Projectile shape optimization using neural network surrogates learnt via exploitation and exploration

Alain Uwadukunze, Xavier Bombois, Marion Gilson, Marie Albisser

Projectile Shape Optimization using Neural Network Surrogates learnt via Exploitation and Exploration

UWADUKUNZE Alain[a,b], BOMBOIS Xavier[c], GILSON Marion[a], ALBISSER Marie[b]

[a] *Department of Control, Identification and Diagnostics, Centre de Recherche en Automatique de Nancy, CNRS, Université de Lorraine, 2 Rue Jean Lamour, Nancy, 54500, , France*
[b] *Department of Aerodynamic and Exterior Ballistics, French-German Research Institute of Saint-Louis, 5 rue du Général Cassagnou, Saint-Louis, 68300, , France*
[c] *Département AIS, Laboratoire Ampère UMR CNRS 5005, Ecole Centrale de Lyon, 36 Av. Guy de Collongue, Ecully, 69134, , France*

Abstract

Aerodynamic design of projectiles is crucial to ensure that projectiles have the best performance during their flight. Performing aerodynamic design boils down to determining the optimal values of certain design variables of the projectile as the solution of a nonlinear optimization problem involving the stability derivatives of the projectile. Solving such an optimization problem involves a heavy procedure since either costly experimental tests or computationally intensive simulations are needed to obtain the stability derivatives for different values of the design variables. In this paper, a (cost-effective) neural network surrogate model is used to model the stability derivatives. A procedure balancing exploitation and exploration is then devised to determine, based on that surrogate model, the values of the design variables for which the stability derivatives have to be evaluated to both improve the surrogate model and approach the optimal design of the projectile. This framework is applied to optimize the geometrical configuration of a rectangular finner for a classical flight scenario and is shown to perform better than the classical *Bayesian Optimization* approach.

Keywords:
Identification of static functions, Neural Network, Optimization, Aerospace Applications, Machine Learning, Experimental Design

1. Introduction

In many aerospace applications, the flight behaviour highly depends on the shape, the dimension, the materials, ... of the considered flying vehicles. Consequently, the flight behaviour can be optimized by adjusting these quantities. In the aerospace literature, such a problem is known as aerodynamic design or aerodynamic optimization (see, e.g., Driver and Zingg (2007))). In this paper, we will perform such task for (gun-launched) projectiles. In particular, the geometrical configuration of a rectangular finner (a classical type of projectiles (Dupuis and Hathaway, 1997)) will be optimized to obtain an optimal flight behaviour for a flat fire trajectory, i.e., for the case where the finner is launched with an initial total angle of attack equal to zero. The flight behaviour will be here deemed optimal when the projectile reaches its target in a minimum time.

As will be shown in the sequel, this aerodynamic optimization problem can be formulated as a constrained optimization problem having as decision variable the geometrical configuration X of the projectile and involving a number of the so-called stability derivatives of the projectile (which are functions of the geometry X of the projectile and of its Mach number M (McCoy, 1999; Anderson and Bowden, 2005)). Our contribution will be to show that learning techniques balancing exploration and exploitation along with the identification of a neural network based on available data can be beneficial for such aerodynamic optimization problem.

The constrained optimization problem described in the previous paragraph is a nonlinear optimization problem involving the static nonlinear function $F(X, M)$ relating X and M on the one hand and the stability derivatives on the other hand. In the literature, two approaches are generally used to determine the stability derivatives for a given configuration, i.e., to evaluate $F(X, M)$. The first approach is an experimental one involving wind-tunnel experiments or free flight tests (Dupuis and Hathaway, 1997; Albisser et al., 2017). The second approach is to perform high-fidelity simulations (Burnett et al., 1981; Renganathan et al., 2021). Evaluating $F(X, M)$ for a given configuration using these approaches is thus either financially costly or time-consuming. In other words, the absence of simple and accurate models for the complex function $F(X, M)$ makes it extremely difficult to address the non-linear optimization problem discussed in the previous

paragraph using a classical gradient-based algorithm. Our first contribution will be to enable a gradient-based optimization algorithm by deriving a simple model for $F(X, M)$ using an existing database. Many institutions have indeed built such databases gathering, for a number of configurations X, the value $F(X, M)$ of the stability derivatives at different Mach numbers M. For the database considered in this paper, $F(X, M)$ has been evaluated via the high-fidelity ballistic simulation tool *PRODAS*, i.e., *Projectile Rocket Ordnance Design & Analysis System* (See *www.prodas.com* for further informations).

Using the data contained in this database, a simple mathematical model $\hat{F}(X, M)$ of the static function $F(X, M)$ can be derived. This simple mathematical model (also called *surrogate model* in the literature (Forrester et al., 2008)) allows to predict $F(X, M)$ for any values of X and M in a fast and efficient way. Therefore, it can be used to solve the considered optimization problem via gradient-descent. Surrogate models can take different forms. We have here chosen a neural network for its relative simplicity, its flexibility and its efficient adaptability to multivariable problems. Moreover, neural networks have proven their efficiency in modeling the aerodynamic coefficients of diverse flying vehicles (see, e.g., Gomec and Canibek (2017); Rajkumar and Bardina (2002)).

As mentioned above, the neural network model $\hat{F}(X, M)$ can be used to solve the considered optimization problem via gradient-descent. However, if the database does not cover the space of possible configurations X in a sufficient manner, the model derived from the database may be inaccurate and the optimal geometrical configuration deduced based on that model may not have the desired properties. We therefore propose a methodology that supposes that, besides the access to the original database, we have the possibility of determining, via an experiment or via a simulation software, the values of $F(X, M)$ for a number N_{sup} of additional configurations X. Every time a new configuration is tested, the database contains an additional data point and the neural network model can therefore be updated. Given this fact, we will determine the additional configurations with the aim of improving the model accuracy (by selecting values of X in the regions that have not yet been covered by the database), while keeping in mind the objective of finding an optimal configuration (by selecting values of X that the current model predicts to be close to the optimum). These two contradictory objectives

are respectively called *exploration* and *exploitation* in the literature (Frazier, 2018).

When the surrogate model is a Gaussian Process model, i.e., a probabilistic model derived based on Gaussian prior assumptions on the unknown static function (see, e.g., Rasmussen (2003)), an optimization approach as the one presented in the previous paragraph is known as *Bayesian Optimization* (Frazier, 2018). This method has already been applied to address aerodynamic optimization problems for other flying vehicles than gun-launched projectiles (see, e.g., Jeong et al. (2005); Renganathan et al. (2021); Arnoult et al. (2020)) and to many control engineering problems (see, e.g., Roveda et al. (2020); Savaia et al. (2021); Dettù et al. (2023); Baheri et al. (2017)).

As shown in our recent contribution (Uwadukunze et al., 2023), a neural network can also be considered as surrogate model in an optimization algorithm balancing the exploration and exploitation objectives. A second contribution of the present paper is to extend these results to the case where the considered optimization problem is a *constrained* optimization problem. We also show that, for the aerodynamic optimization problem considered in this paper, the use of this modified version of Bayesian Optimization, i.e, with a neural network surrogate, yields better results than the classical version of Bayesian Optimization with a Gaussian Process surrogate.

Note finally that we have here decided to use the high-fidelity[1] ballistic simulation tool *PRODAS* to evaluate the function $F(X, M)$ at the N_{sup} configurations X which are selected by the proposed optimization algorithm. An experimental evaluation of this function via wind-tunnel experiments or free flight tests would indeed entail a financial cost which is much too important.

The sequel of this paper will be organized as follows. In Section 2, the considered aerodynamic design problem will be presented in details. In Sections 3 and 4, we will present our procedure to address this aerodynamic design problem via the identification of a neural network and via an approach bal-

[1] *PRODAS* is built upon an extensive database of experimental data, ensuring that the outcomes from its simulations closely align with those derived from wind-tunnel experiments or free flight tests.

ancing the exploration and exploitation objectives. Finally, in Section 5, the results of this optimization procedure will be presented.

2. Aerodynamic design problem

We here consider a so-called rectangular finner of caliber 28 mm with four rectangular fins. Such a projectile can be built with different geometrical configurations, i.e., for different values of the five parameters[2] X_1, X_2, ..., X_5 defined in Table 1 and represented in Figure 1 (X_1, X_3, X_4, X_5 will be expressed in calibers and X_2 in degrees).

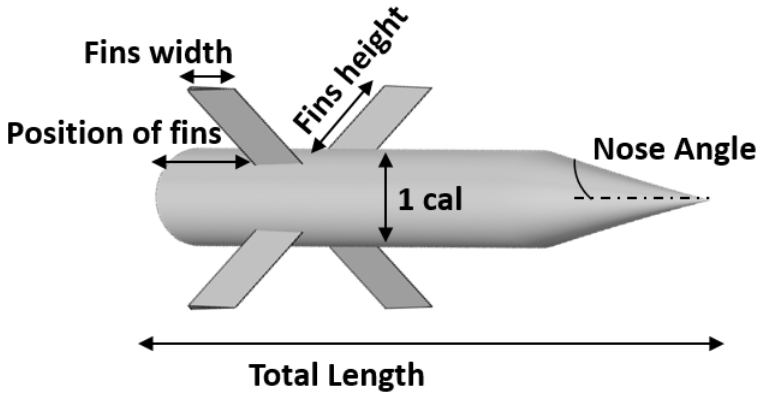

Figure 1: Finner projectile with rectangular fins

In this paper, we wish to determine the geometrical configuration of the finner in such a way that it reaches its target in a minimum time when the finner is fired with an initial velocity M_{max} (e.g., Mach 5) and then flies towards its target with a total angle of attack of zero.

To address this problem, we have to consider the stability derivatives of the finner which are coefficients associated to the forces and moments acting on the projectile during its flight. They depend on both the geometrical configuration of the finner (determined by the vector $X = (X_1, X_2, \ldots, X_5)^T$) and its velocity (the Mach number M). In particular, to minimize the time to reach the target, it is well known that one of the solutions is to minimize the drag coefficient (McCoy, 1999) of the projectile. Since we suppose that

[2]Note that we will suppose that the building material of the finner is fixed. Its center of gravity is thus entirely determined by the five parameters in Table 1.

Table 1: Main characteristics of a rectangular finned projectile

Characteristic X	Design Parameter
X_1	Total Length
X_2	Nose Angle
X_3	Fins Height
X_4	Fins Width
X_5	Position of fins

the finner flies with a total angle of attack equal to zero, the drag coefficient is equal to the *axial force coefficient first order stability derivative at zero yaw*, one of the stability derivatives (denoted C_{A_0} in the sequel) (McCoy, 1999). Since C_{A_0} not only depends on the geometrical configuration X, but also on the Mach number M, we will in fact minimize the average value of C_{A_0} over a velocity range $[M_{min}\ M_{max}]$ with M_{min} chosen in such a way that the interval $[M_{min}\ M_{max}]$ covers the possible velocities of the finner during its flight (e.g., $M_{min} = 2$ and $M_{max} = 5$).

The stability derivative C_{A_0} is not the only stability derivative that is important for an optimal flight. We will also wish to guarantee strong flight stability by making sure that the projectile will be dynamically and statically stable (McCoy, 1999). To ensure static stability, the *pitch moment coefficient slope* C_{m_α} will have to be smaller than $C_{m_\alpha}^{max}$ and to ensure dynamic stability the *pitch damping moment coefficient* C_{m_q} will have to remain smaller than $C_{m_q}^{max}$. These two conditions have to be respected for all velocities in the interval $[M_{min}\ M_{max}]$.

Our objective in this paper is therefore to determine the geometrical configuration of the finner (i.e., the value of the five parameters in Table 1) that leads to the lowest average value of the drag coefficient C_{A_0} while guaranteeing that, at all times, C_{m_α} and C_{m_q} remains smaller than $C_{m_\alpha}^{max}$ and $C_{m_q}^{max}$, respectively.[3]

Let us formulate this objective in a mathematical form. For this purpose, let us denote $Y = (C_{A_0}, C_{m_\alpha}, C_{m_q})^T$ the vector containing the stability derivatives considered in this study. Since Y is a function of X and M, we

[3]The quantities C_{A_0}, C_{m_α}, C_{m_q} and M are dimensionless.

can therefore use the following notation:

$$Y = F(X, M), \tag{1}$$

where F is a static function taking as inputs the vector X and the scalar M and giving as output the vector Y. For the i^{th} entry of Y ($i = 1, ..., 3$), the notation $Y_i = F_i(X, M)$ will also be used.

For a given geometrical configuration X, the average value $J(X)$ of C_{A_0} over the interval $[M_{min}\ M_{max}]$ can thus be expressed as:

$$J(X) = \frac{1}{M_{max} - M_{min}} \int_{M_{min}}^{M_{max}} F_1(X, M) dM. \tag{2}$$

Likewise, the stability constraints can be expressed by $F_2(X, M_{max}) \leq C_{m_\alpha}^{max}$ and $F_3(X, M_{max}) \leq C_{m_q}^{max}$. We here use the fact that, for rectangular finners in supersonic regime, both $C_{m_\alpha} = Y_2$ and $C_{m_q} = Y_3$ always reach their maximum at the highest velocity M_{max} (i.e., the initial velocity of the flight).

The optimization problem considered in this paper can therefore be formulated as follows:

$$X^* = \arg \max_{X \in \mathcal{X}} -J(X) \tag{3a}$$

$$\text{s.t. } F_2(X, M_{max}) \leq C_{m_\alpha}^{max} \text{ and } F_3(X, M_{max}) \leq C_{m_q}^{max} \tag{3b}$$

Note that we here formulate the optimization problem (3) as a maximization to facilitate the presentation in the sequel. Minimizing $J(X)$ is indeed equivalent to maximizing $-J(X)$. In (3), $\mathcal{X}$ represents the allowed search space: the set $\mathcal{X}$ can, e.g., contain all geometrical configurations X that are achievable in practice or, alternatively, the set $\mathcal{X}$ can also be a subset of the latter if, for some reasons, we wish to restrict the possible geometrical configurations.

Solving the optimization problem (3) is not straightforward. Indeed, the static function F is unknown, i.e., there is no readily available physical model for this function (at least not one which does not require computationally intensive simulations). However, there exist databases containing values of this unknown static function, i.e., a set of data giving, for a number of geometry configurations, the corresponding vector Y for different Mach numbers.

These databases are generally determined using experimental campaigns or computationally intensive simulations.

In this paper, we will assume that we have access to such a database. This database contains N_X different geometrical configurations for which the stability derivatives Y have been computed for N_M different Mach numbers in the interval $[M_{min}\ M_{max}]$ [4]. In other words, the database contains the value of the vector $Y = F(X^k, M^l)$ for all pairs (X^k, M^l) with M^l $(l = 1, \dots N_M)$ different values of M in the interval $[M_{min}\ M_{max}]$ and with X^k $(k = 1, \dots, N_X)$ different geometrical configurations. Let us denote by $\mathcal{D}_X$ the set containing these N_X geometrical configurations for which we know Y at N_M different Mach numbers, i.e., $\mathcal{D}_X = \{X^k | k = 1, \dots N_X\}$.

Using this database, $J(X)$ can be approximated for all $X \in \mathcal{D}_X$ using:

$$J(X) = \frac{1}{N_M} \sum_{l=1}^{N_M} F_1(X, M^l) \tag{4}$$

From now onwards, the expression (4) for $J(X)$ will be used and we will assume that M_{max} is a velocity for which Y is given in the database. Then, by restricting attention to the geometrical configurations $X \in \mathcal{X}$ that are present in $\mathcal{D}_X$ (i.e., in the database), we can derive an estimate $X^*_{\mathcal{D}_X}$ of the solution X^* of the optimization problem (3) :

$$X^*_{\mathcal{D}_X} = \arg \max_{X \in \mathcal{D}_X \cap \mathcal{X}} -J(X) \tag{5a}$$

$$\text{s.t. } F_2(X, M_{max}) \leq C^{max}_{m_\alpha} \text{ and } F_3(X, M_{max}) \leq C^{max}_{m_q} \tag{5b}$$

It is to be noted that $\mathcal{D}_X \cap \mathcal{X} = \mathcal{D}_X$ if $\mathcal{X}$ is the set of all achievable geometrical configurations.

The solution $X^*_{\mathcal{D}_X}$ of the optimization problem (5) is only an approximation of X^*. However, it is the best approximation that can be derived given the information in the database. It is also clear that the larger the database, the better the approximation $X^*_{\mathcal{D}_X}$ of X^*. In this paper, the objective is to improve $X^*_{\mathcal{D}_X}$ by extending the set $\mathcal{D}_X$, i.e., the set of geometrical configurations for which the value of Y, at different Mach numbers, is known.

[4] The database may of course also contain the value of the stability derivatives for velocities outside this interval.

Extending the set $\mathcal{D}_X$ comes with a cost. Indeed, if the objective is to know the value of Y at different Mach numbers for a geometrical configuration $X \notin \mathcal{D}_X$, a flight experiment has to be carried out with a projectile having that particular geometrical configuration or, alternatively, a computationally intensive simulation has to be performed. Consequently, we wish to extend the set $\mathcal{D}_X$ in a smart way, i.e., intelligently determining which configuration(s) will be added to $\mathcal{D}_X$ and for which a flight experiment or a simulation has to be performed. Two cases are considered:

- In the first case, the situation where only one geometrical configuration can be added to the set $\mathcal{D}_X$ is considered.
- In the second case, it is supposed that a number $N_{sup} > 1$ of geometrical configurations can be added to the set $\mathcal{D}_X$ (which will finally contain $N_X + N_{sup}$ configurations).

To progressively introduce the concepts, these two cases will be presented separately. Note however that the first case is obviously equivalent to the second case with $N_{sup} = 1$.

Remark 1. The case study concerns a projectile with 4 rectangular fins (see Figure 1) and a given flight scenario. However, the methodology presented in this paper can easily be extended to other types of projectiles and for other flight scenarios.

Remark 2. If $\mathcal{X}$ and/or $\mathcal{D}_X$ is a small set, it may happen that the optimization problem (5) does not have any solution (i.e., the constraints are not respected for all X in $\mathcal{D}_X \cap \mathcal{X}$). In this case, the approaches presented in the next sections are absolutely necessary to determine the geometry of a projectile satisfying the stability constraints.

3. Extending the database with one geometrical configuration

3.1. Procedure

In this section, we wish to add one geometrical configuration X_{new} to $\mathcal{D}_X$ yielding the set $\mathcal{D}_{X,new} = \mathcal{D}_X \cup X_{new}$ with $N_X + 1$ configurations.

The extended database allows to determine $X^*_{\mathcal{D}_{X,new}}$, solution of the following optimization problem:

$$X^*_{D_{X,new}} = \arg \max_{X \in \mathcal{D}_{X,new} \cap \mathcal{X}} -J(X), \tag{6a}$$

$$\text{s.t. } F_2(X, M_{max}) \leq C^{max}_{m_\alpha} \text{ and } F_3(X, M_{max}) \leq C^{max}_{m_q}, \tag{6b}$$

which is the optimization problem (5) where $\mathcal{D}_{X,new}$ is used instead of $\mathcal{D}_X$.

The new configuration X_{new} should be chosen in such a way that $X^*_{\mathcal{D}_{X,new}}$ is a better approximation of X^* than $X^*_{\mathcal{D}_X}$. To find such a geometrical configuration X_{new}, the database is used to derive a black-box model $\hat{F}$ of the unknown function F (see (1)):

$$\hat{Y} = \hat{F}(X, M) \tag{7}$$

Using this model $\hat{F}$, the value of Y can now be predicted for configurations X that are not in the database. In other words, for a configuration $X \notin \mathcal{D}_X$, we can estimate $F_2(X, M_{max})$ and $F_3(X, M_{max})$ by, respectively, the second and third entries of $\hat{F}(X, M_{max})$, i.e., $\hat{F}_2(X, M_{max})$ and $\hat{F}_3(X, M_{max})$, and we can estimate $J(X)$ (see (4)) by :

$$\hat{J}(X) = \frac{1}{N_M} \sum_{l=1}^{N_M} \hat{F}_1(X, M^l) \tag{8}$$

Given this model $\hat{F}$ of the unknown function F, the most promising value for X_{new} is:

$$X_{new} = \arg \max_{X \in \mathcal{X}} -\hat{J}(X) \tag{9a}$$

$$\text{s.t. } \hat{F}_2(X, M_{max}) \leq C^{max}_{m_\alpha} \text{ and } \hat{F}_3(X, M_{max}) \leq C^{max}_{m_q} \tag{9b}$$

In other words, X_{new} is chosen as the geometrical configuration that would be equal to X^* if the model would be perfect, i.e., $\hat{F}(X, M) = F(X, M)$ for all X and M.

If the optimization problem (9) leads to a configuration $X_{new} \notin \mathcal{D}_X$, i.e., a configuration for which the model $\hat{F}$ predicts that X_{new} outperforms $X^*_{\mathcal{D}_X}$,

this new configuration X_{new} should be tested via an experiment or a simulation to obtain the actual values of $F(X_{new}, M^l)$ for the velocities M^l ($l = 1, \ldots, N_M$). This allows to extend the original database to a new database with $N_X + 1$ configurations, i.e., the configurations in $\mathcal{D}_{X,new} = \mathcal{D}_X \cup X_{new}$, and to compute the solution $X^*_{\mathcal{D}_{X,new}}$ of the optimization problem (6). This solution $X^*_{\mathcal{D}_{X,new}}$ will be given by X_{new} if, as expected by the model of F, we indeed have that:

$$F_2(X_{new}, M_{max}) \leq C^{max}_{m_\alpha} \text{ and } F_3(X_{new}, M_{max}) \leq C^{max}_{m_q} \tag{10a}$$

$$J(X_{new}) < J(X^*_{\mathcal{D}_X}) \tag{10b}$$

If $X^*_{\mathcal{D}_{X,new}} = X_{new}$ then X_{new} is a better approximation of X^* than $X^*_{\mathcal{D}_X}$.

It may however happen that the solution of the optimization problem (6) remains $X^*_{\mathcal{D}_X}$. This happens when the model $\hat{F}$ of F does not give an adequate estimate of $F(X, M)$ for $X = X_{new}$, which, in turn, happens when $\mathcal{D}_X$ does not contain enough points in the vicinity of X_{new}. In this case, the approach which consists in adding more than one additional configurations to the database (see Section 4) could be considered.

3.2. Identification of $\hat{F}$

Let us now say a few words on how a model $\hat{F}$ of F can be derived using the initial database. For this purpose, a model structure $F_m(X, M, \theta)$ parametrized with a parameter vector θ is chosen. The value of this parameter vector is determined using the following identification criterion :

$$\hat{\theta} = \arg \min_\theta V(\theta), \tag{11}$$

$$V(\theta) = \frac{1}{N_X \ N_M} \sum_{k=1}^{N_X} \sum_{l=1}^{N_M} \|F(X^k, M^l) - F_m(X^k, M^l, \theta)\|^2, \tag{12}$$

where $F(X^k, M^l)$ are the values of Y available in the original database and

$\|A\| = \sqrt{A^T A}$ is the Euclidean norm[5] of the vector A.
We have then $\hat{Y} = \hat{F}(X, M) = F_m(X, M, \hat{\theta})$.

The model structure $F_m(X, M, \theta)$ has to be chosen as the simplest model structure allowing to explain the data in the database [6]. As mentioned in the introduction, in this paper, the model structure is chosen as a neural network. More details on the neural network model structure are given in Appendix A.

4. Extending the database with N_{sup} geometrical configurations

4.1. Procedure

For the procedure described in Section 3.1, the quality of the model $\hat{F}$ of F which depends on the database with which $\hat{F}$ has been identified is crucial. This quality will be high if $\mathcal{D}_X$ covers every region of $\mathcal{X}$ in a sufficient manner.

When we are allowed to extend the database with N_{sup} geometrical configurations ($N_{sup} > 1$), this opportunity can be used to enrich the coverage of the set $\mathcal{D}_X$ by adding configurations in regions that have not been explored yet, allowing in this way model improvement. Therefore, two contradictory objectives have to be distinguished when adding new geometrical configurations X_{new} to the database, i.e, the exploitation and the exploration objectives:

- When X_{new} is chosen according to (9), it is selected according to the exploitation objective (we choose the most promising configuration according to the available model).
- When X_{new} is chosen according to the exploration objective, it is selected in such a way that $\mathcal{D}_X \cup X_{new}$ covers the space of geometrical configurations in a better way to improve the model.

[5] We can also use a weighted norm in the definition of $V(\theta)$ to account for differences in the magnitude of the different entries of F.

[6] We can, e.g., first split the data in the database in a training data set and a validation data set and determine, using, e.g., a grid search approach, the model structure for which the model $F_m(X, M, \hat{\theta}_{training})$ identified with the training data leads to the smallest value of the cost function (12) when it is computed with the validation data. Once this model structure has been determined, we re-identifiy the model using the whole data in the database (via (11)-(12)).

Inspired by Bayesian Optimization, we here propose a framework where new geometrical configurations are added to $\mathcal{D}_X$ based on a trade-off between the exploitation and exploration objectives. This can be achieved using the following optimization problem instead of the optimization problem (9) :

$$X_{new} = \arg \max_{X \in \mathcal{X}} A(X), \tag{13}$$

where $A(X)$ is the so-called acquisition function that will be precisely defined in the sequel. For the moment, it is sufficient to say that $A(X)$ will be large:

- for values of X for which the current model predicts that the constraints in (9) are respected and that $\hat{J}(X)$ is small (exploitation objective) ;
- for values of X far away from the configurations that are already in the database (exploration objective).

The computation of the value of the acquisition function for a given X thus requires the use of the current model and the use of the configurations present in the current database.

The procedure to determine the N_{sup} new configurations is summarized in Algorithm 1. We start with the initial database with the configurations in the set $\mathcal{D}_X$ from which we can deduce the model $\hat{F}$ (see Section 3.1). Using that model $\hat{F}$ and the set $\mathcal{D}_X$, the acquisition function $A(X)$ can thus be constructed and X_{new} determined according to (13). Using a flight experiment or a simulation, the actual values of $F(X_{new}, M^l)$ can be obtained for the velocities M^l $(l = 1, \ldots, N_M)$. This allows to extend the original database to a new database with $N_X + 1$ configurations (i.e., the configurations in $\mathcal{D}_{X,new} = \mathcal{D}_X \cup X_{new}$). Based on this new extended database, we can re-identify the model $\hat{F}$ of F. Using this updated model and the set $\mathcal{D}_{X,new}$, the value of the acquisition function $A(X)$ is modified and so is the solution X_{new} of the optimization problem (13). For this new geometrical configuration, a flight experiment or a simulation is performed, leading to a database containing the values of $F(X, M)$ for $N_X + 2$ configurations. Using this extended database, the model $\hat{F}$ is once again updated etc. This procedure is followed up to the moment where the database contains the values of F for $N_X + N_{sup} - 1$ configurations and where the model $\hat{F}$ has been identified using this database. The last configuration is then determined based on a

purely exploitation objective, i.e., using the optimization problem (9) such as in Section 3.1. An eventual better model would indeed be of no use and exploration is therefore no longer useful.

Algorithm 1 yields a database $\mathcal{D}_{X,new}$ containing the information $F(X, M^l)$ $(l = 1, \ldots, N_M)$ for $N_X + N_{sup}$ geometrical configurations X. Optimization problem (6) can then be used to determine $X^*_{\mathcal{D}_{X,new}}$, i.e., the optimal configuration within this extended database.

Algorithm 1 Selection of N_{sup} new configurations

Initialize: : Determine the neural network model $\hat{F}(X, M)$ using the database with the geometrical configurations in $\mathcal{D}_X$ and pose $\mathcal{D}_{X,new} = \mathcal{D}_X$

Repeat $N_{sup} - 1$ **times the following steps**

1. $X_{new} = \arg\max_X A(X)$ where the acquisition function $A(X)$ for a given configuration X is computed using the current model $\hat{F}$ and using the configurations in $\mathcal{D}_{X,new}$

2. Determine $F(X_{new}, M)$ for different Mach numbers and replace $\mathcal{D}_{X,new}$ by $\mathcal{D}_{X,new} \cup X_{new}$

3. Determine the neural network model $\hat{F}(X, M)$ using the database with the geometrical configurations in $\mathcal{D}_{X,new}$

End Repeat

Do

X_{new} is determined via the optimization problem (9)

Determine $F(X_{new}, M)$ for different Mach numbers and replace $\mathcal{D}_{X,new}$ by $\mathcal{D}_{X,new} \cup X_{new}$

Remark 3. For $N_{sup} = 1$, Algorithm 1 reduces to the procedure presented in Section 3.1. As already mentioned, we have nevertheless decided to first present the case where $N_{sup} = 1$ for the sake of clarity.

4.2. Definition of the acquisition function

Let us now define precisely the acquisition function $A(X)$. For this purpose, let us suppose as in Algorithm 1 that the current database contains the values of $F(X, M^l)$ $(l = 1, \ldots, N_M)$ for the configurations X in $\mathcal{D}_{X,new}$ and that the model $\hat{F}$ has been determined based on this current database. Using this model, we can evaluate, for every X, $\hat{J}(X)$ (see (8)) as well as $\hat{F}_2(X, M_{max})$ and $\hat{F}_3(X, M_{max})$. Then, we define $A(X)$ as:

$$A(X) = A_{const}(X)\ A_{obj}(X), \tag{14}$$

where

$$A_{obj}(X) = -\hat{J}(X) + \beta\ \delta_{min}(X) + \rho, \tag{15}$$

with

$$\delta_{min}(X) = \min_{X_{database} \in \mathcal{D}_{X,new}} \|X - X_{database}\|, \tag{16}$$

$$A_{const}(X) = \begin{cases} 2 - \delta_{min}^{scaled}(X) & \text{if } \hat{F}_2(X, M_{max}) \leq C_{m_\alpha}^{max} \text{ and } \hat{F}_3(X, M_{max}) \leq C_{m_q}^{max} \\ \delta_{min}^{scaled}(X) & \text{otherwise,} \end{cases} \tag{17}$$

where

$$\delta_{min}^{scaled}(X) = \frac{\delta_{min}(X)}{\max\limits_{\tilde{X} \in \mathcal{X}} \delta_{min}(\tilde{X})} \tag{18}$$

In the expression of $A_{obj}(X)$, ρ is an offset chosen to ensure that $A_{obj}(X) > 0$ for all $X \in \mathcal{X}$. This offset is necessary since $A_{const}(X)$ will be a positive scalar with a value close to zero when it is likely that the constraints on C_{m_α} and C_{m_q} are not respected for a particular X (see below). Otherwise, we note that A_{obj} is made up of two other terms:

- The first term, $-\hat{J}(X)$,reflects the exploitation objective (beneficial to configurations X for which the model predicts a small average value for C_{A_0}).
- The second term, $\delta_{min}(X)$, reflects the exploration objective (beneficial to configurations X in regions where few data points are available in the database).

The quantity $\delta_{min}(X)$ is indeed small for configurations X that are close (in an Euclidean norm[7] sense) to the configurations in $\mathcal{D}_{X,new}$ and increases

[7]Other distance measures than the Euclidean norm can also be considered.

for configurations X that are far from the configurations in $\mathcal{D}_{X,new}$. The user-chosen scalar β realizes the trade-off between the exploration and exploitation objectives.

The term $A_{const}(X)$ in (14) is a penalty term pertaining to the constraints. As mentioned above, $A_{const}(X)$ is a positive scalar with a value close to zero when it is likely that the constraints on C_{m_α} and C_{m_q} are not respected for a particular X. The expression (17) of $A_{const}(X)$ uses the quantity $\delta_{min}^{scaled}(X)$ which is a scaled version of $\delta_{min}(X)$ ($\delta_{min}^{scaled}(X)$ varies between 0 and 1) and is based on the rationale that, if $\delta_{min}^{scaled}(X)$ is close to 1 for a given X, the current model $\hat{F}$ may not be very accurate to predict $F(X, M)$ since the model has been identified with data far away from X.

Let us now explain expression (17). The variable $A_{const}(X)$ will be close to zero for a given X if the current model $\hat{F}$ predicts that the constraints on C_{m_α} and C_{m_q} are not respected for that value of X and if $\delta_{min}^{scaled}(X)$ is close to 0 (i.e., if X is close to the data in $\mathcal{D}_{X,new}$). Indeed, in this case, it is very likely that X does not respect the constraints and this value of X must be strongly penalized in the acquisition function $A(X)$.

For values of X further away from the data in $\mathcal{D}_{X,new}$ and for which the current model predicts that the constraints on C_{m_α} and C_{m_q} are not respected, this penalization will be less strong, since $A_{const}(X) = \delta_{min}^{scaled}(X)$ will be larger (while being always smaller than one).

Let us now analyse the value of $A_{const}(X)$ for values of X for which the current model predicts that the constraints on C_{m_α} and C_{m_q} are respected. We observe that $A_{const}(X)$ will be then maximal (i.e., close to 2) if the value of X for which the current model predicts that the constraints are respected is such that $\delta_{min}^{scaled}(X)$ is close to zero (i.e., a value of X close to the data in $\mathcal{D}_{X,new}$ and thus for which $\hat{F}(X)$ is likely to be close to $F(X)$). These are the values of X that will be less penalized by $A_{const}(X)$. The values of X for which the current model predicts that the constraints are respected, but that are further away from the data in $\mathcal{D}_{X,new}$ will be (slightly) more penalized (value of $A_{const}(X)$ between 1 and 2).

Remark 4. As mentioned above, Algorithm 1 is inspired from Bayesian Optimization. The main differences between Bayesian Optimization and

the approach in this paper are the type of model $\hat{F}$ that is considered and the definition of the acquisition function $A(X)$. Indeed in classical Bayesian Optimization, the model of the static function F is a Gaussian Process (Rasmussen, 2003), i.e., a probabilistic model. The probabilistic nature of the Gaussian Process model allows to define the acquisition function $A(X) = A_{const}(X)A_{obj}(X)$ in a different way than in (14)-(18). More precisely, $A_{const}(X)$ is defined as the probability that the constraints are respected for a given value of X, while $A_{obj}(X)$ can be defined in a similar way as in (15), but with $\delta_{min}(X)$ replaced by the standard deviation of $\hat{J}(X)$ (Frazier, 2018). As explained before, a neural network is chosen here as surrogate model instead of a Gaussian Process. However, since a neural network is not a probabilistic model, the definition of $A(X)$ needed to be modified. Our definition of $A(X)$ in (14)-(18) nevertheless shows strong similarities with the definition of $A(X)$ in Bayesian Optimization. As an example, the quantity $\delta_{min}(X)$ used in (15) is a relevant alternative for the standard deviation of $\hat{J}(X)$ used in Bayesian Optimization since this standard deviation for Gaussian Process models will also be large for values of X far away from the data with which the Gaussian Process model has been identified.

Remark 5. The scalar β in the definition of A_{obj} (see (15)) must be chosen with care. One possibility is to choose β in such a way that the term $\hat{J}(X)$ and the term $\beta\ \delta_{min}(X)$ have the same order of magnitude for all $X \in \mathcal{X}$. Moreover, while respecting this general objective, we can also opt for a different value of β at each of the $N_{sup} - 1$ iterations of the *repeat loop* in Algorithm 1. One could, e.g., start with a relatively high value of β in the first iterations (to enforce more exploration when the model is less accurate) and decrease β in the subsequent iterations (to enforce more exploitation when the model becomes more accurate).

5. Numerical illustration

5.1. Setup and initial database

In this section, the methodology presented in Sections 3 and 4 is applied to determine the optimal geometrical configuration X of a rectangular finner (see Figure 1) for the flight scenario described in Section 2 with $M_{min} = 2$ and $M_{max} = 5$. The search set $\mathcal{X}$ is here limited to enable that the projectile can be fired by a specific field gun (see Table 2 for the formal definition of the set $\mathcal{X}$). For the stability constraints in (3), the upper bounds are fixed

to the following realistic values:

- $C_{m_\alpha}^{max}$ is fixed at -10.
- $C_{m_q}^{max}$ is fixed at -100.

For this type of finners and this flight scenario, we have a database giving the vector Y for $N_X = 324$ different configurations X and for $N_M = 8$ different Mach numbers in the interval [2 5]. As mentioned in the introduction, this database has been here generated using the high fidelity simulation code *PRODAS*.

Note that out of the $N_X = 324$ different configurations in $\mathcal{D}_X$, only 144 are in the restricted set $\mathcal{X}$. Among these configurations $X \in \mathcal{X} \cap \mathcal{D}_X$, none has a value of $F_2(X, M) \leq -10$. Consequently, $X^*_{\mathcal{D}_X}$ does not exist. However, among the configurations $X \in \mathcal{X} \cap \mathcal{D}_X$ that almost respect the constraints, the one leading to the smallest value of $J(X)$ is the configuration :

$$X^*_{\mathcal{D}_{X,almost}} = (20, 20, 1.5, 1.5, 0)^T. \tag{19}$$

More specifically for $X = X^*_{\mathcal{D}_{X,almost}}$, we have:

$$J(X) = 0.45 \tag{20a}$$
$$F_2(X, M = 5) = -8 \tag{20b}$$
$$F_3(X, M = 5) = -1400, \tag{20c}$$

which shows that C_{m_α} is only slightly too high.
Let us now see whether the procedures presented in Section 3 and in Section 4 allows to determine configurations X that are more acceptable than $X^*_{\mathcal{D}_{X,almost}}$.

5.2. Adding one additional configuration

Let us first consider the procedure of Section 3 consisting in determining one single new configuration. This procedure relies on a neural network model identified with the data in the original database. As a first step, we

Table 2: Definition of the set $\mathcal{X}$ for each characteristic X_i of the geometry (X_1, X_3, X_4 and X_5 are expressed in calibers and X_2 in degrees)

Dimension X	Minimum	Maximum
X_1 : Total Length	10	25
X_2 : Nose Angle	10	34.5
X_3 : Fins Height	0.5	2.5
X_4 : Fins Width	0.5	1.72
X_5 : Position of fins	0	1

split these data to determine a suitable structure $F_m(X, M, \theta)$ for the neural network (see the footnote in Section 3.2). This leads to a neural network with 1 hidden layer, 128 neurons and a *reLu* activation function on each neuron output (see Appendix A). The criterion (11)-(12) is subsequently used to determine the values of the different weightings and offsets in this neural network structure (i.e., the parameter vector $\hat{\theta}$). This criterion is solved using a machine learning library known as *Scikit-learn* (Buitinck et al., 2013).

The model $\hat{F}(X, M) = F_m(X, M, \hat{\theta})$ can now be used in the procedure of Section 3. The new configuration X_{new} is thus determined using the optimization problem (9) that we here solve using the Sequential Least Square Programming algorithm given in (Virtanen et al., 2020). This leads to:

$$X_{new} = (12.6859, 10, 1.34, 1.72, 0.43)^T. \tag{21}$$

This solution X_{new} is such that the model $\hat{F}(X, M)$ predicts a value of 0.32 for $J(X_{new})$, i.e., $\hat{J}(X_{new}) = 0.32$ (see (8)). By construction, we have also that $\hat{F}_2(X_{new}, M = 5) \leq -10$ and $\hat{F}_3(X_{new}, M = 5) \leq -100$. More precisely:

$$\hat{F}_2(X_{new}, M = 5) = -10 \tag{22a}$$

$$\hat{F}_3(X_{new}, M = 5) = -419. \tag{22b}$$

See the blue curve in Figures 2, 3 and 4 for other characteristic values of $\hat{F}_i(X_{new}, M^l)$ $(i = 1, \ldots 3, l = 1, \ldots, N_M)$.

Let us now verify whether the predictions of $\hat{F}$ about X_{new} are confirmed in reality. For this purpose, we use *PRODAS* to compute $Y^{new} = F(X_{new}, M)$ at different Mach numbers M (see the green curve in Figures 2, 3 and 4 for characteristic values).

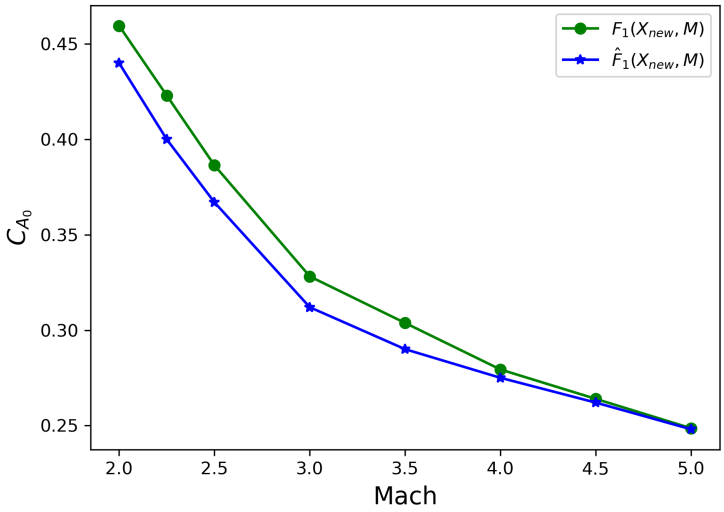

Figure 2: Case $N_{sup} = 1$: neural network prediction $\hat{F}_1(X_{new}, M)$ (blue) and $F_1(X_{new}, M)$ evaluated with PRODAS (green) for $X_{new} = (12.6859, 10, 1.34, 1.72, 0.43)^T$ and for characteristic values of $M \in [2\ 5]$.

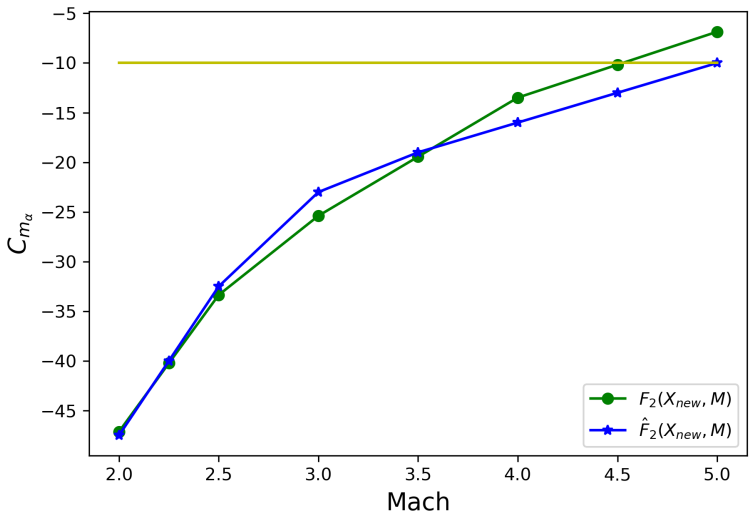

Figure 3: Case $N_{sup} = 1$: neural network prediction $\hat{F}_2(X_{new}, M)$ (blue) and $F_2(X_{new}, M)$ evaluated with PRODAS (green) for $X_{new} = (12.6859, 10, 1.34, 1.72, 0.43)^T$ and for characteristic values of $M \in [2\ 5]$. The yellow line represents the threshold $C_{m_\alpha}^{max} = -10$.

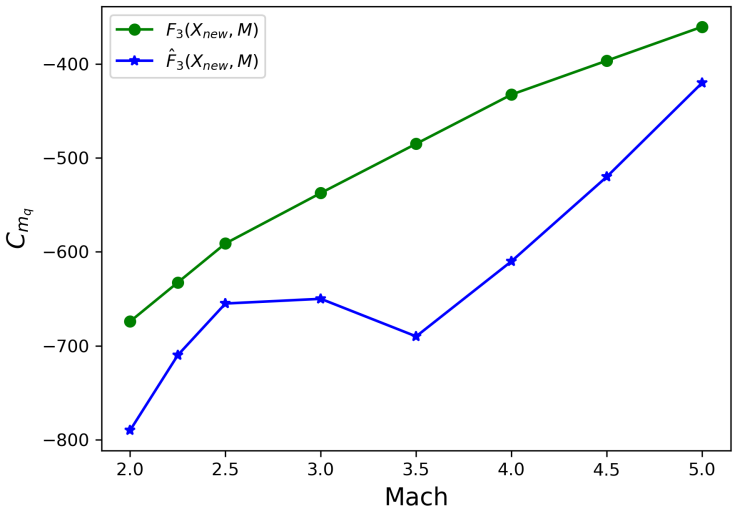

Figure 4: Case $N_{sup} = 1$: neural network prediction $\hat{F}_3(X_{new}, M)$ (blue) and $F_3(X_{new}, M)$ evaluated with PRODAS (green) for $X_{new} = (12.6859, 10, 1.34, 1.72, 0.43)^T$ and for characteristic values of $M \in [2\ 5]$.

This allows to observe that:

$$J(X_{new}) = 0.33 \text{ (see (4))} \tag{23a}$$
$$F_2(X_{new}, M = 5) = -6 \tag{23b}$$
$$F_3(X_{new}, M = 5) = -360 \tag{23c}$$

In other words, as opposed to what was predicted by the model $\hat{F}$, the constraint on C_{m_α} is not respected for the configuration X_{new}. Since the optimization problem (5) did not have a solution, the optimization problem (6) pertaining to $\mathcal{D}_{X,new} = \mathcal{D}_X \cup X_{new}$ has also no solution. The apparent contradiction between the results of optimization problems (9) and (6) can certainly be explained by the fact that the model $\hat{F}$ has been trained with too few data in the vicinity of X_{new}.

We therefore proceed with the methodology of Section 4. We here decide to determine $N_{sup} = 20$ new configurations, allowing for model improvement.

Note that this corresponds to an increase of 6% of the number of configurations in the database.

5.3. Adding 20 new additional configurations using Algorithm 1

Algorithm 1 is now applied[8] with $N_{sup} = 20$. To apply Algorithm 1, we have to specify two parameters in the definition of the acquisition function $A(X)$: the offset ρ and the parameter β balancing the exploitation and exploration objectives. While the choice of the offset ρ is not crucial, we here choose $\rho = 1$, the parameter β has to be chosen with care. Following the philosophy introduced in Remark 5 in Section 4, β must be chosen in such a way that the term $\hat{J}(X)$ and $\beta\ \delta_{min}(X)$ have the same order of magnitude for all $X \in \mathcal{X}$. In our case, this means that β must be chosen in the interval [0.5 4]. As proposed in Remark 5, we therefore choose to start with $\beta = 4$ in the first iteration of the *repeat loop* of Algorithm 1 and linearly decrease this value at each iteration in such a way that, at the last iteration, $\beta = 0.5$.

With this choice for ρ and β, Algorithm 1 delivers an extended database containing the values of Y for $N_X + 20 = 344$ configurations gathered in the set denoted as $X^*_{\mathcal{D}_{X,new}}$.

The optimization problem (6) can thus be used to determine the best configuration $X^*_{\mathcal{D}_{X,new}} \in \mathcal{D}_{X,new}$. The solution of this optimization problem is:

$$X^*_{\mathcal{D}_{X,new}} = (10.8, 13.42, 1.45, 1.72, 0)^T. \tag{24}$$

This configuration $X^*_{\mathcal{D}_{X,new}}$ is such that :

$$J(X^*_{\mathcal{D}_{X,new}}) = 0.34 \tag{25a}$$
$$F_2(X^*_{\mathcal{D}_{X,new}}, M = 5) = -10 \tag{25b}$$
$$F_3(X^*_{\mathcal{D}_{X,new}}, M = 5) = -313 \tag{25c}$$

Other characteristic values of $F_i(X^*_{\mathcal{D}_{X,new}}, M^l)$ $(i = 1, \dots 3, l = 1, \dots, N_M)$ are given in green in Figures 5, 6 and 7.

[8]The optimization of the acquisition function $A(X)$ (see (14)) in Step 1 of the *repeat loop* of Algorithm 1 is performed using the *dual annealing* algorithm (Xiang et al., 1997) in its *Scipy* Implementation (Virtanen et al., 2020).

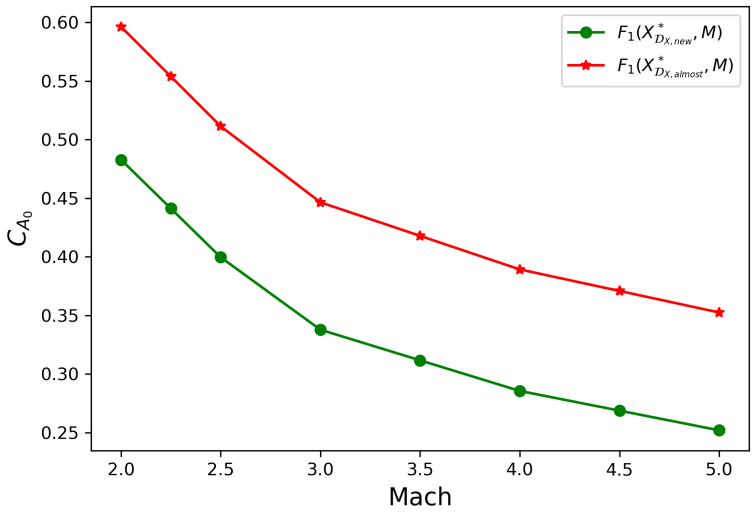

Figure 5: $F_1(X, M)$ (evaluated with PRODAS) for characteristic values of $M \in [2\ 5]$ and for two values of X, i.e., $X = X^*_{\mathcal{D}_{X,new}}$ (green) and $X = X^*_{\mathcal{D}_{X,almost}}$ (red).

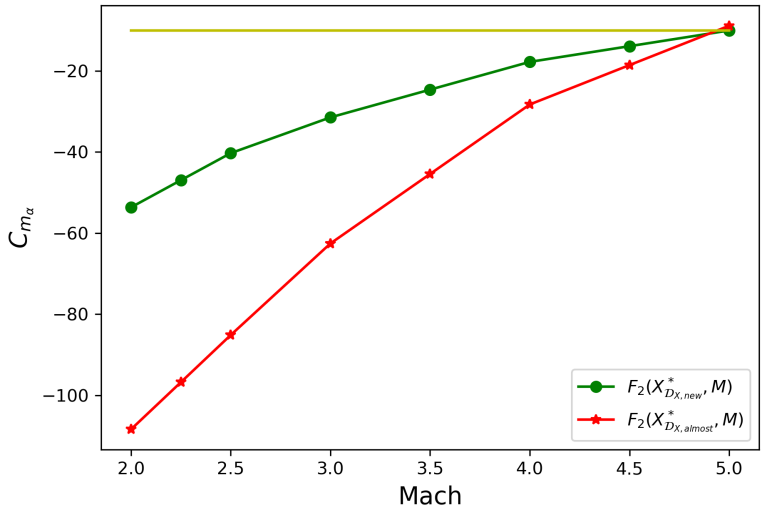

Figure 6: $F_2(X, M)$ (evaluated with PRODAS) for characteristic values of $M \in [2\ 5]$ and for two values of X, i.e., $X = X^*_{\mathcal{D}_{X,new}}$ (green) and $X = X^*_{\mathcal{D}_{X,almost}}$ (red). The yellow line represents the threshold $C^{max}_{m_\alpha} = -10$.

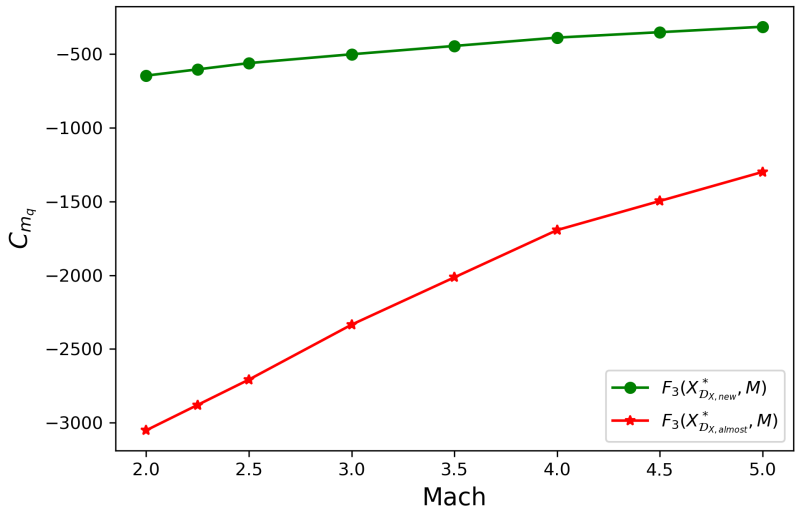

Figure 7: $F_3(X, M)$ (evaluated with PRODAS) for characteristic values of $M \in [2\ 5]$ and for two values of X, i.e., $X = X^*_{\mathcal{D}_{X,new}}$ (green) and $X = X^*_{\mathcal{D}_{X,almost}}$ (red).

Using Algorithm 1 with $N_{sup} = 20$, we have thus been able to determine, in the restricted search set $\mathcal{X}$, a configuration $X^*_{\mathcal{D}_{X,new}}$ that respects the stability constraints. Recall that the initial database did not contain any projectile with $X \in \mathcal{X}$ and that respected the constraints. Moreover, with respect to $X^*_{\mathcal{D}_{X,almost}}$ (which is, among the configurations in the initial database that **almost** respect the constraints, the one with the least average drag), the average drag with $X^*_{\mathcal{D}_{X,new}}$ is 25% smaller ($J(X^*_{\mathcal{D}_{X,new}}) = 0.34$ and $J(X^*_{\mathcal{D}_{X,almost}}) = 0.45$). This improvement of the drag is also evidenced by comparing the green and red curves in Figure 5.

In Figure 8, we represent, in the right plot, the optimal finner, i.e., the finner corresponding to the configuration $X^*_{\mathcal{D}_{X,new}}$ and we compare it to the one having the configuration $X^*_{\mathcal{D}_{X,almost}}$. With respect to $X^*_{\mathcal{D}_{X,almost}}$, the optimal finner has a smaller body and a larger nose length which helps decrease the drag, while its fin length remains large enough to satisfy the stability constraints.

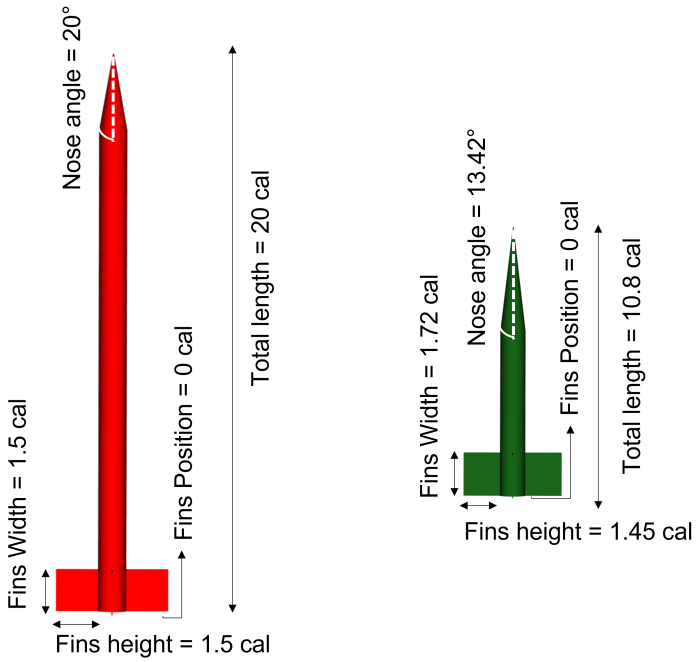

Figure 8: Representation of the projectile with configuration $X^*_{\mathcal{D}_{X,almost}}$ (left) and with configuration $X^*_{\mathcal{D}_{X,new}}$ (right).

As mentioned above, the average drag for $X^*_{\mathcal{D}_{X,new}}$ is 25% smaller than the one obtained with $X^*_{\mathcal{D}_{X,almost}}$. It is important to note that this major improvement in the capacity of the projectile is obtained by considering the set $\mathcal{D}_{X,new}$ which merely contains 6% more configurations ($N_{sup} = 20$) than the initial set $\mathcal{D}_X$. It should also be emphasized that, among the $N_{sup} = 20$ additional configurations that are added to the database via Algorithm 1, 14 configurations respect the stability constraints. Consequently, if we would have chosen $N_{sup} < 20$, we would have also obtained an acceptable projectile, e.g., with $N_{sup} = 2$, we would have obtained a stable projectile achieving an average drag of 0.47. With $N_{sup} = 11$ and $N_{sup} = 17$ this average drag would have been of 0.41 and 0.37, respectively.

5.4. Robustness of Algorithm 1 w.r.t β

To analyse the robustness of Algorithm 1 with respect to the choice of the parameter β, we have run this algorithm with other choices of β in the

interval [0.5 4]. More specifically, it is ran for $N_{sup} = 20$ with different constant values of β at each iteration in the *repeat loop*. Table 3 presents the values of β considered, the values of the average drag J obtained with each value of β and the number of configurations which respect the constraints among the ones tested for each value of β.

Table 3: Robustness of Algorithm 1 w.r.t β

β	Average drag J	Number of configurations which respect the constraints
0.5	0.35	8
1	0.37	12
2	0.38	17
4	0.36	18

As shown in table 3, when Algorithm 1 is performed with different choices of β, we can observe that they all lead to an optimal configuration respecting the stability constraints with a good average drag J. Moreover, among the $N_{sup} = 20$ additional configurations that are added to the database via Algorithm 1, a relatively large number respect the stability constraints.

5.5. Comparison to alternative approaches

As shown above, the procedure introduced in Section 4 and implementing, via Algorithm 1, a trade-off between exploitation and exploration to generate $N_{sup} = 20$ additional configurations for the database, delivers efficient results. To determine how efficient these results really are, let us compare the results obtained with Algorithm 1 with alternative approaches to generate the $N_{sup} = 20$ additional configurations. Recall for comparison purpose that, using Algorithm 1 with $N_{sup} = 20$ and a decaying β, 14 of the 20 additional configurations (i.e., 70%) respect the stability constraints and, among these configurations respecting the constraints, we have a projectile for which the average drag is equal to 0.34.

Alternative approach 1 (Random generation). In this first alternative approach, we choose to generate, in a random manner, 60 new configurations $X \in \mathcal{X}$, i.e., three times more than $N_{sup} = 20$. Only 20% of the randomly

generated configurations respect the constraints (which is much smaller than the 70% obtained with Algorithm 1) and, among these configurations respecting the constraints, the smallest J is 0.42 (which is 20% larger than what is obtained with Algorithm 1).

Alternative approach 2 (Exploration only). Instead of balancing exploitation and exploration, we choose in this second alternative approach to uniquely favour the exploration objective. This can, e.g., be achieved by performing Algorithm 1 with an alternative definition for the acquisition function $A(X)$, i.e., $A(X) = \delta_{min}(X)$. When we follow this approach with $N_{sup} = 20$, the extended database does not improve the initial one. Indeed, none of the additional configurations respects the stability constraints.

Alternative approach 3 (Exploitation only). Instead of balancing exploitation and exploration, we choose in this third alternative approach to uniquely favour the exploitation objective. This can be achieved by performing a modified Algorithm 1 where, in step 1 of the *repeat loop*, X_{new} is determined using optimization problem (9). When we follow this approach with $N_{sup} = 20$, we obtain 20 additional configurations of which only one respects the constraints. This configuration $X = (13.23, 10, 1.34, 1.72, 0)$ also achieves an average drag of 0.34. Since this particular configuration (the only one respecting the constraints) is obtained at iteration 19 of the *repeat loop*, this means that, as opposed to the initially proposed approach, this third alternative would not lead to any configuration respecting the constraints if $N_{sup} < 19$.

Alternative approach 4 (classical Bayesian Optimization). Another alternative approach to generate the $N_{sup} = 20$ additional configurations is to use classical *Bayesian Optimization* with a Gaussian Process model as surrogate model (see the remark 4 at the end of Section 4). We will here use the *Bayesian Optimization* formulation given in (Gardner et al., 2014). When this approach is applied[9] to generate $N_{sup} = 20$ additional configurations, we obtain a configuration for which the average drag is equal to 0.37 which is approximately 10% higher than the optimal average drag obtained with

[9]We have run the corresponding algorithm a number of times and we here report the best case result.

Algorithm 1. Moreover, among the 20 configurations selected via *Bayesian Optimization*, only 3 respect the constraints while there were 14 with Algorithm 1.

Table 4: Comparison between the different alternatives

Approach	$J(X)$ at the optimum	% of tested configurations which respect the constraints
Proposed approach	0.34	70%
Random generation	0.42	20%
Exploration only	-	0%
Exploitation only	0.34	5%
Bayesian Optimization	0.37	15%

Table 4 summarizes the results obtained from the different alternatives. As we can see from this table and from the discussions above, it is clear that the proposed approach balancing exploitation and exploration has the best overall results for this particular geometrical configuration. Indeed, it is the one which provides the configuration with the lowest average drag $J(X)$ and where there are the most tested configurations which respect the constraints.

6. Conclusion

In this paper, we optimize the geometrical configuration of a rectangular finner to obtain the least drag under some stability constraints, for a flat trajectory fire. This particular aerodynamic design problem is formulated as an optimization problem involving the stability derivatives of the projectile. Using an initial database, a (cost-effective) neural network surrogate model is used to model the stability derivatives. A procedure balancing exploitation and exploration is then devised to determine, based on that surrogate model, the values of the design variables for which the stability derivatives have to be evaluated to both improve the surrogate model and approach the optimal design of the projectile. By increasing the size of the database by a mere 6%, the proposed procedure allows to reduce by 25% the drag of the projectile with respect to the best projectile in the initial database. Moreover, on this particular application, the proposed procedure is shown to yield a projectile

with 10% less drag than the one that would have been obtained using classical *Bayesian Optimization*.

References

Albisser, M., Dobre, S., Berner, C., Thomassin, M., Garnier, H., 2017. Aerodynamic coefficient identification of a space vehicle from multiple free-flight tests. Journal of Spacecraft and Rockets 54, 426–435.

Anderson, J.D., Bowden, M.L., 2005. Introduction to flight. volume 582. McGraw-Hill Higher Education New York.

Arnoult, G., Zeidler, M., Garnier, E., 2020. Control surface geometry surrogate-based optimization for spin-stabilized projectile course correction. AIAA journal 58, 550–560.

Baheri, A., Bin-Karim, S., Bafandeh, A., Vermillion, C., 2017. Real-time control using bayesian optimization: A case study in airborne wind energy systems. Control Engineering Practice 69, 131–140.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., Varoquaux, G., 2013. API design for machine learning software: experiences from the scikit-learn project, in: ECML PKDD Workshop: Languages for Data Mining and Machine Learning, pp. 108–122.

Burnett, J., Hathaway, W., Whyte, R., 1981. Projectile Design and Analysis System (PRODAS-81). Technical Report. AFATL-TR-81-43, April 81.

Dettù, F., Corno, M., D'Ambrosio, D., Acquistapace, A., Taroni, F., Savaresi, S.M., 2023. Modeling, control design and experimental automatic calibration of a leveling system for combine harvesters. Control Engineering Practice 132, 105411.

Driver, J., Zingg, D.W., 2007. Numerical aerodynamic optimization incorporating laminar-turbulent transition prediction. AIAA journal 45, 1810–1818.

Dupuis, A.D., Hathaway, W., 1997. Aeroballistic range tests of the basic finner reference projectile at supersonic velocities. Technical Report. Defence Research Establishment Valcatier (Quebec).

Forrester, A., Sobester, A., Keane, A., 2008. Engineering design via surrogate modelling: a practical guide. John Wiley & Sons.

Frazier, P.I., 2018. A tutorial on bayesian optimization. arXiv preprint arXiv:1807.02811 .

Gardner, J.R., Kusner, M.J., Xu, Z.E., Weinberger, K.Q., Cunningham, J.P., 2014. Bayesian optimization with inequality constraints., in: ICML, pp. 937–945.

Gomec, F., Canibek, M., 2017. Aerodynamic database improvement of aircraft based on neural networks and genetic algorithms, in: 7th European Conference for aeronautics and space sciences (Eucass).

Jeong, S., Murayama, M., Yamamoto, K., 2005. Efficient optimization design method using kriging model. Journal of aircraft 42, 413–420.

McCoy, R., 1999. Modern exterior ballistics: the launch and flight dynamics of symmetric projectiles. Schiffer.

Rajkumar, T., Bardina, J.E., 2002. Prediction of aerodynamic coefficients using neural networks for sparse data., in: FLAIRS Conference, pp. 242–246.

Rasmussen, C.E., 2003. Gaussian processes in machine learning, in: Summer school on machine learning, Springer. pp. 63–71.

Renganathan, S.A., Maulik, R., Ahuja, J., 2021. Enhanced data efficiency using deep neural networks and gaussian processes for aerodynamic design optimization. Aerospace Science and Technology 111, 106522.

Roveda, L., Forgione, M., Piga, D., 2020. Robot control parameters autotuning in trajectory tracking applications. Control Engineering Practice 101, 104488.

Ruder, S., 2016. An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747 .

Savaia, G., Sohn, Y., Formentin, S., Panzani, G., Corno, M., Savaresi, S.M., 2021. Experimental automatic calibration of a semi-active suspension controller via bayesian optimization. Control Engineering Practice 112, 104826.

Uwadukunze, A., Bombois, X., Gilson, M., Albisser, M., 2023. Neural networks smart grid based optimisation for expensive functions. Submitted to ECC2024 and available on the HAL repository via https://hal.science/hal-04052060.

Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., et al., 2020. Scipy 1.0: fundamental algorithms for scientific computing in python. Nature methods 17, 261–272.

Xiang, Y., Sun, D., Fan, W., Gong, X., 1997. Generalized simulated annealing algorithm and its application to the thomson model. Physics Letters A 233, 216–220.

Appendix A. Neural networks

A neural network is a static mapping between a vector u of inputs (of dimension n_u) and a vector y of outputs (of dimension n_y)[10]. Let us focus on neural networks with one hidden layer since it is the structure that is used in Section 5. In such a network, each entry y_i ($i = 1, \ldots, n_y$) of y is expressed as an affine combination of so-called neurons $\nu_k(u)$ ($k = 1, \ldots, n_\nu$) (the number of neurons is denoted n_ν):

$$y_i = \sum_{k=1}^{n_\nu} w_{ik}\nu_k(u) + b_i, \tag{A.1}$$

with b_i a scalar offset and w_{ik} scalar weightings. The quantity $\nu_k(u)$ ($k = 1, \ldots, n_\nu$) is a nonlinear mapping of the entries of the input vector $u = (u_1, \ldots, u_{n_u})^T$:

$$\nu_k(u) = \Phi\left(\tilde{b}_k + \sum_{l=1}^{n_u} \tilde{w}_{kl}u_l\right), \tag{A.2}$$

[10]In the study case considered in this paper, we have that $n_u = 6$ and $n_y = 3$.

where $\Phi(x)$ is generally the so-called rectified linear unit (reLu) activation function, i.e., $\Phi(x) = max(0, x)$ and where $\tilde{w}_{kl}$ and $\tilde{b}_k$ are scalar coefficients. Since the coefficients b_i, w_{ik}, $\tilde{w}_{kl}$, $\tilde{b}_k$ are all free coefficients (that we can gather in a vector θ as proposed in Section 3.2), the equations (A.1) and (A.2) represent a parametrized (static) mapping between the input vector and the output vector. As explained in Section 3.2, the value of the parameter vector θ will be determined based on data (see the criterion (11) - (12)). The criterion (11) - (12) is generally solved via the back-propagation method which uses Gradient Descent (Ruder, 2016) to determine in an iterative manner the solution $\hat{\theta}$ of (11) - (12).