Article

Not peer-reviewed version

# Optimization of Airfoils for the Design of Long Endurance Aircrafts Using Deep Learning Models and Metaheuristics Algorithms

Jose Gabriel Quijada Pioquinto [*] , Valentin Shakhov , Evgenii Minaev , Evgenii Kurkin , Oleg Lukyanov

Posted Date: 25 July 2024

doi: 10.20944/preprints202407.1992.v1

Keywords: airfoil optimization; OpenFOAM; deep learning models; metaheuristics algorithms; method CST

*Article*

# Optimization of Airfoils for the Design of Long Endurance Aircrafts Using Deep Learning Models and Metaheuristics Algorithms

**Jose Gabriel Quijada Pioquinto [1],*, Valentin Shakhov [1], Evgenii Minaev [2], Evgenii Kurkin [1] and Oleg Lukyanov [1]**

[1] Department of Aircraft Construction and Design, Samara National Research University, 34 Moskovskoe Shosse, Samara 443086, Russian Federation; shakhov.vg@ssau.ru (V.S.); eugene.kurkin@mail.ru (E.K.); lukyanovoe@mail.ru (O.L.)
[2] Department of Supercomputers and General Informatics, Samara National Research University, 34 Moskovskoe Shosse, Samara 443086, Russian Federation; eminaev@gmail.com
* Correspondence: hosekihada@yandex.ru; Tel.: +7 909 371-75-19

**Abstract:** This paper presents a methodology based on deep learning models and metaheuristic algorithms for the optimization of airfoils for the design of aircraft wings with large endurance. The use of AZTLI-NN (a neural network with an architecture composed of a multilayer perceptron and a variational autoencoder) is implemented for the prediction of graphs of the aerodynamic coefficients of the airfoil as a function of the angle of attack. This neural network presents good predictions of the aerodynamic coefficients, similar to the coefficients obtained with computational fluid dynamics simulations. AZTLI-NN in combination of metaheuristic algorithms and the CST profile parameterization method show excellent performance in single-objective and multi-objective profile optimization tasks.

**Keywords:** airfoil optimization; OpenFOAM; deep learning models; metaheuristics algorithms; method CST

## 1. Introduction

As a key component of aircraft, an excellent aerodynamic configuration of the airfoil plays a crucial role in fuel economy, reducing polluting emissions and improving performance [1–3]. In the design of unmanned aerial vehicles (UAV) and gliders, the development of wing configurations with a high aspect ratio is also being opted for, because these wing configurations provide a better range and flight duration. In UAVs there is a greater interest in improving the flight duration, especially in subsonic flight missions, in missions such as: reconnaissance, target tracking, pseudo-satellites, etc [4].

Taking into account the aircraft powered by propeller engines, aerodynamically, an aircraft to achieve a large endurance its $C_L^{1.5}/C_D$ ($C_L$, lift coefficient; $C_D$, drag coefficient) ratio values should be the highest possible [5]. This $C_L^{1.5}/C_D$ ratio is usually called the endurance parameter. As mentioned above, this is mainly achieved by having a wing with a high aspect ratio, but the shape of the airfoils along the wingspan is also of utmost importance, which should also provide a high $c_l^{1.5}/c_d$ value [4,6]. The design of this type of airfoils is obtained by optimizing the shape of the airfoil, which must produce a high value of the endurance parameter with respect to a design lift coefficient. It is important to mention that airfoils with high values of $c_l^{1.5}/c_d$ and $c_l$ usually have convex shapes that generate very negative values of pitch moment coefficient ($c_m$). This drawback is compensated by the empennage of the aircraft. Although in aircraft configurations that lack a horizontal empennage (such as flying wings) it is necessary that the $c_m$ be as close to zero, for this reason in these cases an extra objective function is usually added to the optimization task, minimizing $|c_m|$ [4,7].

Computational fluid dynamics (CFD), wind tunnel testing and theoretical analysis have gradually become the main approaches to aerodynamic design, advancing the progress of aerodynamic optimization of airfoils. However, in real engineering design, wind tunnel tests are expensive, and theoretical analysis is less applicable when dealing with complex engineering problems, so CFD has gradually become the main method for aerodynamic analysis [8].

The basic element of aerodynamic optimization of the airfoil mainly includes the design object, the design objective, the design constraints and the design method (optimization method). Design object refers to the aerodynamic geometric configuration. The design objective refers to the expected aerodynamic performance, the characteristics of the flow field, etc. The design constraints illustrate the interdependence and constraints between the design variables of the design object. And optimization methods provide the strategies and means to achieve the design goal [8].

Currently there are a large number of optimization methods that are widely used. These methods can be generally classified into gradient-based and gradient-free optimization methods. Current gradient-based approaches use discrete adjunct sensitivity analysis, scale approximately linearly with the number of design variables, and are very capable of handling thousands of design variables and constraints. The advantages of gradient-based optimization to address airfoil shape optimization are low computational costs in large design spaces and a track record of successful implementations in aerodynamics. However, there are some drawbacks to this approach, such as the tendency to converge on local optima and a high sensitivity to the starting point, poor efficiency for nonlinear cost functions and the need for continuous shapes (the shape gradient must exist at all points). The gradient-based approach may be more appropriate for a detailed aerodynamic design, as they are only able to offer a limited range of solutions. Gradient-free methods can be more complex to implement than gradient-based methods, but they do not require continuity or predictability in the design space and usually increase the probability of finding a global optimum. Optimization methods known as metaheuristics can offer robust methods for finding a solution and increase the probability of converging on a solution at the global optimum. It is known that these gradient-free methods are able to address numerically noisy optimization problems that are difficult for gradient-based methods. And unlike gradient methods, derivatives of cost functions are not necessary. In addition, no predefined baseline design or knowledge of the design space is required and gradient-free methods usually optimize several solutions in parallel. However, their convergence speeds are usually slow enough that they are unable to cooperate directly with time-consuming CFD analysis. To counteract this disadvantage that metaheuristic methods have, in the first instance, more robust computing equipment is used (which implies raising the costs of the optimization process). Another alternative is to make use of methods based on surrogate models [8–11].

Traditional methods for training surrogate models include the polynomial response surface method, the support vector regression method, and the Kriging model. The traditional substitute model has adjustable parameters; however, the main problem is that it is not suitable for handling large-scale training data, so it is usually trained with a small amount of data in a relatively limited space [8–12]. Recent advances in machine learning (especially in deep learning methods: artificial neural networks) have provided a new option for generating surrogate models. Compared to traditional surrogate models, the advantages of applying neural networks in aerodynamic data modeling lie in the fact that: a) neural networks, as a data-driven model, are not based on aerodynamic theories or physical models, which means that it does not require in-depth knowledge of aerodynamics; b) neural networks can be used to address high-dimensional, multi-scale and nonlinear problems that are difficult for traditional surrogate models; and c) some neural networks have the ability to process time sequence data [13].

In the present work, a methodology is developed to accelerate the process of airfoil optimization under conditions of subsonic, incompressible and turbulent flow, making use of current deep learning models and metaheuristic algorithms. This research work is specifically aimed at the design of aircraft with long endurance operating in subsonic flight regime.

## 2. Materials and Methods

### *2.1. Mathematical Formulation of the Problems*

Two cases are considered:

1. determination of the airfoil shape producing a long endurance parameter ($c_l^{1.5}/c_d$) with respect to a specified lift coefficient;
2. determination of the airfoil shape producing a long endurance parameter (cl1.5/cd) and a near-zero value of the pitch moment coefficient (cm) with respect to a specified lift coefficient.

The purpose of the above cases is to contemplate aircraft configurations with empennage and without empennage.

For the first case, it is only necessary to maximize the endurance parameter and consider design constraints such as the design lift coefficient.

$$\begin{gathered} \max f_1(\xi) = c_l^{1.5}/c_d, \\ \text{such that } h_a(\xi) = 0,\ a = 1, 2, \ldots, A, \\ g_b(\xi) \le 0,\ b = 1, 2, \ldots, B, \\ \xi \in \Omega. \end{gathered} \tag{1}$$

For the second case, the objective of minimizing the magnitude of $c_m$ is added.

$$\begin{gathered} \max f_1(\xi) = c_l^{1.5}/c_d, \\ \min f_2(\xi) = |c_m| \\ \text{such that } h_a(\xi) = 0,\ a = 1, 2, \ldots, A, \\ g_b(\xi) \le 0,\ b = 1, 2, \ldots, B, \\ \xi \in \Omega. \end{gathered} \tag{2}$$

where $\xi$ is the vector of design parameters; $h(\xi)$ are constraint functions (equalities); $g(\xi)$ are constraint functions(inequalities); $\Omega$ is the decision space of the design parameters.

To provide a solution to the above optimization tasks, the following methodology is proposed:

1. Selecting a method of parameterization of the aerodynamic profile, the method should allow modeling aerodynamic profiles used for the design of aircraft with long-span wings;
2. Select a robust method that allows the aerodynamic coefficients of the airfoils to be calculated. The method should take into account the viscosity and turbulence effects of the flow;
3. Propose a model based on deep learning models to predict the values of the objective functions, this in order to speed up the process of calculating the objective functions;
4. Finally, select the appropriate optimization algorithms for each task.

### *2.2. Selection of the Method of Airfoils Parameterization*

To determine the design parameters, two of the best airfoil parameterization methods were evaluated: CST (see Appendix A) and Bezier-PARSEC (see Appendix B) methods. The way to evaluate the methods is to evaluate the capabilities they have when reconstructing the geometry existing airfoils (such as NACA, Eppler, TSAGI, etc.). Obtaining the original coordinates of the airfoils is carried out using the profile database of the University of Illinois at Urbana-Champaign (UIUC) [14]. Only the selection of asymmetric airfoils that are used in general aviation (aircraft operating in subsonic and/or transonic flight regime) is contemplated [15].

The reconstruction of the airfoils is carried out through the use of the evolutionary algorithm Differential Evolution (DE), whose objective is to find the values of the design parameters of each method which provide the least geometric deviation when comparing the reconstructed profiles and the original ones [16]. The value of the geometric deviation is obtained by the error norm $L_2$:

$$\left\|y_n - y_{tar,n}\right\|_{L_2} = \sqrt{\sum_{n=1}^{P}\left(y_n - y_{tar,n}\right)^2}, \tag{3}$$

where $y_{tar}$ is the set of heights of the original points of the target profile; y is the set of heights of the reconstructed points; P is the number of points that make up the geometry of the airfoil.

### *2.3. Creation of a Database of Airfoils and Their Aerodynamic Coefficients*

#### 2.3.1. Geometries of the Airfoils

In the first instance, it is proposed to make use of the selected profiles from the UIUC database, in case these are not enough, the use of a Generative Adversarial Network (GAN) is suggested for the creation of new profiles, which will be based on the selected profiles from the UIUC database. GANs use a learning scheme that allows the defining attributes of the probability distribution to be encoded in a neural network, making it possible to generate instances that resemble the original probability distribution [17]. The GAN architecture is formed by two constituent neural networks: one called discriminator (D) and the other generator (G). The G network is responsible for generating new instances of the same domain as that of the source dataset. The network D is in charge of discriminating whether the input data are real, that is belonging to the input data set or whether they are fictitious, that is artificially generated. Both networks are trained together in such a way that G maximizes its chances of not being detected by D and D in such a way that it makes its methods of detecting the artificially generated data by G more and more sophisticated. These two adversarial networks compete in a zero-sum game in which it is hypothesized that they eventually reach a Nash equilibrium (see Figure 1) [18].

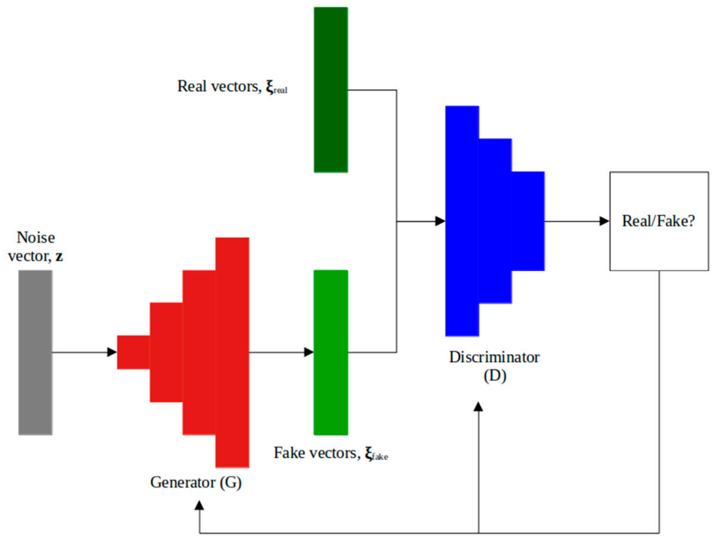


**Figure 1.** General architecture of the architecture of a GAN.

#### 2.3.2. Obtaining the Aerodynamic Coefficients

The aerodynamic coefficients of the profiles are obtained by means of Computational Fluid Dynamics, using the CFD package OpenFOAM 11 [19] and the GMSH mesh generator [20]. It is considered a viscous, turbulent, steady and incompressible flow.

$$\frac{\partial u_i}{\partial x_i} = 0, \tag{4}$$

$$u_j \frac{\partial u_i}{\partial t} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \nu \frac{\partial^2 u_i}{\partial x_j \partial x_j} - \frac{\partial u'_i u'_j}{\partial x_j}, \tag{5}$$

where $u_i$ is the average velocity; ν is the kinematic viscosity of the air; ϱ is the density of the air; p is the pressure; $u'_iu'_j$ is the Reynolds stress tensor.

To model the turbulence the k-ω SST model is considered. This is a two-equation turbulence model that is widely used to calculate the aerodynamic coefficients in subsonic and transonic flows. It combines the low Reynolds number version of the k-ω model and the high Reynolds number version of the k-ε model to obtain accurate results under a wide range of flow conditions. The k-ω

SST model is computationally more expensive than other two-equation models, but can provide more accurate results for more complex flows [21–23]. OpenFOAM uses the version of the k-ω SST model exposed by Menter in [24]. The turbulence specific dissipation rate (ω) equation is given by:

$$\frac{\partial(\rho\omega)}{\partial t}+\frac{\partial(\rho u_i\omega)}{\partial x_i}=\alpha\rho S^2-\beta\rho\omega^2+\frac{\partial}{\partial x_i}\left[(\mu+\sigma_{\omega1}\mu_t)\frac{\partial\omega}{\partial x_i}\right]+2(1-F_1)\rho\sigma_{\omega2}\frac{1}{\omega}\frac{\partial k}{\partial x_i}\frac{\partial\omega}{\partial x_i}, \tag{6}$$

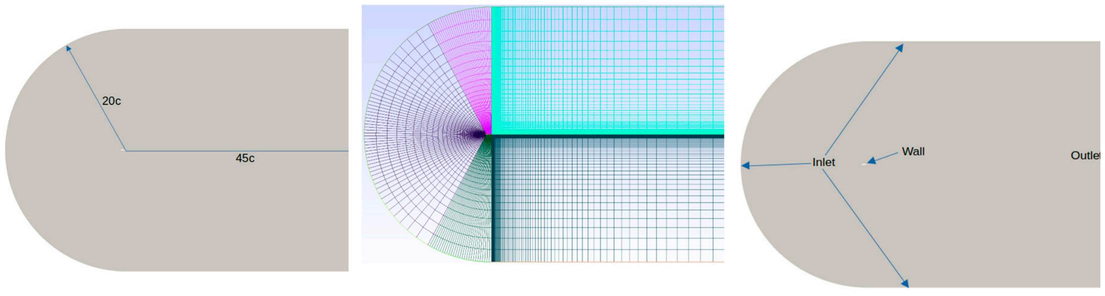
and the kinetic energy of turbulence (k) is defined by:

$$\frac{\partial(\rho k)}{\partial t}+\frac{\partial(\rho u_i k)}{\partial x_i}=\widetilde{P}_k-\beta^*\rho k\omega+\frac{\partial}{\partial x_i}\left[(\mu+\sigma_{k1}\mu_t)\frac{\partial k}{\partial x_i}\right], \tag{7}$$

where F1 is the blending function; S is the invariant measure of the strain rate; $\widetilde{P}_k$ is a production limiter. All constants are computed by a blend from the corresponding constants of the k - ε and the k - ω models via $\alpha = \alpha_1 F + \alpha_2(1 - F)$. The constants for this model are: $\beta^* = 0.09$, $\alpha_1 = 5/9$, $\beta = 3/40$, $\sigma_{k1} = 0.85$, $\sigma_{\omega1}=0.5$, $\alpha_2 = 0.44$, $\sigma_{\omega2} = 0.856$ [24].

The k-ω SST model allows the use of wall functions, which allows the use of meshes that are not so thin next to the wall (y+ values between 10 and 100) without losing as much precision [24]. By having this tolerance in the y+ values, greater flexibility is possible when developing the mesh.

For the creation of the mesh, a control volume with a type C configuration [25] (see Figure 2(a)) and the use of hexahedral elements is considered (OpenFOAM does not work with two-dimensional meshes). Each mesh has 21360 hexahedral elements, which allows y+ values close to 10 when using turbulent flows with Re values between 1 million to 6 million (see Figure 2(b)). Most of the boundary conditions used are shown in Figure. 2(c) . The "wall" condition is applied to the faces that limit the geometry of the profile; the "outlet" condition is assigned to the vertical face located to the right of the control volume; the "inlet" condition is assigned to the arc faces and the upper and lower horizontal faces of the control volume; finally, the "empty" condition is applied to the front and rear faces of the control volume. This last condition is what indicates to OpenFOAM that the simulation is two-dimensional [19]. The entire control volume is defined as "internal field".



(**a**) (**b**) (**c**)

**Figure 2.** (**a**) Dimensions of the control volume, c is the chord length of the profile; (**b**) General view of the mesh; (**c**) Boundary conditions of the control volume.

In the simulations, five physical fields are mainly evaluated: pressure (p) (in incompressible flow simulations, OpenFOAM, uses a specific pressure, p/ϱ [19]), velocity (U), turbulent kinematic viscosity ($\nu_t$), turbulent kinetic energy (k) and specific turbulence dispersion velocity (ω). Table 1 shows the initial conditions for each physical field at each boundary condition and for the internal field.

**Table 1.** Initial conditions.

| | airfoil | inlet | outlet | frontBack | internalField |
|---|---|---|---|---|---|
| U [m/s] | noSlip | $U_\infty$ | $U_\infty$ | empty | $U_\infty$ |
| p [$m^2/s^2$] | zeroGradient | zeroGradient | 0 | empty | 0 |
| k [$m^2/s^2$] | kqRWallFunction($k_0$) | $k_0$ | $k_0$ | empty | $k_0$ |
| ω [1/s] | omegaWallFunction($\omega_0$) | $\omega_0$ | $\omega_0$ | empty | $\omega_0$ |
| $\nu_t$ [$m^2/s$] | nutUWallFunction(0) | 0 | 0 | empty | 0 |

The values of ω and $\nu_t$ at the input, output and internal field are specified by default to work together with the wall function implemented on the profile wall [19,26]. The initial values of k are determined by the following equation [19]:

$$k_0 = 1.5(I\|U_\infty\|)^2, \tag{8}$$

where I is the intensity of the turbulence and $U_\infty$ is the velocity of the free flow.

The indicated solver for this type of simulations is the SIMPLE, in the version 11 of OpenFOAM this solver is called incompressibleFluid [19]. The recommended schemes for working with the SIMPLE solver and the k-ω SST turbulence model are as shown in Table 2.

**Table 2.** List of schemes used [19,22].

| Type of scheme | OpenFOAM scheme |
|---|---|
| Temporary derivatives | *steadyState* |
| Gradients | *Gauss linear* |
| Divergence(φ, U) | *bounded Gauss linearUpwind limited* |
| Divergence(φ, k) | *bounded Gauss upwind* |
| Divergence(φ, ω) | *bounded Gauss upwind* |
| Laplacians | *Gauss linear corrected* |
| Interpolation | *linear* |

φ denotes the (volumetric) flux of velocity on the cell faces for constant-density flows, for example Divergence(φ, U) indicates the advection of velocity.

### 2.3.3. Encoding of the Data

When training neural networks, it is often useful to make sure that the data is normalized at all stages of the network. Normalization helps to stabilize and accelerate the training of the network by means of a downward gradient. If the data has been scaled badly, the loss function value may become an undefined value (NaN) and the network parameters may diverge during training. Common ways to normalize data include rescaling data so that its interval is [0, 1] or that it has a mean of zero and a standard deviation of one [27,28].

For normalization of the input vectors, which are the design parameters of the airfoils, the MinMax scaler is used [29], where the transformation is given by:

$$\xi_{j,std} = \frac{\xi_j - \xi_{j,min}}{\xi_{j,max} - \xi_{j,min}}, \tag{9}$$

$$\xi_{j,scaled} = \xi_{j,std}(d_{max} - d_{min}) + d_{min}, \tag{10}$$

where $\xi_{j,min}$ and $\xi_{j,max}$ are the lower and upper limits of the calculated interval of the parameter $\xi_j$; $d_{min}$ and $d_{max}$ are new design intervals, 0 and 1, respectively.

For the case of the output data, the aerodynamic coefficients are represented in graphs, that is, images, which can be represented in matrices (three matrices representing the RGB layers), and whose components (pixels) have values between 0 and 1 (the original ranges are between 0 and 255, but the Python graphers also detect the range between 0 and 1 as valid) [30]. The graphs of interest are: endurance parameter vs angle of attack, pitch moment coefficient vs angle of attack and lift coefficient vs angle of attack. Each graph was normalized in images of 256 x 256 pixels, the axes and labels are eliminated, only the curve representing the values is left, the pixels that make up the curve have values of 1 (white color), while the pixels that make up the background have values of 0 (black color). Finally, the three images become layers of an RGB-encoded image (see Figure 3).

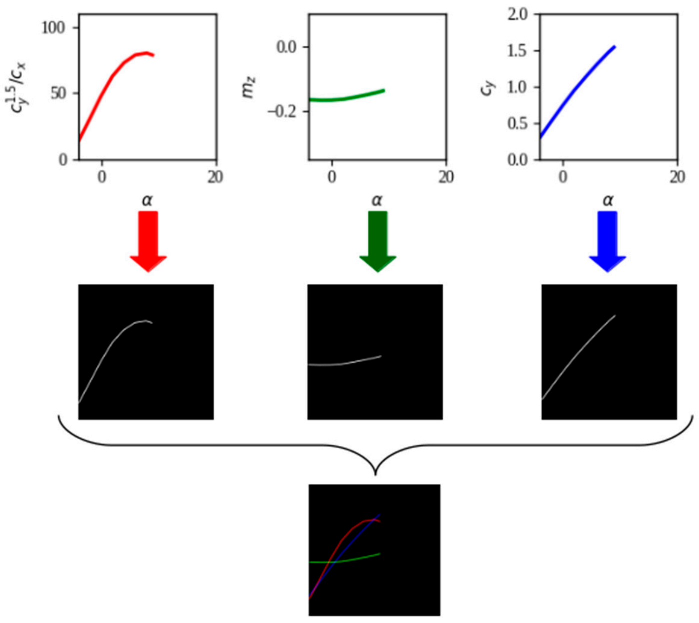


**Figure 3.** Creation of the output data (output images) representing the aerodynamic coefficients of the airfoils.

*2.4. Design of the Neural Network for the Prediction of Aerodynamic Coefficients of Airfoils*

The method for the design of the neural network is depicted in Figure 4, this method can be divided into the following three parts. 1 - the Principal Component Analysis (PCA) is used to perform the reduction of representative features of the output images and to obtain an approximation of the optimal dimension of the features involved in a Variational Autoencoder (VAE). 2 - VAE is used to extract representative features from the output images by unsupervised reconstruction-based learning. 3 - an Multi-layer Perceptron (MLP) network is developed to construct a non-linear mapping of the shapes of the aerodynamic surfaces to the features extracted by the VAE. The graphs of the aerodynamic coefficients of the profiles can be predicted by the composite network connecting the MLP and the decoder of the VAE [31].

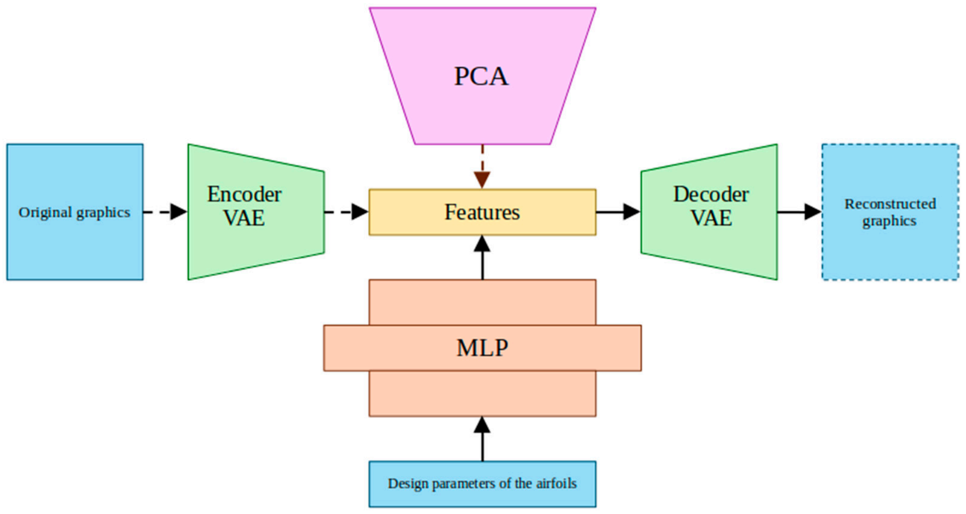


**Figure 4.** Methodology of design of a neural network to predict aerodynamic coefficients.

### 2.4.1. Approximation of the Representative Features of the Output Images

The main goal of PCA is to find the principal component (PC) space, which is used to transform information from the original higher-dimensional space to the lower-dimensional space. The PC space consists of k components that are orthonormal, uncorrelated and represent the directions of the maximum variance of the data. The first principal component of the PCA space represents the direction of the maximum variance of the data, the second component has the second largest variance, and so on [32]. For image compression, the PCA method is used together with the information loss method (LOSSY). In the LOSSY method, some of the redundant information of the original file is lost, that is, when the file is restored, only the most important information is restored [29,32].

### 2.4.2. VAE Configuration

A VAE is a model consisting of an encoder and a decoder (see Figure 5). The encoder, also known as the output model or the recognition model, approximates the posterior distribution of the hidden variables of the decoder. The decoder is a deep hidden variable model and is a variation of the generative model. Both components, the encoder and the decoder, are directional graphic models that are totally or partially parameterized by deep neural networks [33].

In the case of an automatic encoder (AE), with a given input of ζ, the encoder converts ζ into a hidden representation of **z** that has a smaller dimension compared to ζ. The decoder then uses **z** to reconstruct ζ'. The recovery loss function L is calculated by comparing the differences between ζ and ζ'. VAEs are probabilistic modifications of AEs that combine Bayesian variational inference and deep learning. The encoder serves as an inference model that attempts to infer the latent variables **z** from the input data ζ through a learned distribution q(**z**|ζ); the decoder serves as a generative model that generates similar input data ζ' under the given latent vector **z** by sampling from learned distribution p(ζ|**z**). Assuming that the posterior q(**z**|ζ) of the m latent variables is a diagonal Gaussian with mean **μ** and standard deviation **σ**, and the prior p(ζ|**z**) is a normal Gaussian with **μ**=0 and **σ**=1 [31,33]. The Kullback-Leiber divergence is used to reduce the gap between the prior and posterior distributions, i.e.,

$$\mathrm{KL}(q(z|\zeta)||p(\zeta||z)) = \frac{1}{2}\sum_{i=1}^{m}(e^{\sigma_i} - (\sigma_i + 1) + \mu_i^2), \tag{11}$$

where the vector **z** can be assembled as follows:

$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\varepsilon}, \tag{12}$$

where ε is a random vector of R(0, 1) and ⊙ denotes component multiplication.

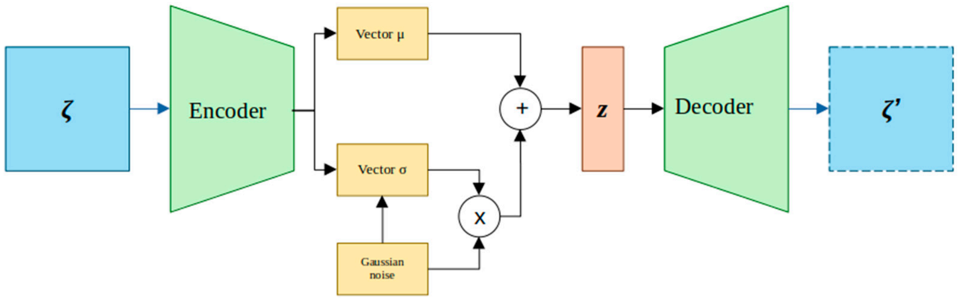


**Figure 5.** General architecture of a VAE.

The binary cross entropy recovery loss function BCE(ζ', ζ) is used to minimize the differences between the generated image and the original image, and is expressed as follows [30]:

$$\mathrm{BCE}(\boldsymbol{\zeta}', \boldsymbol{\zeta}) = -\boldsymbol{\zeta}\log(\boldsymbol{\zeta}') - (1-\boldsymbol{\zeta})\log(1-\boldsymbol{\zeta}'). \quad (13)$$

Therefore, the loss function to be minimized has the form of:

$$\mathrm{L}_{\mathrm{VAE}} = \mathrm{BCE} + \mathrm{KL}. \quad (14)$$

To minimize the loss function, network parameters are adjusted through back-forward propagation in every optimization iteration.

The proposed configuration for the encoder is composed of 6 convolutional layers and 6 MaxPooling layers (see Figure 6). After several blocks of convolutional residuals, the mean and standard deviation are extracted for each input image, in order to apply (12) and determine the **z** vector. The first 5 convolutional layers use the rectified linear unit (ReLU) activation function and the sixth layer uses the sigmoid activation function.

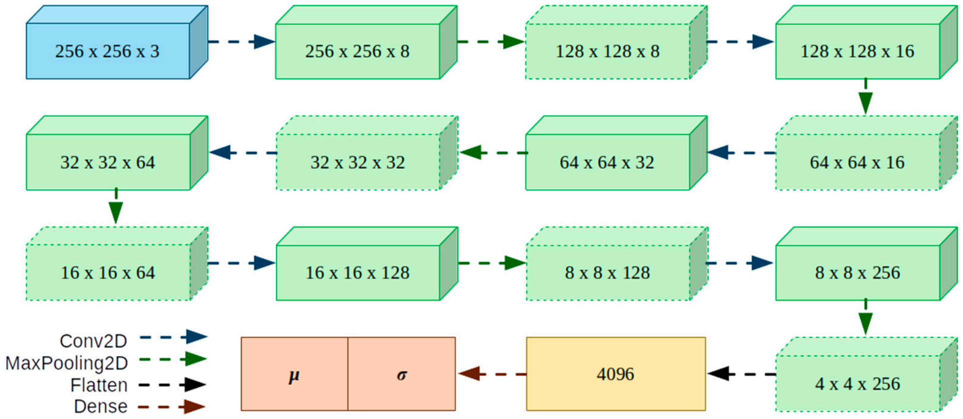


**Figure 6.** Architecture of the encoder.

The architecture of the decoder is similar to that of the encoder, as shown in Figure 7, with the only difference that the convolutional layers are replaced by deconvolutive layers (or transposed convolution layer) and the layers of the MaxPooling operation are replaced by UpSamplin layers. The first five deconvolutive layers use ReLU activation function, and the last layer uses the sigmoid activation function.

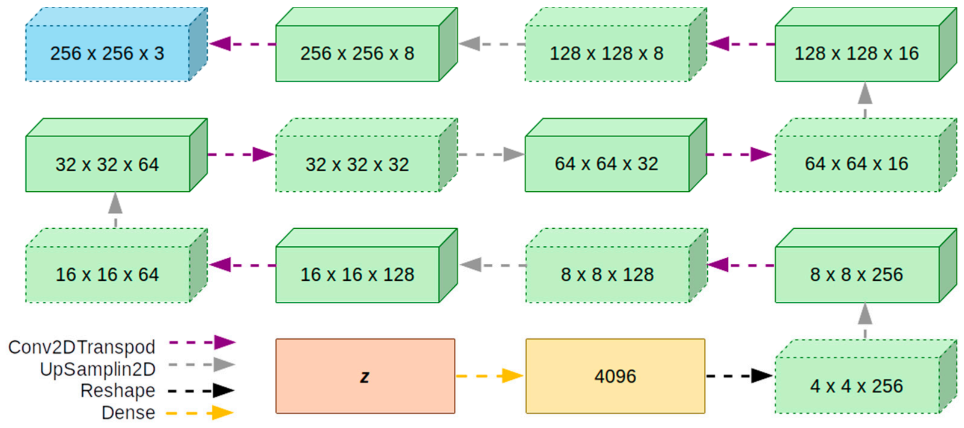


**Figure 7.** Architecture of the decoder.

To determine the performance in image reconstruction by the VAE, the mean absolute error (MAE) metric is used. The MAE describes the difference between the output data and the original input data [31]:

$$\mathrm{MAE} = \frac{1}{n_P}\sum_{p=1}^{n_P}\left|\zeta'_p - \zeta_p\right|. \quad (15)$$

where $n_P$ is the number of pixels of the images; $\zeta'_P$ is the value of the pixel p of the reconstructed image; $\zeta_P$ is the value of the pixel p of the real image.

Having the reconstructed images of the graphs of the aerodynamic coefficients, it is required to evaluate how efficient these images are to perform the readings of each coefficient in comparison to the original graphs. For this case, the use of the determination factor ($R^2$) as a metric is proposed. $R^2$ measures the fraction of variance of a dependent variable explained by independent variables. The values of $R^2$ vary between 0 and 1, the closer the values of $R^2$ to 1, indicates a better fit of the model [34–36].

### 2.4.3. MLP Configuration

The MLP is in charge of performing a non-linear mapping of the vector that indicates the characteristics associated with the graphs derived from the VAE and the normalized parameters of the airfoil parameterization method. The loss function to be minimized is the mean square error (MSE) between the difference between the **z** vectors provided by the encoder and the **z**' vectors predicted by the MLP [31]:

$$L_{MLP} = \mathrm{MSE}(\mathbf{z}',\mathbf{z}) = \frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\|z'_i - z_i\|^2\right). \quad (16)$$

The proposed MLP architecture has an input layer with j neurons (corresponding to the number of parameters of the airfoil parameterization method), an output layer with m neurons (corresponding to the size of the z vector), and three hidden layers. To determine the number of neurons and the activation function of each hidden layer an optimization algorithm is used, where the optimization task is defined as:

$$\min f_1(\boldsymbol{\eta}) = \mathrm{MSE}(\mathbf{z}', \mathbf{z}),$$
$$\text{such that } \boldsymbol{\eta} \in \Omega. \quad (17)$$

In this case $\boldsymbol{\eta}$ represents the vector formed by the parameters $n_1$, $n_2$, $n_3$ representing the number of neurons in each of the hidden layers respectively, and by $a_1$, $a_2$, $a_3$, representing the activation function in each hidden layer. The optimization algorithm chosen to solve this optimization task is the Integer Encoding Differential Evolution (IEDE) [34]. IEDE is a version of the standard DE algorithm where only the mutation operator is modified, so that it can operate with integer type values [37].

$$v_{i,j} = \begin{cases} \eta_{r1,j} + \text{int}\uparrow(\delta_{i,j}) & \text{if } r < \delta_{i,j} - \text{int}\uparrow(\delta_{i,j}) \\ \eta_{r1,j} + \text{int}\downarrow(\delta_{i,j}) & \text{otherwise} \end{cases}, \tag{18}$$

$$\boldsymbol{\delta}_i = F(\boldsymbol{\eta}_{r2} - \boldsymbol{\eta}_{r3}). \tag{19}$$

where F is the mutation factor; r is a randomly selected number based on the normal distribution; $\boldsymbol{\eta}_{r1}$, $\boldsymbol{\eta}_{r2}$, $\boldsymbol{\eta}_{r3}$ are three different vectors randomly selected in the current population; int↑(), int↓() are functions that round a number to the nearest upper and lower integer, respectively.

Having defined the architecture of the MLP, it is connected with the decoder of the VAE in order to have a neural network that is capable of predicting the graphs of the aerodynamic coefficients of a profile from a vector of design parameters (see Figure 8).

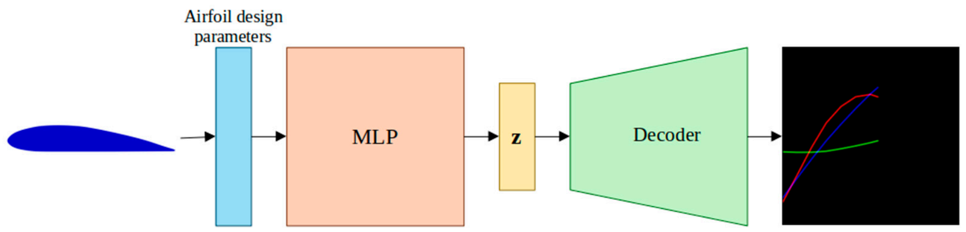


**Figure 8.** Final architecture of the neural network "AZTLI-NN" used to predict aerodynamic coefficients of the airfoils.

## 2.5. Optimization Algorithms

### 2.5.1. Single-Objective Optimization Algorithm

To solve single-objective optimization tasks, we opted to use the Success-History based Adaptive Differential Evolution (SHADE) algorithm [38]. This variant of the DE algorithm has the advantage that it adapts its evolutionary operators as it finds better values of the mutation and crossover factors, that is, it evolves as it searches for the optimal solution to the ongoing problem. Another advantage of the SHADE algorithm is that it can be adapted to the use of population size reduction (PSR) methods [38–40]. These methods help to solve part of the disadvantage of evolutionary algorithms, which is to reduce the number of objective functions to be evaluated in each generation, without having to lose performance in the search for an optimal global value. The variant of the SHADE algorithm to be used in this work incorporates the - Continuous Adaptive Population Reduction (CAPR) [41] method. The performance of the CAPR-SHADE algorithm has been tested in aerodynamic optimization tasks [42,43].

In CAPR-SHADE, the mutation factor F ∊ [0, 1] controls the magnitude of the differential mutation operator and CR ∊ [0, 1] is the crossover rate. This algorithm maintains a historical memory ($M_F$ and $M_{CR}$) with H entries for both control parameters F and CR. In the beginning, the contents of $M_{F,k}$, $M_{CR,k}$ (k = 1, ..., H) are initialized to 0.5. In each generation g, the control parameters $CR_i$ and $F_i$ used by each individual ri are generated by randomly selecting the index ri from [1; H], and then applying the formulas below:

$$CR_i = \begin{cases} 0 & \text{if } M_{CR,ri} = -1 \\ \text{randn}_i(M_{CR,ri}, 0.1) & \text{otherwise} \end{cases}, \tag{20}$$

$$F_i = \text{randc}_i(M_{F,ri}, 0.1). \tag{21}$$

Here, $\text{randn}_i(M, 0.1)$, $\text{randc}_i(M, 0.1)$ are randomly selected values of normal and Cauchy distributions respectively, with mean of M ($M_{CR}$ or $M_F$) and variance 0.1. In equation (20), in case a value for $CR_i$ outside [0, 1] is generated, it is replaced by the limit value (0 or 1) closest to the generated value. When $F_i > 1$, $F_i$ is rounded to 1, and when $F_i \leq 0$, (21) is applied repeatedly until a valid value is generated. In (20), if MCR,ri has been assigned the -1 value, $CR_i$ is set to 0.

The mutation strategy used by CAPR-SHADE is current-to-pbest/1 which is a variant of the current-to-best/1 strategy in which the operator is adjustable using a parameter p:

$$\mathbf{v}_{i,g} = \boldsymbol{\xi}_{i,g} + F_i\left(\boldsymbol{\xi}_{pbest,g} - \boldsymbol{\xi}_{i,g}\right) + F_i\left(\boldsymbol{\xi}_{r1,g} - \boldsymbol{\xi}_{r2,g}\right). \tag{22}$$

The individual $\xi_{pbest,g}$ is randomly selected from the upper p·NP members in the g-th generation ($p \in [0, 1]$). $F_i$ is the parameter F used by individual $\xi_i$. The elitism of the current-to pbest/1 depends on the control parameter p, which provides a balance between exploitation and exploration (small values of p indicate greater discrimination in the elite set). In case any component j of the mutated vectors $\mathbf{v}_i$ is out of the decision range, the value will be adjusted to the nearest extreme value.

The crossover operator uses a binary strategy, similar to the one used in DE standard:

$$u_{j,i,g} = \begin{cases} v_{j,i,g} & \text{if } \mathrm{rand}(0,1) \leq CR_i \\ \xi_{j,i,g} & \text{otherwise} \end{cases}. \tag{23}$$

The selection operator compares each individual $\xi_{i,g}$ with its corresponding trial vector $\mathbf{u}_{i,g}$, where the best one passes to the next generation:

$$\boldsymbol{\xi}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g} & \text{if } f(\mathbf{u}_{i,g}) \leq f(\boldsymbol{\xi}_{i,g}) \\ \boldsymbol{\xi}_{i,g} & \text{otherwise} \end{cases}. \tag{24}$$

In each generation, in (24), $CR_i$ and $F_i$ values that succeed in generating a trial vector $\mathbf{u}_{i,g}$ which is better than the parent individual $\xi_{i,g}$ are recorded as $S_{CR}$, $S_F$ , and at the end of the generation, the $M_{CR}$ and $M_F$ memories content are updated using Algorithm 1.

**Algorithm 1.** Memory update algorithm in SHADE [39].

```
    Input: $S_{CR}$, $S_F$, $M_{CR,k,g}$, $M_{F,k,g}$, k , H
    Output: $M_{CR,k,g+1}$, $M_{F,k,g+1}$, k
1   if $S_{CR} \neq \emptyset$ and $S_F \neq \emptyset$ then
2       if $M_{CR,k,g} = -1$ or $\max(S_{CR}) = 0$ then
3           $M_{CR,k,g+1} = -1$;
4       else
5           $M_{CR,k,g+1} = \mathrm{mean}_{WL}(S_{CR})$;
6       $M_{F,k,g+1} = \mathrm{mean}_{WL}(SF)$;
7       k++;
8       if k > H then
9           k = 1;
10  else
11      $M_{CR,k,g+1} = M_{CR,k,g}$;
12      $M_{F,k,g+1} = M_{F,k,g}$;
13  return Output
```

In Algorithm 1, the index k determines the position in the memory to be updated. In generation g, the element k in the memory is updated. At the beginning of the search, k is initialized to 1. k is incremented every time a new item is inserted in the history. If k > H, k is set to 1. In Algorithm 1 it is observed that when all individuals of generation g do not generate a trial vector that is better than the parent, i.e. $S_{CR} = S_F = \emptyset$, the memory is not updated. The weighted mean of Lehmer $\text{mean}_{WL}(S)$ is calculated as follows:

$$\text{mean}_{WL}(S) = \frac{\sum_{m=1}^{|S|} w_m S_m^2}{\sum_{m=1}^{|S|} w_m S_m}, \tag{25}$$

$$w_m = \frac{\Delta f_m}{\sum_{l=1}^{|S|} \Delta f_l}, \tag{26}$$

$$\Delta f_m = \left| f(\mathbf{u}_{m,g}) - f(\boldsymbol{\xi}_{m,g}) \right|, \tag{27}$$

The amount of improvement $\Delta f_m$ is used to influence the adaptation parameter S.

The CAPR method gradually reduces the size of the population according to the change in the gradient of the fitness value [41]:

$$NP_{g+1} = \begin{cases} \sqrt[\gamma]{\frac{\Delta_g}{\Delta_{g-1}}} & \text{if } 0 < \frac{\Delta_g}{\Delta_{g-1}} < 1 \\ NP_g & \text{otherwise} \end{cases}, \tag{28}$$

$$NP_{g+1} = \begin{cases} NP_{g+1} & \text{if } NP_{g+1} > NP_{min} \\ NP_{min} & \text{otherwise} \end{cases}, \tag{29}$$

where

$$\Delta_g = \frac{f_{avg}(\xi_g) - f_{avg}(\xi_{g-1})}{f_{avg}(\xi_g)}, \quad \Delta_{g-1} = \frac{f_{avg}(\xi_{g-1}) - f_{avg}(\xi_{g-2})}{f_{avg}(\xi_{g-1})}, \tag{30}$$

The above equations are used from the third generation, where $f_{avg}$ indicates the averaging of the values of the objective functions of all individuals in each generation.

The CAPR-SHADE algorithm as well as the standard DE algorithm are created to be used in unconstrained optimization tasks. A penalty function is used to convert a constrained optimization task to a unconstrained one. The penalty function is expressed as follows [44]:

$$L(\boldsymbol{\xi}) = \begin{cases} f(\boldsymbol{\xi}) & \text{if } \psi(\boldsymbol{\xi}) = 0 \\ R\psi(\boldsymbol{\xi}) + U^* & \text{if } \psi(\boldsymbol{\xi}) > 0 \text{ and } f(\boldsymbol{\xi}) \leq U^*, \\ R\psi(\boldsymbol{\xi}) + f(\boldsymbol{\xi}) & \text{if } \psi(\boldsymbol{\xi}) > 0 \text{ and } f(\boldsymbol{\xi}) > U^* \end{cases} \tag{31}$$

where

$$\psi(\boldsymbol{\xi}) = \sum_{k=1}^{K} \max\{0, g_k(\boldsymbol{\xi})\}, \tag{32}$$

where $U^*$ is the minimum value of the values of the objective functions of those vectors that satisfied the constraints. Initially, a large value is assigned to $U^*$, but the value must be updated with the current best-known function value at feasible points. Therefore, the initial $U^*$ is not updated until a feasible solution has been found. R is a penalty parameter that allows to regularize the obtained values of $\psi(\xi)$ and $U^*$. The constraints expressed as equalities ($h(\xi)$) have to be reformulated as inequalities ($g(\xi)$).

Using the penalty function, optimization tasks as expressed in (1) are expressed as:

$$\min L(\xi), \\ \text{such that } \xi \in \Omega. \tag{33}$$

### 2.5.2. Multi-Objective Optimization Algorithm

The Non-Dominated Classification Genetic Algorithm (NSGA-II) is a powerful decision space exploration engine based on Genetic Algorithm (GA) to solve multi-objective optimization problems

[45]. The philosophy of NSGA-II is based on four fundamental principles, which are: non-dominated classification, elite preservation operator, crowding distance and selection operator [45,46].

The non-dominated classification process begins with the assignment of the first rank to the non-dominated members (based on Pareto dominance) of the initial population. These first classified members are then placed on the first front and removed from the initial population. After that, the non-dominant classification procedure is performed on the remaining members of the population. In addition, the non-dominated members of the remaining population are assigned the second rank and placed on the second front. This process continues until all the members of the population are placed on different fronts according to their ranks, as shown in Figure 9(a). Algorithm 2 shows the process of the non-dominated classification.

**Algorithm 2.** Non-dominated classification [45].

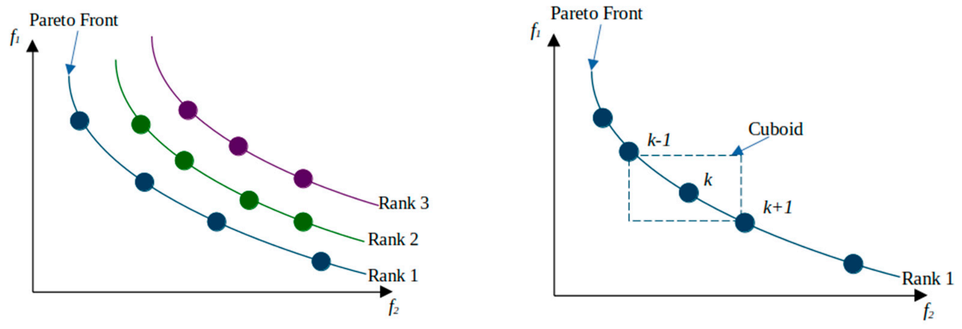
```
Input: $P_t$, $f_1(P_t)$, $f_2(P_t)$
Output: $F_i$
for p in $P_t$ do;
    $S_p = \emptyset$;
    $n_p = 0$;
    for q in $P_t$ do
        if $p \prec q$ then
            $q \rightarrow S_p$;
        else
            $n_p$++;
    if $n_p == 0$ then
        $p_{rank} = 1$;
        $p \rightarrow F_1$;
i = 1
while $F_i \neq \emptyset$ do
    $Q = \emptyset$;
    for p in $F_i$ do
        for q in Sp do
            $n_q = n_q - 1$;
            if $n_q == 0$ then
                $q_{rank} = i+1$;
                $q \rightarrow Q$;
    i++;
    $F_i = Q$;
return Output
```

The elite preservation strategy retains the elite solutions of a population by transferring them directly to the next generation. In other words, the non-dominated solutions found in each generation pass to the next generations until some new solutions dominate them.

The crowding distance (CD) is calculated to estimate the density of the solutions surrounding a particular solution. It is the average distance of two solutions on either side of the solution along each of the objectives (see Figure 9(b)). The CD of the k-th individual is defined as the average distance of two closest solutions on each side:

$$\mathrm{CD_k} = \sum_{l=1}^{2} \frac{f_{l,k+1} - f_{l,k-1}}{f_{l,max} - f_{l,min}}, \tag{34}$$

where $f_{l,max}$ and $f_{l,min}$ are the maximum and minimum values, respectively, of the l-th objective function among all vectors.



(**a**) (**b**)

**Figure 9.** (**a**) non-dominated classification; (**b**) crowding distance.

Algorithm 3 shows the process for determining the crowding distance on each front.

**Algorithm 3.** Crowding distance [45].

```
Input: F
Output: CD
1  $n_k$ = |F|;
2  CD = ∅;
3  $CD_1 = CD_{nk} = \infty$;
4  for k = 2 to $n_k$-1 do
5      Apply (34);
6      $CD_k \rightarrow$ CD;
7  return Output
```

The encoding of the individuals to be used in NSGA-II is floating-type. The genetic operators selected to operate with floating-type coding are: binary tournament selection, simulated binary crossover (SBX) operator and polynomial mutation.

Algorithm 4 describes the binary tournament selection.

**Algorithm 4.** Binary tournament selection.

```
Input: $P_t$, pt
Output: p
1  Select two vectors at random from $P_t$, $\xi_{r1} \neq \xi_{r2}$;
```

```
2   Select a random number r; from a normal distribution;
3   **if** r < pt **then**
4       **if** $\xi_{r1} \prec \xi_{r2}$ **then**
5           **p** = $\xi_{r1}$;
6       **else**
7           **p** = $\xi_{r2}$;
8   **else**
9       **if** $\xi_{r1} \prec \xi_{r2}$ **then**
10          **p** = $\xi_{r2}$;
11      **else**
12          **p** = $\xi_{r1}$;
13  **return Output**
```

The SBX operator simulates the principle of operation of the single-point crossing operator on binary chains [47].

$$\begin{aligned} c_{1,j} &= \frac{1}{2}\left(\left(1-\beta_j\right)p_{1,j} + \left(1+\beta_j\right)p_{2,j}\right) \\ c_{2,j} &= \frac{1}{2}\left(\left(1+\beta_j\right)p_{1,j} + \left(1-\beta_j\right)p_{2,j}\right)' \end{aligned} \tag{35}$$

where $c_1$ and $c_2$ are the child vectors of the parent vectors $p_1$ and $p_2$; the subscript j represents the components of each vector; and $\beta$ is obtained from:

$$\beta_j(u) = \begin{cases} (2u)^{\frac{1}{\eta_c+1}} & \text{if } u < 0.5 \\ \frac{1}{(2(1-u))^{\frac{1}{\eta_c+1}}} & \text{otherwise}' \end{cases} \tag{36}$$

where u is a randomly selected number from a normal distribution from 0 to 1; $\eta_c$ is the distribution index for the crossover.

The mutation operator uses the polynomial mutation strategy [48]:

$$v_{i,j} = c_{i,j} + (\Omega_j^u - \Omega_j^l)\delta_j, \tag{37}$$

where $\mathbf{v}_i$ is the mutated vector of the child vector $\mathbf{c}_i$; $\Omega^l_j$ and $\Omega^u_j$ are the minimum and maximum values of the design parameter j, respectively; $\delta_j$ is a small deviation that is calculated from:

$$\delta_j(u) = \begin{cases} (2u)^{\frac{1}{\eta_m+1}} & \text{if } u < 0.5 \\ \frac{1}{1-(2(1-u))^{\frac{1}{\eta_m+1}}} & \text{otherwise}' \end{cases} \tag{38}$$

where $\eta_m$ is the index of the mutation distribution.

The NSGA-II procedure is generally shown in Figure 10.

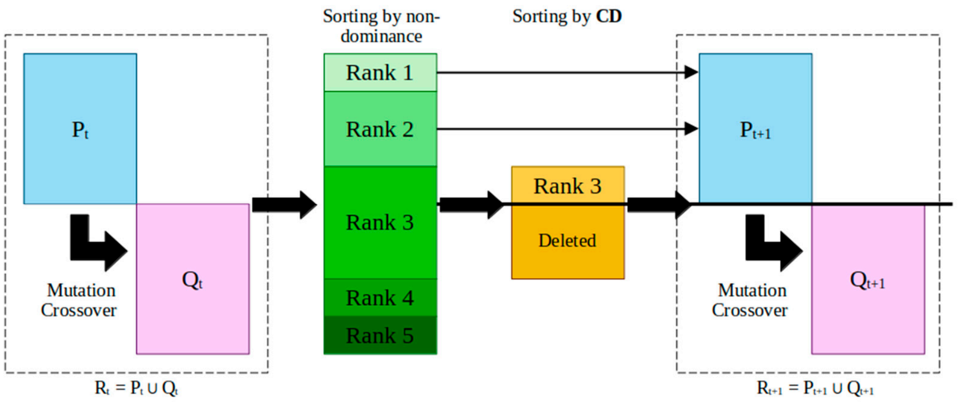


**Figure 10.** The procedure for conducting NSGA-II.

## 3. Results and Discussion

The calculations that were performed in each of the stages were performed with a computer with the following characteristics:

- hardware model - Gigabyte Technology Co., Ltd. B760 GAMING X;
- processor-13th Gen Intel® CoreTM i5-13400f x 12;
- RAM memory - 64 GB;
- GPU - NVIDIA GeForce RTX 4090;
- OS - Ubuntu 22.04.4 LTS.

### *3.1. Selection of the Method of Airfoils Parameterization*

In the airfoil reconstruction tests using the CST and Bezier-PARSEC methods, 322 airfoils extracted from the UIUC database were used. The DE variant “current-to-best/1” was selected as the optimization algorithm [49]. In the optimization tests a value of F = 0.85 was used, the crossover operation was omitted, an initial population of NP = 10D was taken (D is the size of the vector of design parameters) and a maximum of 200 generations were evaluated for each airfoil [16]. The decision spaces for each method are defined in Tables 3 and 4.

**Table 3.** Design intervals of the parameters of the CST method.

| Parameter | Design interval | Parameter | Design interval |
|---|---|---|---|
| $A_{u,0}$ | [0,07, 0,35] | $A_{l,0}$ | [-0,30, -0,05] |
| $A_{u,1}$ | [0,04, 0,55] | $A_{l,1}$ | [-0,26, 0,05] |
| $A_{u,2}$ | [0,00, 0,45] | $A_{l,2}$ | [-0,36, 0,05] |
| $A_{u,3}$ | [0,00, 0,55] | $A_{l,3}$ | [-0,47, 0,05] |
| $A_{u,4}$ | [0,00, 0,55] | $A_{l,4}$ | [-0,47, 0,05] |
| $A_{u,5}$ | [0,00, 0,50] | $A_{l,5}$ | [-0,42, 0,10] |
| $A_{u,6}$ | [-0,01, 0,50] | $A_{l,6}$ | [-0,28, 0,10] |

**Table 4.** Design intervals of the parameters of the Bezier-PARSEC method.

| Parameter | Design interval | Parameter | Design interval |
|---|---|---|---|
| $r_{le}$ | [-0.030, -0.001] | $\gamma_{le}$ | [-0,01, 0,32] |
| $x_t$ | [0,23, 0,50] | $x_c$ | [0,20, 0,85] |
| $y_t$ | [0,030, 0,095] | $y_c$ | [0,010, 0,065] |
| $k_t$ | [-0.9, -0.2] | $k_c$ | [-1,000, 0,025] |

| | | | |
|---|---|---|---|
| $\beta_{te}$ | [0,01, 0,40] | $\alpha_{te}$ | [0,01, 0,70] |

Figure 11 shows a histogram where the results obtained from the reconstruction tests of existing airfoils can be visualized.

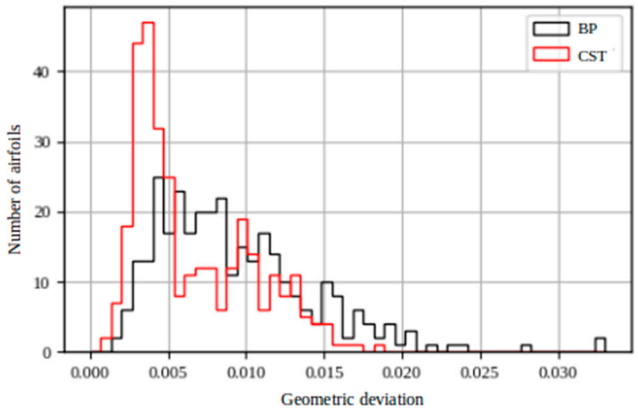


**Figure 11.** Results obtained from the airfoil reconstruction tests.

It can clearly be seen that the CST method was able to reconstruct a greater number of profiles with the smallest possible geometric deviation. Figure 12 shows 4 examples of profiles reconstructed with both methods, comparing them with the original profile. The graphs show the local deviations in the upper curve and in the lower curve of the profile. In all cases it is visualized that the CST Method achieves a better reconstruction of the profile.

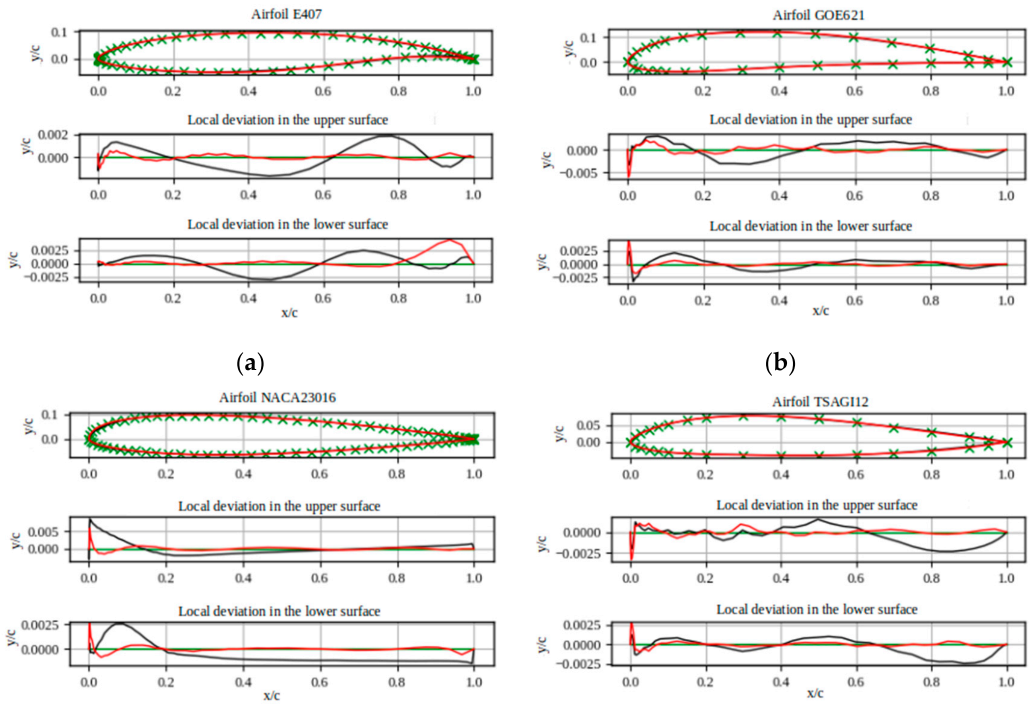


**Figure 12.** Airfoils reconstructed using the Bezier-PARSEC method (black) and the CST method (red). The original coordinates are shown in green. (**a**) Eppler 407 airfoil, (**b**) Gottingen 621 airfoil, (**c**) NACA airfoil of the 5-digit series NACA23016, and (**d**) TSAGI 12 airfoil.

Based on the results obtained, the CST method was selected to demonstrate better flexibility than the CST method, the only disadvantage that this model has is the lack of intuitiveness, since the parameters used in this method lack an aerodynamic meaning of the profile.

### *3.2. New Geometries of the Airfoils Using a GAN*

The architecture of the GAN that was used for the generation of new profiles is shown in Figure 13 and its specifications are shown in Tables 5 and 6. Because the profiles to be generated can be encoded as $\xi_{real}$ vectors (each vector consisting of the profile parameters CST), then the generator and the discriminator can be MLP. To achieve a better training of the GAN the design intervals of the CST parameters (see Table 3) were normalized between the values of -1 to 1.

**Table 5.** Generator (G).

| Layer | Number of neurons | Activation function |
|---|---|---|
| Input layer, IL | 5 | ————————— |
| Hidden layer 1, HL1 | 16 | Leaky RELU |
| Hidden layer 2, HL2 | 32 | Leaky RELU |
| Output layer, OL | 14 | Hyperbolic tangent |

**Table 6.** Discriminator (D).

| Layer | Number of neurons | Activation function |
|---|---|---|
| Input layer, IL | 14 | ————————— |
| Hidden layer 1, HL1 | 32 | Leaky RELU |
| Hidden layer 2, HL2 | 16 | Leaky RELU |
| Output layer, OL | 1 | Sigmoid |

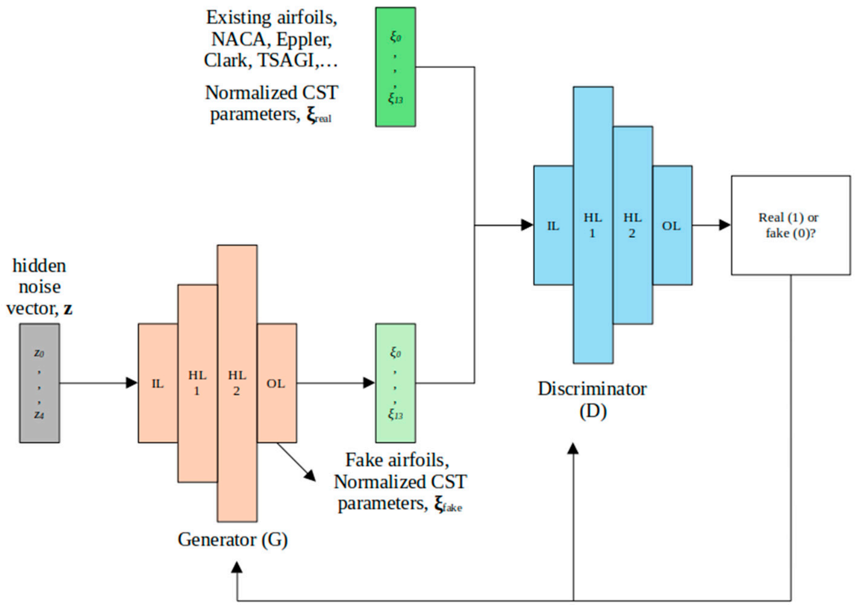

Existing airfoils, NACA, Eppler, Clark, TSAGI,…
Normalized CST parameters, $\xi_{real}$
$\xi_0$ , , , $\xi_{13}$
IL
HL 1
HL 2
OL
Real (1) or fake (0)?
hidden noise vector, z
$z_0$ , , , $z_4$
Fake airfoils, Normalized CST parameters, $\xi_{fake}$
Discriminator (D)
Generator (G)

**Figure 13.** Architecture of the GAN used to create new airfoils.

1200 epochs were required for training. Due to the fact that at this point of the research there are only 322 profiles, a convergence could not be guaranteed, for this reason the results were monitored every 200 epochs. In Figure 14 it can be seen that between 800 and 1000 or at the end of the 1200 evaluated epochs, the values of the loss function of the generator and the discriminator tend to be the same, in these epochs it is where it is required to save the weights of the neural network.

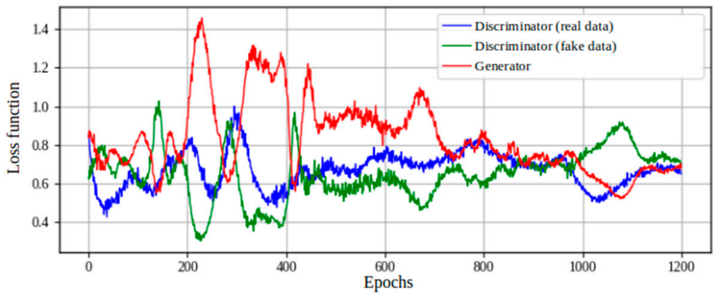


**Figure 14.** Training of the GAN.

Figure 15 shows geometries that are clearly similar to the airfoils used for the GAN training, of course there are airfoils that have no resemblance to the real airfoils, but still retain the geometric characteristics that can identify them as an airfoil.

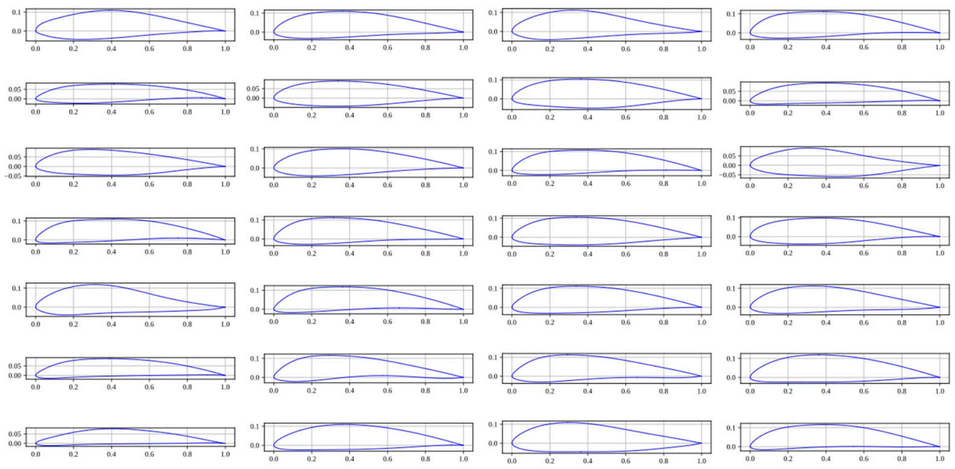


**Figure 15.** Airfoils obtained after 1200 learning epochs.

When analyzing 678 new airfoils, it was found that the values of the maximum profile thickness ($y_{t,max}$) are in the range from 0.066 to 0.17, and the values of the maximum curvature of the profile ($y_{c,max}$) are in the range from 0.0117 to 0.065. The ranges of the above parameters, for the real profiles, are $y_{t,max}$ from 0.06 to 0.19 and $y_{c,max}$ from 0.0055 to 0.065.

## *3.3. Validation of CFD Simulations*

To support the configuration of the simulation in OpenFOAM and the mesh generated in GMSH, the aerodynamic coefficients of the NACA 0012 profile with different Reynolds and Mach numbers were obtained, comparing them with the coefficients obtained experimentally in the laboratories of the National Aeronautics and Space Administration of the United States [50]. At the suggestion of OpenFOAM, the comparison between the experimental results and those obtained by CFD, the data obtained in the tests with fixed transition of the boundary layer should be used, the tests with grain size of 80, 120 and/or 180 are suggested [51]. The characteristics of the tests and the coefficients obtained at different Re and M are shown in detail in Table 7 and Figure 16.

**Table 7.** Characteristics of the simulated flows in the NACA 0012 airfoil.

| Test | M [Re] |
|---|---|
| 1 | 0.15 [2x$10^6$] |
| 2 | 0.15 [4x$10^6$] |
| 3 | 0.15 [6x$10^6$] |
| 4 | 0.30 [4x$10^6$] |
| 5 | 0.30 [6x$10^6$] |

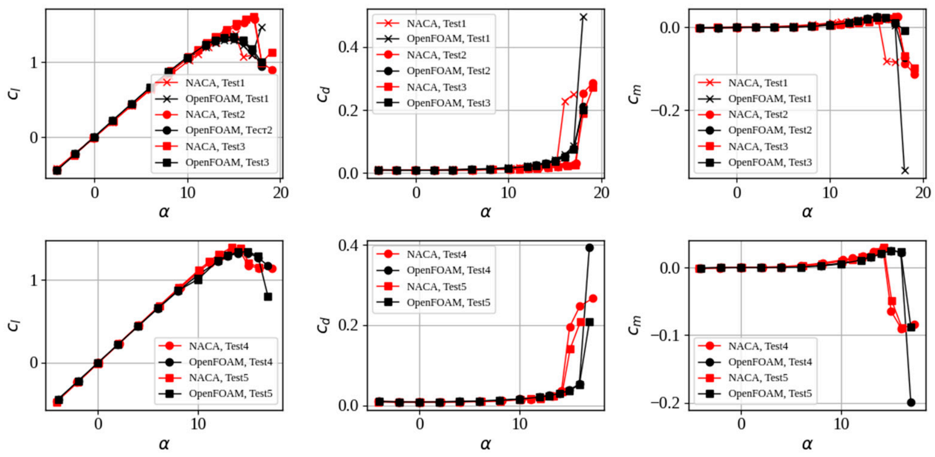


**Figure 16.** Aerodynamic coefficients of the NACA 0012 airfoil.

The results obtained using OpenFOAM are suitable for generating the aerodynamic coefficient database of the profiles. It is important to mention that in this work the CFD simulations will be limited until the maximum value of the endurance parameter is found, this because for the moment the evaluation of the profile loss region will be avoided due to its complexity.

## *3.4. Approximation of the Representative Features of the Output Images*

Figures 17, 18 and 19 show 4 tests with different numbers of PC (5, 10, 15, 20) applied to the three layers of one of the output images. From 10 PC, the shape of the curves can be recreated, only having variation in the color saturation in the pixels, which is not of great importance for reading values.

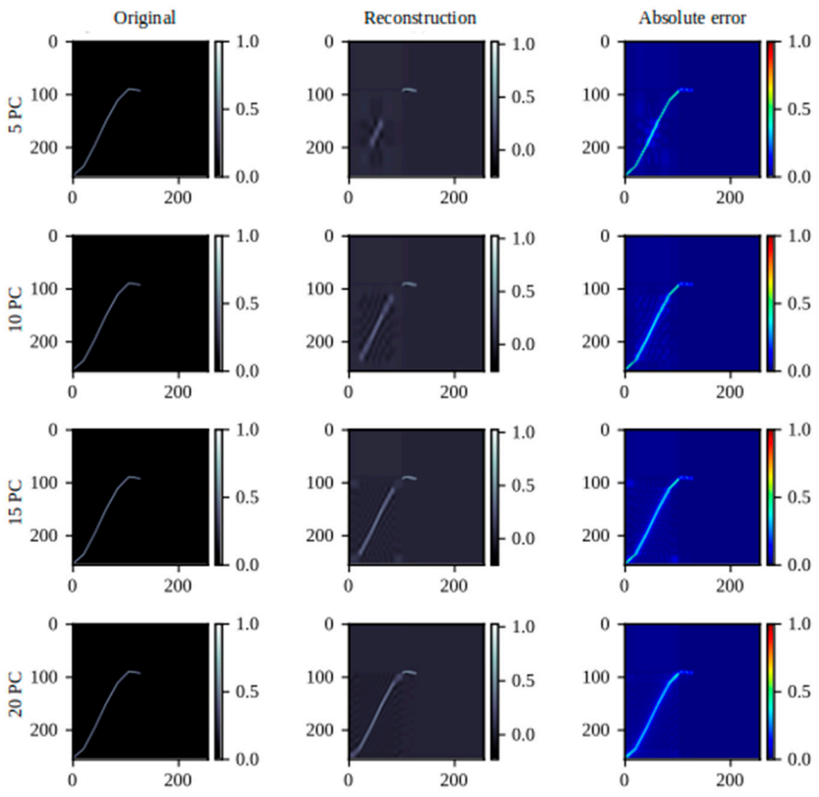


**Figure 17.** Reconstruction of the graph of $c_l^{1.5}/c_d$ vs $\alpha$ by PCA using different number of PCs.

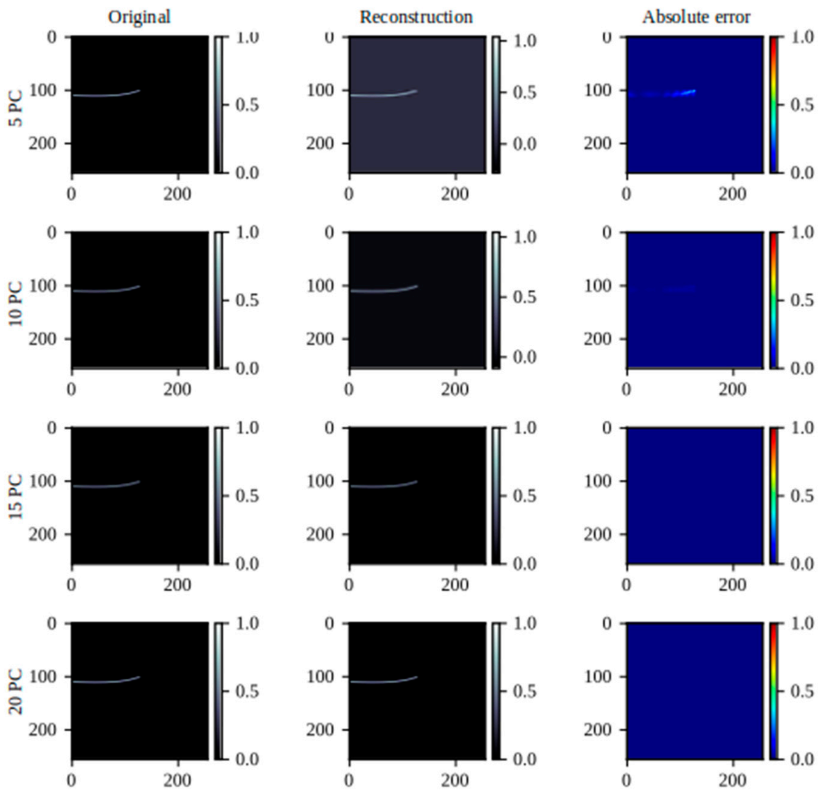


**Figure 18.** Reconstruction of the graph of $c_m$ vs $\alpha$ by PCA using different number of PCs.

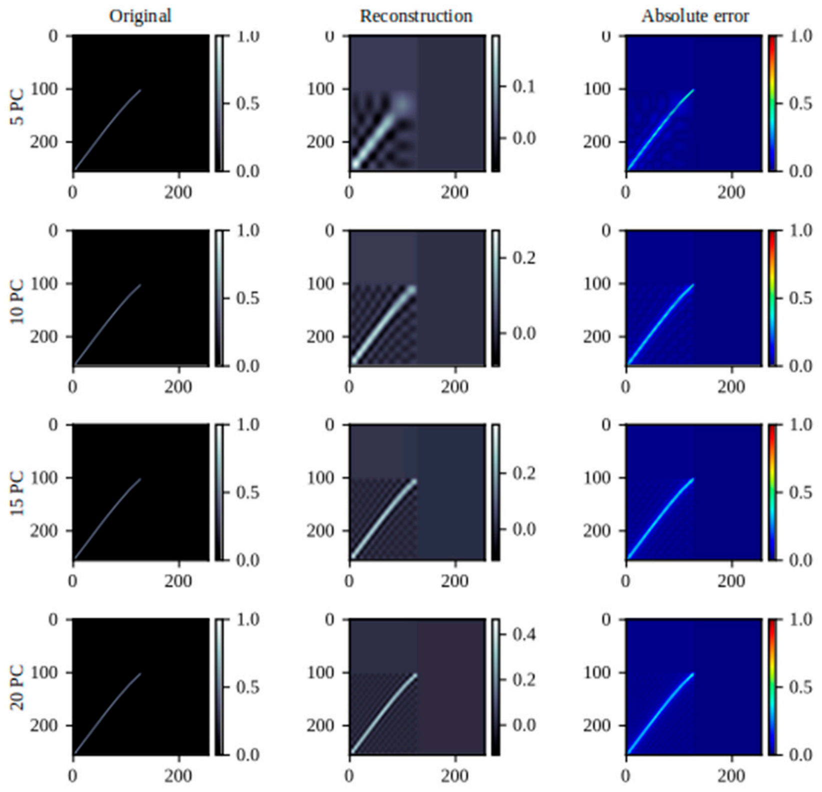


**Figure 19.** Reconstruction of the graph of cl vs $\alpha$ by PCA using different number of PCs.

## 3.5. Evaluation of the VAE

Considering a database of 1000 profiles, 90% of the data was selected for the VAE training and the remaining 10% was used for testing. In the PCA tests it was determined that at least 10 main components are required to be able to reconstruct the images. A VEA being a more robust model requires fewer components, for this reason 5 different sizes of the z vector will be tested, 5, 6, 7, 8 and 9. The results of the reconstruction of the images in both stages are shown in Figures 20 and 21 .

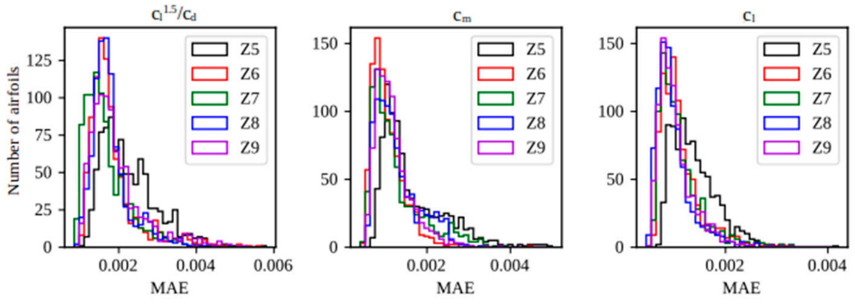


**Figure 20.** Statistical distribution of MAE in the reconstruction of graphs with a VAE. Stage of training.

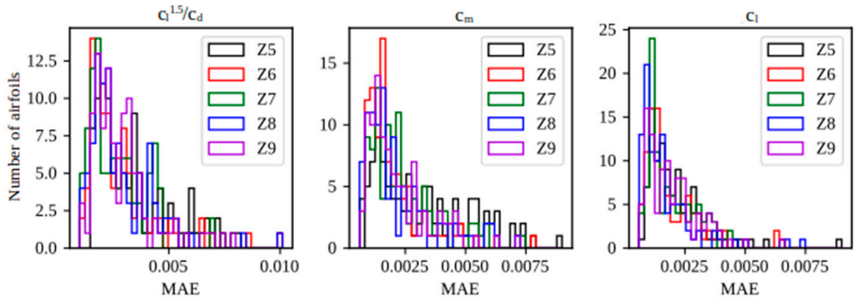


**Figure 21.** Statistical distribution of MAE in the reconstruction of graphs with a VAE. Stage of testing.

In the testing stage, the generalization capacity of a neural network is defined. The average MAE values of the test data for each **z** size are shown in Table 8. The best values are highlighted in green. The **z** vectors of size 7 and 8 come to generalize in a similar way, in this case the vector of smaller size is chosen.

**Table 8.** Average values of MAE in the reconstruction of graphs in the VAE testing stage.

| Size of z | MAE $c_l^{1.5}/c_d$ | MAE $c_m$ | MAE $c_l$ | $MAE_{avg}$ |
|---|---|---|---|---|
| **9** | 0.00291 | 0.00218 | 0.00196 | 0.00235 |
| **8** | 0.00283 | 0.00220 | 0.00165 | 0.00223 |
| **7** | 0.00281 | 0.00223 | 0.00185 | 0.00223 |
| **6** | 0.00294 | 0.00203 | 0.00199 | 0.00232 |
| **5** | 0.00341 | 0.00331 | 0.00212 | 0.00294 |

An example of graph reconstruction using the VAE with a 7-component z vector is shown in Figure 22. Figure 23 shows a linear fit analysis between the read values of the aerodynamic coefficients obtained from the real graphs (those obtained from the CFD simulations) and the graphs reconstructed by the VAE. The values of $R^2$ are indicated for each case, the ideal fit line is indicated (when the predicted values (P) are equal to the target values (T)) and the line of the best linear fit (BLF) of the model.

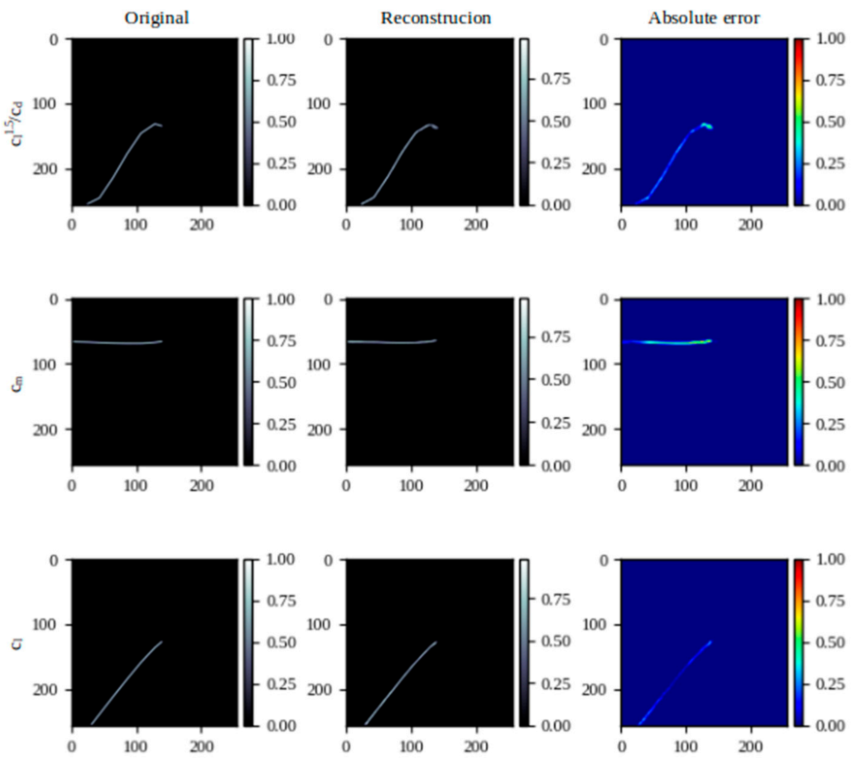


**Figure 22.** Reconstruction of the graphs of the aerodynamic coefficients of the Eppler 502 airfoil (Re = 1.5x106, M = 0.15).

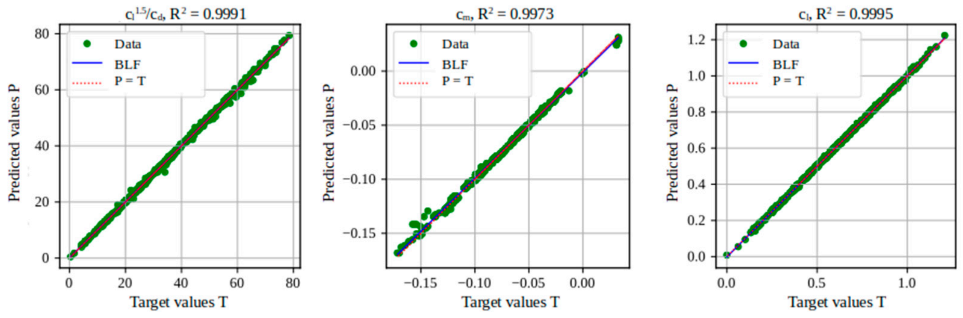


**Figure 23.** Analysis of the performance in the reading of aerodynamic coefficients in the graphs reconstructed by the VAE.

### *3.6. Design of the MLP and Performance Evaluation of AZTLI-NN*

To solve the optimization task shown in (17), the decision space Ω shown in Table 9 is considered.

**Table 9.** Parameters and their design ranges to optimize MLP hyperparameters.

| η | Design ranges |
|---|---|
| $n_1$ | |
| $n_2$ | [128; 512] |
| $n_3$ | |
| $a_1$ | |
| $a_2$ | {0; 1; 2}* |
| $a_3$ | |

*0 - hyperbolic tangent, 1 – RELU, 2 – Leaky RELU.

In the optimization task the output layer has a sigmoidal activation function, the optimizer is ADAM, out of 90% of the data assigned for training 10% is assigned for validation during training, and 500 epochs were considered. The IEDE algorithm has a F = 0.85, the population size of 30, and 30 generations were evaluated. The optimal architecture is shown in Figure 24. In the input layer there are 14 neurons, corresponding to the parameters of the CST method, the first hidden layer has 161 neurons, the second with 320 and the third layer with 511, finally the output layer has 7 neurons corresponding to the size of the z vector of the VAE.

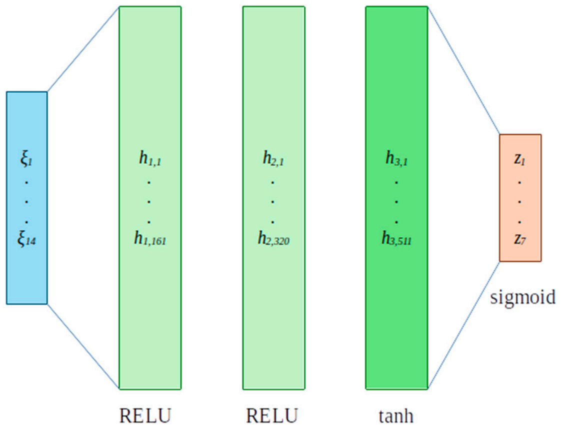


**Figure 24.** The Architecture of MLP.

The results of $R^2$ of the coefficient reading with the graphs predicted by the AZTLI-NN assembled network are shown in Figure 25.

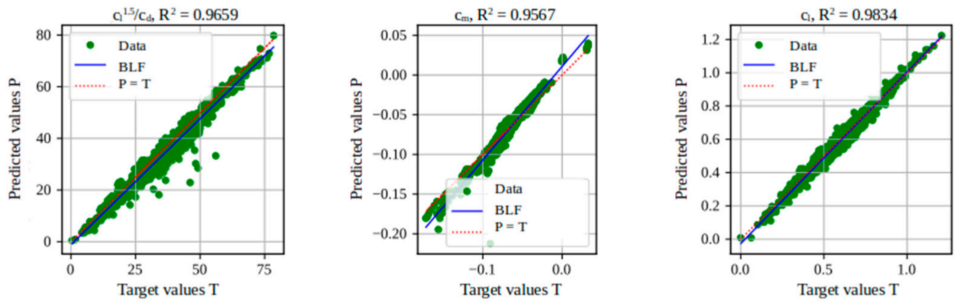


**Figure 25.** Performance analysis in predicting aerodynamic coefficients using AZTLI-NN (1000 data).

Statistically, it can be considered that the neural network achieves a good generalization, but at present, many of the neural networks dedicated to the prediction of aerodynamic coefficients achieve values of $R^2$~0.99. To analyze whether it is possible to increase the accuracy of the neural network, the number of data was increased to 1500 with the help of the GAN. As the number of training data increased, the predictions improved, the R2 value for $c_l^{1.5}/c_d$ improved by 2.2%, for the $c_m$ improved by 3.2%, and for the $c_l$ improved by 1.2%. In all three cases the $R^2$ values were closer to 0.99, and even for the prediction of $c_l$ a value greater than 0.99 was obtained (see Figure 26).

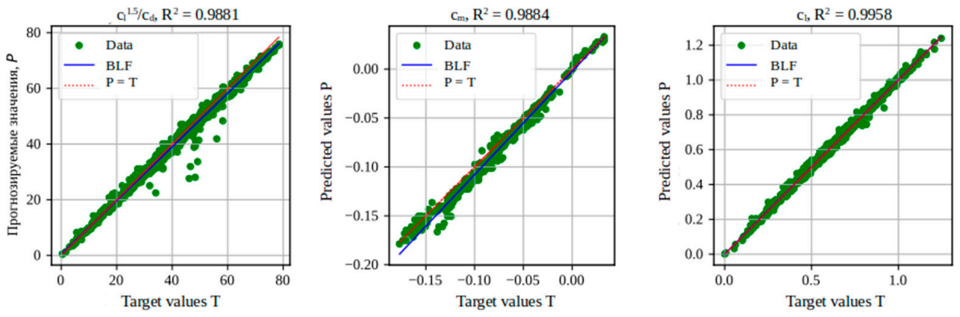


**Figure 26.** Performance analysis in predicting aerodynamic coefficients using AZTLI-NN (1500 data).

From the profiles used for testing, information was sought about experimental tests with similar characteristics to which the neural network was trained. Information on experimental tests of the FX 66-S-161 profile was found in [52]. The experimental tests were developed in a laminar wind tunnel with Re = $1.5\text{x}10^6$ and M = 0.25. The aerodynamic coefficients corresponding to the FX 66-S-161 profile are shown in Figure 27.

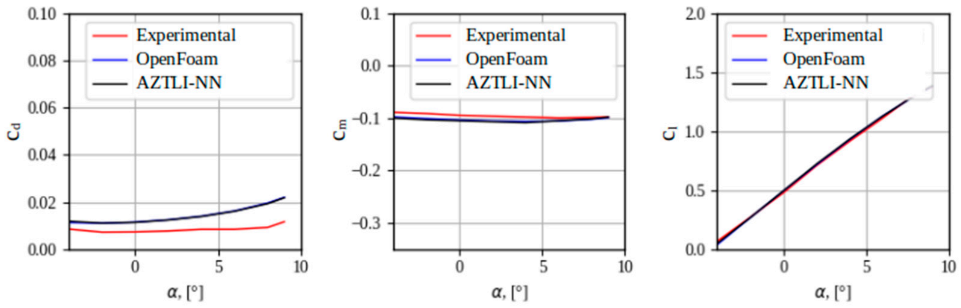


**Figure 27.** Aerodynamic coefficients of the FX 66-S-161 profile obtained with a laminar wind tunnel, with OpenFOAM and with AZTLI-NN.

## *3.7. Solving the First Optimization Case*

The first proposed optimization case is described in detail as follows: during the conceptual design process of an aircraft it is necessary to find an aerodynamic profile that has the following characteristics: design lift coefficient, $c_{l,d}$ = 0.59; minimum maximum thickness of the airfoil, $y_{t,min}$ = 11%; maximum permissible angle of attack, $\alpha_{max}$ = 4°. The flow conditions are: Re = $1.5\text{x}10^6$ and M = 0.15. The optimization algorithm used to solve this task is shown in Appendix C.

In the first instance, the convergence of the algorithm was evaluated based on the number of generations, for this an initial population size $NP_0 = 10|\xi|$, a minimum population $NP_{min}$ = 4 (recommended in [41]) was proposed. 500 generations were evaluated. The test was performed 5 times to analyze the repeatability of the results. The results of the 5 tests performed are shown in Figure 28 .

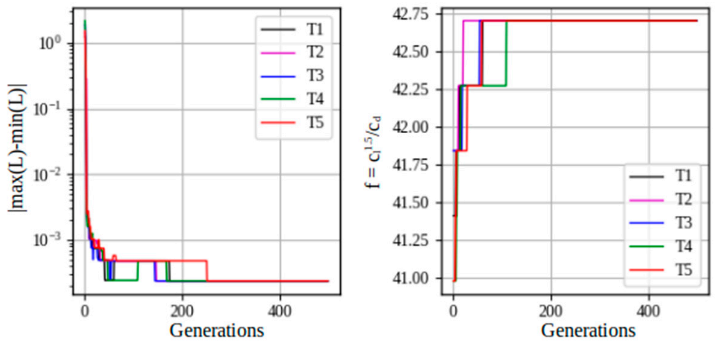


**Figure 28.** Convergence analysis of the CAPR-SHADE algorithm.

As can be visualized in Figure 29 the algorithm requires less than 100 generations to converge. In the next series of tests, a value of 100 generations was maintained to be evaluated. In the following series of tests the size of the initial population was varied, with sizes of 10|ξ|, 20|ξ| and 50|ξ|. The results of this series of tests are shown in Table 10. The 15 airfoils obtained in the different tests are shown in Figure 29 .

**Table 10.** Repeatability analysis of the obtained values obtained by the optimization algorithm.

| $NP_0$ | Test | $c_l^{1.5}/c_d$ | α [°] | $y_{tmax}$ | $c_d$ | $c_m$ |
|---|---|---|---|---|---|---|
| 10\|ξ\| | 1 | 42.7059 | 3.9 | 0.1127 | 0.0106 | -0.0342 |
| | 2 | 42.7059 | 3.9 | 0.1102 | 0.0106 | -0.0342 |
| | 3 | 42.7059 | 3.9 | 0.1136 | 0.0106 | -0.0323 |
| | 4 | 42.2745 | 3.9 | 0.1148 | 0.0107 | -0.0342 |
| | 5 | 42.7059 | 3.9 | 0.1100 | 0.0106 | -0.0323 |
| 20\|ξ\| | 1 | 42.7059 | 3.9 | 0.1148 | 0.0106 | -0.0341 |
| | 2 | 42.7059 | 3.9 | 0.1136 | 0.0106 | -0.0376 |
| | 3 | 42.7059 | 3.9 | 0.1117 | 0.0106 | -0.0341 |
| | 4 | 42.7059 | 3.9 | 0.1118 | 0.0106 | -0.0341 |
| | 5 | 42.7059 | 3.9 | 0.1122 | 0.0106 | -0.0341 |
| 50\|ξ\| | 1 | 42.7059 | 3.9 | 0.1158 | 0.0106 | -0.0341 |
| | 2 | 42.7059 | 3.9 | 0.1117 | 0.0106 | -0.0341 |
| | 3 | 42.7059 | 3.9 | 0.1113 | 0.0106 | -0.0359 |
| | 4 | 42.7059 | 3.9 | 0.1127 | 0.0106 | -0.0341 |
| | 5 | 42.7059 | 3.9 | 0.1104 | 0.0106 | -0.0359 |

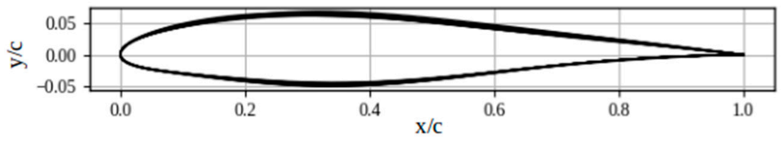


**Figure 29.** Optimal airfoils obtained in different tests.

The optimal airfoils that obtained the maximum and minimum value of $y_{tmax}$ are shown in Figure 30 . The aerodynamic coefficients of both profiles obtained with AZTLI-NN and with OpenFOAM are compared in Table 11.

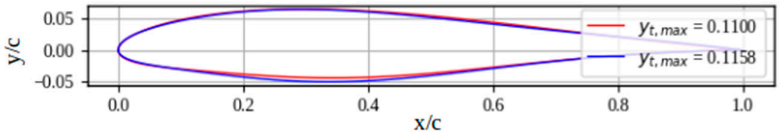


**Figure 30.** Optimal airfoils with maximum and minimum $y_{tmax}$.

**Table 11.** Comparison of the aerodynamic coefficients of optimal profiles obtained using OpenFOAM and AZTLI-NN.

| Airfoil | Method | α [°] | $c_l$ | $c_d$ | $c_m$ | $c_l^{1.5}/c_d$ | $y_{t,max}$ |
|---|---|---|---|---|---|---|---|
| Kfoil_1 | OpenFOAM | 3,9 | 0,5847 | 0,0109 | -0,0293 | 41,0178 | 0,1100 |
| Kfoil_1 | AZTLI-NN | 3,9 | 0,59 | 0,0106 | -0,0323 | 42,7059 | 0,1100 |
| Kfoil_2 | OpenFOAM | 3,9 | 0,5801 | 0,0110 | -0,0303 | 40,1662 | 0,1158 |
| Kfoil_2 | AZTLI-NN | 3,9 | 0,59 | 0,0106 | -0,0341 | 42,7059 | 0,1158 |

## *3.8. Solving the Second Optimization Case*

The second proposed optimization case is described in detail as follows: during the conceptual design process of an aircraft wing, it is necessary to provide profile variants that provide high values of $c_l^{1.5}/c_d$ and close to zero values of $c_m$; and a design lift coefficient ($c_{l,d}$) of 0.59 is considered. The flow conditions are: Re = 1.5x10$^6$ and M = 0.15.

The default values for NSGA-II of the population size (NP = 100) and the number of generations (T = 250) were used [45]. This test was performed 5 times to analyze the repeatability of the algorithm shown in Appendix D.

Figure 31 shows how the Pareto front of each of the 5 tests evolves. After evaluating 250 generations, it can be observed that the Pareto boundaries tend to have the same shape, which implies that there is repeatability in the results.

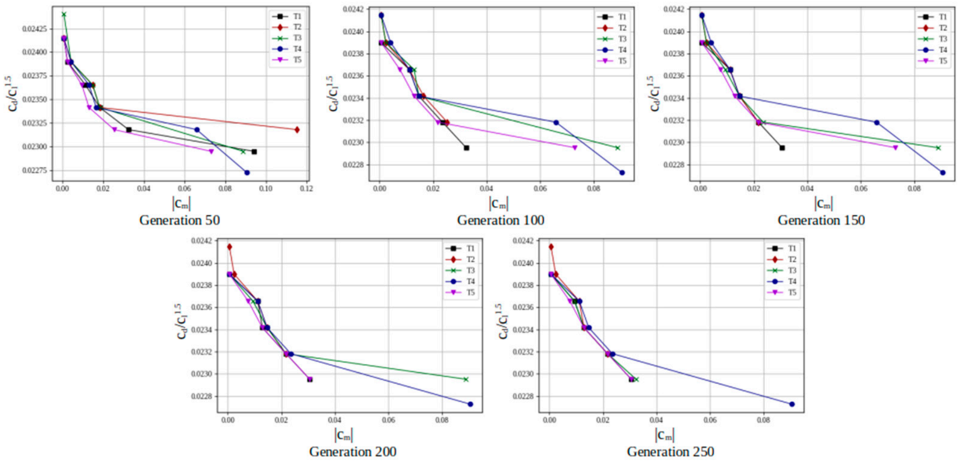


**Figure 31.** Evolution of the Pareto front of the 5 optimization tests.

The computation time of each test is approximately 120 s.

The best Pareto boundary obtained is shown in Figure 32 (In test 5, a boundary with the highest number of dominant individuals was obtained). In addition, the geometries of the airfoils that are part of the Pareto front are shown in Figure 32.

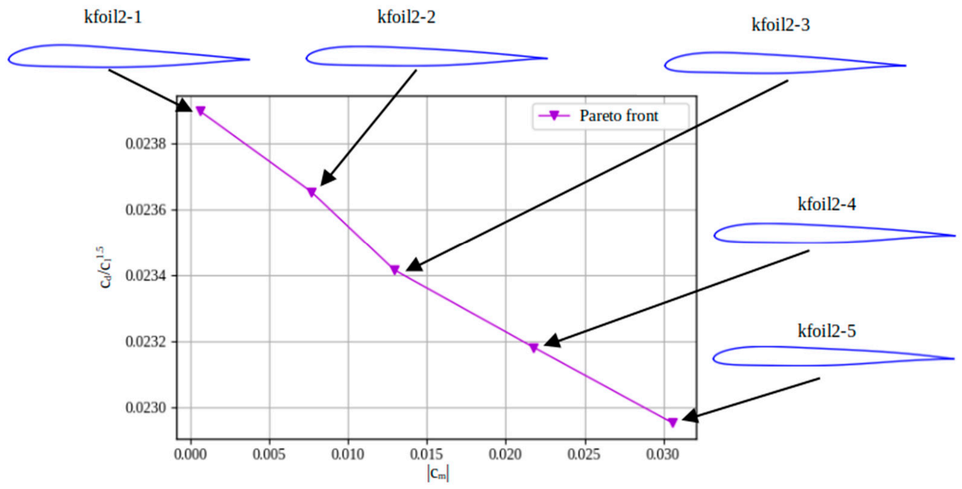


**Figure 32.** Best Pareto front obtained in the tests.

Figure 33 shows in more detail the comparison of the geometries of the airfoils shown in Figure 32 . It can be seen that the position of the maximum thickness (~25% of the chord) and the position of the maximum camber (~0.20 of the chord) of the airfoils are similar. The main difference between the airfoils is shown in the camber at the trailing edge of the airfoil. As $|c_m|$ decreases (or $c_l^{1.5}/c_d$ decreases) the camber decreases more rapidly at the trailing edge of the airfoil.

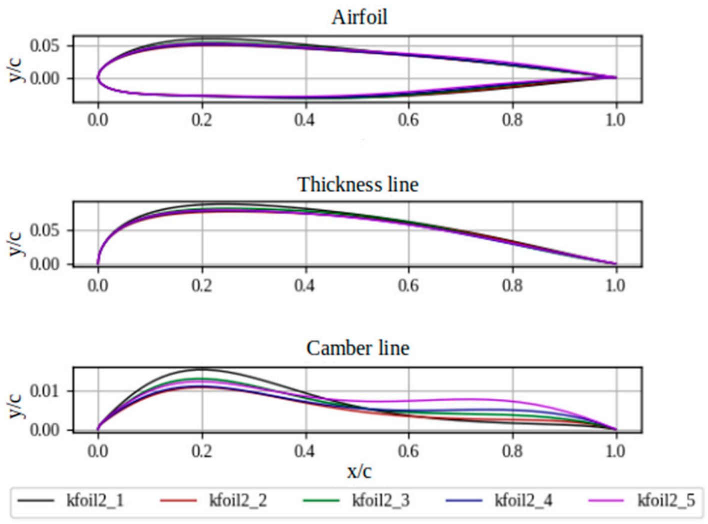


**Figure 33.** Comparison of the geometries of the Pareto front profiles.

The graphs of the aerodynamic coefficients of each of the airfoils of the Pareto front are shown in Figure 34. Among the most remarkable things that can be observed in Figure 34 is the following: in the graph of $c_l$ vs $\alpha$ it can be observed that all the airfoils have a linear behavior within the range of 0 to 10 degrees; the maximum values of $c_l^{1.5}/c_d$ of the airfoils are between 8 and 9 degrees; from 0 to 7 degrees the $c_m$ values of each airfoil do not show a significant change.

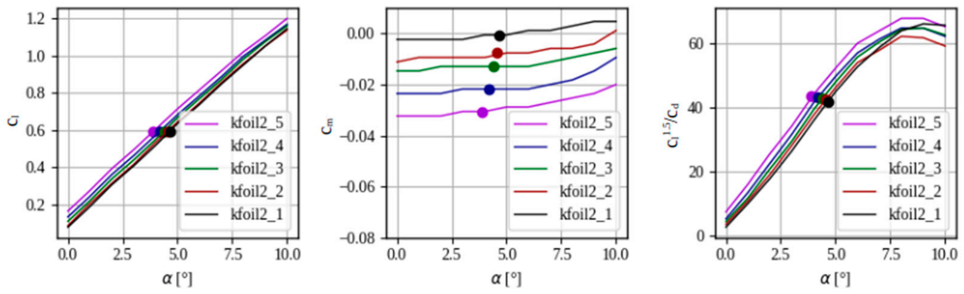


**Figure 34.** Aerodynamic comparison of the Pareto front airfoils.

In the graphs of Figure 34 the design condition is indicated with a dot ($c_{l,d}$ = 0.59), in Table 12 the results obtained under this design condition are shown in more detail, in addition it is compared with the aerodynamic coefficients obtained by OpenFOAM.

**Table 12.** Aerodynamic characteristics of the Pareto front profiles under the condition of design.

| Airfoil | α [°] | $c_l$ | $c_d$ | $c_m$ | $c_l^{1,5}/c_d$ | $y_{c,max}$ | $y_{t,max}$ |
|---|---|---|---|---|---|---|---|
| kfoil2_1 | 4,6 | 0,5900<br>0,5933* | 0,0109<br>0,0110* | -0,0006<br>-0,0029* | 41,5431<br>41,5450* | 0,0154 | 0,0888 |
| kfoil2_2 | 4,5 | 0,5900<br>0,5907* | 0,0107<br>0,0108* | -0,0079<br>-0,0117* | 42,2745<br>42,0365* | 0,0108 | 0,0776 |
| kfoil2_3 | 4,3 | 0,5900<br>0,5905* | 0,0106<br>0,0107* | -0,0129<br>-0,0150* | 42,7059<br>42,4078* | 0,0130 | 0,0823 |
| kfoil2_4 | 4,1 | 0,5900<br>0,5878* | 0,0105<br>0,0107 | -0,0218<br>-0,0231* | 43,1373<br>42,5146* | 0,0111 | 0,0791 |
| kfoil2_5 | 3,9 | 0,5900<br>0,5930* | 0,0104<br>0,0106* | -0,0305<br>-0,0303* | 43,5686<br>43,0800* | 0,0123 | 0,0800 |

*values obtained using OpenFOAM.

## 4. Conclusions

As part of this work, a methodology was developed to generate optimization algorithms, aimed at the aerodynamic optimization of airfoils under conditions of subsonic, incompressible and turbulent flow. In the development process, metaheuristic algorithms (specifically evolutionary algorithms) and different artificial neural network architectures were evaluated. The purpose of the algorithms developed is that they can be used in the early stages of aircraft design with high elongation and long flight duration. To achieve the established goal, the following research and developments were carried out:

1. An artificial neural network architecture, AZTLI-NN, (composed of a multilayer perceptron and a variational autoencoder) was developed for aerodynamic response prediction. A new image-based coding was proposed for the output parameters of the neural network, the neural network has the ability to generate the diagrams of the aerodynamic coefficients as a function of the angle of attack. For the training of the neural network, a database of wing airfoils was generated with their respective aerodynamic coefficients (lift coefficient, drag coefficient, pitch moment coefficient) using computational aerodynamics. The procedure of the numerical simulations was validated with experimental cases. Airfoil parameterization methods were evaluated to determine which provided the best performance when reconstructing wing airfoils.
2. Evolutionary algorithms of mono-objective and multi-objective optimization were adapted so that they could be used in conjunction with the AZTLI-NN network.
3. The performance of each of the algorithms was evaluated. Several tests were carried out to evaluate the repeatability of the results and the consistency in the computation times. In both

cases, repeatability of the results was obtained, and the computation times are suitable for the algorithms to be considered in early-stage design processes.

Therefore, the work carried out achieves the practical importance raised, since not only was it possible to meet the established objective, but also provided a background to achieve the development of neural network architectures and adapt evolutionary algorithms to be used in optimization tasks of airfoils aimed at different tasks. For example, the proposed methodology can be used in the design of aircraft operating in different flight regimes, in the design of propellers, in the design of wind turbines, etc.

**Author Contributions:** Conceptualization, J.G.Q.P.; methodology, J.G.Q.P.; software, J.G.Q.P.; validation, J.G.Q.P.; formal analysis, J.G.Q.P.; investigation, J.G.Q.P.; resources, E.K., and O.L.; data curation, J.G.Q.P.; writing—original draft preparation, J.G.Q.P.; writing—review and editing, V.S., E.M., E.K., and O.L.; visualization, V.S., and E.K.; supervision, V.S., E.M., E.K., and O.L.; project administration, J.G.Q.P.; funding acquisition, E.M. and E.K.,. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported by the Analytical Center for the Government of the Russian Federation (agreement identifier 000000D730324P540002, grant No 70-2023-001317 dated 28.12.2023).

**Data Availability Statement:** The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author/s J.G.Q.P.

**Acknowledgments:** We would like to thank Prof. A. Nikonorov for valuable discussions during the preparation of the manuscript.

**Conflicts of Interest:** The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analysis, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

## Appendix A

*CST Method*

Method proposed by Kulfan in 2006 [53]. The upper and lower curves of the airfoil are represented as the product of the class C function(x) and a shape function S(x) plus a component indicating the thickness of the trailing edge of the profile $\Delta y_{te}$ (in case the trailing edge of the profile has no thickness this term is omitted):

$$y(x) = C(x)S(x) + x\Delta y_{te}. \quad (A1)$$

The class function for defining the airfoil geometries is expressed as:

$$C(x) = \sqrt{x}(1 - x). \quad (A2)$$

To generate an arbitrary aerodynamic shape, the Bernstein polynomial is chosen as the shape function. The definition of the Bernstein polynomial (BPn) of order n is:

$$BPn = K_{r,n}x^r(1 - x)^{n-r}, \quad (A3)$$

$K_{r,n}$ are binomial coefficients defined as:

$$K_{r,n} = \binom{n}{r} = \frac{n!}{r!(n-r)!}. \quad (A4)$$

Then, S(x) is defined as:

$$S(x) = \sum_{r=0}^{n} A_r K_{r,n} x^r (1 - x)^{n-r}. \quad (A5)$$

The extended equations for the upper and lower curves are:

$$\begin{aligned} y_u &= \sqrt{x}(1 - x) \textstyle\sum_{r=0}^{n} \left[A_{u,r} K_{r,n} x^r (1 - x)^{n-r}\right] + x\Delta y_{u,te} \\ y_l &= \sqrt{x}(1 - x) \textstyle\sum_{r=0}^{n} \left[A_{l,r} K_{r,n} x^r (1 - x)^{n-r}\right] + x\Delta y_{l,te} \end{aligned}, \quad (A6)$$

where u subscript indicating the upper curve of the profile; l subscript indicating the lower curve of the profile; $A_{u,r}$ and $A_{l,r}$ are the coefficients of the components of the Bernstein polynomial, and in turn are the design variables for numerical design optimization; x is a succession of values between 0 and 1.

The following equations are used to determine the thickness line and the sag line of the profile:

$$y_t(x) = 2C(x)\sum_{r=0}^{n}\left(\frac{A_{u,r}-A_{l,r}}{2}\right) BPn, \tag{A7}$$

$$y_c(x) = C(x)\sum_{r=0}^{n}\left(\frac{A_{u,r}+A_{l,r}}{2}\right) BPn. \tag{A8}$$

To be able to reconstruct most of the existing profiles it is recommended to use Bernstein polynomials of order 6, which implies having a total of 14 design variables.

## Appendix B

*Bezier-PARSEC Method*

These parameters were created by Derksen and Rogalsky in 2004 [54]. The profiles created with the Bezier-PARSEC parameters in their BP3333 version are represented by four third-degree Bezier curves, two for determining the thickness line (leading edge and trailing edge) and two for determining the middle line (leading edge and trailing edge). The parameters used to create the profiles are the following: $r_{le}$ - radius of the leading edge; $\alpha_{te}$ - angle of the camber line at the trailing edge; $\beta_{te}$ - angle of the thickness line at the trailing edge; $z_{te}$ - vertical displacement of the trailing edge; $\gamma_{le}$ - angle of the camber line at the leading edge; ($x_c$, $y_c$) - position of the maximum value of the camber line; $k_c$ - curvature at camber line; ($x_t$, $y_t$) - position of the maximum value of the thickness line; kt - curvature at the maximum point of the thickness line; $dz_{te}$ – half the thickness of the trailing edge.

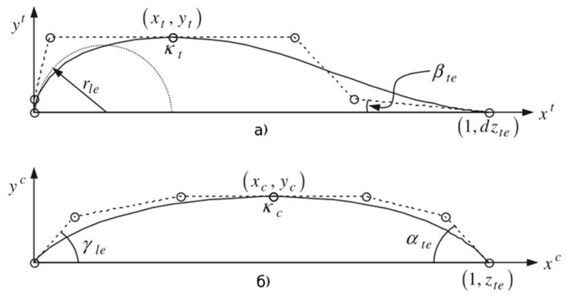


**Figure B1.** BP parameters and Bezier reference points for: (a) thickness line; (b) camber line.

The third-degree Bezier curves are given by the following parametric equations (x(u) and y(u)):

$$x(u) = x_0(1-u)^3 + 3x_1u(1-u)^2 + 3x_2u^2(1-u) + x_3u^3, \tag{B1}$$

$$y(u) = y_0(1-u)^3 + 3y_1u(1-u)^2 + 3y_2u^2(1-u) + y_3u^3, \tag{B2}$$

where u is a parameter that changes from 0 to 1 under its own conditions.

The Bezier control points for the leading edge thickness line are calculated by the following equations:

$$\begin{cases} x_0 = 0 \\ x_1 = 0 \\ x_2 = b_9 \\ x_3 = x_t \end{cases}, \quad \begin{cases} y_0 = 0 \\ y_1 = 3k_t\frac{(x_t-b_9)^2}{2} + y_t \\ y_2 = y_t \\ y_3 = y_t \end{cases}, \tag{B3}$$

and the control points of the trailing edge thickness line are calculated using:

$$\begin{cases} x_0 = x_t \\ x_1 = 2x_t - b_9 \\ x_2 = 1 + \left(dz_{te} - (3k_t\frac{(x_t-b_9)^2}{2} + y_t)\right)\cot(\beta_{te}) \\ x_3 = 1 \end{cases}, \quad \begin{cases} y_0 = y_t \\ y_1 = y_t \\ y_2 = 3k_t\frac{(x_t-b_9)^2}{2} + y_t \\ y_3 = dz_{te} \end{cases}, \tag{B4}$$

the parameter b₉ is the smallest root of:

$$\begin{gathered} 27k_t^2\frac{b_9^4}{4} - 27k_t^2x_tb_9^3 + (9k_ty_t + 81k_t^2x_t^2/2)b_9^2 \\ +(2r_{le} - 18k_tx_ty_t - 27k_t^2x_t^3)b_9 = 0 \end{gathered}, \tag{B5}$$

and fulfill the condition of

$$\max(0, x_t - \sqrt{-2y_t/3k_t}) < b_9 < x_t. \tag{B6}$$

The Bezier control points for the camber line at the leading edge are calculated using the following relations:

$$\begin{cases} x_0 = 0 \\ x_1 = b_1\cot(\gamma_{le}) \\ x_2 = x_c - \sqrt{2(b_1 - y_c)/3k_c} \\ x_3 = x_c \end{cases}, \quad \begin{cases} y_0 = 0 \\ y_1 = b_1 \\ y_2 = y_c \\ y_3 = y_c \end{cases}, \tag{B7}$$

the parameter $b_1$ is calculated from:

$$\begin{gathered} b_1 = (16 + 3k_c(\cot(\gamma_{le}) + \cot(\alpha_{te}))(1 + z_{te}\cot(\alpha_{te})))/(3k_c(\cot(\gamma_{le}) + \cot(\alpha_{te}))) \\ \pm 4\sqrt{16 + 6k_c(\cot(\gamma_{le}) + \cot(\alpha_{te}))(1 - y_c(\cot(\gamma_{le}) + \cot(\alpha_{te})) + z_{te}\cot(\alpha_{te}))}, \end{gathered} \tag{B9}$$

and must comply with $0 < b_1 < y_c$.

And the curvature line control points at the trailing edge are calculated using:

$$\begin{cases} x_0 = x_c \\ x_1 = x_c + \sqrt{2(b_1 - y_c)/3k_c} \\ x_2 = 1 + (z_{te} - b_1)\cot(\alpha_{te}) \\ x_3 = 1 \end{cases}, \quad \begin{cases} y_0 = y_c \\ y_1 = y_c \\ y_2 = b_1 \\ y_3 = z_{te} \end{cases} \tag{B10}$$

## Appendix C

**Algorithm C1.** CAPR-SHADOW + GAN + AZTLI-NN algorithm [55].

**Inputs:** $\Omega$, $c_{y,d}$, $\alpha_{max}$, $y_{t,min}$ M, Re, G, NP, $NP_{min}$, U*, $\gamma$, H, p

**Outputs**: $L(\xi_{opt})$, $\xi_{opt}$

```
1   Download the parameter normalization model;
2   Download AZTLI-NN architecture and Weights;
3   g = 1;
```

```
4   Initialize the Pg aggregate using GAN;
5   Normalize the parameters of the CST vectors of the initial population;;
6   Create images of graphs of aerodynamic coefficients using AZTLI-NN;
7   **for** i = 1 **to** NP **do**
8       Divide the images into three layers;
9       Get $\alpha_i(c_{l,d})$ with the graph $(c_l \text{ vs } \alpha)_i$;
10      Get $\frac{c_d}{c_l^{1.5}}(\alpha_i(c_{l,d}))$ with the graph $(c_l^{1.5}/c_d \text{ vs } \alpha)_i$;
11      Apply (A7), to obtain $\max(y_t(x))_i$;
12  Get $\psi(\xi_g)$ with (32);
13  Get $L(\xi_g)$ with (31);
14  Update U*;
15  mean$(L(\xi_g)) \rightarrow L_{avg}$;
16  Assign values of 0,5 in memories $M_{CR}$ and $M_F$;
17  Create $A = \emptyset$, $|A| = \text{round}(2{,}6NP)$;
18  k = 1;
19  **for** g = 2 **to** G **do**
20      $S_{CR} = \emptyset$, $S_F = \emptyset$, $\Delta f = \emptyset$;
21      **for** i = 1 **to** NP **do**
22          ri = choose randomly from [1, H];
23          Get $CR_{i,g}$ with (20);
24          Get $F_{i,g}$ with (21);
25          Get the mutated vector $\mathbf{v}_{i,g}$ with(22);
26          **for** j = 1 **to** $|\xi|$ **do**
27              Get the component of the trial vector $u_{j,i,g}$ with (23);
28      Create images of graphs of aerodynamic coefficients of $P(\mathbf{u}_g)$ using AZTLI-NN;
29      **for** i = 1 **to** NP **do**
30          Divide the images into three layers (from $\mathbf{u}_g$);
31          Get $\alpha_i(c_{l,d})$ with the graph $(c_l \text{ vs } \alpha)_i$;
32          Get $\frac{c_d}{c_l^{1.5}}(\alpha_i(c_{l,d}))$ with the graph $(c_l^{1.5}/c_d \text{ vs } \alpha)_i$;
33          Apply (A7), to obtain $\max(y_t(x))_i$;
34      Get $\psi(\mathbf{u}_g)$ with (32);
35      Get $L(\mathbf{u}_g)$ with (32);
36      Update U*;
37      **for** i = 1 **to** NP **do**
38          **if** $L(\mathbf{u}_{i,g}) \leq L(\xi_{i,g})$ **then**
```

```
39                $\xi_{i,g+1}$ = $\mathbf{u}_{i,g}$;
40                $\xi_{i,g} \rightarrow$ A;
41                $CR_{i,g} \rightarrow S_{CR}$, $F_{i,g} \rightarrow S_F$;
42                $|L(\mathbf{u}_{i,g}) - L(\xi_{i,g})| \rightarrow \Delta f$;
43            **else**
44                $\xi_{i,g+1} = \xi_{i,g}$;
45        Update the memories $M_{CR}$ and $M_F$ with Algorithm 1;
46        mean($L(\xi_{g+1})$)$\rightarrow L_{avg}$;
47        **if** $g \geq 3$ **then**
48            Get $\Delta_g$ and $\Delta_{g-1}$ with (30);
49            Get $NP_{g+1}$ with (28);
50            **if** $NP_{g+1} < NP_{min}$ **then**
51                Apply (29);
52            $(NP_g - NP_{g+1})$ worst elements $\rightarrow$ A;
53        **if** $|A| >$ round(2,6NP) **then**
54            Delete ($|A|$ - round(2,6NP)) elements randomly;
55        k++;
56    **return Output**
```

## Appendix D

**Algorithm D1.** NSGA-II + GAN + AZTLI-NN algorithm [55].

**Inputs:** $\Omega$, $c_{y,d}$, M, Re, T, NP

**Outputs**: $F_{Pareto}(\xi)$, $\xi_{Pareto}$

```
1   Download the parameter normalization model;
2   Download AZTLI-NN architecture and weights;
3   t = 1;
4   Initialize the $P_t$ aggregate using GAN;
5   Normalize the parameters of the CST vectors of the initial population;;
6   Create images of graphs of aerodynamic coefficients using AZTLI-NN;
7   **for** i = 1 **to** NP **do**
8       Divide the images into three layers;
9       Get $\alpha_i(c_{l,d})$ with the graph $(c_l$ vs $\alpha)_i$;
10      Get $\frac{c_d}{c_l^{1.5}}(\alpha_i(c_{y,d}))$ with the graph $(c_l^{1.5}/c_d$ vs $\alpha)_i$;
11      Get $|c_m|(\alpha_i(c_{y,d}))$ with $(c_m$ vs $\alpha)_P$;
12  Get ranks $F_i$ from $P_t$ with Algorithm 2;
```

```
13 for $i = 1$ to $max_{rank}$ do
14     Get $CD_i$ with Algorithm 3;
15 Sort $P_t$ based on $F_i$ and $CD_i$;
16 while $|Qt| < NP$ do
17     Get $\mathbf{p}_1$ with Algorithm 4;
18     Get $\mathbf{p}_2$ with Algorithm 4;
19     $\mathbf{v}_1 = \mathbf{v}_2 = \emptyset$;
20     for $j = 1$ to $|\mathbf{p}_1|$ do
21         Get $c_{1,j}$ and $c_{2,j}$ with (35) and (36);
22         Get $v_{1,j}$ and $v_{2,j}$ with (37) and (38)
23         $v_{1,j} \rightarrow \mathbf{v}_1$;
24         $v_{2,j} \rightarrow \mathbf{v}_2$;
25     $\mathbf{v}_1 \rightarrow Q_t$;
26     $\mathbf{v}_2 \rightarrow Q_t$;
27 $R_t = P_t \cup Q_t$;
28 Get ranks $F_i$ from $R_t$ with Algorithm 2;
29 for $i = 1$ to $max_{rank}$ do
30     Get $CD_i$ with Algorithm 3;
31 Sort $R_t$ based on $F_i$ and $CD_i$;
32 Best NP vectors pass $R_t \rightarrow P_{t+1}$;
33 $P_t = P_{t+1}$;
34 for $t = 2$ to $T$ do
35     while $|Qt| < NP$ do
36         Get $\mathbf{p}_1$ with Algorithm 6;
37         Get $\mathbf{p}_2$ with Algorithm 6;
38         $\mathbf{v}_1 = \mathbf{v}_2 = \emptyset$;
39         for $j = 1$ to $|\mathbf{p}_1|$ do
40             Get $c_{1,j}$ and $c_{2,j}$ with (35) and (36);
41             Get $v_{1,j}$ and $v_{2,j}$ with (37) and (38);
42             $v_{1,j} \rightarrow \mathbf{v}_1$;
43             $v_{2,j} \rightarrow \mathbf{v}_2$;
44         $\mathbf{v}_1 \rightarrow Q_t$;
45         $\mathbf{v}_2 \rightarrow Q_t$;
46     $R_t = P_t \cup Q_t$;
47     Get ranks $F_i$ from $R_t$ with Algorithm 2;
48     for $i = 1$ to $max_{rank}$ do
```

```
49            Get $CD_i$ with Algorithm 3;
50        Sort $R_t$ based on $F_i$ and $CD_i$;
51        Best NP vectors pass $R_t \rightarrow P_{t+1}$;
52        $P_t = P_{t+1}$;
53    return Output
```

## References

1. Ma, Y.; Elham, A. Designing high aspect ratio wings: a review of concepts and approaches. *Progress in Aerospace*. **2024**. *145*. pp. 100983.
2. Martins, J.R.; Kennedy, G.; Kenway, G.K. High aspect ratio wing design: optimal aero structural trade offs for the next generation of materials. In 52nd Aerospace Sciences Meeting. USA. 13-17 January 2014.
3. Vassberg, J.C.; Jameson, A. Industrial applications of aerodynamic shape optimization. In VKI Lecture-II. Belgium. 11 September 2018.
4. Nikolaev, N.V. Optimization of airfoils along high-aspect-ratio wing of long-endurance aircraft in trimmed flight. *Journal of Aerospace Engineering*. **2019.** *32(6)*. pp. 04019090.
5. Anderson, J.D.; Bowden, M.L. *Introduction to flight*, 9th ed.; McGraw-Hill Higher Education: USA, 2021; pp. 508-513.
6. Steinbuch, M.; Marcus, B.; Shepshelovich, M. Development of UAV wings-subsonic designs. In 41st Aerospace Sciences Meeting and Exhibit. USA. 6-9 January 2003.
7. Park, K.; Kim, B.S. Optimal design of an airfoil plataform shapes with high aspect ratio using genetic algorithms. *International Journal of Aerospace and Mechanical Engineering*. **2013**. *7(4)*. pp. 584-590.
8. Wang, L.; Zhang, H.; Wang, C.; Tao, J.; Lan, X.; Sun, G.; Feng, J. A review of intelligent airfoil aerodynamic optimization methods based on data-driven advanced models. *Mathematics*. **2024**. *12*. pp 1417.
9. Dussage, T.P.; Sung, W.J.; Pinon Fischer, O.J.; Mavris, D.N. A reinforcement learning approach to airfoil shape optimization. *Scientific Reports*. **2023**. *13(1)*. pp 9753.
10. Yan, X.; Zhu, J.; Kuang, M.; Wang, X.; Aerodynamic shape optimization using a novel optimizer based on machine learning techniques. *Aerospace Science and Technology*. **2019**. *86*. pp 826-835.
11. Skinner, S.N.; Zare-Behtash, H. State-of-the-art in aerodynamic shape optimization methods. *Applied Soft Computing*. **2018**. *62*. pp 933-962.
12. Li, J.; Du, X.; Martins, J.R. Machine learning in aerodynamic shape optimization. *Progress in Aerospace Sciences*. **2022**. *134*. pp 100849.
13. Hu, L.; Zhang, J.; Xiang, Y.; Wnag, W. Neural networks-based aerodynamic data modeling: a comprehensive review. *IEEE Access*. **2020**. *8*. pp. 90805-90823.
14. UIUC Airfoil coordinates database. Available online: https://m-selig.ae.illinois.edu/ads/coord_database.html (20 February 2024).
15. The incomplete guide to airfoil usage. Available online: https://m-selig.ae.illinois.edu/ads/aircraft.html#conventional (19 February 2024).
16. Derksen, R.W.; Rogalsky, T. Bezier-PARSEC: an optimized aerofoil parameterization for design. *Advances in Engineering Software*. **2010**. *41*. pp. 923-930.
17. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Bengio, Y. Generative adversarial networks. *Communications of the ACM*. **2020**. *63(11)*. pp. 139-144.
18. Creswell, A.; White, T.; Dumoulin, V.; Arulkumuran, K.; Sengupta, B.; Bharath, A.A. Generative adversarial networks: an overview. *IEEE Signal Processing*. **2018**. *35(1)*. pp. 53-68.
19. OpenFOAM v11 User Guide. Available online: https://doc.cfd.direct/openfoam/user-guide-v11/index (20 March 2024).
20. Welcome to pygmsh' documentation!. Available online: https://pygmsh.readthedocs.io/en/latest (10 March 2024).
21. Eleni, D.C.; Athanasios, T.I.; Dionissios, M.P. Evaluation of the turbulence models for the simulation of the flow over a National Advisory Committe for Aeronautics (NACA) 0012 airfoil. *Journal of Mechanical Engineering Research*. **2012**. *4(3)*. pp. 100-111.
22. Suvanjumrat, C. Comparison of turbulence models for flow past NACA0015 airfoil using OpenFOAM. *Engineering Journal*. **2017.** *21(3)*. pp. 207-221.
23. Khan, S.A.; Bashir, M.; Baig, M.A.A.; Ali, F.A.G.M. Comparing the effect of different turbulence models on the CFD predictions of NACA0018 airfoil aerodynamics. *CFD Letters*. **2020.** *3*. pp. 1-10.
24. Menter, F.R.; Kuntz, M.; Lagtry, R.; Ten years of industrial experience with the SST turbulence model. *Turbulence, heat and mass transfer*. **2003**. *4(1)*. pp. 625-632.

25. Lu, S.; Liu, J.; Hekkenberg, R. Mesh properties for RANS simulations of airfoil-shaped profiles: a case study of ruder hydrodynamics. *Journal Marine Science and Engineering*. **2021**. *9*. pp. 1062.
26. Liu, F. A thorough description of how wall functions are implemented in OpenFOAM. In CFD with OpenSource Sotfware. 22 January 2017.
27. Sola, J.; Sevilla, J. Importance of input data normalization for the application of neural networks to complex industrial problems. *IEEE Transactions on Nuclear Science*. **1997**. *44(3)*. pp. 1464-1468.
28. Gokhan, A.K.S.U.; Guzeller, C.O.; Eser, M.T. The effect of the normalization method used in different sample sizes on the success of artificial neural network model. *International Journal of Assessment Tools in Education*. **2019**. *6(2)*. pp. 170-192.
29. Pedregosa, F.; Varoquax, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; Duchesnay, E. Scikit-learn: machine learning in Python. *Journal of Machine Learning Research*. **2011**. *12*. pp. 2825-2830.
30. Keras. Available online: https://keras.io (15 April 2024).
31. Wang, J.; He, C.; Li, R.; Chen, H.; Zhai, C.; Zhang, M. Flow field predictions of supercritical airfoils via variational autoencoder based deep learning framework. *Physics Fluids*. **2021**. *33*. pp. 086108.
32. Tharwat, A. Principal component analysis – a tutorial. *International Journal of Applied Pattern Recognition*. **2016**. *3(3)*. pp. 197-240.
33. Kingma, D.P.; Welling, M. An introduction to variational autoencoders. *Foundations and Trends in Machine Learning*. **2019**. *12(4)*. pp. 307-392.
34. Espinosa Barcenas, O.U.; Quijada Pioquinto, J.G.; Kurkina, E.; Lukyanov, O. Surrogate aerodynamic wing modeling based on a multilayer perceptron. *Aerospace*. **2023**. *10*. pp. 149.
35. Sun, G.; Sun, Y.; Wang, S. Artificial neural network based inverse design: airfoils and wings. *Aerospace Science and Technology*. **2015**. *42*. pp. 415-428.
36. Moin, H.; Khan, H.Z.I.; Mobeen, S.; Riaz, J. Airfoil's aerodynamic coefficients prediction using artificial neural network. In 2022 19th International Bhurban Conference on Applied Sciences and Technology (IBCAST). Pakistan. 16 - 20 August 2022.
37. Deng, C.; Zhao, B.; Yang, Y.; Deng, A. Integer encoding differential evolution algorithm for integer programming. In 2010 2nd International Conference on Information Engineering and Computer Science. China. 30 December 2010.
38. Tanabe, R.; Fukunuga, A. Success-history based parameter adaptation for differential evolution. In IEEE Congress on Evolution Computation (CEC). Mexico. 15 July 2013.
39. Tanabe, R.; Fukunuga, A.S. Improving the search performance of SHADE using linear population size reduction. In 2014 IEEE Congress on Evolution Computation (CEC). China. 22 September 2014.
40. Renkavieski, C.; Parpinelli, R.S. L-SHADE with alternative population size reduction for unconstrained continuous optimization. *Anais do Computer on the Beach*. **2020**. *11*. pp. 351-358.
41. Wong, I.; Liu, W.; Ho, C.M.; Ding, X. Continuous adaptive population reduction (CAPR) for differential evolution optimization. *SLAS Technology*. **2017**. *22(3)*. pp. 289-305.
42. Pioquinto, J.G.Q.; Moreno, R.A.F. Methods for increasing the efficiency of the differential evolution algorithm for aerodynamic shape optimization applications. In the XXVI All-Russian Seminar on Motion Control and Navigation of Aircraft. Russia. 14-16 June 2023.
43. Sedelnikov, A.; Kurkin, E.; Quijada Pioquinto, J.G.; Lukyanov, O.; Nazarov, D.; Chertykovtseva, V.; Kurkina, E.; Hoang, V.H. Algorithm for propeller optimization based on differential evolution. *Computation*. **2024**. *12*. pp. 52.
44. Ali, M.M.; Zhu, W.X. A penalty function-based differential evolution algorithm for constrained global optimization. *Comput. Optim. Appl*. **2013**. *54*. pp. 707-739.
45. Deb, K.; Pratab, A.; Agarwal, S.; Meyarivan, T.A.M.T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*. **2002**. *6(2)*. pp. 182-197.
46. Verma, S.; Pant, M.; Snasel, V. A comprehensive review on NSGA-II for multi-objective combinatorial optimization problems. *IEEE Access*. **2021**. *9*. pp. 57757-57791.
47. Deb, K.; Sindhya, K.; Okabe, T. Self-adaptive simulated binary crossover for real-parameter optimization. In the 9th Annual Conference on Genetic and Evolutionary Computation. 7 July 2007.
48. Kakde, M.R.O. Survey on multiobjective evolutionary and real coded genetic algorithms. In the 8th Asia Pacific Symposium on Intelligent and Evolutionary Systems. January 2004.
49. Das, S.; Mullick, S.S.; Suganthan, P.N. Recent advances in differential evolution – an update survey. *Swarm and Evolutionary Computation*. **2016**. *27*. pp. 1-30.
50. Ladson, C.L. Effects of independent variation of Mach and Reynolds numbers on the low-speed aerodynamic characteristics of the NACA 0012 airfoil section. National Aeronautics and Space Administration, Scientific and Technical Information Division. 1988. № 4074.
51. Turbulent flow over NACA0012 airfoil (2D). Available online: https://www.openfoam.com/documentation/guides/latest/doc/verification-validation-naca0012-airfoil-2d.html (20.03.2024).

52. Althaus, D.; Wortmann, F.X. Experimental results from laminar wind tunnel of the Institut fur Aero- und Gasdynamic der Universitat Stuttgart. Fried. Vieweg & Sohn: Germany, 1981; pp. 1-20.
53. Kulfan, B.M.; Bussoletti, J.E. "Fundamental" parametric geometry representations for aircraft component shapes. In 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference. USA. 6-8 September 2006.
54. Rogalsky, T. Acceleration of differential evolution for aerodynamic design. PhD Thesis. Winnipeg, Canada, March 2004.
55. OpenVozduj/AZTLI-NN. Available online: https://github.com/OpenVozduj/AZTLI-NN (22 July 2024)