26th International Congress of Mechanical Engineering
November 22-26, 2021, Virtual Congress, Brazil

# COB-2021-0586
# A STUDY ON AIRFOIL OPTIMIZATION WITH GENETIC ALGORITHMS

**Giovana Weffort Fernandes**
**Manuel Nascimento Dias Barcelos Júnior**
Universidade de Brasília - Faculdade UnB Gama - St. Leste Projeção A - Gama Leste - Brasília - Brasil
giwf015@gmail.com, manuelbarcelos@aerospace.unb.br

***Abstract.*** *The work in hand deals with the optimization of aeronautical use airfoils through genetic algorithms. Optimization techniques are essential in a wide range of applications. In the field of aeronautics, specifically, an aircraft's performance may improve by optimizing various aspects, including the design of the airfoils. The objective is to find the most efficient airfoils possible for specific flight conditions defined in terms of Reynolds numbers and angles of attack. These efficiencies are measured with the lift to drag ratio, or $^{L}/_{D}$. Two types of geometric parametrization were used: the NACA 4-digit configuration and the CST (Class and Shape function Transformation) method, which allows for more design freedom. The algorithm runs on MATLAB in conjunction to XFOIL, which is used to provide the aerodynamic data for the airfoils. After performing tests regarding the reliability and functionality of the algorithm, we ran it under a set of case studies using both geometric parametrizations. The CST parametrization proved to provide better solutions than the NACA airfoils due to its more extensive design capabilities. The results demonstrate the promising capabilities of a genetic algorithm with CST implemented, allowing for further improvements of the optimization process according to the design necessity.*

***Keywords:*** *Airfoils, Genetic algorithm, CST, NACA, XFOIL*

## 1. INTRODUCTION

In any project in the field of aeronautics, the design and/or choice of the airfoils used in wings, stabilizers or any other kind of aerodynamic surface is a critical step in the complete aerodynamic design of the craft. Airfoils affect lift, drag and pitching moment characteristics, altering the performance of the craft, with said characteristics being directly influenced by the geometry of the airfoil. Therefore, since the objective is to always employ the best solutions to the project (limited by the referent trade-offs, of course), this is clearly a situation that would be greatly facilitated through the use of optimization algorithms. That is the objective of this paper - the development of a genetic algorithm airfoil optimization code integrated with XFOIL.

Genetic algorithms are widely used to solve optimization problems due to their simple structure, versatility and reliability. XFOIL is a high order panel method software used to simulate flows around airfoils. Due to its simplicity, fast calculations, and sufficiently accurate results for preliminary design stages, XFOIL is an excellent tool for airfoil optimization. In fact, similar methodologies to the one adopted in this work have already been put to test in previous papers, such as those by Ram *et al.* (2014) and Dina *et al.* (2019), both of which use genetic algorithms to optimize airfoils with the aid of XFOIL. The distinction between such papers and the present work are mostly the geometry parametrization methods and the objective functions. For example, Ram *et al.* (2014) optimized a NACA 0012 airfoil based upon the lift, drag and pressure coefficients, while the present work searches for airfoils with the highest lift-to-drag ratios possible, not starting from any specific airfoils.

This work was conducted mainly with aircraft design in mind. However, the methodology covered in this paper may also be applied to airfoils used in other contexts, such as wind turbines, racing vehicles, or watercraft (in which case, the object of study is named hydrofoil) as long as the flows are subsonic.

## 2. GENETIC ALGORITHMS

One of the main objectives of computer science was, since the onset of its existence, the solution of problems. A type of particular recurring problem in engineering is the problem of optimization, for which numerous techniques were developed in the past. One such technique, the genetic algorithm, a branch of evolutionary computing, is employed in this study.

The platform used to run these algorithms was MATLAB. Additionally, the optimization of airfoils requires that they are somehow evaluated in regards to the optimization problem in hands. These evaluations are performed by means of numerical aerodynamic simulations. The tool chosen for the task is XFOIL, a numerical aerodynamic simulation code well established in the field of aeronautics. The MATLAB-XFOIL integration is simple due to XFOIL being a command prompt based software, which MATLAB may easily summon and execute. Therefore, when it is desired to evaluate an airfoil, the algorithm executes XFOIL, inputting all required information - geometry, angle of attack, Reynolds number, and other parameters. When the simulation ends, XFOIL is closed and all results are returned to the algorithm.

As such, the tasks of the two software employed in this work may be categorized as follows: MATLAB runs the complete process of the genetic algorithm, including the generation of the initial population, the crossovers, mutations, among other steps, and XFOIL provides the necessary data with which the airfoils are evaluated. No specific MATLAB modules intended for optimization were used - all genetic algorithms used for this paper were written from scratch in standard architecture, that is, with no mechanisms of elitism.

## 2.1 Tests and validations

Brand new optimization algorithms must be validated in order to prove their performance. This is done with the use of test functions which allows for the evaluation of the algorithm's capability and efficiency to solve optimization problems.

Below are presented the test functions (Yang, 2010) used to validate the algorithm.

### 2.1.1 De Jong's function

The first of the De Jong's functions is Eq. 1.

$$f(x) = \sum_{i=1}^{n} x_i^n \tag{1}$$

Where $i = 1, 2, ..., n$ and the terms $x_i$ are defined in the domain $-5.12 \leq x_i \leq 5.12$. The global minimum is $f_* = 0$ at $\mathbf{x}_* = (0, 0, ..., 0)$.

Among the validation equations used in this work, this is the simplest, in which the algorithm always reaches or approaches the global minimum as long as the population size increases as $n$ increases.

## 2.2 Ackley's function

Ackley's function is Eq. 2.

$$f(x) = -20 \exp\left[-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right] - \exp\left[\frac{1}{n}\sum_{i=1}^{n}\cos\left(2\pi x_i\right)\right] + 20 + \exp\left[1\right] \tag{2}$$

Where $i = 1, 2, ..., n$ and the terms $x_i$ are defined in the domain $-32.768 \leq x_i \leq 32.768$. The global minimum is $f_* = 0$ at $\mathbf{x}_* = (0, 0, ..., 0)$.

Although more complex, the algorithm behaved similarly to the case of De Jong's function.

### 2.2.1 Easom's function

Easom's function is Eq. 3.

$$f(x) = -\cos(x)\cos(y)\exp\left[-(x-\pi)^2 + (y-\pi)^2\right] \tag{3}$$

Where $-100 \leq x, y \leq 100$. The global minimum is $f_* = -1$ at $x = \pi$ and $y = \pi$.

The algorithm consistently reaches the global minimum starting from a population size of about 300 individuals.

### 2.2.2 Griewank's function

Griewank's function is Eq. 4.

$$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \tag{4}$$

Where $i = 1, 2, ..., n$ and the terms $x_i$ are defined in the domain $-600 \leq x_i \leq 600$. The global minimum is $f_* = 0$ at $\mathbf{x}_* = (0, 0, ..., 0)$.

The algorithm also reaches this function's global minimum with no trouble given a large enough population size.

### 2.2.3 Summary of the validations

Table 1 presents one example for each function as a mean to better illustrate the behaviour of the algorithm. The data in the Iterations column refers to the number of iterations the algorithm took until the best solution found stabilized.

Table 1. Validation run examples.

| Function | Length of **x** | Population size | Mutation chance | Iterations | Result | Analytic solution |
|---|---|---|---|---|---|---|
| De Jong | 20 | 100 | 2% | 15 | 0.0032 | 0 |
| Ackley | 20 | 300 | 2% | 80 | 0 | 0 |
| Easom | - | 300 | 2% | 3 | -1.0429 | -1 |
| Griewank | 20 | 300 | 2% | 11 | 0.0159 | 0 |

## 3. AIRFOILS

Airfoil is the name given to the cross section of a wing (or another aerodynamic surface), possessing a geometry conceived to generate lift in efficient fashion - that is, with as little drag as possible (Gudmundsson, 2014). There are other important aspects in regards to airfoil design, such as pitching moment and internal space. However, for this work, only lift and drag were used to qualify the airfoils.

The method to assess the airfoil's performance characteristics in regards to lift and drag is to analyze their respective dimensionless coefficients $C_L$ and $C_D$, calculated here using XFOIL. XFOIL uses a high-order panel method with Karman-Tsien compressibility corrections and a two-equation system to represent viscous layers (Drela, 1989) (Gudmundsson, 2014). The robustness of XFOIL's code and its near instant results make it an invaluable asset in optimization.

A common practise is to measure the aforementioned aerodynamic efficiency by means of the lift-to-drag ratio, defined as ${}^{C_L}/_{C_D}$ or as ${}^{L}/_{D}$. In either case, the higher the value, the higher the efficiency. All optimization studies in this work were conducted using the lift-to-drag ratio to evaluate the airfoils the same way - the higher the ${}^{L}/_{D}$, closer to the optimum is the solution. In other words, given a specific flight condition defined by a Reynolds number and an angle of attack, the algorithm searches for the airfoil with the highest ${}^{L}/_{D}$ possible.

In order to evaluate the airfoils, it is necessary to generate them in the first place using some sort of geometric parametrization. Two types of parametrization were used: the NACA 4-digit method and the CST formulations, which are further explained in the following subsections.

### 3.1 NACA 4-digit

One of the most widely used types of airfoils are the NACA 4-digit series, which are parametrized through the use of functions that provide the thickness distribution and the mean camber line of the airfoil. All of these characteristics are inferred from the 4 digits that compose the airfoil's name - the first $(m)$ representing the maximum camber as a percentage of the chord length, the second $(p)$ representing the position of the maximum camber from the leading edge in tenths of the chord, and the last two digits $(t)$ representing the maximum thickness as a percentage of the chord (Abbott and Doenhoff, 1959). All three numbers are integers. When using the NACA 4-digit airfoils, therefore, the algorithm has to handle three design variables - the numbers $m$, $p$ and $t$ for each airfoil.

Because the algorithm evaluates the airfoils in regards to the lift-to-drag ratios, cambered airfoils are always favored in the results, because higher camber provides more lift. Therefore, symmetric airfoils were ignored, so $m$ is never equal to zero, and neither is $p$. Symmetric airfoils would be more relevant when studying negative angles of attack, which is not the case in this paper, or if the objective function was different than the one chosen.

The implementation of the NACA 4-digit parametrization in the genetic algorithm is straightforward, because XFOIL already has a built-in function to generate these airfoils. Therefore, it is not necessary to program a way to generate the airfoils in the algorithm itself and generate coordinate files, as will be described in the following subsection.

As mentioned in Section 2, the first stage of a genetic algorithm is the generation of the initial population. In the case of the NACA 4-digit algorithm, all individuals are generated randomly, with $m$ and $p$ both ranging from 1 to 9 and $t$ ranging from 5 to 50.

In the crossover stage, for every two individuals from the current generation, two "offsprings" are created. In the NACA 4-digit algorithm, crossover is done by exchanging between them one of the three variables. In the mutation stage, the individual has one of its variables randomly chosen for mutation, and this variable is reassigned a new value from its respective range.

## 3.2 CST - Class and Shape Transformation

We searched for an additional method of parametrization other than the NACA 4-digit foils in order to provide more freedom when configuring the shape of the airfoils. The method chosen was the CST (Class and Shape function Transformation). The CST, as demonstrated by Kulfan and Bussoletti (2006) and Kulfan (2008), consists of the following equation:

$$\frac{z}{c}\left({}^{x}\!/_{c}\right) = C\left({}^{x}\!/_{c}\right) \cdot S\left({}^{x}\!/_{c}\right) + \frac{x}{c} \cdot \frac{\Delta Z_{TE}}{c} \tag{5}$$

Where $z$ is the vertical distance from the chord line, $x$ is the horizontal distance from the leading edge ($x = 0$), $c$ is the chord length and $\Delta Z_{TE}$ is the vertical distance between the trailing edge and the horizontal axis. Additionally, C$({}^{x}\!/_{c})$ is the class function and S$({}^{x}\!/_{c})$ is the shape function. They are presented below.

$$C\left({}^{x}\!/_{c}\right) = \left(\frac{x}{c}\right)^{N1}\left[1 - \frac{x}{c}\right]^{N2} \tag{6}$$

Equation (6) is the class function, which defines the general shape of the geometry. By modifying the values of the exponents $N1$ and $N2$ the type of the geometry is changed. For round leading edge airfoils, $N1 = 0.5$ and $N2 = 1.0$.

The shape function uses a mathematical approximation to sculpt the details of the geometry. In this work, the Bernstein polynomial was chosen for the task. The Bernstein polynomial of order $n$ is defined as shown in Eq. (7).

$$BPn = \sum_{r=0}^{n} K_{r,n} x^{r}\left(1 - x\right)^{n-r} \tag{7}$$

Where $K_{r,n}$ are binomial coefficients, defined as shown in Eq. (8).

$$K_{r,n} = \binom{n}{r} = \frac{n!}{r!(n-r)!} \tag{8}$$

Therefore, the shape function in Eq. (5) is simply the Bernstein polynomial from Eq. (7) multiplied by $A_i$, which are weights used as design variables. The number of design variables (not counting $\Delta Z_{TE}$) is equal to the order of the Bernstein polynomial plus 1.

$$S\left({}^{x}\!/_{c}\right) = \sum_{i=0}^{n}\left[A_i \cdot \left[K_{r,n}\left(\frac{x}{c}\right)^{r}\left(1 - \frac{x}{c}\right)^{n-r}\right]\right] \tag{9}$$

The first and last weights are defined, respectively, by Eq. (10) and Eq. (11).

$$A_0 = \sqrt{\frac{2R_{Le}}{c}} \tag{10}$$

$$A_n = \tan\beta + \frac{\Delta Z_{TE}}{c} \tag{11}$$

Where $R_{Le}$ is the radius of the leading edge and $\beta$ is the trailing edge boattail angle. All other weights, the intermediary weights, are values chosen by the user, being valid as long as $A_i \in \mathbb{R}$.

Adopting $\psi = {}^{x}\!/_{c}$, $\zeta = {}^{z}\!/_{c}$, and disregarding the last term of Eq. (5) (because $\Delta Z_{TE}$ was chosen to always be equal to zero in this work), it becomes

$$\zeta\left(\psi\right) = \sqrt{\psi}\left(1 - \psi\right)\sum_{i=0}^{n}\left[A_i \cdot \left[K_{r,n}\psi^{r}\left(1 - \psi\right)^{n-r}\right]\right] \tag{12}$$

Equation (12) gives the upper contour of the airfoil. To obtain the lower contour, it is necessary only to use Eq. (12) again and multiply the resulting $\zeta\left(\psi\right)$ ordinates by -1. It is also necessary to multiply these results by a scaling factor (the scaling factor may also be used as a design variable, but for this work we decided to keep its value fixed at $0.1$ for all airfoils generated by the CST method). The Bernstein polynomial used is of fourth order.

Because the CST airfoils are not generated in XFOIL directly, it is necessary to import them through the use of coordinate files. As such, the algorithm in MATLAB generates the airfoils through the equations above, then generates their respective coordinate files, and finally starts XFOIL in a manner similar to the NACA 4-digit case, but using a command to import the airfoil through the coordinate file.

With the CST parametrization, in the configuration used in this paper, there are 9 design variables - a leading edge radius equal for both upper and lower surfaces, 3 intermediary weights for each surface and a trailing edge boattail angle for each surface.

When in the initial stage of randomly generating individuals, the NACA 4-digit algorithm is simple because virtually any combination of design variables $m$, $p$ and $t$ generates an airfoil with no geometric anomalies. That is not the case for the CST airfoils, because certain combinations of design variables may produce airfoils with one surface intersecting the other, airfoils with hourglass shape, or other less drastic, but still undesirable features. Therefore, the CST algorithm has geometry quality checks to ensure that all airfoils have correct geometries. These quality checks act in the stages of initial population generation, crossover and mutation, and they enforce the following rules: upper and lower surfaces must not intersect, the upper surface must have only one point of slope shift (from positive to negative or vice-versa), the lower surface must have at most three points of slope shift (from positive to negative or vice-versa), and the mean thickness must be larger than ${}^{0.04}/_{c}$.

Similarly to the NACA 4-digit algorithm, the CST algorithm also generates the initial population in a random manner, although with additional settings in order to aid the algorithm in dealing with the increased search space caused by the higher complexity of the CST parametrization. The leading edge radius ranges from 0.5 to 5, the three intermediary weights range from 0 to 6 for the upper surface and from -5 to 5 for the lower surface, and the trailing edge boattail angle ranges from 20 to 60 for the upper surface and from -50 to 20 for the lower surface. The algorithm may also force the generation of airfoils with more camber (in which case the three intermediary weights range from 3 to 6 for the upper surface and from -5 to 1 for the lower surface) or the generation of airfoils with less camber (in which case the three intermediary weights range from 0 to 3 for the upper surface and from -1 to 3 for the lower surface). Whether a given airfoil will be generated freely, with more camber or with less camber is also random. These variables are organized in the form of two vectors, one containing the information referent to the upper surface, and the other, the lower surface. Both contain the leading edge radius, because it is equal for both sides.

Crossover, like in the NACA 4-digit algorithm, also generates two new individuals for every two parents. There are two types of crossover in the CST algorithm. The first consists of exchanging parts of both vectors between both individuals, with the length of these parts determined randomly. The second type switches the entire vectors, which effectively generates individuals with the upper surface of one parent and the lower surface of the other. The selection of the type of crossover is done randomly. For both cases, all offsprings need to have the leading edge radius of the upper surface substituted by the leading edge radius of the lower surface or vice-versa, because, again, this value was established to be equal for both surfaces in this work. Which value is substituted is also determined randomly.

Mutation in the CST algorithm is not performed in the same unrestricted way as the NACA 4-digit algorithm, because, when dealing with parametrization such as the CST, such behaviour has an enormous potential in worsening good individuals - the CST, in comparison to the NACA 4-digit foils, has a much smaller number of combinations of design variable values that produce good airfoils. Considering this, the mutation per say is done as follows. There are four types of mutation in the CST algorithm. In the first type, the airfoil's camber is increased by summing a random number, ranging from zero to one, to the intermediary weights of the upper surface, and subtracting the same number from the intermediary weights of the lower surface. The second type of mutation consists of altering all values of both of the airfoil's vectors. All variables are changed to a value within a range close to its original value. For the leading edge radius and the intermediary weights, this range starts from the original value minus one up to the original value plus one, and the range for the boattail angle starts from the original value minus ten up to the original value plus ten. The third and fourth types of mutation work in the same way as the second type, but altering only the upper surface and the lower surface respectively. The type of mutation is determined in random fashion. After the mutation, it is necessary to substitute the leading edge radius in one of the vectors, in the same way as done in the crossover stage.

## 4. DISCUSSIONS AND RESULTS

### 4.1 Algorithm configuration parameters

We tested the algorithm using four different case studies, varying the Reynolds number and the angle of attack. They are shown in Tab. 2. In regards to the hardware, the computer used had a third generation Intel I7 processor and 8 Gb of RAM.

Table 2. Algorithm case studies.

| **Case study** | **Reynolds number** | **Angle of attack** |
|---|---|---|
| 1 | 1e5 | 0° |
| 2 | 1e5 | 5° |
| 3 | 1e6 | 0° |
| 4 | 1e6 | 5° |

The airfoil optimization algorithm has parameters which may change the results depending on how they are configured. It is important, therefore, to understand this relation. Along the development of this work, different configurations were tested until a good compromise between results and performance was found. The standard configurations chosen for the algorithm are shown in Tab. 3.

Table 3. Standard algorithm configuration.

| Geometric parametrization method | Population size | Mutation chance | XFOIL iterations |
|---|---|---|---|
| NACA 4-digit | 300 | 10% | 50 |
| CST | 300 | 20% | 10 |

To document the effects of these configurations, we ran the algorithm multiple times in the first case study keeping the standard configurations shown in Tab. 3, changing only the parameter whose effect is under analysis.

Tables 4 to 6 present the results of these analyses in regards to the NACA 4-digit algorithm.

Table 4. NACA 4-digit tests: population size.

| Population | NACA airfoil | L/D | Algorithm iterations | Time |
|---|---|---|---|---|
| 50 | 4317 | 29.0190 | 10 | 4 min and 36.05 s |
| 150 | 5311 | 36.4782 | 10 | 13 min and 56.03 s |
| 300 | 5311 | 36.4782 | 10 | 28 min and 37.56 s |

In a genetic algorithm, a larger population size provides a larger range of initial variation to be used by the algorithm, which assists the algorithm in finding the optimum individual. However, this has a cost in the form of computational requirements - the larger the population, the longer the algorithm takes to run. Both of these tendencies may be observed in Tab. 4. Despite the fact that, in the population size tests, using a population size of 150 also resulted in the best solution, we decided to keep the standard population size at 300 individuals to ensure that the algorithm is more consistent in its solutions.

Table 5. NACA 4-digit tests: mutation chance.

| Mutation chance | NACA airfoil | L/D | Algorithm iterations | Time |
|---|---|---|---|---|
| 2% | 5313 | 35.2479 | 10 | 28 min and 8.67 s |
| 10% | 5311 | 36.4782 | 10 | 28 min and 12.97 s |
| 20% | 5311 | 36.4782 | 10 | 29 min and 0.96 s |

Mutation a way of adding more variation after the population is generated. In the tests for the NACA 4-digit algorithm, documented in Tab. 5, the mutation chance configurations did not strikingly affect the results, probably due to the large population size - that is, due to the large initial genetic variation available to the algorithm. Anyhow, the value of 10% was chosen, for it provides just enough additional genetic information while not adding too much randomization to the optimization process.

Table 6. NACA 4-digit tests: XFOIL iterations.

| XFOIL iterations | NACA airfoil | L/D | Algorithm iterations | Time |
|---|---|---|---|---|
| 10 | 8412 | 32.2738 | 10 | 23 min and 35.76 s |
| 50 | 5311 | 36.4782 | 10 | 28 min and 34.19 s |
| 100 | 5311 | 36.4782 | 10 | 36 min and 31.55 s |

Depending on the complexity of the geometry of the airfoil, XFOIL may require additional simulation iterations in order to converge to a solution (namely, the aerodynamic data). Not all NACA 4-digit airfoils simulation converge within

10 iterations, which is initial value XFOIL is configured - however, more XFOIL iterations increase the time the algorithm takes to run. The effect of this parameter was studied and, as shown in Tab. 6, 50 iterations already provides the best solution, so it was chosen as the standard value for the NACA 4-digit algorithm.

In the algorithms, XFOIL has an additional function other than simulating the airfoils. When a given airfoil is simulated in XFOIL, but the solution does not converge, XFOIL does not return any aerodynamic data of that airfoil to the algorithm. This means that the algorithm has no way to evaluate the airfoil. Thus, this given airfoil is treated as a dud, which does not take part in the crossover stage. When XFOIL simulations do not converge, the reason, in the majority of the cases, is excessively unconventional or outright undesirable geometry. In the case of the NACA 4-digit airfoils, these anomalies may be odd configurations of camber (for example, $p = 9$) or exaggerated thickness. Therefore, due to this behavior, XFOIL helps the algorithm to filter out bad airfoils - because, as mentioned, these individuals do not pass on their genes. This process is even more important in the CST algorithm, as shall be described afterward.

Now, the same analyses are presented for the CST algorithm.

Table 7. CST tests: population size.

| Population | L/D | Algorithm iterations | Time |
|---|---|---|---|
| 100 | 35.6995 | 40 | 38 min and 42.03 s |
| 300 | 45.2408 | 40 | 116 min e 8.99 s |
| 500 | 40.0441 | 40 | 192 min 36.48 s |

The effects of population size are the same as observed in Tab. 4, at least in regards to the necessary computation time. Increasing the population size tends to improve the results, but that is not necessarily the norm every time, as Tab. 7 shows.

Due to the increased number of design variables, common sense says that the CST algorithm should have much larger population sizes than the NACA 4-digit algorithm for the best results. However, due to the results of Tab. 7, and due to time constraints, the population size of 300 individuals was chosen as standard for these tests.

The higher complexity of the CST parametrization requires that the CST algorithm runs for more iterations than the NACA 4-digit algorithm in order to consolidate better results, which naturally increases the running time. Also, the processes of geometry quality check and coordinate generation, which are not present in the NACA 4-digit algorithm, add to the running time of the CST algorithm.

Table 8. CST tests: mutation chance.

| Mutation chance | L/D | Algorithm iterations | Time |
|---|---|---|---|
| 2% | 28.6364 | 40 | 115 min and 47.76 s |
| 10% | 45.5166 | 40 | 120 min and 22.30 s |
| 20% | 44.5890 | 40 | 127 min and 6.73 s |

As explained previously, the mutation mechanism of the CST algorithm is more subtle than the one in the NACA 4-digit algorithm. It also tends to facilitate the search for the airfoils that achieve the objective function as a way to compensate for the much larger search space of the CST airfoil optimization problem. Therefore, the highest value of mutation chance was chosen as standard as a mean to offset the relatively low population size, despite the longer run time caused by the more frequent mutation processes.

Table 9. CST tests: XFOIL iterations.

| XFOIL iterations | L/D | Algorithm iterations | Time |
|---|---|---|---|
| 10 | 43.4532 | 40 | 123 min and 31.35 s |
| 50 | 46.2883 | 40 | 138 min and 20.81 s |
| 100 | 47.0372 | 40 | 161 min and 19.47 s |

The secondary role of XFOIL (to serve as an additional quality check), mentioned previously, is more critical in the CST algorithm. The CST quality checks ensure that the individuals have reasonable airfoil shapes, in the sense that these shapes must at least resemble airfoils - but whether these airfoils have realistic geometries is up to the rest of the algorithm to evaluate by means of the aerodynamic simulations and the objective function. Therefore, because the CST parametrization is more prone to generating airfoils with unconventional geometries, XFOIL was configured with the standard 10 simulation iterations as a more severe countermeasure against bad airfoils. The small number of iterations also considerably reduces running time, as shown in Tab. 9.

### 4.2 General tests

The NACA 4-digit algorithm was executed, for 10 iterations, under each of the case studies presented in Tab. 2. The results are presented in Tab. 10.

Table 10. NACA 4-digit algorithm: general tests.

| Case study | NACA airfoil | L/D | Algorithm iterations | Time |
|---|---|---|---|---|
| 1 | 5311 | 36.4782 | 10 | 28 min and 42.51 s |
| 2 | 3710 | 61.2614 | 10 | 27 min and 7.07 s |
| 3 | 9610 | 167.4271 | 10 | 22 min and 37.69 s |
| 4 | 9510 | 187.3425 | 10 | 23 min and 43.53 s |

Next, again under the same case studies from Tab. 2, we ran the CST algorithm for sufficient iterations until it could improve the airfoils no further.

Table 11. CST algorithm: general tests.

| Case study | L/D | Algorithm iterations | Time |
|---|---|---|---|
| 1 | 47.2991 | 80 | 244 min and 2.52 s |
| 2 | 64.2192 | 100 | 280 min and 30.29 s |
| 3 | 178.6068 | 80 | 246 min and 49.65 s |
| 4 | 193.5139 | 60 | 181 min and 32.19 s |

Figure 1 provides a visualization of the airfoils obtained by both algorithms, where the NACA 4-digit airfoils correspond to those presented in Tab. 10 and the CST airfoils correspond to those presented in Tab. 11.

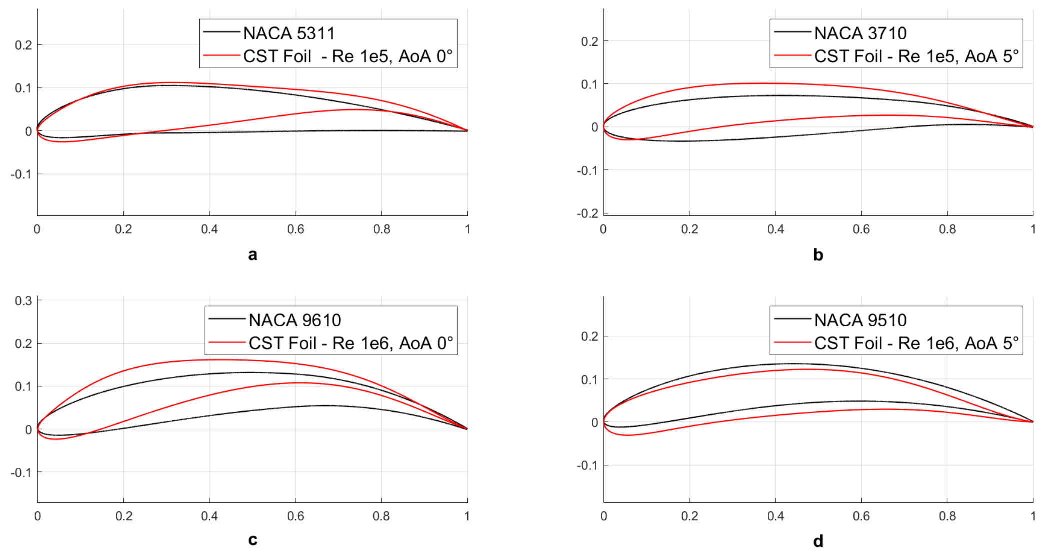

Figure 1. Comparison of the NACA 4-digit airfoils and the CST airfoils.

## 5. CONCLUSIONS

When analysing the results from the algorithms, regardless of the geometrical parametrization method used, two tendencies in the airfoils may be observed: low thickness and high camber. Both aspects are justified, because thin airfoils have less drag (both viscous and pressure drag) and higher curvature contributes to the generation of lift. Therefore, both tendencies contribute to higher lift-to-drag ratios. However, despite sharing these general characteristics, the way each airfoil was molded considerably changes with each case study, which is especially noticeable in regards to the CST airfoils. The design freedom of such a thorough parametrization method allows for interesting optimization results, demonstrating how specific flight conditions mold the airfoil geometries in specific ways. Such behavior is not exclusive to genetic algorithms and the CST method, as Srinath and Mittal (2008) show.

Additionally, the algorithm may evaluate the airfoils with objective functions different than the one employed in this work. For instance, instead of searching for an airfoil with the highest possible $L/D$ at a given flight condition, the algorithm may be configured to search for an airfoil that best matches a value of $L/D$ specified by the designer. Alternatively, the airfoils could be evaluated by their lift coefficients only, or perhaps be evaluated by a combination of parameters - for example, given a Reynolds number and an angle of attack, find an airfoil with the highest possible $L/D$, which at the same time has a specified value of moment coefficient.

It is relevant to note that the airfoils obtained in this paper may not be practical to use, or even safe, for there are other aspects that were not covered by the algorithm - for instance, internal space for fuel tanks or control surface mechanisms, moment characteristics, stall characteristics. The characteristic cusp shape of the trailing edges of the CST airfoils obtained are not ideal from both an aerodynamic perspective and a structural perspective. In other words, it was optimization simply for the sake of understanding the algorithm. However, the results obtained by analyzing only the lift-to-drag ratio show that the methodology itself is reliable and consistent, which, considering the thoughts presented in the previous paragraph, means that the algorithm may be adapted according to necessity and used as a design tool, allowing for actual, mindful design of the airfoil considering the aircraft as a whole and its complete flight envelope. As for the aforementioned cusp shaped trailing edges of the CST airfoils, it is a problem that may be solved by specifying a minimum separation in terms of the boattail angles, and further analysis regarding the effects of the configuration parameters (population size, mutation chance, XFOIL iterations, and others not considered in this paper) would be ideal as well.

## 6. REFERENCES

Abbott, I.H. and Doenhoff, A.E.V., 1959. *Theory of Wing Sections: Including a Summary of Airfoil Data*. Dover Publications, INC, New York.

Dina, A., Danaila, S., Pricop, M.V. and Bunescu, I., 2019. "Using genetic algorithms to optimize airfoils in incompressible regime". *INCAS BULLETIN*, Vol. 11, pp. 77–90.

Drela, M., 1989. "Xfoil: An analysis and design system for low reynolds number airfoils". MIT Dept. of Aeronautics and Astronautics, Cambridge, Massachusetts.

Gudmundsson, S., 2014. *General Aviation Aircraft Design: Applied Methods and Procedures*. Butterworth-Heinemann, Oxford.

Kulfan, B.M., 2008. "Universal parametric geometry representation method". *Journal of Aircraft*, Vol. 45, pp. 142–158.

Kulfan, B.M. and Bussoletti, J.E., 2006. ""Fundamental" Parametric Geometry Representations for Aircraft Component Shapes". In *11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. Portsmouth, Virginia.

Ram, R.K.G., Cooper, Y.N., Bhatia, V., Karthikeyan, R. and Periasamy, C., 2014. "Design optimization and analysis of naca 0012 airfoil using computational fluid dynamics and genetic algorithm". *Applied Mechanics and Materials*, Vol. 664, pp. 111–116.

Srinath, D.N. and Mittal, S., 2008. "Optimal airfoil shapes for low reynolds flows". *International Journal for Numerical Methods in Fluids*, Vol. 61, pp. 355–381.

Yang, X., 2010. *Engineering Optimization: An Introduction with Metaheuristic Applications*. John Wiley & Sons, Inc, New Jersey.

## 7. RESPONSIBILITY NOTICE

The authors are solely responsible for the printed material included in this paper.