Thermal Engineering & Renewable Energies

Project Report

Optimization Study of NACA Airfoil using Nonlinear Programming & Genetic Algorithms

Authors
Anass EL HOUD
Yasser HALLOU

Supervisor
Prof. Rabii EL MAANI

Department of Energy,
Ecole Nationale Supérieure d'Arts et Métiers
ENSAM Meknes, Morocco

Printed and submitted
May 20, 2019

Acknowledgements

The completion of this work would not have been possible without the support and guidance of various people. We are privileged to get this assistance up to the achievement of our project. We wouldn't forget to thank them as all we've done up to this point is only thanks to such supervision and help.

In the first place, we would like to express our deep gratitude to **Mr. EL MAANI Rabii**, the project supervisor, for his interest on our project work, for his patient guidance all along and for his useful critiques. We have very much appreciated his willingness to give his time and energy so generously. His explanations were very helpful in our understanding of the purpose of this project. And his useful and constructive recommendations were essential in choosing the methods and algorithms involved in our work.

We would also thank all the teaching staff of Department of Energy for their encouragement, support and professional guidance, which helped us in successfully completing our project work. We would extend our sincere esteems to **Mr. MASROUR Tawfik**, professor in Department of Mathematics, for his help by providing key information in Nonlinear Programming algorithms, which were essential for the completion of our work. We would also thank him for his enthusiastic encouragement.
Last but not least, as no person is more important in the pursuit of this project than them, it should not forgotten to thank our families for their unconditioned support and love in whatever we pursue.

Contents

Introduction **1**

1 Airfoil Parametrization **4**

1.1 Introduction 4

1.2 Mathematical and numerical generation 5

1.2.1 Procedure of generation 6

1.2.2 Coding 7

1.3 Results of airfoil generation 9

1.4 PARSEC Method 10

2 Optimization with Discrete Nonlinear Programming **11**

2.1 Mathematical formulation 11

2.1.1 Calculation of the area 12

2.2 Implementation in MATLAB 13

2.2.1 PARSEC Transformation 13

2.2.2 Fmincon MATLAB function 14

2.3 Results 15

3 Optimization with Genetic Algorithms **16**

3.1 Principle of Genetic Algorithms 16

3.2 Implementation in MATLAB 17

3.3 Result 21

Conclusion **23**

List of Figures

1 Genetic operators: selection, crossover and mutation. 2

1.1 Airfoil nomenclature 5
1.2 Difference between symmetrical and asymmetrical airfoils 5
1.3 Plot of NACA 2412 airfoil after generation 9
1.4 PARSEC parameters for an airfoil representation. 10

2.1 Initial and optimized airfoil with NL Programming 15

3.1 Initial and optimized airfoil with GA 21
3.2 Evolution of fitness through generations 22

List of Tables

1.1 Sampling of the results of the airfoil generation 9
1.2 PARSEC parameters of NACA 2412 10

2.1 Area values of original and optimized airfoil 15

3.1 Genetic parameters values 17
3.2 Area values of original and optimized airfoil using GA 22

List of acronyms

CFD	*Computational Fluid Dynamics*
NLP	*Nonlinear Programming*
GA	*Genetic Algorithms*
NACA	*National Advisory Committee for Aeronautics. Refers to airfoil shapes*
Airfoil	*The cross-sectional shape of a wing of planes in American English.*
RWS	*Roulette wheel selection or Fitness proportionate selection*
PARSEC	*PARametric SECtion method*

Introduction

Optimization methods in various fields of technology have been studied for several years. The most common general approach is the calculation of the gradient. The reliability and success of gradient methods generally require a regular state space and the cost function can have multiple local minimum. However, the gradient methods converge to the first found minimum that can be characterized by a value of the cost function that is mediocre compared with the absolute minimum.

In order to reduce fuel consumption of airplanes, the efficiency of airplanes and its components needs to be improved. In particular, wings or airfoils play a major role in lifting the plane. The design of the airfoils affects directly the drag coefficient and the lift coefficient, thus the aerodynamics performance of the plane. In general, CFD models are used to simulate the air flow around the airfoil in order to optimize its aerodynamic performance using optimization algorithms (i.e. simplex-based, gradient-based ...) that try to minimize the aerodynamic cost function, generally the Lift-to-drag ratio. Unlike the typical methodology of shape optimization, direct equations are used to generate the initial airfoil. In this study, we will solely focus on optimizing the geometry and area of an airfoil using Nonlinear programming solver and genetic algorithms.

All the optimization algorithms are based on the same mathematical model. The idea is that we minimize or maximize a function subject to linear or nonlinear constraints. But each algorithm has a specific way to reach his goal. We can mention, for example, Ant colony optimization (ACO), Differential evolution (DE) and Swine flow Optimization Algorithm. However, the objective and mathematical form remains the same, and can be written in the following form

$$\begin{aligned} \min_{x_1,x_1,\ldots,x_n} \quad & f_0(x_1, x_2, \ldots, x_n) \\ \text{s.t.} \quad & f_j(x) \leq b_i,\ j = 1, \ldots, m. \\ & x_i \geq 0,\ i = 1, \ldots, n. \end{aligned}$$

Where f_0 is the cost function, $(x_1, x_1, \ldots, x_n)$ are the decision variables and below it we find the linear (or non linear) constraints. A matrix representation is often used to gather all the linear constraints related to the decision variables which will be called control points.

As mentioned above, this study is based on two different optimization algorithms, which we found more suitable for our case, nonlinear programming solver and Genetic algorithms.

Nonlinear Programming. it's another generalization of applied mathematics that amounts to minimizing a nonlinear operate subject to nonlinear constraints. Many methods proposed so far for solving this type of problems lead to replace the problem given by a series of problems without constraints, whose resolutions use conventional methods especially so-called gradient methods. In our study, we will focus on the nonlinear programming MATLAB solver in order to manipulate its code to write a new one that will help us solve the problem we face.

Genetic Algorithms. The basic idea behind the genetic algorithms approach is to search for optimal solutions using an analogy to Darwin's theory of evolution "Survival of the fittest". During solution iteration, decision variables or genes are manipulated using various genetic operators (selection, crossover and mutation) to create a new generation. The method consists in using a population of individuals, each individual being a design vector, and in simulating evolution by selection with this population. Starting from an initial population drawn randomly in the research space, the solution is evolved using the various genetic operators of the algorithm.

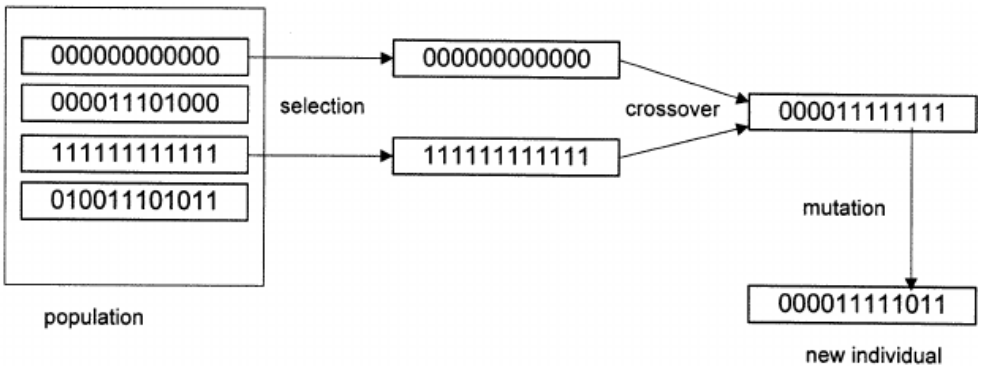

Figure 1: Genetic operators: selection, crossover and mutation.

- Selection: based on Darwin's evolution theory, the best individuals should survive and create new generation. In the literature, there are several selection methods: Wheel Selection [1] , Stochastic Universal Sampling, the tournament selection and the selection of Boltzmann and others [2].

- Crossover: combine selected individuals (parents) to get new different individuals (fig. 1). Examples of some methods, one-point crossover, n-point crossover and uniform crossover.

- Mutation: The main purpose of this GA operator is to generate random population by random changes in order to avoid local optimum (fig. 1). Many types are used such as Bit Flip Mutation [3] , Random Resetting, Swap Mutation and Inversion Mutation [4].

At the beginning of each iteration, the algorithm must calculate the **fitness function** which measures the ability of a solution to compete with other solutions. The probability that an individual (a solution) will be selected for reproduction is based on its fitness score.

In the following study, we will use this two methods, Nonlinear Programming and Genetic Algorithms in order to optimize the area of the airfoil, starting from the initial population of points of NACA 2412.

Chapter 1

Airfoil Parametrization

1.1 Introduction

The NACA airfoils are developed by the National Advisory Committee for Aeronautics (NACA). They are the most popular and notorious profile series in aircraft construction. The shape of the NACA airfoils is described using a series of digits following the word NACA. Before the National Advisory Committee for Aeronautics (NACA) developed these series, airfoil design was rather arbitrary with nothing to guide the designer except experience with known shapes and experimentation with modifications to those shapes.

This methodology began to change in the early 1930s with the publishing of a NACA report entitled "The Characteristics of 78 Related Airfoil Sections from Tests in the Variable Density Wind Tunnel". In this landmark report, the authors noted that there were many similarities between the airfoils that were most successful, and that the two primary variables that affect those shapes are the slope of the airfoil (mean camber line) and the thickness distribution above and below this line (fig. 1.4). They then presented a series of equations incorporating these two variables that could be used to generate an entire family of related airfoil shapes.

The early NACA airfoil series, the 4-digit, 5-digit, and modified 4-/5-digit, were generated using analytic equations that describe the camber (curvature) of the mean-line (geometric center-line) of the airfoil section as well as the section's thickness distribution along the length of the airfoil.

The parameters in the numerical code can be entered into equations to precisely generate the cross-section of the airfoil and calculate its properties. All dimensions in % are understood as % of chord length, with the line connecting the leading and trailing edges, unless otherwise specified. (fig. 1.4)

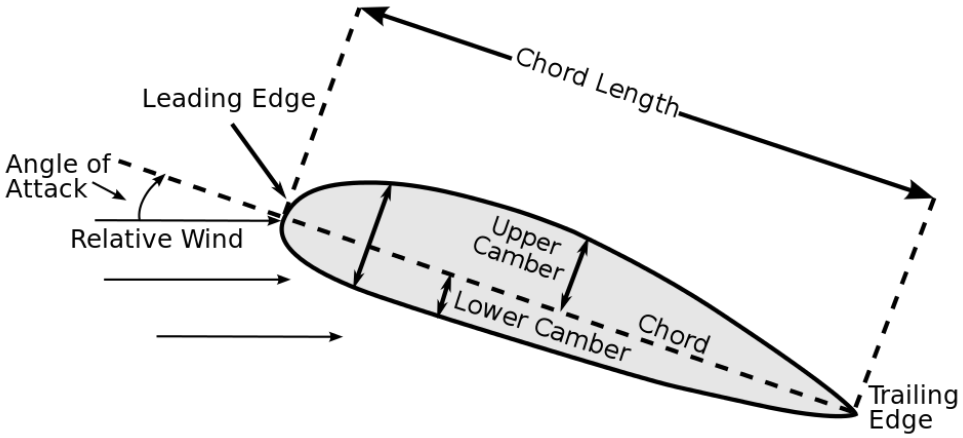

Figure 1.1: Airfoil nomenclature

NACA Four-Digit. The first family of airfoils designed using this approach became known as the NACA Four-Digit Series. The first digit specifies the maximum camber (m) in percentage of the chord (airfoil length), the second indicates the position of the maximum camber (p) in tenths of chord, and the last two numbers provide the maximum thickness (t) of the airfoil in percentage of chord. Using these m, p, and t values, we can compute the coordinates for an entire airfoil. And when changing them, a new airfoil shape is resulted. Our purpose is to change this parameters in an iterative way to get the optimum solution or shape.

1.2 Mathematical and numerical generation

The first thing to note is that there are two main categories of the four-digit series, namely the symmetrical airfoils and cambered ones (non-symmetrical). In our case, the NACA 2412 is a cambered four-digit since its formula is different from 00XX, which is a signature of a symmetrical airfoil.

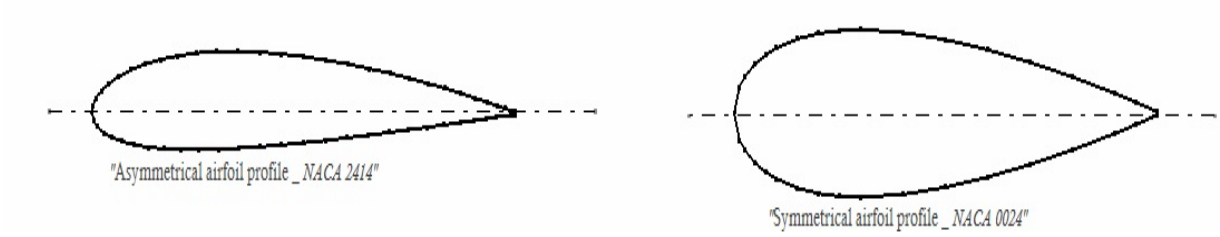

Figure 1.2: Difference between symmetrical and asymmetrical airfoils

The NACA Four-Digit sections define the profile by:
1. First digit describing maximum camber as percentage of the chord.
2. Second digit describing the distance of maximum camber from the airfoil leading edge in tens of percent of the chord.
3. Last two digits describing maximum thickness of the airfoil as percent of the chord.

1.2.1 Procedure of generation

The procedure to generate a cambered four-digit profile: (using MATLAB)

- Firstly, we define the parameters of the profile to be generated i.e. *c, m, p, t.*
- We define the vector x=(x_i) containing points from 0 to 1 with a step of 0.01, as all dimension are understood as % of the chord length *c*.
- Then we calculate the mean camber line using:

$$y_c = \begin{cases} \frac{m}{p^2} \times \left(2p \times \left(\frac{x}{c} - (\frac{x}{c})^2 \right) \right) & \text{if } 0 \leq x \leq p \times c \\ \frac{m}{(1-p)^2} \times \left(\left(1 - 2p \right) + 2p \times \left(\frac{x}{c} - (\frac{x}{c})^2 \right) \right) & \text{if } p \times c \leq x \leq c \end{cases}$$

Where m is the maximum camber ($100 \times m$ is the first of the four digits), p is the location of maximum camber ($10 \times p$ is the second of the four digits). **Note** that the vector y_c has the same length as the vector x.

- Then we calculate the thickness distribution above (+) and below (-) the mean line by plugging the value of t into the following equation for each of the x coordinates:

$$y_t = 5 \times t \times \left[0.02969 \times \sqrt{\frac{x}{c}} - 0.1260 \times \frac{x}{c} - 0.3516 \times \left(\frac{x}{c}\right)^2 + 0.2843 \times \left(\frac{x}{c}\right)^3 - 0.1015 \times \left(\frac{x}{c}\right)^4 \right]$$

Where y_t is the half thickness at a given value of x (mean line to surface), t is the maximum thickness as a fraction of the chord (so t is equal to the last two digits in the NACA 4-digit denomination divided by 100). **Note** that a zero-thickness trailing edge is required, one of the coefficients should be modified such that they sum to zero. Modifying the last coefficient (i.e. to 0.1036) will result in the smallest change to the overall shape of the airfoil.

The previous equation, the thickness of the airfoil, is sufficient to generate or describe a symmetrical airfoil. Since the coordinates (x_U, y_U) of the upper airfoil surface, and (x_L, y_L) of the lower airfoil surface are:

$$x_U = x_L = x \quad , \quad y_U = +y_t \quad \text{and} \quad y_L = -y_t$$

- For cambered four-digit airfoil, to calculate (x_U, y_U) and (x_L, y_L), we need to calculate another parameter namely

$$\theta = arctan(\frac{dy_c}{dx})$$

- We can calculate the coordinates of the upper and lower airfoil surfaces, as follows:

$$x_U = x - y_t \sin(\theta) \ ; \ x_L = x + y_t \sin(\theta)$$

$$y_U = y_c + y_t \cos(\theta) \ ; \ y_L = y_c - y_t \cos(\theta)$$

Note that (x_U, y_U) and (x_L, y_L) all have the same length, that is the length of x, but in the optimization process, due to the large length, we will use a restricted amount of points (coordinates) to characterize the profile, with the help of SPLINE function.

1.2.2 Coding

Using MATLAB and the last part related to the mathematical generation, we coded the equations into a numerical model where we can generate numerically each NACA profile using basically the four parameters that we have mentioned before: m as the maximum camber, p as the position of the maximum camber, c as the chord length and t as the maximum thickness. The input is the vector x which represents some chosen points of the horizontal axis. The outputs of the MATLAB function are the checkpoints of airfoil, they represent the coordinates of points of the perimeter or section of the airfoil.

For NACA 2412, knowing that we work for a chord length of c=1, we calculate the other three parameters as follows

$$100 \times m = 2 \Rightarrow m = 0.02$$

$$10 \times p = 4 \Rightarrow p = 0.4$$

$$100 \times t = 12 \Rightarrow t = 0.12$$

Using all the equations presented in the previous part, we write a MATLAB script as a parametric function to generate NACA airfoils. We set the four parameters to their values calculated above for NACA 2412. The MATLAB script has the following form

```
function [xu,xl,yu,yl]=generatenaca(x)
n=length(x);
yc=zeros(1,n); % coord de la partie stable
yt=zeros(1,n); % coord de la partie dynamique
theta=zeros(1,n); % angle dynamique
xu=zeros(1,n); % x de lextrados
xl=zeros(1,n); % x de lintrados
yu=zeros(1,n); % x de lextrados
yl=zeros(1,n); % x de lintrados

m=0.02; % cambrure maximale p=0.4; % position de cambrure max
c=1; % longueur de la corde t=0.12; % max thickness

for i=1:n
    if 0<=x(i)<(p*c)
        yc(i)=m*x(i)*(2*p-(x(i)/c))/(c*(p^2));
    else (p*c)<=x(i)<c
        yc(i)=m*(1-2*p+2*p*(x(i)/c)-(x(i)/c)^2)/((1-p)^2);
    end
end

for i=1:n
    yt(i)=5*t*(0.2969*sqrt(x(i)/c)-0.126*(x(i)/c)-0.3516*(x(i)/c)^2 +
        0.2843*(x(i)/c)^3 - 0.1015*(x(i)/c)^4);
end
for i=1:n
    if 0<=x(i)<=(p*c)
        theta(i)=atan(2*m*(p-(x(i)/c))/(c*(p^2)));
    else (p*c)<=x(i)<c
        theta(i)=atan(2*m*(p-(x(i)/c))/(c*(1-p)^2));
    end
end

for i=1:n
   xu(i)=x(i)-yt(i)*sin(theta(i));
   xl(i)=x(i)+yt(i)*sin(theta(i));
   yu(i)=yc(i)+yt(i)*cos(theta(i));
   yl(i)=yc(i)-yt(i)*cos(theta(i));
end
end
```

1.3 Results of airfoil generation

After running the script above on MATLAB, we get the coordinates of the checkpoints of NACA 2412. These points are considered as the initial population for our optimization study. We gather a sampling of the values in the table bellow

x_U	y_U	x_L	y_L
0.0000	0.000	0.000	0.000
0.0094	0.0287	0.0108	-0.0055
0.0192	0.0357	0.0212	-0.0117
0.0491	0.0489	0.0315	-0.0161
0.1197	0.0657	0.1429	-0.0362
0.1298	0.0673	0.7564	-0.0179
0.1399	0.0688	0.7766	-0.0172
0.4549	0.0754	0.8880	-0.0091
0.7588	0.0437	0.8982	-0.0084
0.8998	0.0207	0.9083	-0.0077
0.9098	0.0189	0.9795	-0.0027
1.0000	0.0013	1.0000	-0.0013

Table 1.1: Sampling of the results of the airfoil generation

We plot the approximative section of NACA 2412 airfoil using MATLAB to check its shape and its section. The following figure shows the result

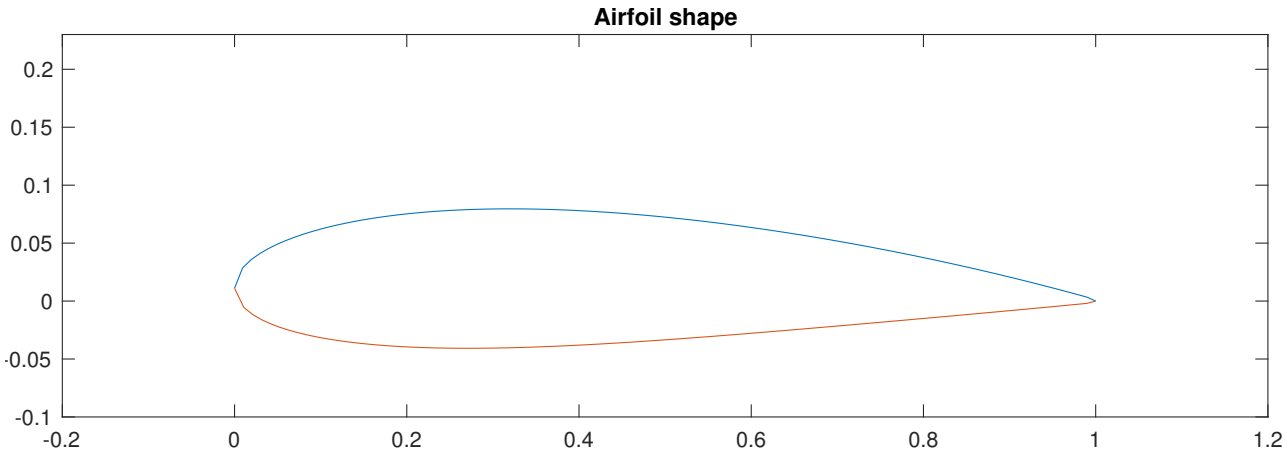

Figure 1.3: Plot of NACA 2412 airfoil after generation

1.4 PARSEC Method

As a result of the previous parametrization, using directly analytic functions that describe both the upper and the lower curvature, we got one hundred point for each curvature. Dealing with this number of points is difficult, henceforth the number of parameters should be reduced. In order to parametrize the airfoil surface using less parameter number, we employ PARSEC Method. [5] In fact, PARSEC Method is a robust method, allowing to create analytically defined airfoils. With each of the 11 parameters representing a geometric property of the airfoil. as shown by the following figure [6]

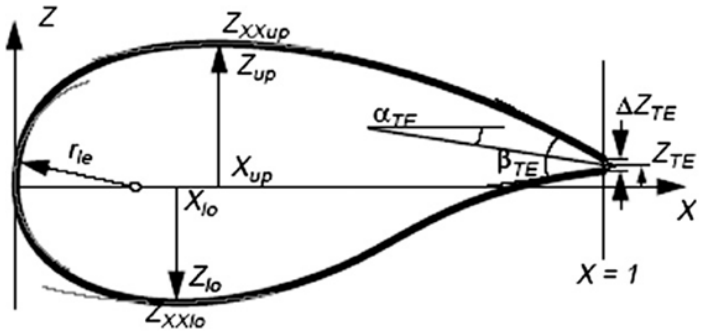

Figure 1.4: PARSEC parameters for an airfoil representation.

The 11 PARSEC parameters represent:
- P1: the radius of the leading edge (rle). P2: the position of the upper crest (Xup). P3: the value of the upper crest (Zup), P4: the upper curvature (Zxxup). P5: the position of the lower crest (Xlo), P6: the value of the lower crest (Zlo). P7: the lower curvature (Zxxlo), P8: the trailing edge coordinate (Zte). P9: the trailing edge thickness (ΔZte), P10: the trailing edge direction (αte). P11: the trailing edge angle (βte).

Depending on the resulting values of the airfoil generation, the PARSEC parameters of our initial airfoil NACA 2412 are calculated and presented in the table bellow

p_1	0.015	p_7	0.400
p_2	0.400	p_8	0.000
p_3	0.080	p_9	0.000
p_4	0.080	p_{10}	0.000
p_5	0.2736	p_{11}	0.000
p_6	-0.04	–	–

Table 1.2: PARSEC parameters of NACA 2412

Chapter 2

Optimization with Discrete Nonlinear Programming

Nonlinear programming refers, generally, to the field of mathematics handling the problems of optimization for a given set of variables satisfying certain constraints, typically given by equalities or inequalities and also a certain criterion, called objective function, which depends on such variables, is optimized, that is it attains its minimum value among all the combination of feasible variables.
In reality, many problems met in practice are formulated as constrained problems. This is because in most instances a complex problem such as, for example, the design of a complex device cannot be directly treated in its entirety accounting for all possible choices, but instead must be decomposed into separate sub-problems, each sub-problem having constraints that are imposed to restrict its scope. [7]

2.1 Mathematical formulation

Before elaborating the optimization algorithm, we start by formulating our problem using mathematical equations. This formulation refers to translating our problem into the form of mathematical equations which could be solved. It usually requires a thorough understanding of the problem. In our case, we are trying to minimize the area of the airfoil based on an initial control points generated by the PARSEC method.

We can write the mathematical formulation of our problem as given bellow

$$\begin{aligned} \min_{p_1,x_1,\ldots,p_{11}} \quad & A(p_1, p_2, \ldots, p_{11}) \\ \text{s.t.} \quad & p_j \leq b_j,\ j = 1, \ldots, 11. \\ & p_j \geq c_j,\ j = 1, \ldots, 11. \\ & A(p_1, p_2, \ldots, p_{11}) > 0. \end{aligned}$$

Where A is the area of the airfoil, it can be calculated using the PARSEC parameters $(p_1, x_1, \ldots, p_{11})$ of the airfoil. b_i and c_i are respectively the maximum and minimum limit of the PARSEC parameters in order to respect the global airfoil shape.

In order to respect the form of an airfoil, we set the limit vectors of the PARSEC parameters to the values bellow

b_i=-c_i=[0.0015, 0.025, 0.015, 0.01, 0.02, 0.015, 0.075, 0, 0, 0.175, 0.05]

2.1.1 Calculation of the area

As mentioned above, our objective function is not analytic. In each iteration, we need to calculate its value in order to compare it with its values the previous iterations. In this case, we are going to transform the analytic form of the supposed function to a numerical form, easy to evaluate. The area of a 2D geometry is given by

$$A_{x,y} = \iint_{\mathbf{R}^2} f_a(x, y)\, \mathrm{d}x\, \mathrm{d}y \tag{2.1}$$

Where $f_a(x, y)$ is the analytic function describing the 2D shape of the airfoil. Or, mathematically, it is difficult to calculate this equation. Therefore, we will use the Trapezoidal rule to approximate the value of the area. The equation (2.1) becomes

$$A_{x,y} \approx \sum_{n=1}^{N} \frac{f(x_n)) + f(x_{n+1})}{2} \Delta x_n \tag{2.2}$$

Where x_n are discrete points of the airfoil in X-axis. and $y_n = f(x_n$ are their images in the Y-axis. This equation will be implemented and programmed in MATLAB in order to calculate numerically the value of the area of the airfoil in each generation or iteration.

2.2 Implementation in MATLAB

2.2.1 PARSEC Transformation

We created two MATLAB scripts that serve to transform the PARSEC parameters into X-axis and Y-axis coordinates. The first script or MATLAB function works on transforming the input vector containing the PARSEC parameters into polynomial coefficients a_i. This function determines a=[a1, a2, ...an] to solve the airfoil polynomial. The number n in the function represents the number of coordinates for the upper or lower surface. The code of the first script is given by (it is also attached with the digital version of this document)

```
function a=parsec(p)
c1=[1,1,1,1,1,1];
c2=[p(2)^(1/2),p(2)^(3/2),p(2)^(5/2),p(2)^(7/2),p(2)^(9/2),p(2)^(11/2)];
c3=[1/2, 3/2, 5/2, 7/2, 9/2, 11/2];
c4=[(1/2)*p(2)^(-1/2),(3/2)*p(2)^(1/2),(5/2)*p(2)^(3/2),(7/2)*p(2)^(5/2),
(9/2)*p(2)^(7/2),(11/2)*p(2)^(9/2)];
c5=[(-1/4)*p(2)^(-3/2),(3/4)*p(2)^(-1/2),(15/4)*p(2)^(1/),(35/4)*p(2)^(3/2),
(53/4)*p(2)^(5/2),(99/4)*p(2)^(7/)];
c6=[1,0,0,0,0,0];
c7=[1,1,1,1,1,1];
c8=[p(5)^(1/2),p(5)^(3/2),p(5)^(5/2),p(5)^(7/2),p(5)^(9/2),p(5)^(11/2)];
c9=[1/2, 3/2, 5/2, 7/2, 9/2, 11/2];
c10=[(1/2)*p(5)^(-1/2),(3/2)*p(5)^(1/2),(5/2)*p(5)^(3/),(/2)*p(5)^(5/2),
(9/2)*p(5)^(7/2),(11/2)*p(5)^(9/2)];
c11=[(-1/4)*p(5)^(-3/2),(3/4)*p(5)^(-1/2),(15/4)*p(5)^(1/2),(35/4)*p(5)^(3/2),
(53/4)*p(5)^(5/2),(99/4)*p(5)^(7/2)];
c12=[0,0,0,0,0,1];
Cup=[c1; c2; c3; c4; c5; c6];
Clo=[c7; c8; c9; c10; c11; c12];
bup=[p(8)+p(9)/2;p(3);tand(-p(10)+p(11)/2);0;p(4);(sqrt(2*p(1)))];
blo=[p(8)-p(9)/2;p(6);tand(p(10)+p(11)/2);0;p(7);-(sqrt(2*p(1)))];
aup=linsolve(Cup,bup);
alower=linsolve(Clo,blo);
a(:,1)=aup;
a(7:12,1)=alower;
end
```

The output of the first script will be used as the input of the second script. More specifically, this function calculates the y-coordinates for a given PARSEC coefficients and X-coordinate points. The second script is given by

```
function [yu,yl]=yCoord2(a,x)
yu=a(1)*x^.5+a(2)*x^(1.5)+a(3)*x^(2.5)+a(4)*x^(3.5)+a(5)*x^(4.5)+a(6)*x^(5.5);
yl=a(7)*x^.5+a(8)*x^(1.5)+a(9)*x^(2.5)+a(10)*x^(3.5)+a(11)*x^(4.5)+a(12)*x^(5.5);
end
```

2.2.2 Fmincon MATLAB function

We are using the Fmincon function included in MATLAB to find the optimal solution. The cost function is the area of the airfoil that we try to minimize. The main script is presented below. The range vector represents the max and min value of the variation of the PARSEC parameters. We created two vectors, C1 and C2, related to the linear inequalities.

```
p0=[0.015 0.4 0.08 0.08 0.2736 -0.04 0.4 0 0 0 0]; %%NACA 2412
range=[0.0015 0.025 0.015 0.01 0.02 0.015 0.075 0 0 0.175 0.05];

C1=p0-range; %min inequality constraints
C2=p0+range; %max inequality constraints
A=[]; %empty linear equality
b=[]; %empty linear equality
Aeq=[]; %empty equality constraints
beq=[]; %empty equality constraints

%main solver
[p,fval]=fmincon(@(p)airenaca1(p),p0,A,b,Aeq,beq,C1,C2);

plotairfoil(p,'k') %plotairfoil function already programmed to plot the airfoil
    section based on its parsec parameters
hold on
plotairfoil(p0,'r')
axis('equal')
legend('Optimized','Original')

%calculate the original area value of the initial airfoil
Sorig=airenaca(p0);

%print the area value of the original and optimized airfoil
fprintf(' Original Surf= %f \n Optimized Surf= %f \n',Sorig,fval)
```

2.3 Results

As result of the Nonlinear programming study above, we managed to optimize the airfoil's area. The figure below shows the difference in shape between the initial and optimized airfoils

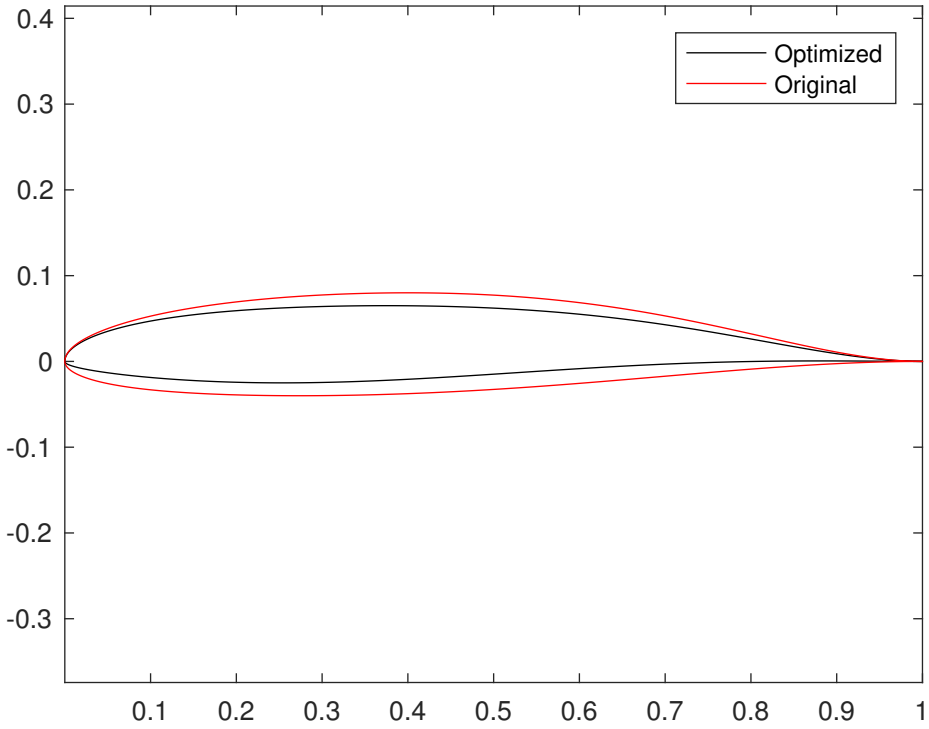

Figure 2.1: Initial and optimized airfoil with NL Programming

The area of the original and optimized airfoils are represented in the following table

Airfoil	Surface
Initial	0.077
Optimized	0.055

Table 2.1: Area values of original and optimized airfoil

Chapter 3

Optimization with Genetic Algorithms

Influenced by Charles Darwin's theory of evolution, genetic algorithms are a range of algorithms that solve both constrained and unconstrained optimization problems, using natural selection and mutation; the driving processes of evolution. Owing to their most interesting characteristics namely efficiency, simple programmability and robustness, it is mathematically possible for such algorithms to retrieve a solution regardless of the correctness of the initial population. Generally, genetic algorithms are usually applied in a variety of optimization problems where classical methods are not suitable, especially when the objective function is not analytic (discrete), non-differential, stochastic or highly nonlinear with many local optima. Unlike classical algorithms, where, on the one hand, at each iteration, we generate a new single point and the points sequence converges to an optimal solution, and on the other hand, the selection approach or rules are deterministic. Genetic algorithms mimic the natural process, and so, instead of a single point, a whole population of points is generated at each iteration in which only the fittest points approach the optimal solution, and the next population of points is selected through sampling search, blind search and using stochastic operators. [8].

3.1 Principle of Genetic Algorithms

Diving directly to how such algorithms work, we encounter 5 phases: Initial population, Fitness function, Selection, Crossover and Mutation.
- To start off, we dispose of an initial population constituted by a set of individuals, with each one being a solution to the optimization problem. Each one of these individual's chromosomes are represented using an array of numbers which themselves represent genes.
Note. In our case, each individual is defined by a specific PARSEC parameters array which represent the chromosome. While each of these parameters represent a gene. For each individual, the area of the airfoil can be calculated.

- After the generation of initial population of individuals, these individuals enter a competition to determine which are the fittest ones for reproduction. In fact, using a fitness function, each individual is attributed a fitness score depending on the solution it leads to. Individuals with higher fitness score have more chance to reproduce.
Note. In our case, the fitness score depends on the area of the airfoil induced by the individual's chromosome. And so individuals that lead to smaller areas possess the bigger fitness score.
- The selection phase consists on choosing the fittest individuals and letting them reproduce so their higher quality genes can pass to the next generation.
- Crossover is the most important phase as within it the next generation comes to life. In fact, a cross point shall be chosen randomly within the genes for each parent pair. And so, the offspring are created by exchanging the genes of the parents pair. Then these offspring replace the least fitted individuals in the population.
- To prevent monotony and to ensure diversity, while the genes are being exchanged, mutations may randomly occur. And so, some of the offspring genes can change drastically from their ancestors in just some few generations. It is thanks to this mutation that we can attain a global optima.
Note. In our case, mutation is guaranteed by the randomize function. The PARSEC parameters are randomized, consequently we obtain new individuals that add-up to the population at the detriment of ones who have the lower fitness.

3.2 Implementation in MATLAB

The implementation of Genetic Algorithms in MATLAB are a little bit difficult, specially that we worked on a whole genetic algorithm instead of using the GA function on MATLAB. First, we set the genetic parameters (such as crossover parameter, mutation parameter...). Those parameters control the accuracy and the precision of the algorithm, we have chosen the following values

Selection percentage	0.05
Crossover probability	0.75
Mutation probability	0.2

Table 3.1: Genetic parameters values

We created over 7 MATLAB scripts, each one of them has a specific function to be called in the main script that we will run:

- **GAairfoil.m**: this is the main script of the genetic algorithms. It contains all the

genetic operations that produce the evolution of individuals until getting the optimal solution.
- **randp.m**: this function is called inside the GAairfoil script. It helps in generating new population via mutation and crossover, by giving pseudo-random numbers in a specific range of values.
- **parsec.m**: this function is also called inside the GAairfoil script. It helps in generating the parsec coefficients for the airfoil parametrization based on parsec parameters.
- **yCoord2.m**: this function takes the parsec parameters and the output of parsec.m function and transforms them into X and Y coordinates, ready to plot and visualize.
- **plotairfoil.m**: this function calls the yCoord2.m function in order to plot the airfoil based on the output coordinates.
- **airenaca.m**: this function calculates the area value for each input airfoil.
- **runit.m**: this is the main script of the optimization. We set the initial airfoil, and run the genetic algorithm and other instructions.

All the files are attached with the digital version of this report. You find below the code of the main script (runit.m):

```
%initial individual
p0=[0.015 0.4 0.08 0.08 0.2736 -0.04 0.4 0 0 0 0]; %%NACA 2412
range=[0.0015 0.025 0.015 0.01 0.02 0.015 0.075 0 0 0.175 0.05];
%Solver parameters
genNo=100;      %number of generations
%Genetic solution
[AAoriginal,AAfittest,fittest,fitness]=GAairfoil(genNo,p0,range);
%ploting and graphing
fprintf(' Original Area= %f \n Optimized Area= %f \n',AAoriginal,AAfittest)
axis('equal')
legend('Optimized','original')
```

Also the code of the main script of the genetic algorithm (GAairfoil.m):

```
function [AAoriginal,AAfittest,fittest,fitness]=GAairfoil(genNo,p0,range)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%genNo     number of generations to mate
%p0        Original airfoil to oprimize
%range     Randomizer range to vary the PARSEC parameters
%Npanel    number of panels for the fitness function
%fitness   represents the fitness rate of each individual
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%genetic parameters
[AAoriginal,~]=airenaca(p0);
popsize=40; %population size
transprob=0.05; %transcendence percentage
crossprob=0.75;  %cross over percentage
mutprob=0.2;     %mutation percentage
newpop=[];
fitness=[];
for k=1:genNo
AA=[];
p=[];
%population evaluation (starting from the second generation)
for i=1:length(newpop)
    p1=newpop(i,:);
    [AAnew, ~]=airenaca(p1); %fitness evaluation
    AA=[AA;AAnew];
    p=[p;p1];
end
%first population initialization
for i=1:popsize-length(newpop)
    p1=randp(p0,range);
    [AAnew, maxThickness]=airenaca(p1); %fitness evaluation
     %geometric constrain
    if maxThickness>0.12
        AAnew=AAoriginal;
    elseif maxThickness<0.01
        AAnew=AAoriginal;
    end
    AA=[AA;AAnew];
    p=[p;p1];
end
pop=p;
%constraining the surface
for i=1:length(AA)
if AA(i)>=AAoriginal
        AA(i)=AAoriginal;
end
end
%sorting the individuals by the fittest
fitness=[fitness, 1/(min(AA./AAoriginal))];

fi=AA./AAoriginal;
[fittest,ind]=sort(fi,'ascend');
```

```
fittest=fittest(1:ceil(transprob*popsize));
ind=ind(1:ceil(transprob*popsize));
if k~=genNo
    newpop=pop(ind,:);
    %crossover
    for i=1:ceil(crossprob*popsize)
        indv1=randi([1,popsize],1);
        indv2=randi([1,popsize],1);
        crossindex=randi([1,11],1);
       newpop=[newpop;pop(indv1,1:crossindex) pop(indv2,crossindex+1:end)];
    end
    %mutation
    for i=1:ceil(mutprob*popsize)
        indv=pop(randi([1,popsize],1),:);
        mutindex=randi([1,11],1);
        pmut=randp(p0,range);
        indv(mutindex)=pmut(mutindex);
        newpop=[newpop;indv];
    end
end

end


%choosing the tournemnt winner or the most evolved individual
fittest=pop(ind(1),:);
AAfittest=AA(ind(1));
if AAfittest==AAoriginal
    fittest=p0;
end

x=[1:length(AA)];
%plotting the original airfoil vs. the optimized
plotairfoil(fittest,'k')
hold on
plotairfoil(p0,'r')
legend('Optimized','original')
xlabel('X/C')
ylabel('Y/C')
title('Airfoil shape')

end
```

Code explication

We start off with fixing genetic parameters as seen above. For every generation, the individuals within the population are modified. An initial population is generated by randomly modifying the PARSEC parameters. The area of the shape induced is also measured and so is their fitness. Geometric as well as surface constraints are defined; the maximum thickness vary in a fixed interval otherwise the old individuals are kept. If the newly calculated area exceeds the old one, we keep the latter value. Then individuals of each generation enter a competition to sort the fittest one. To reproduce the next generation, individuals are crossed over, by randomly permuting their genes. Also, some are randomly mutated by varying the values of their PARSEC vector.

3.3 Result

As result of the Genetic Algorithm above, we managed to optimize the airfoil's area. The figure below shows the difference in shape between the initial and optimized airfoils

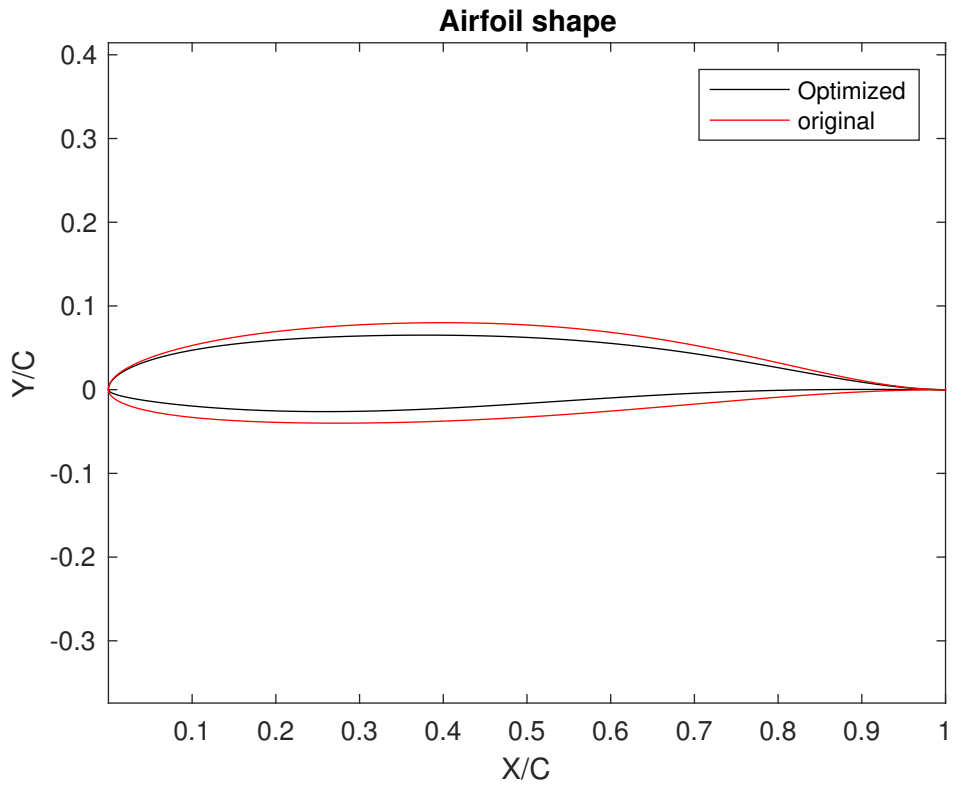

Figure 3.1: Initial and optimized airfoil with GA

The area of the original and optimized airfoils are represented in the following table

Airfoil	Surface
Initial	0.077
Optimized	0.056

Table 3.2: Area values of original and optimized airfoil using GA

The following figure shows the evolution of the fitness values through generations. This is appropriate with what is expected, the fitness increases through generation.
Note. the fitness function we chose is given by

$$fit(i) = \frac{1}{\frac{A(i)}{A_{initial}}} = \frac{A_{initial}}{A(i)}$$

Searching a minimum area $A(i)$ amounts to saying that the fitness function increase.

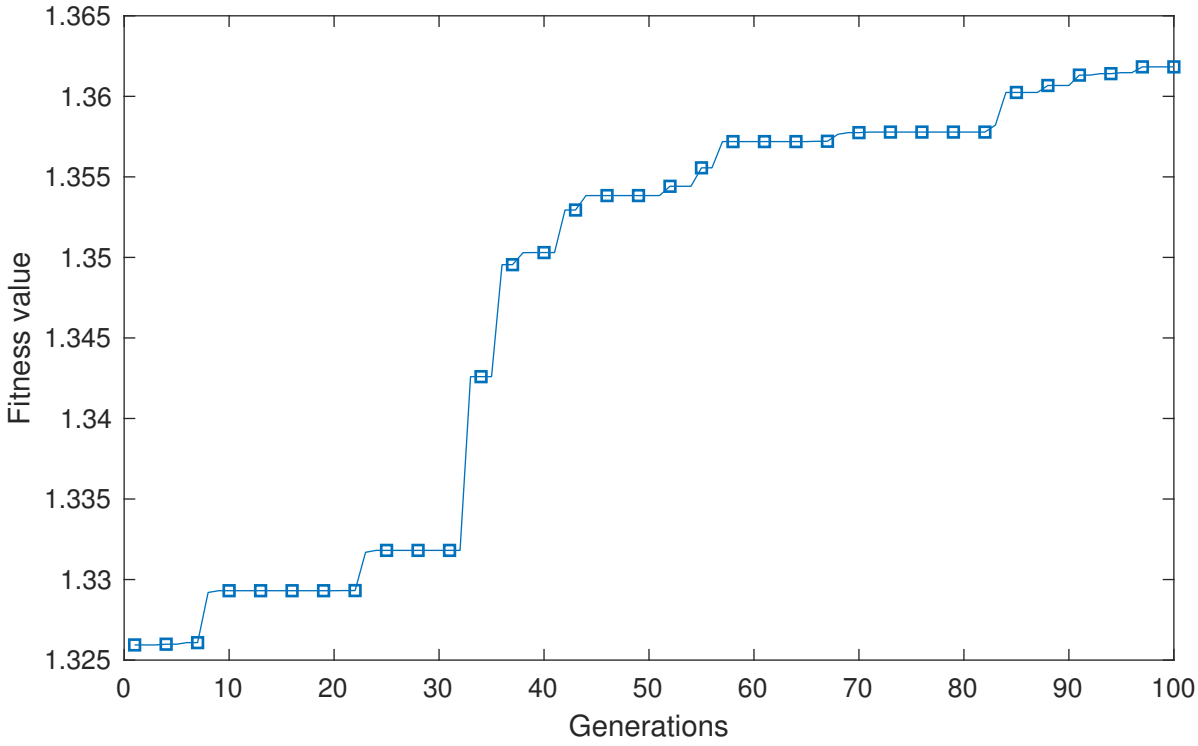

Figure 3.2: Evolution of fitness through generations

Conclusion and perspectives

The result of this study is an optimized airfoil shape based on a NACA 2412 airfoil shape. The optimization process followed in this study started with the initial shape. In this study, we mainly used two optimization methods: nonlinear programming using "fmincon" function in MATLAB and Genetic Algorithm.

To start off, we exploited equations that describe mean camber, the upper and the lower curvature of the NACA shape. We checked the shape obtained with these equations with the one provided by "AirfoilTools" webpage. Thanks to PARSEC parametrization method, decision variables number was reduced to only eleven variable. We designated the area of the airfoil as the objective function to be minimized. Geometrical constraints were also assigned. The first method, using "fmincon" function, consists on the interior-point algorithm that solves even discrete optimization problem, like in our case.
The second method being a genetic algorithm, we reach an optima in just a hundred generation thanks to randomness and the suitability of such a method especially in discrete optimization problems. As a result of each of these two methods, an optimized solution was obtained, inducing an airfoil with a smaller area than the initial one while also preserving its shape. Thus, minimizing the area means less weight of the wing and ultimately reducing the cost.

The remaining question is the feasibility of the production of such a shape. Note that the aerodynamics of the airfoil, i.e. lift coefficient..., weren't taken into account, and so we can't conclude neither on the aerodynamic performance of the optimized airfoil, nor on the utility of such a study; especially as no accompanying experimental study was conducted to validate it.

Bibliography

[1] Thiele. L Blickle. T. A comparison of selection schemes used in genetic algorithms. *CTIK-Report*, 11(1):TIK Institut fur Technische und Kommunikationsnetze, Swiss Federal Institute of Technology, Dec 1995.

[2] Deb.K Goldberg D. E. A comparative analysis of selection schemes used in genetic algorithms. *In Foundations of Genetic Algorithms. Morgan Kaufmann*, pages 69–93, 1991.

[3] Whitley LD. Alba E. Chicano F., Sutton AM. Fitness probability distribution of bit-flip mutation. *Evolutionary Computation*, 23:217–248, 2015.

[4] Andreas Kroll. Chun Liu. Performance impact of mutation operators of a subpopulation-based genetic algorithm for multi-robot task allocation problems. *SpringerPlus*, 5:1361, 2016.

[5] Luciano Gonc¸alves. et Al Rafael Alves. Parsec parametrization methodology for enhancing airfoils geometry using pso algorithm. *Latin-American Congress on Computational Methods in Engineering*, November 6-9, 2016.

[6] N. J. Taylor. et Al D. A. Masters. A geometric comparison of aerofoil shape parameterisation methods. *American Institute of Aeronautics and Astronautics Journal*, Vol. 55:1575–1589, 2017.

[7] Yinyu Ye. et Al David G. Luenberger. Linear and nonlinear programming, fourth edition. *International Series in Operations Research Management Science*, 228:18, 2016.

[8] Volker Schmid Anke Meyer-Baese. Chapter 5 - genetic algorithms. *Pattern Recognition and Signal Analysis in Medical Imaging*, 228:135–149, 2014.

Abstract— The purpose of this study is to optimize the shape of an airfoil using MATLAB. We start by generating and parametrizing the initial profile. Two methods of optimization are used: Nonlinear programming which uses mathematical resolution of optimization problem, and Genetic Algorithms which is an Artificial Intelligence technique that reflects the process of natural selection in order to optimize the well defined cost function.

Keywords : Airfoil optimization, Genetic Algorithms, MATLAB Optimization, Aerodynamic, Shape Optimization

ENSAM
Marjane II, Beni Mhamed
B.P. 4024
50500 Meknès, Maroc