, H., and Searle, T., "ALTAIR Semi-Reusable Air-Launch System -Current Design and Status of Flight Experiments," *8th European Conference for Aerospace Sciences (EUCASS)*, Madrid, Spain, 2019, pp. 1–16. URL https://hal.archives-ouvertes.fr/hal-03272113.

[10] Timmermans, L., Bernving, N., Van Kleef, A., Haemmerli, B., Kuhn, M., Muller, I., Petrozzi, M., and Psoni, G., "Small Innovative Launcher for Europe: Results of the H2020 Project SMILE," *2019 IAC*, 2019, pp. 1–13.

[11] "European Union's Horizon 2020: ENVOL," , 2022. URL https://envol-h2020.eu/, [Online; accessed 27-May-2022].

[12] Schmierer, C., Kobald, M., Fischer, U., Tomilin, K., Petrarolo, A., and Hertel, F., "Advancing Europe's Hybrid Rocket Engine Technology with Paraffin and LOX," *Proceedings of the 8th European Conference for Aeronautics and Space Sciences*, 2019, pp. 1–9. https://doi.org/https://www.eucass.eu/doi/EUCASS2019-0682.pdf.

[13] "NewSpace Index Website," , 2024. URL https://www.newspace.im/launchers/, [Online; accessed 3-May-2024].

[14] Casalino, L., Masseni, F., and Pastrone, D., "Hybrid Rocket Engine Design Optimization at Politecnico di Torino: A Review," *Aerospace*, Vol. 8, No. 8, 2021, p. 226. https://doi.org/10.3390/aerospace8080226.

[15] Casalino, L., and Pastrone, D., "Optimal Design of Hybrid Rocket Motors for Launchers Upper Stages," *J. Propuls. Power*, Vol. 26, No. 3, 2010, pp. 421–427. https://doi.org/10.2514/1.41856, URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-77952557900&doi=10.2514%2f1.41856&partnerID=40&md5=7a5fbb6eaeb5ca0b53448a7eef3298f0.

[16] Casalino, L., Masseni, F., and Pastrone, D., "Optimal Design of Hybrid Rocket Small Satellite Launchers: Ground Versus Airborne Launch," *Journal of Spacecraft and Rockets*, 2022, pp. 1–9.

[17] Taguchi, G., and Chowdhury, S., *Robust Engineering: Learn How to Boost Quality While Reducing Costs & Time to Market*, McGraw Hill Professional, 1999.

[18] Casalino, L., Masseni, F., and Pastrone, D., "Comparison of Robust Design Approaches for Hybrid Rocket Engines," *53rd AIAA/SAE/ASEE Joint Propulsion Conference*, 2017. https://doi.org/10.2514/6.2017-4642, aIAA-2017-4642.

[19] Casalino, L., Masseni, F., and Pastrone, D., "Deterministic and Robust Optimization of Hybrid Rocket Engines for Small Satellite Launchers," *J. Spacecr. Rocket.*, 2021. https://doi.org/10.2514/1.A35007.

[20] Casalino, L., Masseni, F., and Pastrone, D., "Uncertainty Analysis and Robust Design for a Hybrid Rocket Upper Stage," *J. Spacecr. Rocket.*, Vol. 56, No. 5, 2019, pp. 1424–1431. https://doi.org/10.2514/1.A34422, URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85073098326&doi=10.2514%2f1.A34422&partnerID=40&md5=f5f3023a41b702d749a226dd1025dc72.

[21] Arianespace, "VEGA, User's Manual, Issue 4 - Revision 0," , 2014. URL http://www.arianespace.com/wp-content/uploads/2015/09/Vega-Users-Manual_Issue-04_April-2014.pdf, [Online; accessed 2-July-2019].

[22] Sutton, G. P., and Biblarz, O., *Rocket propulsion elements*, John Wiley & Sons, 2001.

[23] McBride, B. J., Reno, M. A., and Gordon, S., "CET93 and CETPC: An Interim Updated Version of the NASA Lewis Computer Program for Calculating Complex Chemical Equilibria with Applications," 1994. NASA TM-4557.

[24] Ellis, R., "Solid Rocket Motor Nozzles-NASA Space Vehicle Design Criteria (Chemical Propulsion)," 1975. NASA SP-8115.

[25] Bianchi, D., and Nasuti, F., "Numerical Analysis of Nozzle Material Thermochemical Erosion in Hybrid Rocket Engines," *J. Propuls. Power*, Vol. 29, No. 3, 2013, pp. 547–558. https://doi.org/10.2514/1.B34813.

[26] Brown, C. D., *Spacecraft Propulsion*, AIAA Education Series, 1992. https://doi.org/10.2514/4.862441.

[27] Barrere, M., Jaumotte, A., Fraeijs de Veubeke, B., and Vandenkerckhove, J., *Rocket Propulsion*, Elsevier Publishing Company, 1960.

[28] Casalino, L., Masseni, F., and Pastrone, D., "Robust Design Approaches for Hybrid Rocket Upper Stage," *J. Aerosp. Eng.*, Vol. 32, No. 6, 2019. https://doi.org/10.1061/(ASCE)AS.1943-5525.0001078.

[29] Colasurdo, G., Pastrone, D., and Casalino, L., "Optimization of rocket ascent trajectories using an indirect procedure," 1995, pp. 1375–1383. https://doi.org/10.2514/6.1995-3323, URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-0347187770&doi=10.2514%2f6.1995-3323&partnerID=40&md5=d336628b5032910d1b7c5ee984b44a81.

[30] Casalino, L., and Pastrone, D., "Optimal Design and Control of Hybrid Rockets for Access to Space," *41st AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, 2005. https://doi.org/10.2514/6.2005-3547, URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85088334585&doi=10.2514%2f6.2005-3547&partnerID=40&md5=0605dc3e747a149a8ff59f023b0c62ec, aIAA-2005-3547.

[31] Park, G., Lee, T., Lee, K., and Hwang, K., "Robust Design: an Overview," *AIAA journal*, Vol. 44, No. 1, 2006, pp. 181–191. https://doi.org/10.2514/1.13639.

[32] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," , 2015. URL https://www.tensorflow.org/, software available from tensorflow.org.

[33] Sentinella, M. R., and Casalino, L., "Hybrid Evolutionary Algorithm for the Optimization of Interplanetary Trajectories," *J. Spacecr. Rocket.*, Vol. 46, No. 2, 2009, pp. 365–372. https://doi.org/10.2514/1.38440.

[34] Trelea, I. C., "The particle swarm optimization algorithm: Convergence analysis and parameter selection," *Inf. Process. Lett.*, Vol. 85, No. 6, 2003, pp. 317–325.