POLITECNICO DI MILANO

Scuola di Ingegneria Industriale e dell'Informazione

Dipartimento di Scienza e Tecnologie Aerospaziali

Corso di Laurea Magistrale in Ingegneria Aeronautica

Multi-fidelity optimization of leading edge surfaces for supersonic drag reduction and high-lift configuration

Advisor: Prof. Maurizio BOFFADOSSI

Master Thesis of:
Tommaso GHIDONI
ID 841839

Academic year 2016 - 2017

Contents

List of Figures

List of Tables

Sommario

In questo lavoro di tesi viene presentato il progetto aerodinamico di superfici di bordo d'attacco atte alla riduzione di resistenza in condizioni di volo supersonico. Lo scopo principale è valutare, in via preliminare, questo tipo di soluzione nella prospettiva di un eventuale utilizzo su un aereo da trasporto supersonico; per far ciò viene considerata la sola sezione *2D* di un profilo adatto al volo transonico; sono omessi aspetti strutturali e cinematici del meccanismo. Il lavoro è diviso in tre fasi; per cominciare, il profilo di partenza viene valutato in modo da ottenere i valori di riferimento per le successive comparazioni. Nella seconda fase si procede ad un'ottimizzazione basata su modello surrogato (*SBO*) per ottenere la configurazione adatta alla crociera supersonica; l'ottimizzazione, che ha come scopo ridurre il coefficiente di resistenza, usa la *Shock Expantion Theory* come modello a bassa fedeltà, un ottimizzatore genetico ed un solutore per le equazioni di *Eulero* come modello ad alta fedeltà. La terza ed ultima fase si propone di testare l'utilizzo della superficie superiore come dispositivo di ipersostentazione. Come per la seconda fase l'ottimizzatore è un *SBO*; l'ottimizzazione surrogata viene svolta da un metodo a gradiente che minimizza la cifra di merito ottenuta dal metodo a pannelli di *Hess-Smith*; i risultati ad alta fedeltà sono ottenuti risolvendo le equazioni *RANS* incomprimibili. Due distinte ottimizzazioni sono eseguite in differenti condizioni di volo; la prima si prefigge di massimizzare il coefficiente di portanza in prossimità dello stallo mentre la seconda lavora sull'indice di salita massimizzandolo ad un'incidenza di 8 gradi. Come prevedibile, la resistenza supersonica della configurazione estesa è stata drasticamente ridotta rispetto al profilo di partenza; il miglioramento rispetto all'ottimo del surrogato è tuttavia modesto. L'ipersostentazione, invece, registra un netto miglioramento sia rispetto al profilo di partenza sia rispetto all'ottimo dato da *Hess-Smith*.

Parole chiave: SST, Ottimizzazione , Manifold Mapping, Multi-fidelity, Dispositivi bordo d'attacco, Profilo supersonico, Ipersostentatore

Abstract

In this work the aerodynamic design of leading edge surfaces to reduce supersonic drag is presented; the aim is to asses the performances of such design in prospect of its implementation for a two-optimum regime supersonic transport. The process is limited to the *2D* wing section and it do not consider structural and cinematic aspects. This works is divided into three main steps. Firstly the baseline airfoil, suitable for transonic flight, is evaluated using the physical models later adopted in the optimizations. In the second step, a multi-fidelity surrogate based optimization (*SBO*) is performed to obtain the supersonic design; the constrained single-objective optimization aims to reduce the drag coefficient in supersonic cruise; it couples a *genetic optimization* and *Shock Expansion Theory* (*SET*) with validation runs that rely on *compressible Euler equation*. The third step tests the possible use as high lift device of the upper surface; two similar optimizations are carried out to obtain its position in different flow conditions. The first one aims to maximize lift coefficient near stall while the second one is focused on the climb index. Again, the optimization uses multi-fidelity *SBO*: *Hess-Smith panel method* (*HSPM*), used as coarse model, is optimized by an *interior point algorithm*. Incompressible *RANS* equations serve as high fidelity Model. The supersonic performance predictably records a great improvement respect the baseline, but only a modest one respect the low-fidelity optimum; the high lift configurations show significant improvement respect both the baseline airfoil and the low-fidelity optimum.

Key words: SST, Surrogate Based Optimization, Manifold Mapping, Multi-fidelity, Leading Edge Devices, Supersonic Airfoil, High lift Devices

List of acronyms

SST	Supersonic transport
SSBD	Shaped Sonic Boom Demonstration
SBO	Surrogate based optimization
SM	Space mapping
MM	Manifold mapping
OMM	Original manifold mapping
CFD	Computational fluid dynamics
RANS	Reynolds average Navier-Stokes
SET	Shock expansion theory
HSPM	Hess-Smith panel method
PDV	Pressure difference value

Chapter 1

Introduction

1.1 Motivation

As *NASA* completed a series of experiments with the *Shaped Sonic Boom Demonstrator* [1], supersonic transport has returned to be of primary interest in the aerospace industry. Despite the tempting capability of half the time needed to fly across the Atlantic ocean, only two *SuperSonic Transport* (*SST*) have been used in regular service; the challenges posed by supersonic cruise for a civilian airplane are, in fact, responsible for the withdraw of numerous projects and studies presented since the 1950s by the most advanced aerospace industries in the world.

A more comprehensive review and future prediction of *SST*s is provided by *Sun et al* [2], in the paragraphs to follow will be reported only some key concepts. Development of *SST*s began almost simultaneously with the operative debut of the first generation of supersonic military aircrafts. The tactical advance given by supersonic capability rapidly pushed the technology to a new evolutionary step in terms of propulsion, materials and aerodynamic. From a commercial point of view, one of the problems to overcome was to make this form of transport profitable. The unusual flight conditions penalize the general efficiency of the plane, especially considering the 1960s technologies; besides the higher initial and maintenance costs, a narrow fuselage, dictated by aerodynamic reasons, implies a reduced capacity and therefore higher cost for the passengers; it is no coincidence that the most enduring *SST*, *Aèrospatiale-BAC Concorde* (1976-2003),had a ticket price that was as much as ten times the price of a conventional long-range subsonic transport aircraft, while its direct competitor, *Tupolev Tu-144* (1977-1983), had a notably shorter operative life due to, among other reasons, economic inef-

Figure 1.1: *Shaped Sonic Boom Demonstration (SSBD)*[1]

ficiency. Nowadays it is technologically possible to design a more efficient transport, but some of the intrinsic limitations are still to be completely overcome. The problem is posed by the environmental impact of supersonic cruise; besides the initial concerns about the effects on the ozone depletion due to the exhaust gas released at unusually high altitude [3], the main limiting factor is noise. Since the operative years of *Concorde*, noise limitations conditioned the overall utilization of the aircraft; the high level of noise generated during take off was an issue to the communities nearby the airports and therefore it relegated the ground operations to limited hours during the day. Most importantly, in order to prevent *sonic-boom* effect over populated areas, the supersonic flight on land was forbidden by *FAA* and *ICAO*. As result, the eligible routes were drastically reduced and the mission profile was forced to incorporate a considerable subsonic flight portion, in which the aircraft underperformed and operated outside the design condition. Under those limitations a *SST* could still be more effective than a conventional transonic aircraft in term of flight time, but the number of possible flights per day would remain similar to the conventional aircraft, while moving just a fraction of the passengers.

In recent years researches have moved toward a way to make supersonic flight possible and profitable under those circumstances; two main approaches can be perused (and eventually combined):

- Reduce the sonic-boom entity in order to comply with the normative. This could be achieved by a combined effort from the legislator, devel-

[1]Photo credit: *Northrup Grumman*: `https://www.nasa.gov/centers/langley/news/releases/2003/03-060.html`

Figure 1.2: *Aèrospatiale-BAC Concorde*[2]

oping new standards specifically for supersonic commercial and business jet, and constructors, focusing the design to shape the wave; this approach has been pursued, among others, by *NASA* in 2003 with the *Shaped Sonic Boom Demonstrator* and in 2017 with *Quiet SuperSonic Transport* (*QueSST*) [4].

- Optimize the aircraft shape such that a two-regime optimal configuration is obtained; recently pursued by *TsAGI* with a scale model presented at the 2017 *MAKS Airshow* [5] This approach would mitigate the handicap of a profile mission with intermittent supersonic legs.

Thanks to exploitation of technologies and numerical simulations, accurate computation fluid dynamic (*CFD*) and state of the art multi-disciplinary optimization technique, unavailable when the design of first *SST*, a new generation of business jet and commercial aircraft is under development and is expected to debut around the second half of the 2020s.

1.2 Aim of the thesis

The first step toward the design of an efficient supersonic wing is the optimization of the *2D* airfoil. In this work is presented a preliminary aerodynamic design of a two-regime airfoil derived by a *NACA 6-series*. In order to design a wing, a multi-disciplinary approach is necessary, but in the process hereby described, only the aerodynamic aspect will be considered; the

[2]Photo credit: *Robert Sullivan*: `https://www.flickr.com/photos/my_public_domain_photos/36193741756`

Figure 1.3: *Tupolev Tu-144*[3]

structural and aeroelastic feasibility is left to be assessed in future development.

The original *NACA* airfoil is considered for the subsonic regime and its performance will be evaluated in chapter 3. The supersonic design is obtained through two leading edge surfaces, implemented to modify the shape of the airfoil and reduce the wave drag in supersonic condition. The surfaces shape and deployed positions are obtained through a constrained optimization process described in chapter 4. Once obtained the new geometry, the aerodynamic coefficients will be evaluated in flow conditions that range from transonic Mach number to the supersonic design condition.

After the definition of the shapes an optimization will be carried to position the upper surface to serve as high lift device ad maximize its utilization during the mission profile. This second optimization frame will be described in detail in chapter 5. Particularly two optimization will be performed in different flight condition; the first will aim to maximize the lift coefficient in a near stall angle of attack allowing to short runway length, the second will focus on the climb index to reduce the time to climb and enlarge the efficiency in this flight phase.

[3]Photo credit: *Milan Nykodym*: `https://www.flickr.com/photos/milannykodym/6080091405`

Chapter 2

Description of the adopted optimization procedures

In every optimization process three major players can be identified: optimizer, model and simulator. The optimizer is the algorithm that governs the logic of the optimization, generate and searches for new candidates and assesses the convergence of the process. The model, mathematical or numerical, is the approximation of the physics of the problem. The simulator,in which is implemented and solved the model, is the component that actually returns the figure of interest; the majority of the optimization running time is occupied by the simulator so it is of great importance that the model is efficiently implemented and that the fidelity of physics is appropriate with the available resources. In the sections to follow will be described each of the component used in the optimizations presented in this work.

2.1 Surrogate Based Optimization (*SBO*)

No algorithm is universal so the choice of the optimization have to be carefully thought; according to the particular problem, the chosen method has to be able to carry on the search, provide the optimum solution in relation to the desired accuracy, be efficient enough to perform under the available resources.
Optimization algorithms can be categorized in several different ways; the main discriminants can be briefly recalled as:

- gradient necessity: methods defined *gradient-based* need derivative information of the fitness function to compute the next iteration of the optimization; these information can be obtained through numerical approximation or analytically. Oppositely, *gradient-free* algorithms rely

on other information, often the fitness value alone is sufficient.

- number of solution tracked: a method can be *trajectory-based* if during the iterations a single point is tracked; otherwise the algorithm is called *population-based*: in this case the iteration takes the name of *generation*.

- randomness level: : if optimization is performed without any random process in it,it is defined *deterministic*; otherwise the method falls under the *stochastic* category. The latter, due to its nondeterministic nature, will return different solutions each time it is run. Different algorithms can be performed in sequence or used within one another in order to overcome the limitations of the single method. The resulting algorithms are defined *hybrid-method*;

- dependency to the previous iteration:if the next iterations of the algorithm is function of the previous ones, the method is defined *history-based*; *memory-free* algorithms need to record only the best individual to carry on the procedure.

- convergence range: *local* algorithms, contrarily to *global* methods, don't have the ability to escape local minimum, so they are not suitable to a global optimization;

- model update: if the procedure optimizes the given fitness function the algorithm it is defined *direct*. According on the complexity of the problem, the properties of the cost function and the available resources a different approach can be more efficiently perused; the optimization is focused on a cheaper representation of the problem: the simpler model is solved and updated iteratively with information gained from the complete model or, in some case, actual experiments. The cheaper model is chosen to be easy to compute and reasonably accurate. Different approaches to choose it, define different method. Every algorithm that relies on this kind of procedure is defined *surrogate based*;

Surrogate Based Optimization (SBO) [6] fits perfectly for problems that normally would need a great number of simulations, but the resources needed for such computations make them not only inconvenient but also unpractical; this is particularly the case of expensive numerical models that, by their nature, could be discontinuous and non-differentiable. The expensive direct problem, find the minimum $\mathbf{x}^*$ of the high fidelity function $f(\mathbf{x})$, is then replaced by a series of inexpensive surrogate model optimization $s^{(i)}(\mathbf{x})$ (eq. 2.1), depending on the particular algorithm a single expensive evaluation is

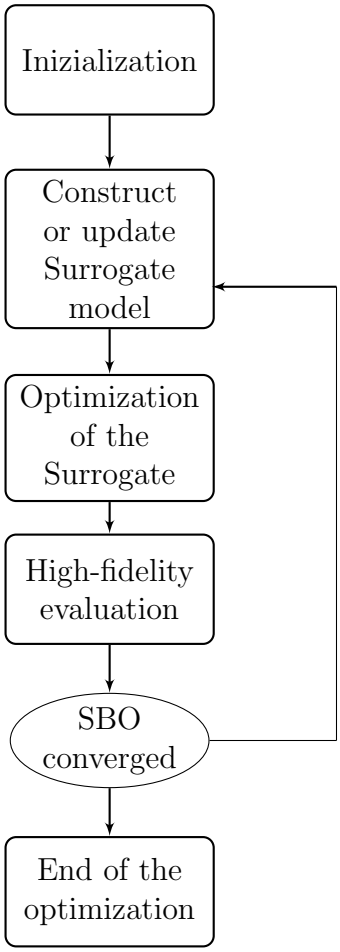


Figure 2.1: General *SBO* workflow [1]

needed for each iteration in order to validate and/or correct the surrogate; this reduce drastically the resources needed to perform the optimization.

$$\begin{gathered} \mathbf{x}^* = arg\,min\, f(\mathbf{x}) \\ \Downarrow \\ until\, convergence\, criteria\, are\, met \\ \mathbf{x^{(i+1)}} = arg\,min\, s^{(i)}(\mathbf{x}) \end{gathered} \tag{2.1}$$

Under the assumption that the surrogate model sufficiently approximates the fine model, the series of design points converges to a solution of the original problem.
Surrogate models can be obtained from different techniques. All of them

[1] Figure from *Koziel et al* [6]

provide inexpensive and/or smooth approximations so rarely the analytic properties of the surrogate are worse than the original problem, therefore the algorithm performed for the surrogate optimization can be normally chosen among a wider range of methods.
Surrogate models can be classified into *physical-based* and *functional*. The latter are simpler and don't need previous knowledge of the physics involved in the problem; they are usually algebraic approximation constructed by sampling the high-fidelity solutions. The benefit given by the virtually negligible optimization cost can be limited by the number of expensive evaluation needed to gain the required accuracy. The specific technique has to be chosen according to how many evaluations are affordable for each iteration; in order to attenuate this drawback, several design of experiments (DOE), have been developed to maximize the amount of information obtained for a given number of experiments; to name but a few, factorial design (full or partial), Latin hypercube,orthogonal array sampling,quasi-Monte Carlo are common strategies to allocate the sample points; not only the technique can be deterministic based or quasi-random, but the DOE can be seen as a minimization problem itself and therefore rely on optimization techniques.
The actual construction of a functional surrogate model can be based on analytic formulas,as for polynomial approximation, or can be the result of an error minimization problem; the latter techniques space from simple least-square fitting for an oversampled polynomial regression to Kriging method and neural networks.
A validation assessment is needed to evaluate the prediction error of the surrogate outside sampled points. This add more expensive computation to the amount needed to obtain and tune the functional surrogate model; the training points are thus only a fraction of the computed samples: according to the specific technique two subset could be used (*simple-sample*) and generate only one model from the training half; otherwise the computed samples could be divided in n subsets and cross-evaluate (*cross-validation*) the n resulting models overt the remain $n-1$ subsets.
Physical-based surrogate, on the other hand, relies on a low-fidelity approximation of the phenomenas involved on the problem; generally they need few fine model evaluations to converge because oppositely from function-based method the problem nature is preserved. The more efficiency in term of evaluation does not always reflect on a better running time because the coarse optimization generally take more time to be performed compared to a *functional based* surrogate.
The model can be simplified by not considering some therms in the governing equation, approximating them with a different analytic formulation or modeling them on empirical formulas. An optimization based on such

surrogate is defined *multi-fidelity*. If the simulation relies on numerical computation implemented over a discretized domain, a different approach can be pursued; since the computational cost usually increases nonlinearly with the discretization refinement, it can be convenient to computed the problem over a coarse discretization. Such surrogate model is called *multi-resolution* and it uses the same simulator, physic model and numerical techniques than the fine model but, depending on the required precision, it is much faster and cheaper to compute. It is worth notice that the improvement is relative only to the fine problem, any other surrogate techniques are always faster than the *multi-resolution*, nevertheless the number of iteration to convergence is usually much lower.
Once obtained the surrogate, a correction method is needed to proceed with the optimization loop; the corrections can be additive, multiplicative or a combination of the two. The level of consistency depends on the particular corrector form.
Several families of surrogate based optimization strategies have been developed and tested; four of the most common are hereby briefly described:

- Approximation Model Management Optimization (AMMO) combines trust-region gradient based optimization and a multiplicative corrector operator.
- Surrogate Management Frameworks (SMF) is based on *pattern-search* that relies on a surrogate model in the *search* step of the algorithm.
- Space Mapping (SM) corrects the surrogate response by optimizing the correction parameters in order to reduce the discrepancy between the fine and coarse model.

In particular, for the *SM* [7] the high-fidelity optimization objective function f is assumed to be the output of a functional U operating on the high fidelity system response $\mathbf{R}_f(\mathbf{x})$ that is approximated by $\mathbf{R}_s(\mathbf{x};\mathbf{p}_{SM})$ somehow related to the coarse model $\mathbf{R}_c(\mathbf{x})$ (eq. 2.2)

$$\mathbf{f}(\mathbf{x}) = U(\mathbf{R}_f(\mathbf{x})) \Rightarrow U(\mathbf{R}_s(\mathbf{x};\mathbf{p}_{SM})) \tag{2.2}$$

$\mathbf{p}_{SM}$ represents the parameter vector of the operator $\mathbf{R}_s$ and depends on which correction is performed; the vector is obtained minimizing the weighted error over the p available sampled points $\mathbf{x}^k$ and their high fidelity computation $\mathbf{R}_f(\mathbf{x}^k)$; The weight vector $\omega^{(k)}$ can used to relax the condition far from the converging region (eq. 2.3)

$$\mathbf{p}_{SM} = arg\,min \sum_{k=1}^{p} \omega^{(k)} ||\mathbf{R}_f(\mathbf{x}^k) - \mathbf{R}_s(\mathbf{x}^k;\mathbf{p}_{SM})|| \tag{2.3}$$

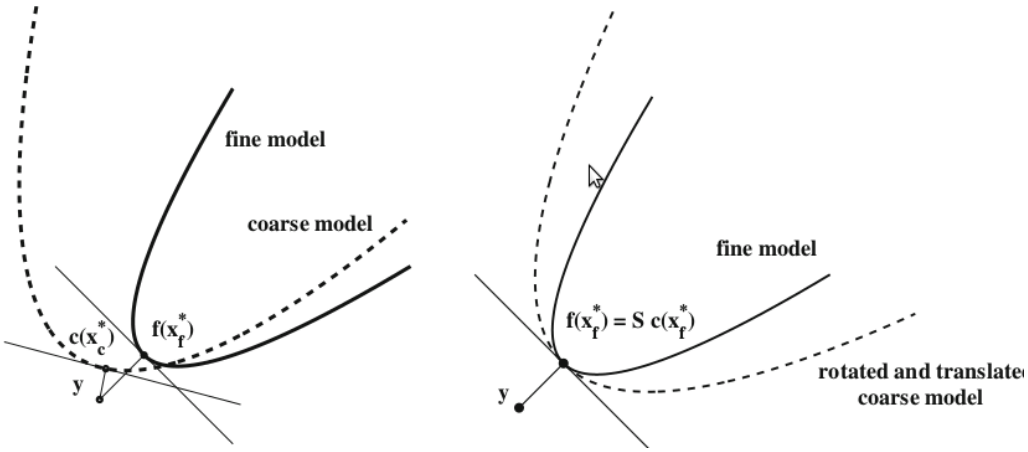


Figure 2.2: *MM* model alignment [2]

Operator $\mathbf{R}_s(\mathbf{x};\mathbf{p}_{SM})$ form depends on the specific type of mapping

- Custom: the structure of given problem is exploited and *ad-hoc* corrections are developed.

- Implicit SM: the coarse model itself is parametrized, the error minimization is performed directly over the operator $\mathbf{R}_s(\mathbf{x};\mathbf{p}_{SM}) = \mathbf{R}_c(\mathbf{x};\mathbf{p}_{SM})$.

- Input SM: the input vector of the low-fidelity model is linearly transformed, the parameter vector contains the needed coefficients, the simpler case consists in $\mathbf{R}_s(\mathbf{x};\mathbf{B},\mathbf{b}) = \mathbf{R}_c(\mathbf{B} * \mathbf{x} + \mathbf{b})$

- Output SM: similar to the input SM but the transformed quantity is the response of the coarse model such as $\mathbf{R}_s(\mathbf{x};\mathbf{B},\mathbf{b}) = \mathbf{B} * \mathbf{R}_c(\mathbf{x}) + \mathbf{b}$

Manifold Mapping is a particular case of output SM that has the advantage of not requiring the parameters extraction. The Original Manifold Mapping (OMM) [8] aims to correct the misalignment between the fine model response function $\mathbf{f}(\mathbf{x})$ and coarse ones $\mathbf{c}(\mathbf{x})$ through operator $\tilde{\mathbf{S}} : \mathbf{c}(Z) \rightarrow \mathbf{f}(X)$ (fig. 2.2), where Z is the space of the coarse model and X is the fine model one; generally if $\mathbf{c}$ is a physic-based surrogate the two spaces coincide otherwise an additional right preconditioning is needed such that $\bar{\mathbf{p}} : X \rightarrow Z$. Both models are assumed C^2 in their respective space and locally similar to each other.

[2] Figure from *Echeverria et al* [8]

In every $i-esim$ iteration of the *SBO*, it is necessary to optimize the $i-esim$ coarse problem.
The coarse problem is formulated as eq. 2.4

$$\begin{gathered}\mathbf{x_i} = arg\,min\,||\tilde{\mathbf{S}}(\mathbf{c}(\mathbf{x})) - y|| := \{\mathbf{x_l} < \mathbf{x} < \mathbf{x_u} \,|\, \tilde{\mathbf{S}_g}(\mathbf{g_c}(\mathbf{x})) < \mathbf{0} \,|\, \tilde{\mathbf{S}_h}(\mathbf{h_c}(\mathbf{x})) = \mathbf{0}\} \\ \tilde{\mathbf{S}}(\bullet) = \mathbf{f}(\mathbf{x_{i-1}}) + \mathbf{S} \cdot (\bullet - \mathbf{c}(\mathbf{x_{i-1}})) \\ \tilde{\mathbf{S}_g}(\bullet) = \mathbf{g_f}(\mathbf{x_{i-1}}) + \mathbf{S_g} \cdot (\bullet - \mathbf{g_c}(\mathbf{x_{i-1}})) \\ \tilde{\mathbf{S}_h}(\bullet) = \mathbf{h_f}(\mathbf{x_{i-1}}) + \mathbf{S_h} \cdot (\bullet - \mathbf{h_c}(\mathbf{x_{i-1}}))\end{gathered} \tag{2.4}$$

$\tilde{\mathbf{S}_g}$ and $\tilde{\mathbf{S}_h}$ are the correction operators needed for the inequality and the equality constrain functions; they are computed from the quantity obtained in the previous steps.
The correction matrices are all computed similarly for the constrains and fitness function and are computed at the end of the $i-esim$ fine evaluation for the $i+1$ coarse optimization (eq. 2.5). $\mathbf{\Sigma_c}^\dagger$ is the result obtained by the inversion of the non-zero entries of the singular values matrix $\mathbf{\Sigma_c}$ (obtained through single value decomposition) of the $\mathbf{\Delta C}$.

$$\begin{gathered}\mathbf{\Delta F} = [\mathbf{f}(\mathbf{x_i}) - \mathbf{f}(\mathbf{x_{i-1}}),...,\mathbf{f}(\mathbf{x_i}) - \mathbf{f}(\mathbf{x_{max(i-n,0)}})]; \\ \mathbf{\Delta C} = [\mathbf{c}(\mathbf{x_i}) - \mathbf{c}(\mathbf{x_{i-1}}),...,\mathbf{c}(\mathbf{x_i}) - \mathbf{c}(\mathbf{x_{max(i-n,0)}})]; \\ \mathbf{\Delta C} \xrightarrow{SVD} \mathbf{V_c \Sigma_c U_c^\intercal} \\ \mathbf{\Sigma_c} \xrightarrow{\text{pseudo-inverse}} \mathbf{\Sigma_c}^\dagger \\ \mathbf{\Delta C}^\dagger = \mathbf{V_c \Sigma_c}^\dagger \mathbf{U_c^\intercal} \\ \mathbf{S} = \mathbf{\Delta F} \cdot \mathbf{\Delta C}^\dagger;\end{gathered} \tag{2.5}$$

Echeverrìa et al [8] define the *OMM* optimization loop as reported in eq. 2.6.

$$
\begin{aligned}
&\mathbf{x_0} = \mathbf{x_c^*} = arg\ min\ ||\mathbf{c}(\mathbf{x}) - \mathbf{y}|| \\
&\tilde{\mathbf{S}_0}(\bullet) \quad = \mathbf{f}(\mathbf{x_0}) + (\bullet - \mathbf{c}(\mathbf{x_0})) \\
&\ do\ k = 0,1,...\ while\ (stop\ criteria) \\
&\qquad \mathbf{x_{k+1}} = arg\ min\ ||\tilde{\mathbf{S}_k}(\mathbf{c}(\mathbf{x})) - \mathbf{y}|| \\
&\qquad break\ if\ converged \\
&\qquad evaluate\ models \\
&\qquad \mathbf{f}(\mathbf{x_{k+1}});\ \mathbf{c}(\mathbf{x_{k+1}}); \\
&\qquad compute\ correction \\
&\qquad \mathbf{\Delta F};\ \mathbf{\Delta C};\ \mathbf{\Delta C}^\dagger \\
&\qquad \mathbf{S}_{k+1} = \mathbf{\Delta F} \cdot \mathbf{\Delta C}^\dagger \\
&\qquad \tilde{\mathbf{S}}(\bullet) = \mathbf{f}(\mathbf{x_{k+1}}) + \mathbf{S}_{k+1} \cdot (\bullet - \mathbf{c}(\mathbf{x_{k+1}}) \\
&\ end
\end{aligned}
\tag{2.6}
$$

The actual algorithm performed in the chapter to follow (eq. 2.7) is implemented in a slightly different way compared to the canonical one. No initialization is needed for the first point,the correction matrices for the first two iteration are imposed to be identities. The initial values for the $\mathbf{f}$ and $\mathbf{c}$ (step -1) do not exist at this point; they are set to be zero and will be overwrite as soon as $\mathbf{x_0}$ will be obtained.

$$
\begin{aligned}
&\mathbf{S}_{-1} = I;\ \mathbf{f}_{-1} = \mathbf{0};\ \mathbf{c}_{-1} = \mathbf{0}; \\
&\ do\ \ i = 0,1...\ while\ (stop\ criteria) \\
&\qquad \mathbf{x_i} = arg\ min\ ||(\mathbf{f}_{i-1} + \mathbf{S}_{i-1} \cdot (\mathbf{c}(\mathbf{x}) - \mathbf{c}_{i-1})) - y|| \\
&\qquad break\ if\ converged \\
&\qquad evaluate\ models \\
&\qquad \mathbf{f}_i = \mathbf{f}(\mathbf{x_i});\ \mathbf{c}_i = \mathbf{c}(\mathbf{x_i}); \\
&\qquad if\ i == 0 \\
&\qquad\quad \mathbf{S}_i = I; \\
&\qquad else\ compute\ correction \\
&\qquad\quad \mathbf{\Delta F},\ \mathbf{\Delta C},\ \mathbf{\Delta C}^\dagger \\
&\qquad\quad \mathbf{S}_i = \mathbf{\Delta F} \cdot \mathbf{\Delta C}^\dagger \\
&\qquad end \\
&\ end
\end{aligned}
\tag{2.7}
$$

From the implementation point of view, this minor change allows to start directly with the execution of the loop and to have a generalized optimization function that depends on the correction matrices.

2.2 Physical Model

In this section will be briefly recalled the physical models used for the two optimizations; only the fine models will be here described. The coarse models will be later discussed in their respective chapters. For a more exhaustive description and the rigorous analytic derivation, reference will be given in the section to follow.

2.2.1 Euler equations

Euler equations govern the physic of compressible inviscid flows, composed by the conservation laws of mass, momentum and energy; the resulting system is hyperbolic. Bi-dimensional Euler equation can be written in conservative form as equation 2.8,

$$\begin{gathered} \frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}_i}{\partial x_i} = 0 \\ \mathbf{U} = \begin{bmatrix} \rho \\ \rho v_i \\ E \end{bmatrix}; \mathbf{F}_i = \begin{bmatrix} \rho v_i \\ \rho v_i v_j + p\delta_{ij} \\ v_i(E+p) \end{bmatrix}; \end{gathered} \tag{2.8}$$

Each variable is a field dependent on time and space; ρ represents the fluid density, v_i represents the velocity component along the $i-esim$ direction, E is the total energy, p is the pressure field and is given by a known function of the other variables. It is assumed chemical and thermodynamic equilibrium in the gas, the specific internal energy is related with pressure and density such that $e = e(p,\rho)$; the particular equation depends on the gas: for a polytropic ideal gas is reported in eq. 2.9

$$\begin{aligned} e &= \frac{p}{(\gamma-1)\rho}; \\ E &= \frac{1}{2}\rho v_i^2 + \rho e; \end{aligned} \tag{2.9}$$

Euler equations are commonly used in transonic and supersonic computation due to the ability to predict wave drag. The lack of viscous effects, and consequently of skin friction, in the considerate physic model is tolerated since wake drag is the predominant contribute to the total drag.

2.2.2 RANS equations

Only *Navier-Stokes* incompressible formulation will be considered. It is a particular case of the compressible equations obtained by fixing the density

$\rho(\mathbf{x},t) = \bar{\rho}$. Under this assumption, the unknown variable are reduced to the fields $\mathbf{u}(\mathbf{x},t)$ and $p(\mathbf{x},t)$. The latter, lost its thermodynamic meaning, can be seen as Lagrange multiplier that ensures the incompressible condition over the field $\mathbf{u}(\mathbf{x},t)$.

The system is composed by two equation: the scalar continuity equation and the vectorial momentum equation; the energy conservation law, present in the compressible formulation as well as the state equation for the thermodynamic variables, are made unnecessary to solve the system.

$$\begin{gathered} \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} + \frac{\nabla p}{\bar{\rho}} = \nu \nabla^2 \mathbf{u} \\ \nabla \cdot \mathbf{u} = 0 \end{gathered} \tag{2.10}$$

Raynolds Averaged Navier-Stokes (RANS) equations govern the mean velocity field; the system is obtained by taking the time average of the Navier-Stokes in which the *Raynolds decomposition* (eq. 2.11) has been applied; each variable is thus split into its main value, depending only on the position, and into its fluctuation, depending on time and position.

$$\begin{gathered} for\ a\ general\ variable \\ \mathbf{a}(\mathbf{x},t) = \bar{\mathbf{a}}(\mathbf{x}) + \mathbf{a}'(\mathbf{x},t) \\ \bar{\mathbf{a}}(\mathbf{x}) = \lim_{T \to \infty} \frac{1}{T} \int_0^T \mathbf{a}(\mathbf{x},t) dt \end{gathered} \tag{2.11}$$

The equations are here reported in eq. 2.12. The full derivation can be found in [9]. For the sake of the notation the constant density will be written as ρ since from now on, the bar values, will represent the mean term.

$$\begin{gathered} \nabla \cdot (\bar{\mathbf{u}}\bar{\mathbf{u}}) + \nabla \cdot (\overline{\mathbf{u}'\mathbf{u}'}) = -\frac{\nabla \bar{p}}{\rho} + \nu \nabla^2 \bar{\mathbf{u}} \\ \nabla \cdot \bar{\mathbf{u}} = 0 \end{gathered} \tag{2.12}$$

The nonlinear convective term generates the *Renolds-Stress tensor* $\overline{\mathbf{u}'\mathbf{u}'}$. it is common to separate it in the isotropic stress tensor and the anisotropic part, in the Einstein notation can be written as eq. 2.13

$$\begin{gathered} \overline{u_i' u_j} = \frac{2}{3} k \delta_{ij} + a_{ij} \\ k = \frac{1}{2} \overline{u_i' u_i} \end{gathered} \tag{2.13}$$

Where k is the kinetic energy associated to the velocity fluctuations. The full tensor is symmetric and thus composed, for a general statistically three-

dimensional formulation, by six independent element; this results in closure problem since the number of unknown variable is greater than the number of available equation. To overcome the problem, a further modeling is needed; two approaches can be pursued:

- *Reynolds-Stress Models* (RSM) are a series of techniques in which a model for the tensor is given directly to complete the closure.
- *Eddy viscosity* approach is based on the *gradient-diffusion* hypothesis. It computes the *Reynolds-Stresses* through the assumption that they are proportional to the *turbulent viscosity* scalar field.

The first approach is relatively more recent and it does not rely on strong assumption as the latter and it is able to represent physic phenomena otherwise neglected. As drawback, it implies the adding of as many (usually differential unless specific algebraic approach are considered) equations as the independent elements of *Reynolds-Stress* tensor. Eddy viscosity approaches are still the most used due the widely development they have been subjected to in the years and due to their cheaper computational cost. Also known as *Bussinnesq methods*, they models in analogy with the molecular diffusion; in particular the two hypothesis assumed are:

- *intrinsic*: a_{ij} depends upon mean velocity gradients only.
- *specific* : a_{ij} specifically have the form of eq. 2.14.

$$a_{ij} = -2\nu_t \left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i} \right) \tag{2.14}$$

The system can be rewritten in the form of eq. 2.15 similar to the laminar formulation; the effective viscosity is now the sum of the kinematic viscosity, property of the fluid, and the turbulent viscosity, a positive quantity property of the flow.

$$\begin{aligned} \nabla \cdot (\bar{\mathbf{u}}\bar{\mathbf{u}}) &= -\frac{\nabla \bar{P}}{\rho} + \nabla \cdot (\nu_e(\mathbf{x}) \nabla \bar{\mathbf{u}}) \\ \nabla \cdot \bar{\mathbf{u}} &= 0 \\ \nu_e(\mathbf{x}) &= \nu_t(\mathbf{x}) + \nu \\ \nu_t &\sim \mathbf{u}^* l^* \end{aligned} \tag{2.15}$$

The asterisk stands for the turbulence scales quantities; now a turbulence model has to be introduced in order define the turbulent viscosity and to proceed to solve the *RANS* system. It is worth to mention that the *RANS*

solution is equal to the time average of the Navier-Stokes equation only if the turbulent stress tensor is exactly modeled; this occurrence is unrealistic so a model induced error will be always to be expected.

Turbulence model: $k-\omega\ SST$

Among the variety of turbulent model available today, the two-equation $k-\omega\ SST$ has been chosen. This model ideally combines the advantage of the $k-\omega$ model, more robust and accurate near wall, and $k-\epsilon$ model, more accurate far from the wall. A cross diffusion term that depends on the blending function F_1 switches the model between the two from which it is derived; the semi-empirical coefficient of the two differential equations are linear combinations of the original ones. In the flow solver is implemented a version derived from [10].

Wall function

To correctly compute skin friction, the dimension of the first cells off the wall, Δs_1, have to be small enough to resolve the viscous sublayer region; this practically implies that in the dimensionless length scale, y_p^+, the height of near-wall region's cells has to be smaller than the unit (eq. 2.16), a dimension comparable to a *DNS* case.

$$y^+ = \frac{yu_\tau}{\mu};\ u_\tau = \sqrt{\frac{\tau_w}{\rho}};\ \tau_w = \frac{1}{2}\rho U^2 C_f(Re); \tag{2.16}$$

Such concentration of fine cells in a relatively small portion of the domain mitigates the (relative) inexpensiveness of *eddy viscosity RANS* formulation. One method to overcome this drawback is to rely on the theory results on wall flows and set the boundary condition further away from it. The near-wall region is thus not computed but modeled after the *law of the wall* through a *wall-function*; this approach has been introduced by *Launder et al* [11]. The first point has now to be placed in the *log region*, approximately at $y^+ \approx 50$; the boundary condition are consequently computed.

2.3 Software used during the project

A description of the software used in this work either for programming the optimization tasks or to solve the physic models introduced in the previous section. When not specified all computations are made using an *Intel Core i7-2600* Processor.

2.3.1 *MATLAB*

MATLAB (*MATrix LABoratory*) [12] is a numerical computing environment based on a proprietary high-level interpreted programming language; it was first developed from the mid-eighties to perform linear algebra (it incorporates *LINPACK* and *EISPACK*). Nowadays it can count on toolboxes for several engineering practices such as control system, curve fitting, optimization, Partial Differential Equation, signal processing, image processing, data acquisition and statistic.
The software is composed by five main parts: language, working environment, graphic handles, function library and application programming interface; together they integrate computation, visualization and programming in an interacting system.

The main algorithms used in this work, as well as several functions and executables, are implemented through *MATLAB*. The algorithm composes the backbone of each *SBO* and performs these main tasks:

- the optimization loop is initialized and carried out in it;
- the dedicate toolbox is used to perform the coarse optimization;
- the interactions with external software (mesh generation, computation, post-process) are managed by it;

The specific optimization strategies will be discussed in their respective chapters.

2.3.2 *Standford University Unstructured* (SU^2)

In this work the software will be used to solve stationary compressible Euler equations and the optimization strategy will rely on a *MATLAB* implemented algorithm, but even if most of the SU^2 potential will be left unexploded a brief description of the software's capability will be hereby presented. Stanford University Unstructured (SU^2) [13] [14] is a software suite developed to solve PDE and PDE-constrained optimization problems. The software is the results of the work conducted at the Stanford University Aerospace Design Lab from a team led by *Francisco Palcios*; released initially in 2012, due to its open-source nature, many other individuals as well as organized teams around the word have contributed to the development of the code; the complete list of the contributors can be found in the official web site.
The software is mainly oriented to fluid dynamic problems: compressible *RANS* solver and adjoint methods are the pivotal core of the software. The

framework is extensible to arbitrary set of governing equations of multi-physics design problems. The suite is composed by high level Python scripts and of C++ executables that can be combined to perform complex tasks including optimization and grid refinement. The C++ modules consist in:

- SU2_CFD solves direct, adjoint and linearized Euler, Navier-Stokes and RANS equations; different methods can be used to perform time and spatial integration. Advanced features are implemented to enlarge convergence and robustness.
- SU2_DEF performs the mesh deformation.
- SU2_DOT computes the partial derivative of a function with respect to the shape design variables; it uses the surface sensitivity at each mesh point provided by adjoin solution of SU2_CFD
- SU2_GEO evaluates constrain during optimization (volume, surface, thickness)
- SU2_MSH performs grid adaptation based on the converge solution
- SU2_SOL generates the solution files

All of them embedded in a Python framework that allows vertical integration of the optimizer structure. Python scripts link the five different levels of the architecture by automatically adapt the input execute lower level components and processing the data.
Most of the bibliographic references that will be cited in the following chapters rely on SU^2 to compute the flow solution.

2.3.3 *OpenFOAM*

Open Source Field Operation and Manipulator (*OpenFOAM*) is an open source $C++$ toolbox for the solution of continuum mechanics problem, particularly computational fluid dynamic (*CFD*). The software has been released in 2004 and developed by the *OpenCFD* until 2011 when the *OpenFOAM* [15] foundation took over the management and distribution to the general public. The applications in the toolbox can be summed into two category:

- *solvers* are designed to compute the solution of particular problems in fluid mechanics
- *utility* performs data manipulation, pre-processing and post-processing

Mesh can be generated using built-in applications or imported from outside the toolbox thanks to the variety of available converter. User defined governing equation can also be implemented with relative ease due to the peculiar syntax used in the libraries. As for the solver application, any other utility can be adapted; this led to an high degree of flexibility. The list of solvers covers a variety of physics models; from basic codes that solve Laplacian equation to multiphase compressible ones; *RANS*, *LES* and *DNS* simulations are implemented in the toolbox along with heat transfer and also financial ones. The flexibility, the extensibility of the applications along with the open-source license make *OpenFOAM* the most widely used *CFD* software.
OpenFOAM will be used to solve *RANS* equations thought *simpleFOAM* a steady-state solver for incompressible, turbulent flow, that relies on *SIMPLE* algorithm [16].

Chapter 3

Original airfoil characterization

The airfoil chosen as baseline is the *NACA 6-series 64212*. T he choice is based on the good performance that this airfoil presents in subsonic flight and it's relatively high critical Mach number,its percentage thickness is comparable with the one of a today's commercial airplane. *NACA 6-series* airfoils were used, with thicknesses usually lower than 10%, on several supersonic fourth generation fighter.

In order to properly asses the variation of the aerodynamic coefficients given by the optimizations, it is first needed a solid baseline solution for each model that will be use.

3.1 Steady compressible Euler solution

In the later stage of the first optimization, the aerodynamic coefficients of the modified geometry will be compared with the ones from the original airfoil. The evaluation will aim to asses the drag reduction over a range of Mach numbers and to find the crossover Mach. According to the optimization, the generated lift for each condition is forced to satisfy the lift equality

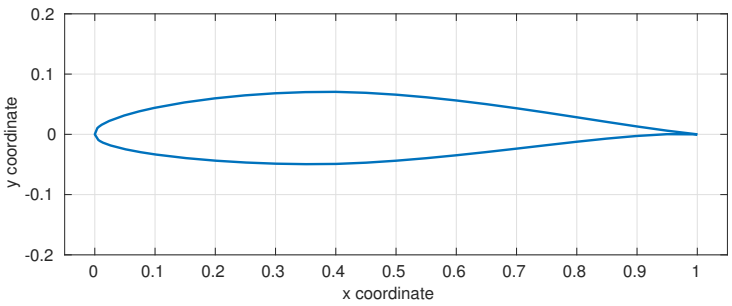


Figure 3.1: *NACA 64212* coordinates

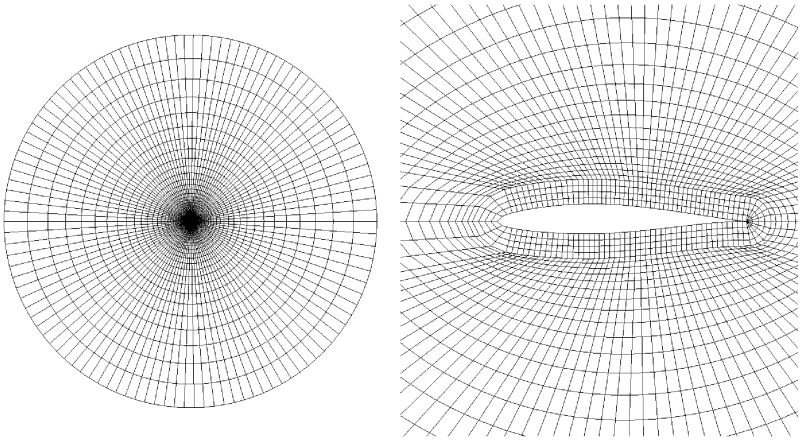


Figure 3.2: SU^2 mesh example of the *NACA 64212*: domain overview (left), zoom near aifoil (right)

constrain using the angle of attack as an independent variable.

3.1.1 Mesh calibration

The overall precision requested over the different Mach evaluation has been set to be 1 drag count; in order to understand how the mesh parameters affect the results, a series of preliminary tests have to be performed.
The flow field is computed in SU^2 over an *O-type* mesh clustered around a *NACA 64212* airfoil (fig 3.2) of unit chord. The mesh, generated by a *MATLAB* function, is composed by quadrilaterals and it is parametrized over the far-field distance and the number of elements over the airfoil.

In order to save time, in the calibration runs, the lift constrain is not enforced and the angle of attack is therefore set to be zero. Focusing on the mesh radius, an initial study has been realized. With a reasonable number of elements on the airfoil, far-field distance has been progressively incremented by ten chords until the difference of computed drag coefficient fulfilled the request precision; the results showed that, especially in the transonic regime, aerodynamic coefficients are greatly affected (fig. 3.3). The most critical Mach number appears to be 0.85, so in order to obtain the proper number of divisions, the following meshes comparison is carried out only at this velocity. The drag coefficient converges within the set tolerance when 1600 divisions have been implemented (fig. 3.4). Figures 3.5 and 3.6 show how a proper number of elements on the airfoil is needed to correctly identify the shock position, and consequently to compute aerodynamic coefficient.
The main parameters of the SU^2 case are reported in table 3.1.

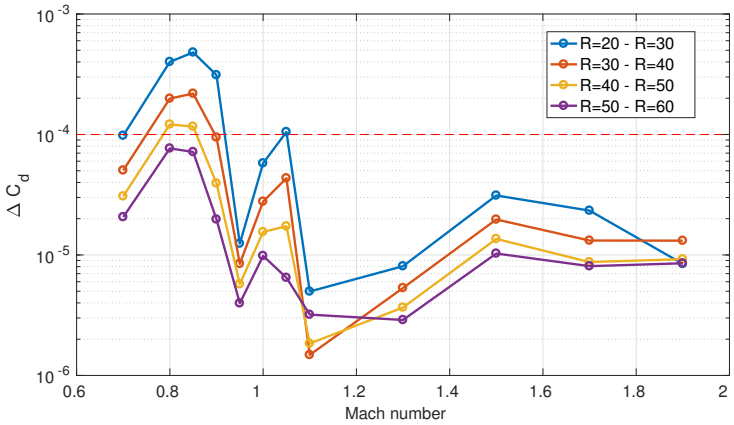


Figure 3.3: SU^2 mesh calibration: ΔC_d with different mesh radius

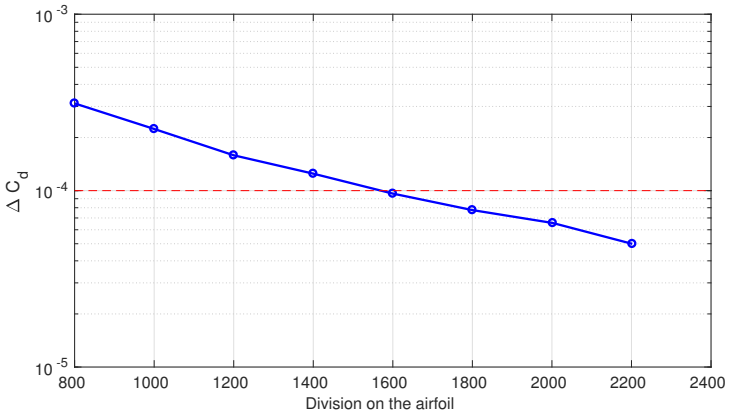


Figure 3.4: SU^2 mesh calibration: ΔC_d with different number of elements

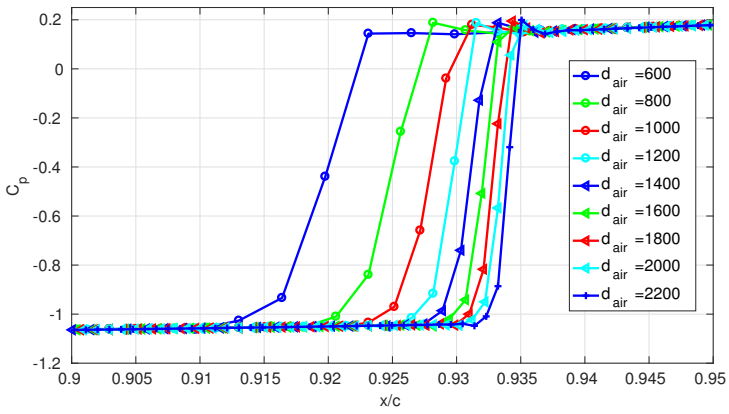


Figure 3.5: SU^2 mesh calibration: shock position, suction side

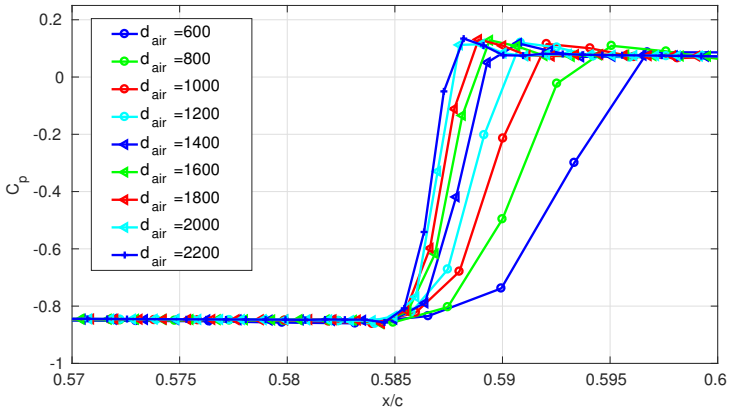


Figure 3.6: SU^2 mesh calibration: shock position, pressure side

Mesh radius	50 chord
Element on the airfoil	1600
Expected precision	1 dc
MGLEVEL	3
CONV_NUM_METHOD_FLOW	JST
CONV_CRITERIA	RESIDUAL
STARTCONV_ITER	10
RESIDUAL_REDUCTION	6
RESIDUAL_MINVAL	-8
ITER_MAX	5000

Table 3.1: SU^2 case parameters

3.1.2 Aerodynamic coefficient

The computed aerodynamic coefficients are showed in figures 3.7 3.8 3.9 and in table 3.2; both lift constrained coefficients and zero incidence ones have been calculated.

M	C_l	C_d	$C_d\|_L$	C_m	$C_m\|_L$
0.80	0.418	0.030	0.0300	0.1393	0.1394
0.85	0.463	0.070	0.0648	0.2408	0.2164
0.90	0.034	0.114	0.1262	0.0623	0.1244
0.95	0.011	0.108	0.1223	0.0465	0.1023
1.00	0.010	0.104	0.1158	0.0442	0.0942
1.05	0.010	0.100	0.1102	0.0425	0.0876
1.10	0.008	0.097	0.1061	0.0404	0.0812
1.30	-0.004	0.092	0.0975	0.0307	0.0605
1.50	-0.010	0.087	0.0908	0.0233	0.0475
1.70	-0.013	0.082	0.0849	0.0185	0.0382
1.90	-0.014	0.078	0.0800	0.0153	0.0314
2.00	-	-	0.0780	-	0.0287

Table 3.2: *NACA 64212*: aerodynamic coefficient obtained with SU^2; L subscript indicates the lift constrained values

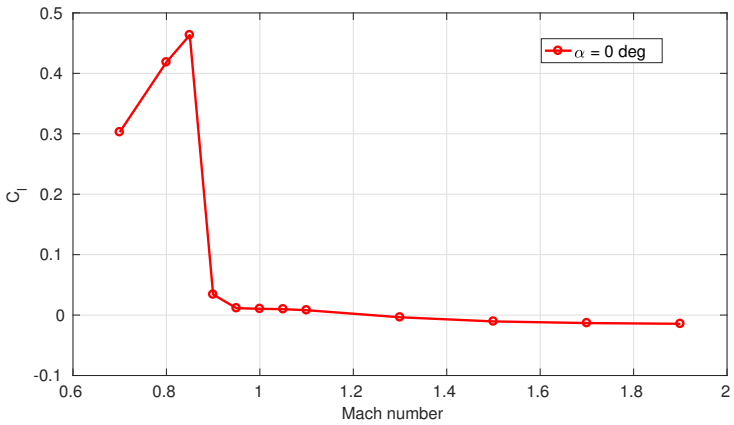


Figure 3.7: *NACA 64212*: C_l - Mach

3.2 RANS solution

In this paragraph the $C_l - \alpha$ curve of the slat-less airfoil will be computed; in particular the aim is to obtain the angle of stall α_{stall} of the numerical

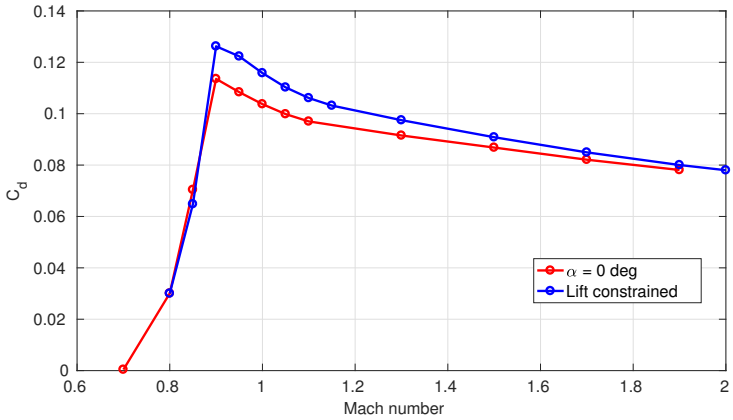


Figure 3.8: *NACA 64212*: C_d - Mach

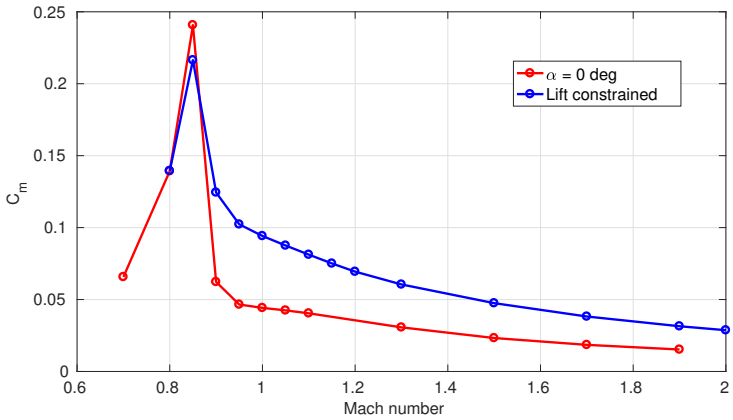


Figure 3.9: *NACA 64212*: C_m - Mach

model. The computations are carried out using *simpleFOAM*, the mesh is generated by *Gmsh* [17] and its parameters are the same used for the fine evaluations in the second optimization; the grid calibration process and the solver settings will be more extensively described in the dedicated section (5.4); a coarser discretization is showed as example in figure 3.10.

3.2.1 Flow solver validation

The validation is carried out using *NACA 0012* airfoil of which, unlikely *NACA 64212*, experimental and computational data are largely available. The report presented by *Ladson* [18] are used as reference; the computation is performed using an incompressible solver so the lowest available Mach number data are considered. The datasets were acquired in different flow conditions covering Mach from 0.15 to 0.36 and Reynolds from $2 \cdot 10^6$ to

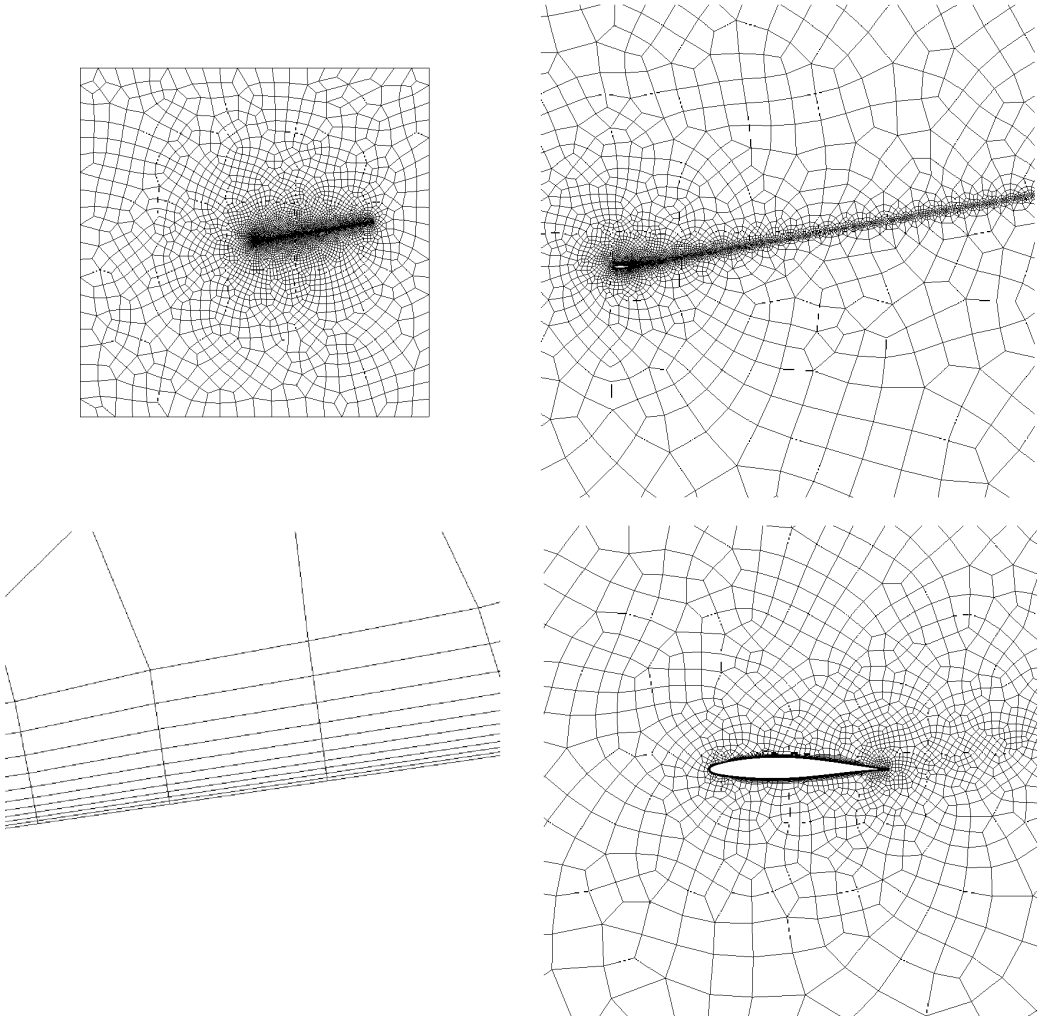


Figure 3.10: *OpenFOAM* mesh example of the *NACA 0012*: domain overview (top left), zoom over the wake refinement (top right), zoom near airfoil (bottom right), close-up in the boundary layer refinement (bottom-left)

Far-field distance	300c
Max element size on the airfoil	0.003c
Solver	*simpleFOAM*
Converge criteria	*residualControl*
p	$1e-7$
U	$1e-9$
endTime	12000

Table 3.3: *simpleFOAM* case parameters

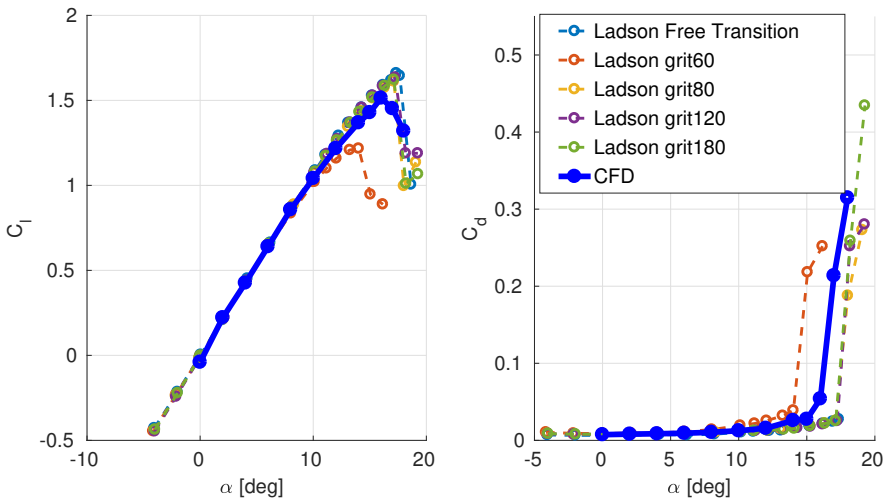


Figure 3.11: *NACA 0012*: *CFD* results compared with *Ladson et al*

$12 \cdot 10^6$. The boundary layer transition was tested both forced and free; in figure 3.11 are reported all available curves at $Re = 6 \cdot 10^6$ and $Mach = 0.15$. Free transition data is reported, but it will not be considered due to its drastically different stall angle.

Depending on the grit size, the experimental curves suffer stall between $\alpha|_{stallexp} = 17.13 - 17.35\ deg$. The computed results show a stall angle between $\alpha|_{stallCFD} = 16 - 17\ deg$ with unstable flow solution already at 17 degree. The aerodynamic coefficient trends are well approximated in the linear portion of the curve, lift coefficient is fairly estimated up to the computed stall. The error regarding drag coefficient in contained under ~ 0.002 for $\alpha < 13$; in the stall proximity drag prediction become unreliable.
In order to get a more realistic stall assessment, a different computational setup may be needed, but, for the purpose of comparison, the current results show sufficient agreement to proceed with the tested mesh.

3.2.2 Aerodynamic coefficients

The $C_l - \alpha$ here computed will become the baseline curve in the later comparison. The coefficients are reported in figure 3.12 and tab. 3.4.

α	C_l	C_d
0.0	0.137	0.008
2.0	0.396	0.009
4.0	0.626	0.009
8.0	1.065	0.011
10.0	1.261	0.014
12.0	1.417	0.018
13.0	1.470	0.024
14.0	1.531	0.027
14.5	1.583	0.049
15.0	1.509	0.071
15.5	1.504	0.075
16.0	1.444	0.087
17.0	1.392	0.393

Table 3.4: *NACA 64212*: aerodynamic coefficients (*RANS*)

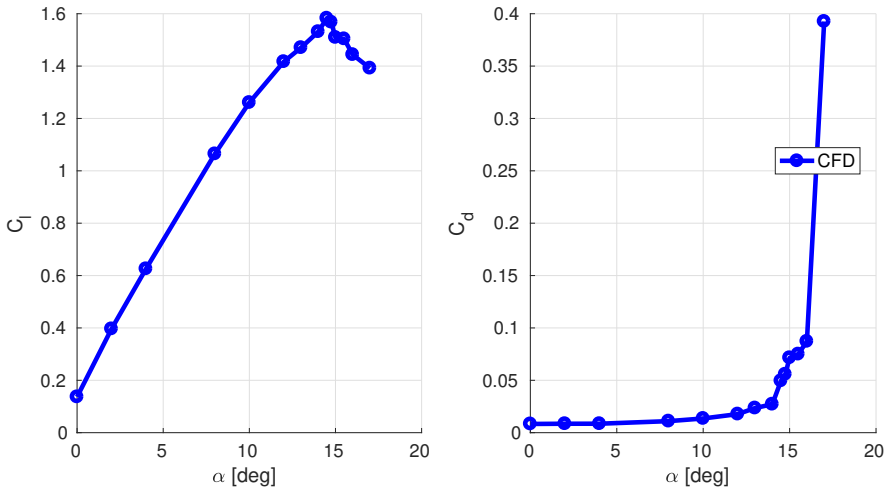


Figure 3.12: *NACA 64212*: *CFD* results

Chapter 4

Supersonic cruise optimization

The first phase of this work regards the supersonic configuration of the airfoil in which the leading edge surfaces are extended. In supersonic flight the airfoil drag is affected by the leading edge sharpness; a sharp edge prevents the formation of a detached shock wave that would cause a drastic increase of the wave drag. Supersonic shape optimization has been studied since 1950s when simplified analytical models were used to design wing sections; nowadays, thanks to the massive computational resources available, the approach has evolved to directly optimize the entire airplane shape thought the use numerical simulations. The use of *MM* for airfoil shape optimization has been recently tested both in transonic (*Ren et al* [19]) and supersonic regime (*Siegel et al* [20]) with satisfactory results, it will be applied also in the current optimization.

4.1 Problem formulation

The aim is minimize the drag of the airfoil in supersonic cruise ($M = 2$). The problem is constrained both linearly and nonlinearly in order to guarantee that:

- the resulting design will be geometrically feasible
- the maximum thickness of the slats will be greater than $\bar{th}_{lim} = 2\%$ of their chords.
- the lift generated by the final configuration will be equal to the one generated by the original airfoil in the reference condition $\alpha = 0$ deg at $M = 0.8$

The optimization is carried out using multi fidelity *OMM* algorithm that rely on *shock expansion theory* (*SET*), as low fidelity model, and Euler equation, as high fidelity one; such combination has been used by *Siegel*, differently from its problem, this case add more demanding geometric constrains. The lift constrain will be enforced thought the angle of attack; surrogate based optimization is used to minimize the number of fine evaluation that the equality constrain would have needed: instead of solve the constrain on the fine model, it will be user to only verify the corrected coarse one.
The optimization ceases if:

- The change in the normalized component input vector is less than 0.001
- No improvement has been registered in the last 5 iteration
- The maximum number of iterations is reached; the number is set to 15

4.2 Geometry description

The design is obtained starting from the unitary chord original airfoil using eight geometric parameters to define the dimensions,the shape and the extended configuration of the slats. No symmetry is imposed to the two surfaces, so the dimensions can vary from one slat to the other. Four out of eight parameters define the dimensions of the two slats; two out of eight are needed to set the inner shape and the remain two establish their position when extended.
The dimensions of the slats are defined by these four parameters (fig. 4.1):

- INC_{UP} represents the distance along x axis between the leading edge of the airfoil and the leading edge of the retrieved upper slat (*point A*)
- INC_{DWN} represents the distance along x axis between the leading edge of the airfoil and the leading edge of the retried lower slat(*point B*)
- C_{UP} represents the distance between (*point A*) and (*point B*) along x axis
- C_{DWN} represents the distance between (*point C*) and (*point D*) along x axis

The outer curves of the slats must coincide with the shape of the clean airfoil; the inner curve of each slat is a *parabola* computed over the slat reference system which is aligned so the slat's chord coincides with the x axis and the y axis feces outward. The curve is defined by three points: the slat's leading edge, the half chord point and the slat's trailing edge (fig. 4.2).

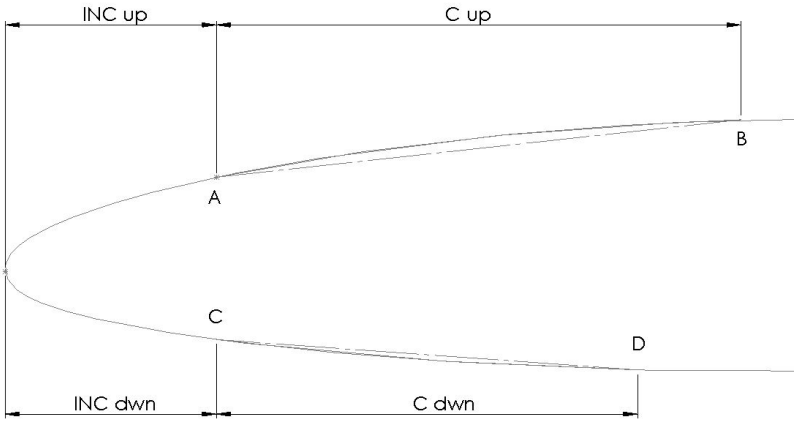


Figure 4.1: Parameters that define the dimension of the slats

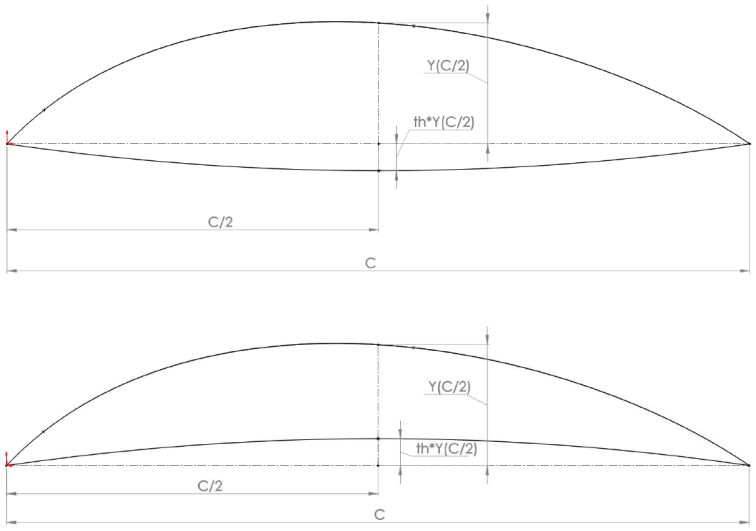


Figure 4.2: Inner shape geometry examples; in case of $th < 0$ (top), in case of $th > 0$ (bottom); both geometries are represented in the slat reference system

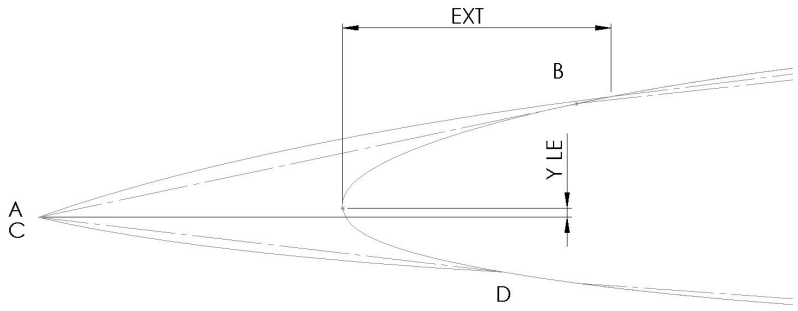


Figure 4.3: Parameters that define the extended configuration

- th_{up} is the height of the mid point expressed in percentage respect the outer curve.
- th_{dwn} is the height of the mid point expressed in percentage respect the outer curve.

The extended configuration is defined by the last two parameters (fig. 4.3):

- EXT defines the extension of the slats; it is the distance along the x axis between the clean airfoil leading edge and the trailing edge of the extended upper slat.
- Y_{LE} is the distance along the y axis of the modified leading edge.

This way defined the geometry unambiguously. The eight parameters are now included into the input array such that:

$$\mathbf{x} = [INC_{UP};\ C_{UP};\ INC_{DWN};\ C_{DWN};\ EXT;\ Y_{LE};\ th_{up};\ th_{dwn}] \quad (4.1)$$

The geometry is computed in a *MATLAB* function. The steps of the process are:

1. interpolate the original airfoil
2. select the portions of the airfoil that will become the outer curves of the slats

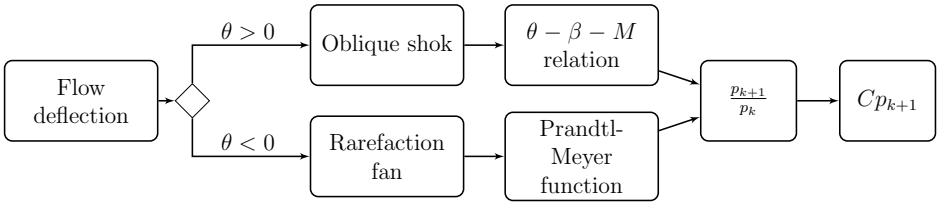


Figure 4.4: *SET* workflow

3. compute the inner shapes of the slats and the slat thicknesses
4. compute the new outer shape of the airfoil
5. position the upper slat as EXT imposes
6. rotate the upper slat until Y_{LE} is verified
7. generate new leading edge moving the lower slat
8. rotate the lower slat to close the new airfoil

At the end of each step, the feasibility of the design is checked; few problems can emerge from the process: for example the slats could penetrate into the airfoil or the slat don't close the geometry .
If any of the checks fails, the geometry is declared *'not feasible'* and no other computation will be done on it.

4.2.1 Test configuration

In order to set up the low fidelity model and the hight fidelity model, tests were conducted on a feasible design defined by the following input vector:

$$\mathbf{x_{test}} = [0.05c,\ 0.25c,\ 0.059c,\ 0.24c,\ 0.051,\ 0.04c,\ 3,\ 3]; \tag{4.2}$$

4.3 Low fidelity model

The shock expansion theory (SET) allows the computation of the pressure distribution over a panel-discretized airfoil in a supersonic flow. The method considers both sides of the airfoil as a succession of compression or expansion corners and computes the pressure of the segments solving the $\theta - \beta - M$ relation or *Prandtl-Mayer* function.

The procedure first needs to split the airfoil into upper side and lower side; then it proceeds to break down the two curves into *n/2* panels each. Once discretized each curve in *n/2 + 1* points, it is possible to compute the pressure distribution along the flow direction. First of all the lengths Δs_k, the normal directions $\mathbf{n_k}$ and the inclinations θ_k of the k segments are computed as eq 4.3.

$$\begin{aligned} \Delta s_k &= \sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2}; \\ \mathbf{n_k} &= \begin{bmatrix} -\sin(\theta_k) \\ \cos(\theta_k) \end{bmatrix}; \\ \theta_k &= atan\left(\frac{y_{k+1} - y_k}{x_{k+1} - x_k}\right); \end{aligned} \tag{4.3}$$

The deflections that the flow have to encounter along the curve are incorporated into the array $\mathbf{\Delta}\theta$ (eq. 4.4)

$$\mathbf{\Delta}\theta = [\theta(1) - \alpha, diff(\theta)]; \tag{4.4}$$

It is now possible to solve k consecutive problems in which the upstream Mach number is the Mach of the $k-1$ panel or, in the case $k = 1$, the far-field Mach number.
Depending of whether the deflection is positive or negative, it is formed an *oblique shock* or an *expansion fan*.
For both cases it is possible to compute the pressure ratio p_{k+1}/p_k that can be re-conduct to the far-field pressure p_0 (eq. 4.5)

$$\frac{p_k}{p_0} = \prod_{j=0}^{k} \frac{p_{k-j}}{p_{k-1-j}} \tag{4.5}$$

It is now possible to calculate the force coefficients acting on the upper side panels as eq. 4.6;

$$\begin{aligned} C_{f-up}(k) &= \frac{2}{\gamma M_\infty^2} \frac{p_k}{p_0} \frac{\Delta s_k}{c}; \\ C_{m-up}(k) &= \frac{1}{c} \begin{bmatrix} x_k + 0.5(x_{k+1} - x_k) - x_{ref} \\ y_k + 0.5(y_{k+1} - y_k) - y_{ref} \end{bmatrix} \times C_{f-up}(k)\mathbf{n_k} \end{aligned}; \tag{4.6}$$

From which it is possible to compute the upper side aerodynamic coefficients (eq 4.7)

$$
\begin{aligned}
C_{x-up} &= \sum_{j=1}^{n/2} C_{f-up}(k) sin(\theta_k) \\
C_{y-up} &= \sum_{j=1}^{n/2} C_{f-up}(k) cos(\theta_k) \\
C_{m-up} &= \sum_{j=1}^{n/2} C_{m-up}(k)
\end{aligned}
\tag{4.7}
$$

A similar process can be applied for the lower side of the airfoil; afterward it is possible to calculate the aerodynamic coefficients of the entire airfoil in the airfoil reference system (eq. 4.8). The *y-coefficient* is obtained thought subtraction due to the algorithm implementation; specifically, the lower curve is mirrored before going thought the same computation of the upper one.

$$
\begin{aligned}
C_x &= C_{x-up} + C_{x-dwn} \\
C_y &= C_{y-dwn} - C_{y-dwn} \\
C_m &= C_{m-up} + C_{m-dwn}
\end{aligned}
\tag{4.8}
$$

The obtained coefficients are then reported in the farfield reference system (eq 4.9)

$$
\begin{aligned}
C_l &= C_y cos(\alpha) - C_x sin(\alpha) \\
C_d &= C_y sin(\alpha) + C_x cos(\alpha)
\end{aligned}
\tag{4.9}
$$

4.3.1 Oblique shock

Under the assumption of calorific perfect gas the oblique shock is characterized by the relation reported in eq. 4.10;

$$
\begin{aligned}
M_{n1} &= M_1 \sin(\beta) \\
M_{n2}^2 &= \frac{M_{n1}^2 + [2/(\gamma-1)]}{[2/(\gamma-1)]M_{n1}^2 - 1} \\
M_2 &= \frac{M_{n2}}{sin(\beta-\theta)} \\
\frac{p_2}{p_1} &= 1 + \frac{2\gamma}{\gamma+1}(M_{n1}^2 - 1)
\end{aligned}
\tag{4.10}
$$

Where β is the shock angle, θ is the deflection of the flow, the subscript n indicates the quantity in the normal direction respect to the shock, the subscript *1* indicates the quantities before the shock and the subscript *2*

indicates the quantities past the shock. In order to compute the pressure ratio, the shock angle has to be obtained by the $\theta - \beta - M$ relation (eq. 4.11);

$$\tan(\theta) - 2\cot(\beta)\left[\frac{M_1^2 sin^2(\beta) - 1}{M_1^2(\gamma + cos(2\beta) + 2}\right] = 0 \tag{4.11}$$

For every value of M_1 there is maximum possible deflection; if $\theta > \theta_{max}$ the shock is detached and in this occurrence the calculation is stopped and the geometry is declared infeasible. It is always considered only the *weak shock solution*.

4.3.2 Expansion fan

The expansion fan is a continuous expansion region where the Mach number increases from the upstream value M_1 to the downstream value M_2. Considering infinite small expansions, the deflection can be written as eq. 4.12;

$$\int_{\theta_1}^{\theta_2} d\theta = \int_{M1}^{M_2} \sqrt{M^2 - 1}\frac{dV}{V} \tag{4.12}$$

Assuming polytropic ideal gas, dV/V can be substituted in order to obtain the eq. 4.13;

$$\int_{\theta_1}^{\theta_2} d\theta = \Delta\theta = \int_{M1}^{M_2} \frac{\sqrt{M^2 - 1}}{1 + \frac{\gamma-1}{2}M^2}\frac{dM}{M} \tag{4.13}$$

The integral on the right end side is called *Prantl-Meyer function* defined by the symbol ν, that can be rewritten as eq. 4.14;

$$\begin{aligned}\nu(M) &= \int \frac{\sqrt{M^2 - 1}}{1 + \frac{\gamma-1}{2}M^2}\frac{dM}{M} \\ &= \sqrt{\frac{\gamma+1}{\gamma-1}}\tan\left(\sqrt{\frac{\gamma+1}{\gamma-1}(M^2-1)} - \arctan(\sqrt{M^2-1})\right)\end{aligned} \tag{4.14}$$

Combining eq. 4.13 and eq.4.14, eq. 4.15 is obtained;

$$\Delta\theta = \nu(M_2) - \nu(M_1) \tag{4.15}$$

M_2 is computed numerically such that $\nu(M_2) - \Delta\theta - \nu(M_1) = 0$; once M_2 is known the isoentropic relation (eq. 4.16) can be used to obtain the pressure ratio between the upstream and downstream.

Panels	275
Expected precision	0.1 *dc*
Expected evaluation time	~ 0.3 s

Table 4.1: Coarse evaluation setting

$$\frac{p_1}{p_2} = \left[\frac{1+\frac{\gamma-1}{2}M_2^2}{1+\frac{\gamma-1}{2}M_1^2}\right]^{\frac{\gamma}{\gamma-1}} \tag{4.16}$$

As for the oblique shock, if the turning angle can't be archived the design is declared unfeasible .

4.3.3 Convergence study

A convergence test has been made to asses the number of panels request to obtain an appropriate precision. The convergence history and the computational time are here reported in figures 4.2 4.5 4.6.
It appears that for archive 1 drag count precision 150 panels are needed, to archive 0.1 drag count precision 275 panels need to be used.

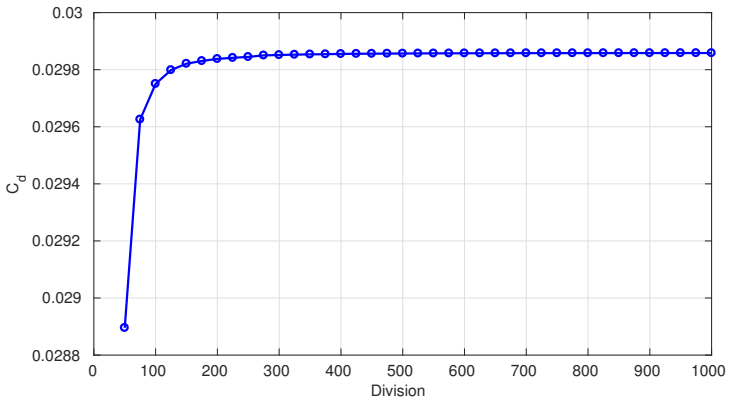


Figure 4.5: *SET* convergence study

4.4 High fidelity model

The solution is obtained solving bi-dimensional stationary Euler equation; as for the characterization of the original airfoil the case is developed from the *NACA 0012* test case available in SU^2 .
An *O-type* (fig. 4.7) mesh has been generated by a *MATLAB* function to

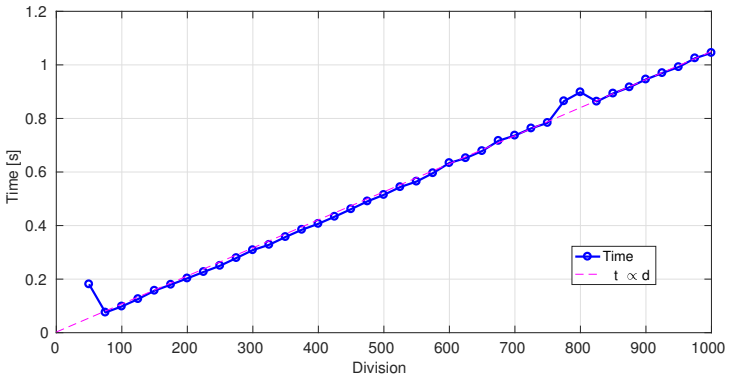


Figure 4.6: *SET* computation time

d	Cd	Time [s]
50	0.0288955	0.180
75	0.0296256	0.075
100	0.0297504	0.098
125	0.0297988	0.125
150	0.0298211	0.157
175	0.0298306	0.179
200	0.0298379	0.202
225	0.0298417	0.226
250	0.0298449	0.250
275	0.0298501	0.278
300	0.0298516	0.308
325	0.0298527	0.328
350	0.0298537	0.357
375	0.0298543	0.384
400	0.0298553	0.406

Table 4.2: *SET* convergence study recap

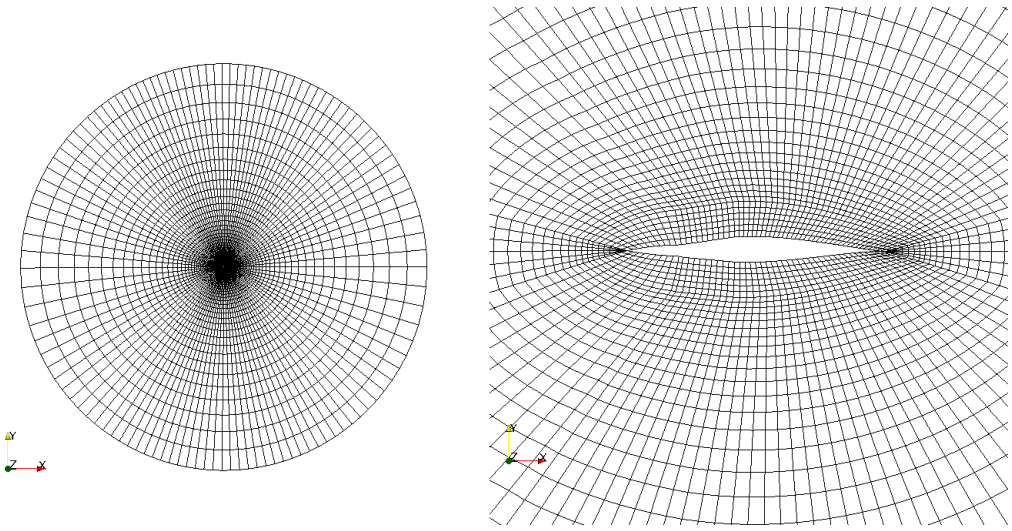


Figure 4.7: SU^2 mesh example: domain overview (left), zoom near airfoil (right)

Mesh size	20c
Elements on the airfoil	2400
Expected precision	0.1 *dc*
Expected evaluation time	$\sim$ 40 min

Table 4.3: Fine evaluation setting

be composed by quadrilaterals. The mesh is defined by two parameters: the mesh radius and the division on the airfoil; the latter parameter is proportionally related to the division on the radial direction. The cells are set to be squares near the airfoil and then stretched approaching the far-field boundary. Differently from what was done for the characterization of *NACA 64212*, the study is focused on Mach 2 only.

Convergence study

A convergence study has been done in order to correctly set the mesh parameters (fig 4.9 4.8 4.10).
For the test airfoil (section 4.2.1) at Mach 2 and zero incidence, 1500 divisions over the airfoil are needed to archive 1 drag count precision; to improve the precision to 0.1 *dc*, 2400 divisions are needed. The mesh radius almost does not influence the results (tab. 4.5).
The mesh setting used are used in the optimization are reported in table 4.3; the other parameters are the same reported in tab 3.1.

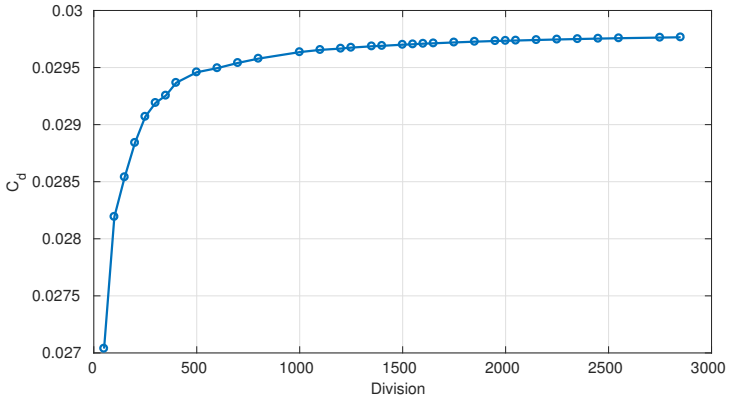


Figure 4.8: SU^2 convergence study

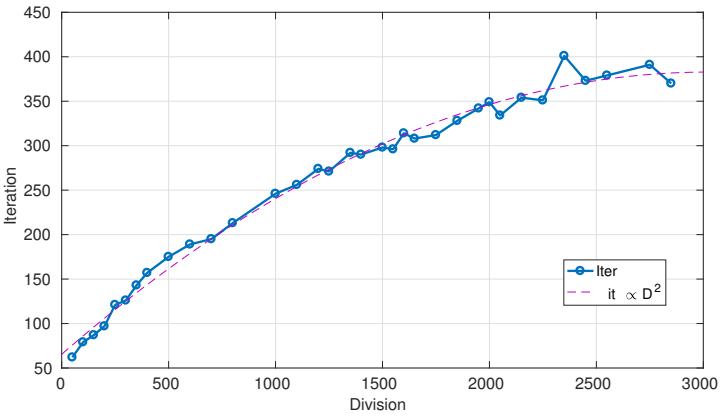


Figure 4.9: SU^2 iteration to reach convergence

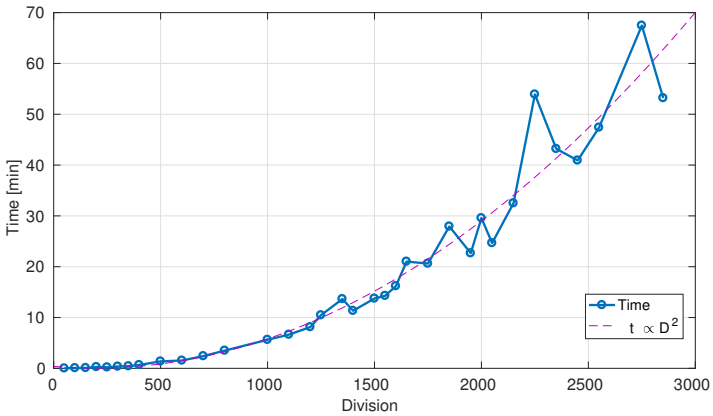


Figure 4.10: SU^2 computation time

d	iteration	C_d	time [min]	$res(\rho)$	$res(\rho u)$	$res(\rho v)$	$res(\rho e)$
50	62	0.027038	0.01	-8.10	-7.62	-8.16	-7.33
100	79	0.028193	0.06	-8.01	-7.56	-8.13	-7.26
150	87	0.028540	0.08	-8.04	-7.63	-8.14	-7.32
200	97	0.028841	0.27	-8.03	-7.73	-8.06	-7.32
250	121	0.029069	0.21	-8.01	-7.76	-8.00	-7.29
300	126	0.029191	0.36	-8.02	-7.75	-8.02	-7.29
350	143	0.029255	0.43	-8.02	-7.77	-8.03	-7.30
400	157	0.029366	0.66	-8.00	-7.75	-8.00	-7.28
500	175	0.029459	1.37	-8.00	-7.74	-8.00	-7.27
600	189	0.029495	1.54	-8.03	-7.76	-8.03	-7.30
700	195	0.029541	2.41	-8.01	-7.74	-8.04	-7.28
800	213	0.029578	3.49	-8.03	-7.76	-8.03	-7.30
1000	246	0.029635	5.62	-8.02	-7.74	-8.03	-7.28
1100	256	0.029655	6.64	-8.00	-7.74	-8.00	-7.27
1200	274	0.029666	8.12	-8.01	-7.75	-8.01	-7.28
1250	271	0.029675	10.46	-8.01	-7.74	-8.01	-7.28
1350	292	0.029686	13.65	-8.01	-7.76	-8.00	-7.28
1400	290	0.029690	11.33	-8.01	-7.75	-8.00	-7.28
1500	298	0.029700	13.74	-8.01	-7.76	-7.99	-7.28
1550	296	0.029704	14.28	-8.01	-7.75	-8.00	-7.27
1600	314	0.029709	16.18	-8.00	-7.75	-8.00	-7.27
1650	308	0.029712	21.01	-8.01	-7.76	-8.00	-7.28
1750	312	0.029720	20.61	-8.01	-7.75	-8.01	-7.28
1850	328	0.029727	27.92	-8.01	-7.76	-7.99	-7.28
1950	342	0.029733	22.66	-8.01	-7.76	-7.99	-7.28
2000	349	0.029735	29.57	-8.00	-7.75	-7.99	-7.27
2050	334	0.029737	24.66	-8.01	-7.76	-7.99	-7.27
2150	354	0.029741	32.50	-8.01	-7.76	-7.99	-7.28
2250	351	0.029746	53.91	-8.01	-7.76	-7.99	-7.28
2350	401	0.029750	43.20	-8.01	-7.76	-7.99	-7.28
2450	373	0.029754	40.92	-8.00	-7.76	-7.98	-7.27
2550	379	0.029757	47.39	-8.00	-7.76	-7.98	-7.27

Table 4.4: SU^2 convergence study recap (division on the airfoil); residuals are reported in log_{10} format

d	iteration	C_d	time [min]	$res(\rho)$	$res(\rho u)$	$res(\rho v)$	$res(\rho e)$
10	299	0.029699	12.48	-8.01	-7.77	-7.99	-7.29
15	302	0.029700	14.48	-8.01	-7.77	-7.99	-7.28
20	298	0.029700	11.54	-8.01	-7.76	-7.99	-7.28
25	299	0.029700	18.08	-8.01	-7.76	-8.00	-7.28
30	302	0.029700	11.20	-8.01	-7.75	-8.00	-7.27
40	305	0.029700	12.59	-8.00	-7.74	-8.00	-7.27
50	309	0.029700	11.34	-8.00	-7.75	-8.00	-7.27
60	312	0.029700	14.04	-8.01	-7.75	-8.01	-7.28

Table 4.5: SU^2 convergence study recap (farfield distance); residuals are reported in log_{10} format

4.5 Coarse optimization

The *MATLAB* genetic algorithm *ga*, from *Global Optimization Toolbox*, has been chosen as optimizer; this due to the fact that it is able to archive the global optimum for discontinuous (fitness function not defined over the entire domain), constrained problem. Genetic algorithms are a family of derivative-free, population based, stochastic optimizers; first presented in the 1960s, their principle is inspired by natural selection of biological systems. To initialize the optimization process an initial population have to be given or generated; the individuals are then encoded in string that unequivocally define them: each string (chromosome) is composed the individual's codified input variable (gene). The optimization generally performs four steps each iteration (generation)

1. a usually large portion (generally between 80 and 95%) of the population obtained from previous generation, composed by individuals codified in strings, is processed by the *crossover function*. Given two current individuals, it generates a new one by mixing the strings of the parents;

2. a small group of individuals (lower than 5%) of the given population, are processed by the *mutation function*; it operates modifying a portion of the individual's string creating a new mutated individual

3. all new individuals are evaluated by the fitness function

4. individuals, newly computed and from previews generations, are sorted by a given criteria (usually their fitness value); the top prospects will compose the future generation while the other will be discarded; this task is carried out by the *selection function*

Population size	2000
Fitness function tolerance	1e-6
Nonlinear inequality tolerance	1e-6
Max generation	250
Max stall generation	50
Scaling function	*Rank*
Selection function	*Stochastic uniform*
Crossover function	*Constrain dependent*
Migration direction	*Forward*
Nonlinear constrain algorithm	*Augmented Lagrangian*

Table 4.6: *ga* function parameters

The particular implementation of the three functions defines the specific algorithm; despite the high level of personalization that *MATLAB Global Optmization toolbox* offers, the majority of *ga* parameters have been left untouched; the specs used in the optimization are reported in the table 4.6.

The algorithm has the disadvantage to be computational expensive: it needs several generation (composed by large amount on individuals) to converge to the global optimum. This aspect is mitigated by the use of a surrogate model and the parallelization of the optimizer. To furthermore reduce the computation time, all components of $\mathbf{x}$ are *integer type*, the geometrical parameters unit refers to a thousandth of the airfoil chord; this implies the use of special creation, crossover and mutation functions. The minimization approach is changed by focusing on penalty function instead of fitness function; the penalty is structured to consider linear constrains, feasibility and fitness of the individuals; the particular *ga* implementation relies on techniques described in references [21] [22].

The *ga* function calls separately the constrain function and the fitness function; the two functions have virtually the same execution time due to the fact the feasibility of the individuals can only be assessed at the end of the *SET* calculation. In order to reduce the computation time, when a new individual is generated, its fitness function and constrain vectors are computed in the same function and stored. This allows a huge time saving because the time needed for the single individual is now the sum of the actual computational time plus the time need to retrieve the individual from the storage array instead of twice the computational time.

The best individuals from previous coarse optimizations are included into the initial population at the beginning of each *OMM* iteration.

4.5.1 Optimization constrains

Linear constrains

The geometric parameters are bounded with both upper and lower limits. A liner inequality has been set in order to prevent the slat from taking up more than the 30% of the airfoil's chord (eq. 4.17).

$$\begin{aligned} \mathbf{x_{min}} &= [0.05c,\ 0.1c,\ 0.05c,\ 0.1c,\ 0.05,\ -0.015,\ -999,\ -999]; \\ \mathbf{x_{max}} &= [0.1c,\ 0.25c,\ 0.1c,\ 0.25c,\ 0.3c,\ 0.015c,\ 99,\ 99]; \\ \mathbf{A_{ineq}} &= \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}; \\ \mathbf{b_{ineq}} &= [0.3c;\ 0.3c]; \\ \text{such that } \mathbf{A_{ineq}}&\mathbf{x} < \mathbf{b_{ineq}}; \end{aligned} \tag{4.17}$$

In order to properly compare two consecutive iterations, the components of the input vector have been normalized respect their interval (eq. 4.18).

$$\mathbf{z} = \frac{\mathbf{x} - \mathbf{x_{min}}}{\mathbf{x_{max}} - \mathbf{x_{min}}} \tag{4.18}$$

The difference in the $\mathbf{z}$ will be evaluated to asses the convergence of the optimization.

Nonlinear inequality constrain

The nonlinear inequality constrain is composed by an array of three elements:

$$\mathbf{g_c}(\mathbf{x}) = [\ 0.5 - IS_{sol}(\mathbf{x});\ \bar{th}_{lim} - th_{up}(\mathbf{x});\ \bar{th}_{lim} - th_{dwn}(\mathbf{x})\ ]; \tag{4.19}$$

where $IS_{sol}(\mathbf{x})$ is logical value indicating whether the configuration is feasible both geometrical and computational wise, th_{up} and th_{dwn} are the percentage thicknesses of the slats.
The coarse model is the most demanding respect the feasibility of the design since the *SET* is the less robust method of the two. Consequently, since all the rest of the components of $\mathbf{g_c}(\mathbf{x})$ are geometrical, there is no need to adapt the constrain function regard the fine model. The constrains can be directly applied in the optimizer. In order to simplify the notation from now on the non linear inequality constrain will be addressed as $\mathbf{g}(\mathbf{x})$ (eq. 4.20).

$$\tilde{\mathbf{S}}_{\mathbf{g}}(\mathbf{g_c}(\mathbf{x})) = \mathbf{g_c}(\mathbf{x}) = \mathbf{g_f}(\mathbf{x}) = \mathbf{g}(\mathbf{x}) < 0 \tag{4.20}$$

Nonlinear equality constrain

MATLAB ga function does not allow to perform an integer optimization with nonlinear equality constrain. To overcome this limitation the nonlinear equality constrain is implemented in the evaluation function. In this case it is a scalar defined in eq. 4.21.

$$h_c(\mathbf{x},\alpha) = C_l(\mathbf{x},\alpha) - \bar{C}_{ltgt}; \tag{4.21}$$

The angle of attack α is used as independent variable to archive the constrain. The constrain function is minimized using *Newton–Raphson method* until the constrain tolerance is matched. The derivative has been computed through central finite difference. This means that at least four *SET* evaluations are needed for each feasible individual.
For each individual the problem is set as showed in eq. 4.22.

$$\begin{aligned}
&find\ \bar{\alpha}\ such\ that: \\
&|\tilde{S}_{hi-1}(h_c(\mathbf{x},\bar{\alpha}))| = |h_f(\mathbf{x_{i-1}},\bar{\alpha}) + S_{h\,i-1}(h_c(\mathbf{x},\bar{\alpha}) - h_c(\mathbf{x_{i-1}},\bar{\alpha}))| < toll \\
&iteratively\ solved\ using: \\
&\alpha_{k+1} = \alpha_k - \frac{\tilde{S}_{hi-1}(h_c(\mathbf{x},\alpha_k))\Delta\alpha}{\tilde{S}_{h\,i-1}\left[h_c(\mathbf{x},\alpha_k + \Delta\alpha/2) - h_c(\mathbf{x},\alpha_k - \Delta\alpha/2)\right]}
\end{aligned} \tag{4.22}$$

4.5.2 Optimization fitness function

The figure of merit for the first iteration is the drag coefficient computed using *SET* evaluated at the angle of attack resulting from eq. 4.22, here showed in 4.23;

$$c(\mathbf{x}) = C_d(\mathbf{x},\bar{\alpha})\Big|_{SET} \tag{4.23}$$

From the second *OMM* iteration, the correction operator became involved, the *SET* drag coefficient is corrected as shown in eq. 4.24.

$$c(\mathbf{x}) = \tilde{\mathbf{S}}\left(C_d(\mathbf{x},\bar{\alpha})\Big|_{SET}\right) \tag{4.24}$$

4.6 Optimization workflow

The *OMM* optimization workflow scheme has been reported in figure 4.11.

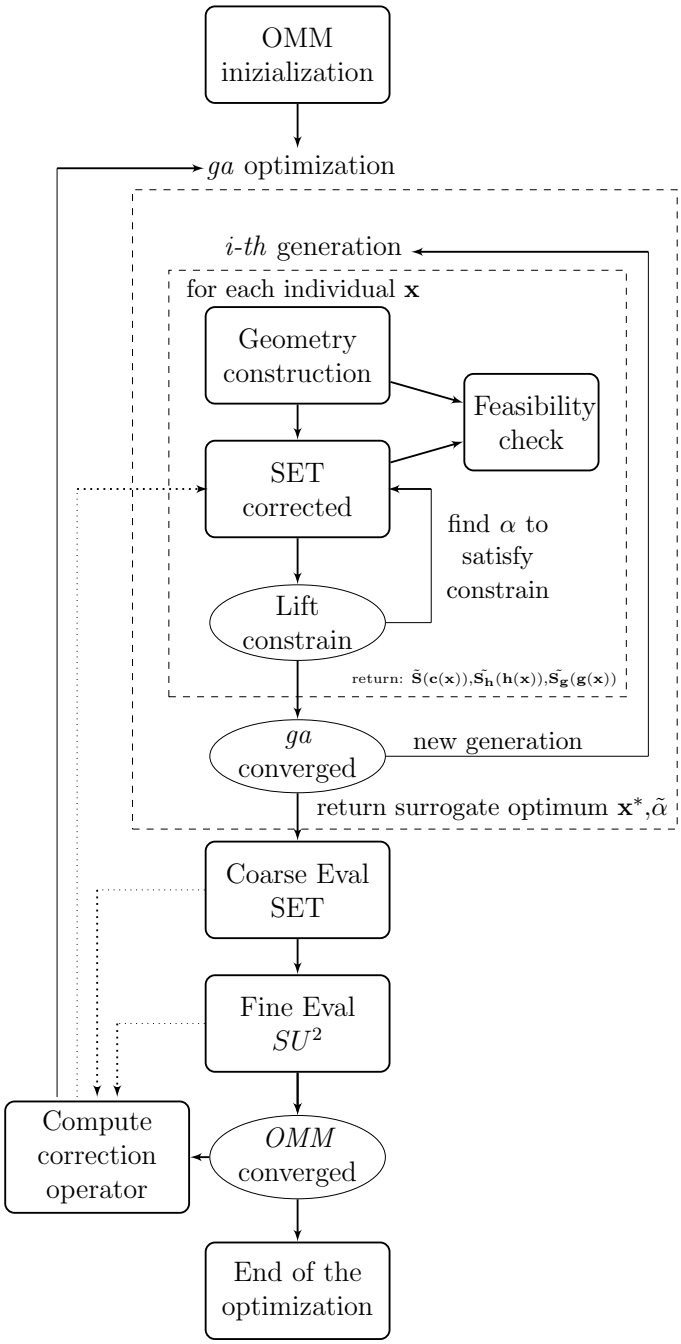


Figure 4.11: Supersonic optimization workflow

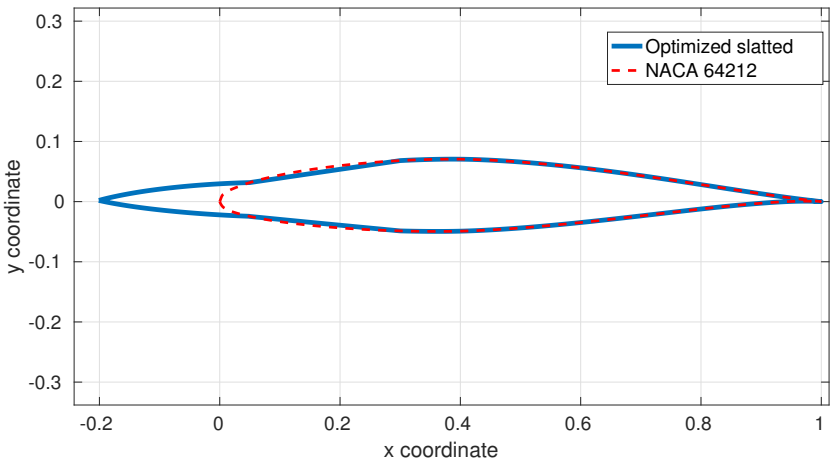


Figure 4.12: Optimized design for supersonic flight

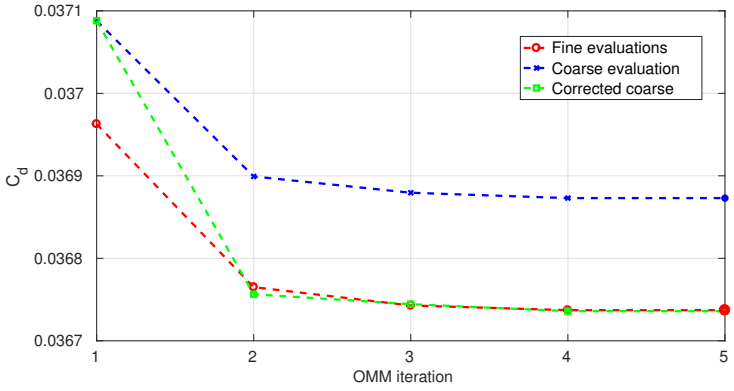


Figure 4.13: *OMM* convergence history

4.7 *OMM* optimization

The algorithm has been run and it appears that five iterations are needed to converge. The convergence has been reached due to the fact that the improvement between the last iteration was lower than the set tolerance. The optimized input array is:

$$\mathbf{x}^* = [0.05c,\ 0.25c,\ 0.05c,\ 0.25c,\ 0.05c,\ 0.02c,\ 2,\ 3]; \tag{4.25}$$

It appears that $\mathbf{x}^*$ has almost all of its component at the optimization bounds: the first four elements are equals to their upper limits while the last two are near the lower ones. The resulting shape is reported in figure 4.12.

Assuming the first iteration (low-fidelity optimum) as baseline for the modified design, an improvement on the result can be seen over the *OMM*

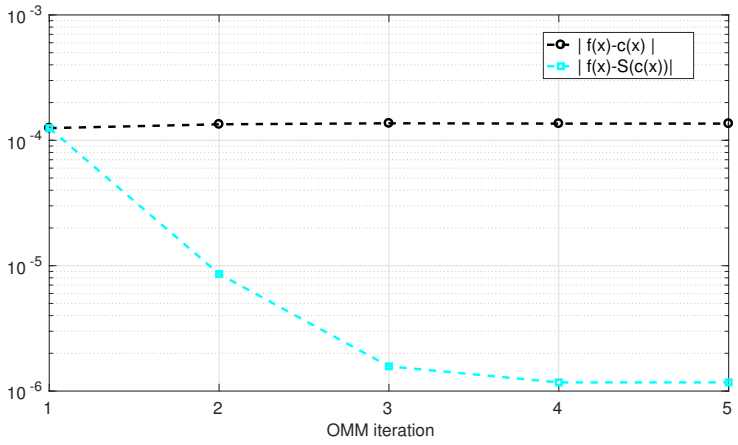


Figure 4.14: Discrepancies between models: fine and coarse (black curve), fine and corrected coarse (cyan curve)

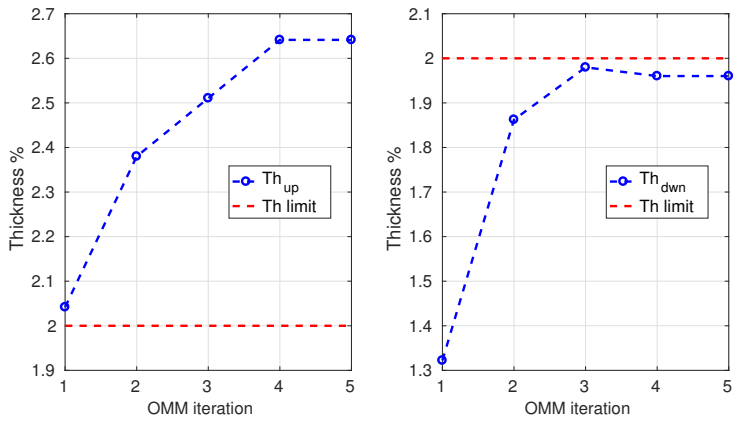


Figure 4.15: Slat thicknesses evolution

iterations (fig. 4.13) . The major improvement has been archived between the first and the second iteration, where the discrepancy between the fine model and the corrected model has been cut by an order of magnitude (fig. 4.14). The correction at the end of the algorithm was able to reduce the error on the merit function by almost two orders of magnitude. The improvement of the coarse fitness function can be attributed to the increasing quality of the initial populations over the *OMM* iterations.

The error on the nonlinear equality constrain is slightly reduced over the course of the optimization, even if it seems that the correction is ineffective, the level of error is still acceptable (fig. 4.16). The geometric constrains for the upper slat are verified since the first iteration; the constrain for the lower one greatly improves from iteration one but it fail to overcome the minimum

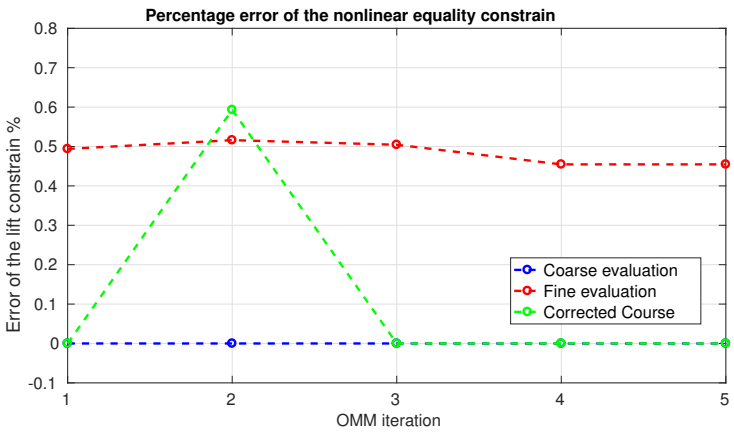


Figure 4.16: Percentage error of the nonlinear equality constrain

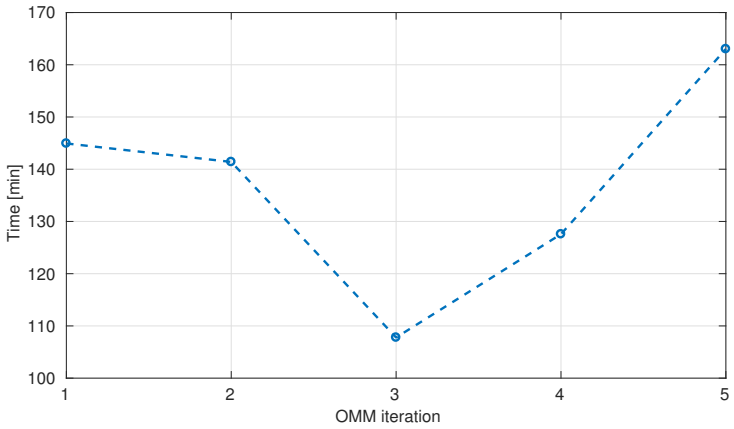


Figure 4.17: Computational time of *ga* for each iteration

thickness. The thickness of the lower slat settles on 1.96% (fig. 4.15).

From the *ga* running time (fig. 4.17) it is possible to roughly estimate how many feasible individuals has been created for each iteration. For the majority of individuals it can be speculated just one iteration of the *Newton-Raphson* algorithm (section 4.22) so the execution time can be assumed as the sum of the time needed to generate the geometry plus the time needed for four *SET* evaluation; neglecting the geometry generation it is possible to approximate:

$$feasible\ elements \sim \frac{GA_{running\ time}}{\frac{4 \cdot SET_{running\ time}}{n_{processors}}}; \tag{4.26}$$

The values and percentages of estimated feasible individuals are reported in figure 4.18; it is possible to notice a positive trend on the percentages due

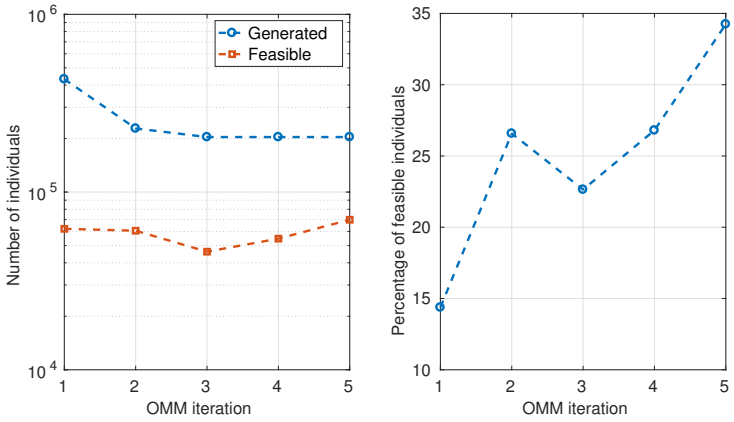


Figure 4.18: Estimated feasible individuals in the *ga* for each iteration

to the increasing quality of initial population.

The time needed for the fine evaluation remains almost constant for every iteration and it's consistent with the value expected from the test runs.

OMM iteration	5
Improvement from first iteration	$2.2571e-04$
Discrepancy between model last iteration	$1.1721e-06$
Error on the lift constrain	0.5%
Percentage thickness of upper slat	2.64
Percentage thickness of lower slat	1.96
Total time of *OMM*	14.9 [h]
Total time of *ga*	11.4 [h]
Equivalent fine model evaluation[1]	$\sim 23\ (17+5)$
Equivalent fine lift constrained individuals[2]	~ 6
Individual completely tested in *ga*	293443
Individual generated in *ga*	~ 1272005
Percentage of feasible individuals	~ 23

Table 4.7: Supersonic *OMM* recap

[2]In the overall optimization time, 5 *CFD* evaluations have been made, 17 evaluations would have been completed withing the overall *ga* running time

[2]Assuming 4 evaluations to satisfy the lift constrain, from a total of 23 evaluations a total of six lift constrained airfoil would have been obtained

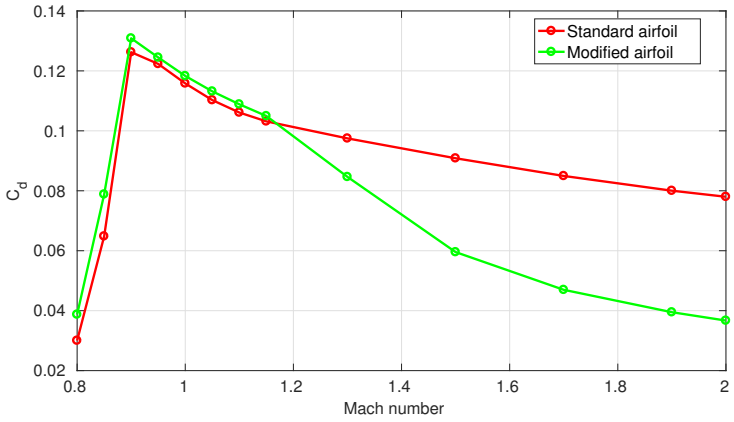


Figure 4.19: Drag coefficient comparison (coefficients are referred to the unit chord for both airfoils)

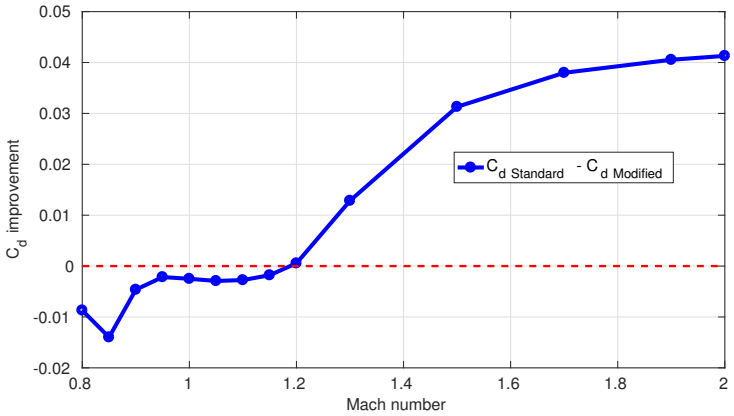
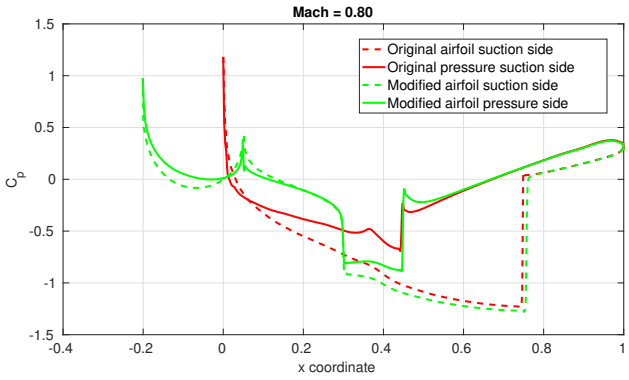


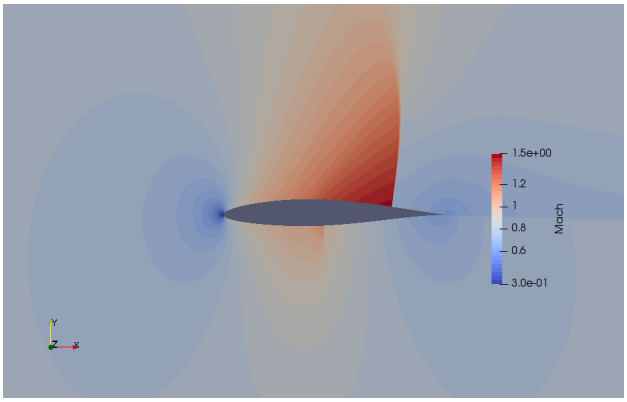
Figure 4.20: Drag coefficient improvement

4.8 Comparison with the original airfoil

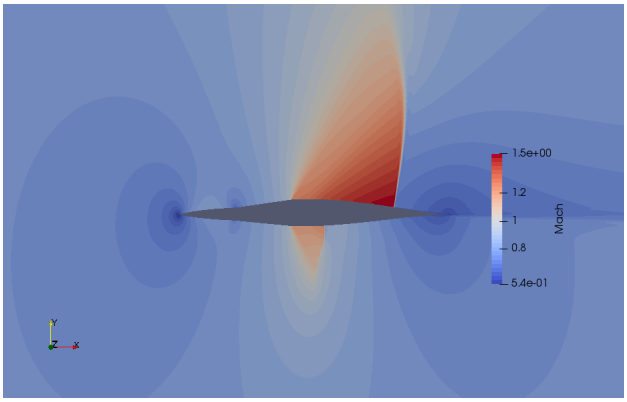
A comparative run (fig. 4.21 to 4.24) has been made in order to asses the performance of the design at different Mach numbers; at each evaluation the lift constrain has been enforced through the angle of attack.
All aerodynamic coefficients of the modified airfoil are reported to the initial unit chord so the values are directly comparable.
The mesh used to perform the calculation is similar to the one used to characterize the original airfoil (section 3.1.1).



(a)

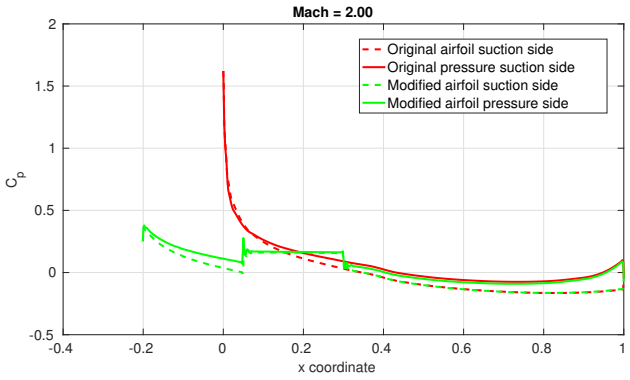


(b)

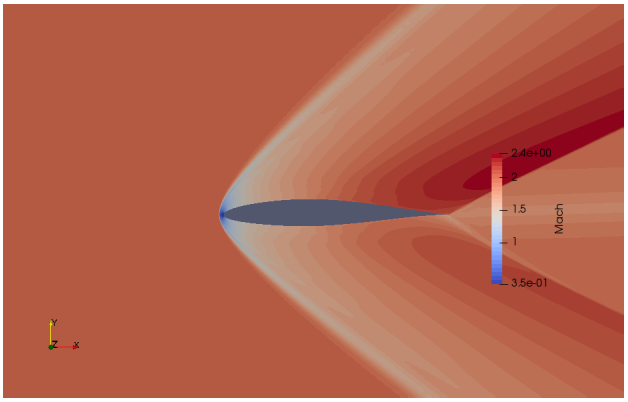


(c)

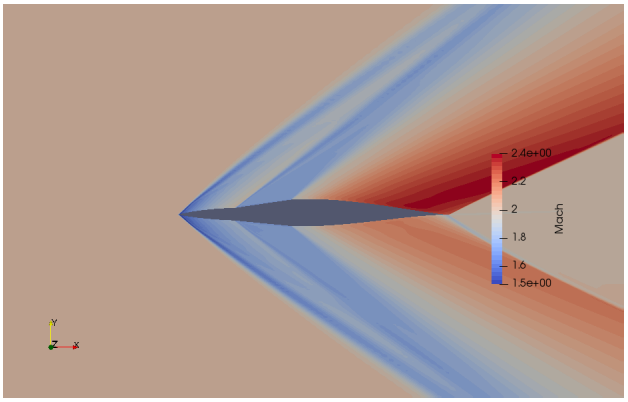
Figure 4.21: (a) Pressure coefficient comparison. (b) Mach contour for standard airfoil at Mach = 0.8. (b) As for (c) but for the modified airfoil.



(a)



(b)



(c)

Figure 4.22: (a) Pressure coefficient comparison. (b) Mach contour for standard airfoil at Mach = 2.0 (c) As for (a) but for the modified airfoil.

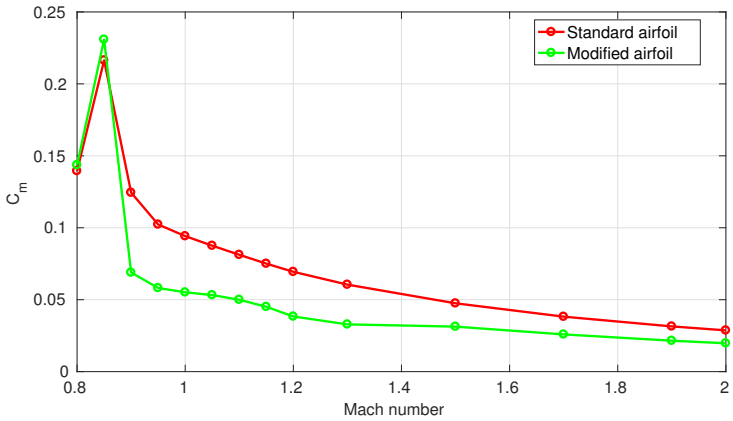


Figure 4.23: Pitching moment coefficient comparison

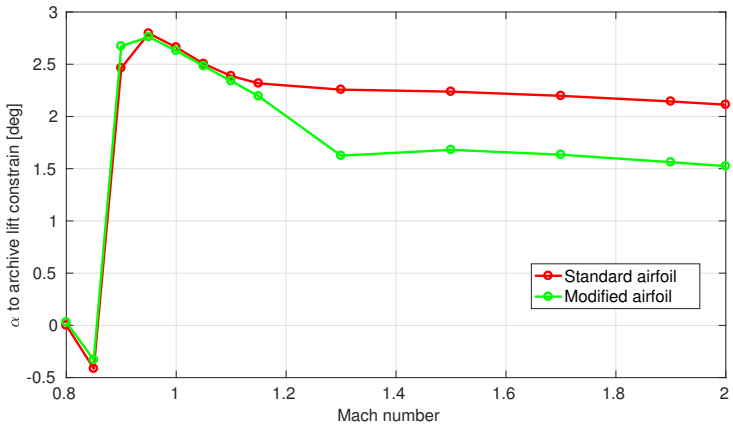


Figure 4.24: α to satisfy lift constrain

Chapter 5

High lift Configuration

The second phase of this work aims to find the position of the upper slat previously obtained in order to perform as high lift device in low speed flight. The design of high-lift configuration is crucial and it directly affects required runway distance, available payload and operative range. An usual optimization of such configuration can comprehend several design variables (the number drastically increases if the elements' shapes has to be optimized and thus the variable vector includes the parametrization variables) and multiple non-linear constrains (structural, cinematic, noise); therefore it can become excessively time demanding and computational expensive. Surrogate based methods have been extensively used in this process since the end of the 1990s; most applications rely on *functional-based* e.g. *Greenman et al* [23] Neural Network, *Kanazaki et al* [24] Kriging Method, *Leifsson et al* [25] Polynomial Response Surface (for marine application). A *physic-based multi-resolution* [26] has been recently tested by *Jonsson et al*; the optimization is carried out using space mapping algorithm to correct the coarse mesh *CFD* results. Here it is proposed a *physic-based multi-fidelity* approach that couples panel method and expensive *CFD*.
Usually, high lift configuration is composed by several elements operating on both leading and trailing edges. In this paragraph, in order to not increase the number of optimization parameters, only the upper slat will be operated; it is reasonable to presume that the high lift performance could still be enlarged coupling its effect with a (or multiple) trailing edge device.

5.1 Problems formulation

The objective is to obtain the optimal displacement of the slat. In this chapter, two optimization will be performed using the same algorithm; the objective function and the flow conditions are different for each optimization:

- Maximum lift condition: at Reynolds $Re = 6 \cdot 10^6$, the aim is to enlarge the baseline lift coefficient in the proximity of its max lift condition ($\alpha = 15\ deg$). This objective aim to reduce the take-off and landing speed and therefore the runaway length. A comparison of the $Cl - \alpha$ curve will be done afterward to assess how the angle of stall is affected by the deployment of the slat.

- Take off condition: at Reynolds $Re = 6 \cdot 10^6$ the aim is to maximize the climb index Cl^3/Cd^2 (see references [27] [28]), this merit figure is directly correlated with the time to climb and therefore its optimization can provide a reduction in term of fuel consumption during this phase of the mission;

The optimization is carried out using multi-fidelity surrogate basic technique, *Hess-Smith panel method* (HSPM) has been chosen as the surrogate model; since it is a potential based method, it is necessary to rely on an additional condition to contemplate the phenomenon of stall. On this matter *Valarezo et al* [29] reported a semi-empirical condition concerning the suction peak in which it can be assumed that the airfoil is approximately at the maximum lift.
The position of the slat is the variable of the optimization: the degrees of freedom are the *x-position*,*y-position* and relative incidence respect the main airfoil. The optimization is constrained so:

- The resulting design will be feasible in therms of clearance between slat and airfoil;

- The *Valarezo* condition is respected both in the main airfoil and the slat;

The optimization ceases if:

- Norm difference of the normalized component input vector is less than 0.005;

- No improvement has been registered in the last 10 iteration;

- The fine evaluation has been performed for more than 15 times;

5.2 Geometry description

The slat and the main airfoil designs (fig 5.1) are obtained from the first part of this work.

The relative position of the slat respect the main airfoil is defined by three geometric parameters, as shown in figure 5.2:

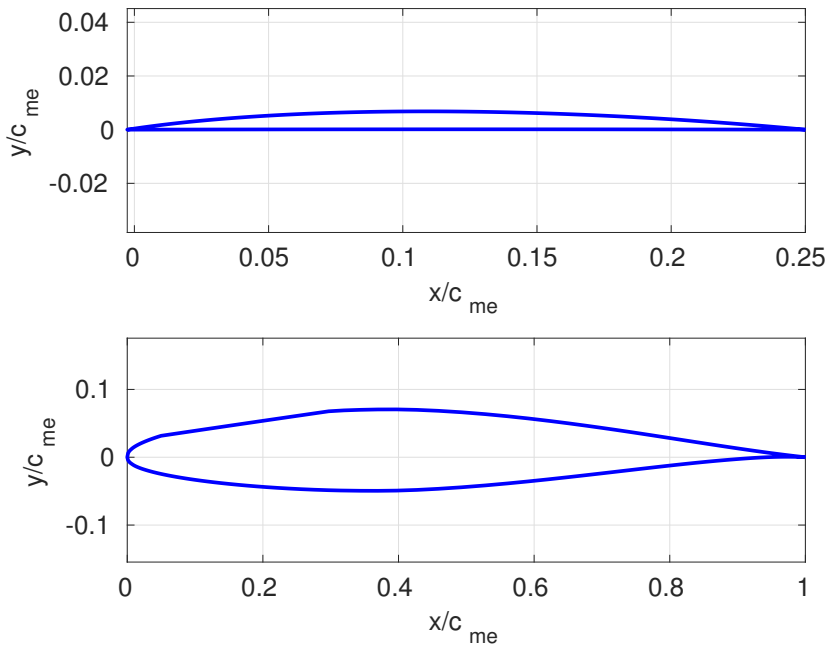


Figure 5.1: Slat (top), Main airfoil (bottom); coordinates normalized respect to the main element chord

- the distance along x-axis between the slat trailing edge and the main airfoil leading edge normalized respect to main airfoil chord
- the distance along y-axis between the slat trailing edge and the main airfoil leading edge normalized respect to main airfoil chord
- the angle of attack of the slat expressed in degrees

All distances are normalized respect the main airfoil chord.

The parameters are included into the input array such that:

$$\mathbf{x} = [\Delta x, \Delta y, \theta]; \tag{5.1}$$

The geometry is computed in a *MATLAB* function that executes a $2D$ translation of the slat followed by a rotation around its trailing edge.

5.3 Low fidelity model

The *Hess-Smith Panel Method* ($HSPM$) has been used as surrogate model. Under the assumption of incompressible, inviscid, irrotational flow,

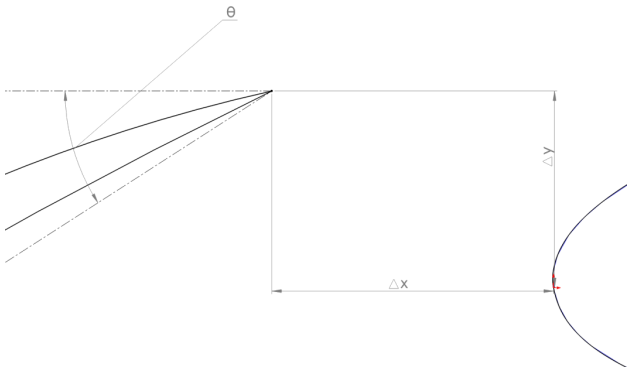


Figure 5.2: Parameters that define slat's position

the velocity potential satisfies the Laplace equation (eq. 5.2); this formulation can be used in low-speed aerodynamic to describe the flow domain outside the region in which viscosity effects and rotational flow are confined (wake and boundary layer).

$$\nabla^2 \Phi^* = 0 \tag{5.2}$$

Where Φ^* is the total potential that can be divided into the free stream potential Φ_∞, which is function of asymptotic velocity components, and into the perturbation potential Φ. The general solution to Laplace Equation for a submerged body can be obtained from Green's identity as distribution of source γ and doublet μ (eq. 5.3), for a compete derivation reference see [30].

$$\Phi^*(x,y,z) = \Phi_\infty - \frac{1}{4\pi} \int_{body} \left[\sigma \left(\frac{1}{r} \right) - \mu \mathbf{n} \cdot \nabla \left(\frac{1}{r} \right) \right] dS \tag{5.3}$$

HSPM considers source distribution and a constant vortex distribution γ in order to satisfy the *Kutta* condition and the body boundary condition ($\partial\Phi/\partial\mathbf{n}|_{body} = 0$); once the geometry is divided into panels the potential can be expressed as

$$\Phi^*(x,y,z) = \Phi_\infty + \sum_{j=1}^{n_{panels}} \frac{1}{2\pi} \int_{panel_j} [\sigma(s) \ln r - \gamma\theta] \, dS \tag{5.4}$$

For a single airfoil configuration, the algorithm divides the body into n panels, starting from the pressure side's trailing edge proceeding clockwise to the suction side's one. Each *i-esim* panel has its own source of strength σ_i, its own vortex of strength γ_i and its own control point located in the middle of the panel. The source strength is assumed to be constant panel-wise and it is determined by satisfying the flow tangency on each panels, the vortices'

intensity is set to be equal over the entire body such that $\gamma_1 = \gamma_i = \gamma$; the Kutta condition is used to set its value.
From the intensity of sources and vortices, it is possible to obtain the perturbation velocity that is directly related to the pressure coefficient (eq. 5.5).

$$Cp = 1 - \left(\frac{u_{pert}}{\|\mathbf{U}_\infty\|} \right)^2; \tag{5.5}$$

In order to compute the intensities of the sources and vortices for each panel the induced velocities are now calculated on each control point . For the *k-esim* control point the influence of the source and the vortex distributed over the *j-esim* panel (within the points *j* and *j+1*) can be expressed in the panel's reference system as function of relative distance and angle (eq. 5.6).

$$\begin{aligned} \tilde{u}_{source} &= -\frac{\sigma_j}{2\pi} log \frac{\|\tilde{\mathbf{r}}_{\mathbf{j+1}}\|}{\|\tilde{\mathbf{r}}_{\mathbf{j}}\|}; & \tilde{v}_{source} &= \sigma_j \frac{\Delta\tilde{\theta}}{2\pi}; \\ \tilde{u}_{vortex} &= -\gamma \frac{\Delta\tilde{\theta}}{2\pi}; & \tilde{v}_{vortex} &= -\frac{\gamma}{2\pi} log \frac{\|\tilde{\mathbf{r}}_{\mathbf{j+1}}\|}{\|\tilde{\mathbf{r}}_{\mathbf{j}}\|}; \end{aligned}$$

$$\begin{aligned} \tilde{\mathbf{r}}_{\mathbf{j}} &= [\tilde{x}_j - \tilde{x}_k, \tilde{y}_j - \tilde{y}_k]; \\ \tilde{\mathbf{r}}_{\mathbf{j+1}} &= [\tilde{x}_{j+1} - \tilde{x}_k, \tilde{y}_{j+1} - \tilde{y}_k]; \\ \Delta\tilde{\theta} &= atan\left(\frac{\tilde{\mathbf{r}}_{\mathbf{j+1}}(2)}{\tilde{\mathbf{r}}_{\mathbf{j+1}}(1)} \right) - atan\left(\frac{\tilde{\mathbf{r}}_{\mathbf{j}}(2)}{\tilde{\mathbf{r}}_{\mathbf{j}}(1)} \right); \end{aligned} \tag{5.6}$$

Where the tilde apex represents the quantity in the *j-esim* panel reference system. Once transported in the global reference system, the projection along the normal direction of the *k-esim* source induced velocity is placed into the *k,j* element of the matrix $\mathbf{A}$. At the same time the projection along the tangent direction is placed into the *k,j* element of the matrix $\mathbf{B}$ and the same procedure is done for the vortex induced velocities for the matrix $\mathbf{C}$ and $\mathbf{D}$ (eq. 5.7).

$$\begin{aligned} &for\ k = \ 1:n,\ for\ j = 1:n \\ \mathbf{A}(k,j) &= \mathbf{n_k} \cdot [u_j, v_j]_{sources}; \\ \mathbf{C}(k,j) &= \mathbf{n_k} \cdot [u_j, v_j]_{vortex}; \\ \mathbf{B}(k,j) &= \tau_\mathbf{k} \cdot [u_j, v_j]_{sources}; \\ \mathbf{D}(k,j) &= \tau_\mathbf{k} \cdot [u_j, v_j]_{vortex}; \\ \mathbf{b}(k) &= -\mathbf{U}_\infty \cdot \mathbf{n_k}; \end{aligned} \tag{5.7}$$

The effect of the vortex induced velocities are considered thought the last

column of $\mathbf{A}$ and $\mathbf{B}$,that are computed from the matrices $\mathbf{C}$ e $\mathbf{D}$ (eq. 5.8).

$$\begin{aligned} \mathbf{A}(k,n+1) &= \sum_{j=1}^{n} \mathbf{C}(k,j); \\ \mathbf{B}(k,n+1) &= \sum_{j=1}^{n} \mathbf{D}(k,j); \end{aligned} \tag{5.8}$$

In order to satisfy the Kutta condition, in the last row of $\mathbf{A}$ as well as the last row of the $\mathbf{b}$, the vector contains the normal projection of the asymptotic velocity (eq. 5.9).

$$\begin{aligned} &for\ j = 1 : n \\ \mathbf{A}(n,j) &= B(1,j) + B(n,j); \\ \mathbf{b}(n) &= -\mathbf{U}_\infty \cdot (\tau_\mathbf{1} + \tau_\mathbf{n}) \end{aligned} \tag{5.9}$$

A linear system has now been formed with the unknown vector composed by the intensity of the n sources and the intensity of the distribute vortices.

$$\begin{aligned} \mathbf{x} &= [\sigma_1,...,\sigma_j,...,\sigma_{n-1},\gamma]'; \\ \mathbf{Ax} &= \mathbf{b}; \end{aligned} \tag{5.10}$$

The discretized perturbation velocity of the *k-esim* panel can be computed as:

$$\mathbf{u_{pert}}(j) = \sum_{k=1}^{n} \mathbf{B}(j,k)\mathbf{x}(k) + \mathbf{U}_\infty \cdot \tau_\mathbf{j}; \tag{5.11}$$

For a multi-element airfoil composed by m airfoils, the matrix $\mathbf{A}$ and $\mathbf{B}$ are replaced by the matrix $\bar{\mathbf{A}}$,$\bar{\mathbf{B}}$ and the vectors $\mathbf{x}$ and $\mathbf{b}$ are replaced by the vectors $\bar{\mathbf{x}}$ and $\bar{\mathbf{b}}$. The matrices are assembled from the m^2 combination of the matrix $\mathbf{A_{p,q}}$ and $\mathbf{B_{p,q}}$, $p = 1 : m$, $q = 1 : m$, that considers the influences over the panels of the body p of the singularities of the body q. The vectors $\bar{\mathbf{x}}$ and $\bar{\mathbf{b}}$ respectively contain the p vectors $\mathbf{x_q}$ and $\mathbf{b_q}$. The Kutta condition and the tangency condition are enforced in each sub-matrices. Once solved the linear system, pressure coefficient is computed in the same way as the single airfoil.

$$\begin{bmatrix} \mathbf{A_{1,1}} & \mathbf{A_{1,2}} & ... & \mathbf{A_{1,m}} \\ \mathbf{A_{2,1}} & \mathbf{A_{2,2}} & ... & ... \\ ... & ... & \mathbf{A_{p,q}} & ... \\ \mathbf{A_{m,1}} & ... & ... & \mathbf{A_{m,m}} \end{bmatrix} \begin{bmatrix} \mathbf{x_1} \\ \mathbf{x_2} \\ ... \\ \mathbf{x_m} \end{bmatrix} = \begin{bmatrix} \mathbf{b_1} \\ \mathbf{b_2} \\ ... \\ \mathbf{b_m} \end{bmatrix} \tag{5.12}$$

$$\bar{\mathbf{u}}_{\mathbf{pert}} = \begin{bmatrix} \mathbf{u_{pert\ 1}} \\ \mathbf{u_{pert\ 2}} \\ \dots \\ \mathbf{u_{pert\ m}} \end{bmatrix} = \begin{bmatrix} \mathbf{B_{1,1}} & \mathbf{B_{1,2}} & \dots & \mathbf{B_{1,m}} \\ \mathbf{B_{2,1}} & \mathbf{B_{2,2}} & \dots & \dots \\ \dots & \dots & \mathbf{B_{p,q}} & \dots \\ \mathbf{B_{m,1}} & \dots & \dots & \mathbf{B_{m,m}} \end{bmatrix} \begin{bmatrix} \mathbf{x_1} \\ \mathbf{x_2} \\ \dots \\ \mathbf{x_m} \end{bmatrix} + \begin{bmatrix} \mathbf{T_1} \\ \mathbf{T_2} \\ \dots \\ \mathbf{T_m} \end{bmatrix} \begin{bmatrix} U_x \\ U_y \end{bmatrix} \tag{5.13}$$

Where the array $\mathbf{T_q}$ has size $n \times 2$ and it contains the n tangential vector of the body q.
The method has been implemented in a *MATLAB* function and it has been be tested (*Appendix* B) with the analytic results obtained by *Suddhoo et al* [31]. The function needs in input the X and Y coordinates (in form of $m \times (n+1)$ matrices) and the far-field velocity vector and it returns the lift, drag and pressure coefficient. The pressure coefficient array resulting from equation 5.5 has dimension $1 \times (m \times n)$ since the nonlinear inequality correction operator will need to be applied to it.

5.3.1 Convergence study

In order to asses the ideal number of panels ti achieve the optimized quantities' precision, a convergence study has been made. The pressure coefficient is computed solving a linear system; the cost of the computation nonlinearly increases with the dimension of the matrix, so it is crucial to not over-discretize the airfoil. The number of divisions along the chord is used as parameter that corresponds to half of the total number of panel. The quantities of interest are the lift coefficient and the pressure difference between the suction peak and the trailing edge. The test has been carried on a *NACA 0012* airfoil on three different incidences, it can be seen that 150 division are sufficient to archive a converged solution for both quantities (figures 5.3 5.4).

5.4 High fidelity model

To obtain more physically accurate results, the geometry resulted from the coarse optimization has been tested in *OpenFOAM*. The case has been set to solve incompressible *2D RANS* equations with the *simpleFOAM* flow solver. The closure equation relies on the eddy viscosity assumption and particularly uses $k-\omega$ *SST* formulation and the use of *wall-function*, both already implemented in *OpenFOAM*. *Gmesh* provides the *box-type* unstructured hexaedrical mesh; this type of mesh is easy implemented and fast computed, the ideal characteristic to be used in an optimization. The main element leading edge is positioned as the origin, the far-field boundary is

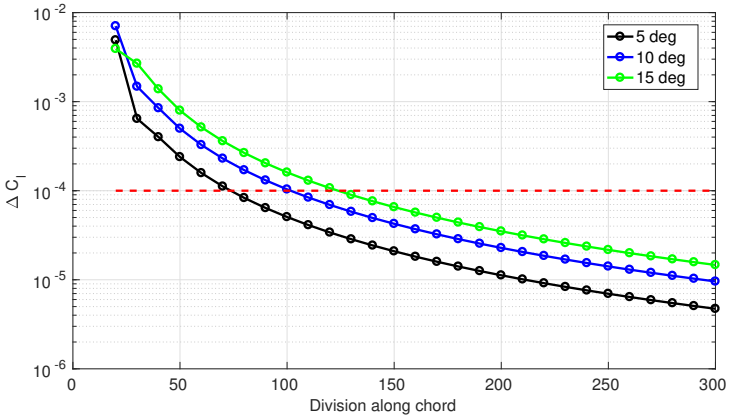


Figure 5.3: *HSPM*: ΔC_l with different number of panels

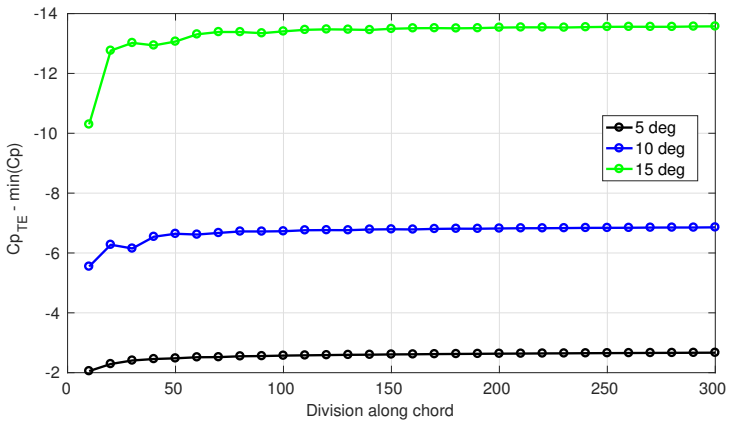


Figure 5.4: *HSPM*: suction peak values with different number of panels

Division	Cl	$min(C_p) - C_{pTE}$	Time [s]
20	1.7831	-12.763	0.337
30	1.7857	-13.021	0.533
40	1.7871	-12.938	1.082
50	1.7879	-13.066	1.479
60	1.7884	-13.307	1.576
70	1.7888	-13.385	2.102
80	1.7890	-13.384	2.737
90	1.7893	-13.344	3.440
100	1.7894	-13.406	4.255
110	1.7895	-13.457	5.235
120	1.7896	-13.473	7.032
130	1.7897	-13.466	8.174
140	1.7898	-13.453	9.557
150	1.7899	-13.492	11.762
160	1.7899	-13.510	12.853
170	1.7900	-13.514	13.944
180	1.7900	-13.508	14.815
190	1.7901	-13.514	16.690
200	1.7901	-13.532	20.419
210	1.7901	-13.540	21.947
220	1.7902	-13.540	24.243
230	1.7902	-13.534	24.342
240	1.7902	-13.547	25.472
250	1.7902	-13.556	27.584
260	1.7903	-13.560	29.269
270	1.7903	-13.559	31.935
280	1.7903	-13.558	36.098
290	1.7903	-13.568	38.279
300	1.7903	-13.574	41.896

Table 5.1: *HSPM* convergence study recap

$\mathbf{x_{test}}$	$[\ -0.0160c,\ -0.0057c,\ -39.993]$
Reynolds number	6e6
α	15°
Boundary at far field	$freestream$
Max iteration	7000
residualControl p	1e-7
residualControl U	1e-9

Table 5.2: *simpleFOAM* parameters

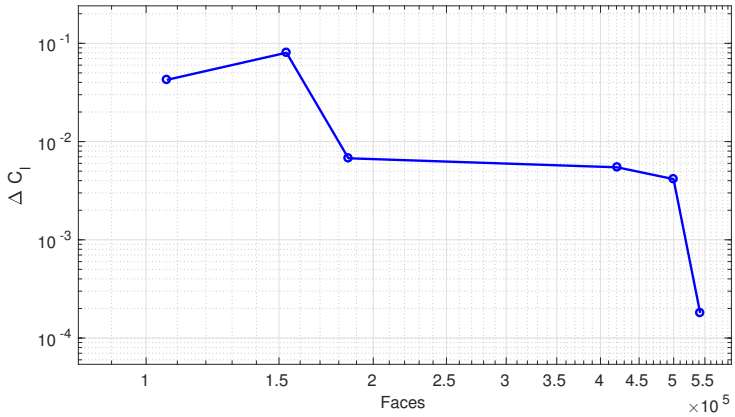


Figure 5.5: *simpleFOAM* mesh calibration: ΔC_l

placed at 300 chords of distance to assure that the finite boundary to have only a minimal influence on the solution; boundary layer's cells expand with a ratio of 1.3 in order to obtain the proper $y+$ for the application of the *wall-function*.

Convergence study

The mesh calibration has been carried out on a test configuration in the same boundary condition as the optimization (tab. 5.2). The computation is set to end when either the residuals are lower than prescribed values or maximum iteration limit is exceeded; all the cases used to asses the mesh calibration are converged under the residual condition.

Different meshes have been tested; the main differences between them consist on the dimension of the elements over the airfoil, the zone in which the further refinement have been applied and the size of the elements on the refinement. To summarize the calibration, the number of faces has been used as main parameter.

The results are shown in tab 5.3 and in figure 5.5 5.7; the chosen mesh

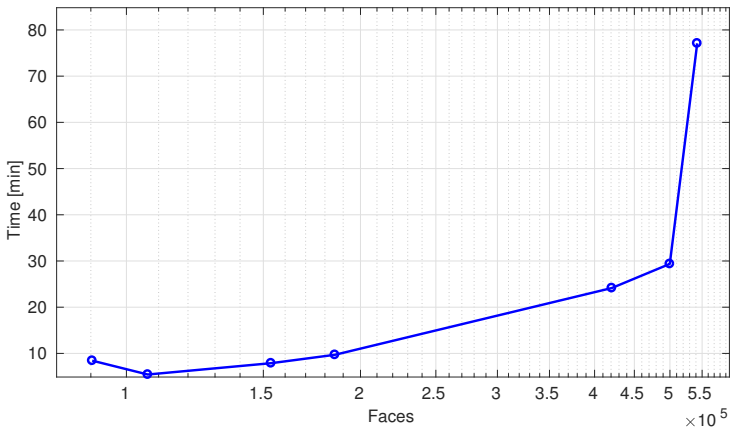


Figure 5.6: *simpleFOAM* mesh calibration: Time to reach convergence

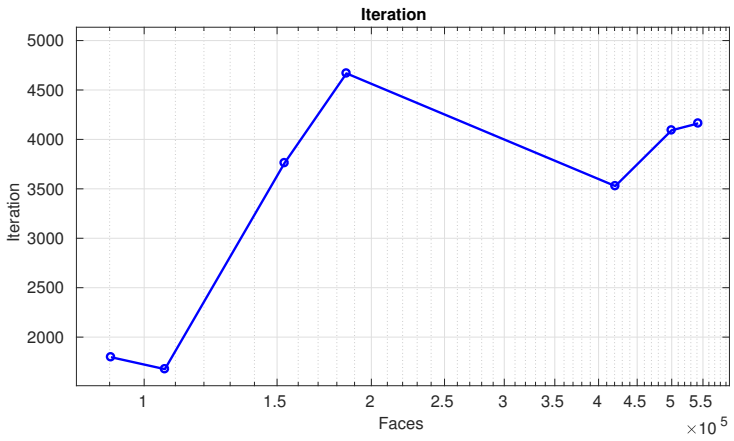


Figure 5.7: *simpleFOAM* mesh calibration: Iteration to reach convergence

n_{faces}	C_l	C_d	t [min]	$res(\rho)$	$res(u)$	$res(v)$	$res(k)$	$res(\omega)$
90365	1.9261	0.03384	8.43	-9.47	-9.00	-8.24	-6.11	-6.01
106530	1.8836	0.02458	5.43	-9.21	-9.00	-8.08	-6.07	-6.02
153460	1.8033	0.02552	7.88	-9.44	-9.00	-8.04	-6.04	-6.01
185380	1.7966	0.02481	9.70	-9.54	-9.14	-8.00	-6.07	-6.03
420930	1.7911	0.02299	24.15	-9.00	-9.40	-8.48	-6.12	-6.00
499800	1.7952	0.02210	29.37	-9.00	-9.11	-8.32	-6.01	-6.00
542080	1.7951	0.02217	77.10	-9.00	-9.09	-8.40	-6.06	-6.01

Table 5.3: *simpleFOAM* mesh convergence study; the number of face is used a parameter to sort the configurations; residuals are printed in log_{10} format

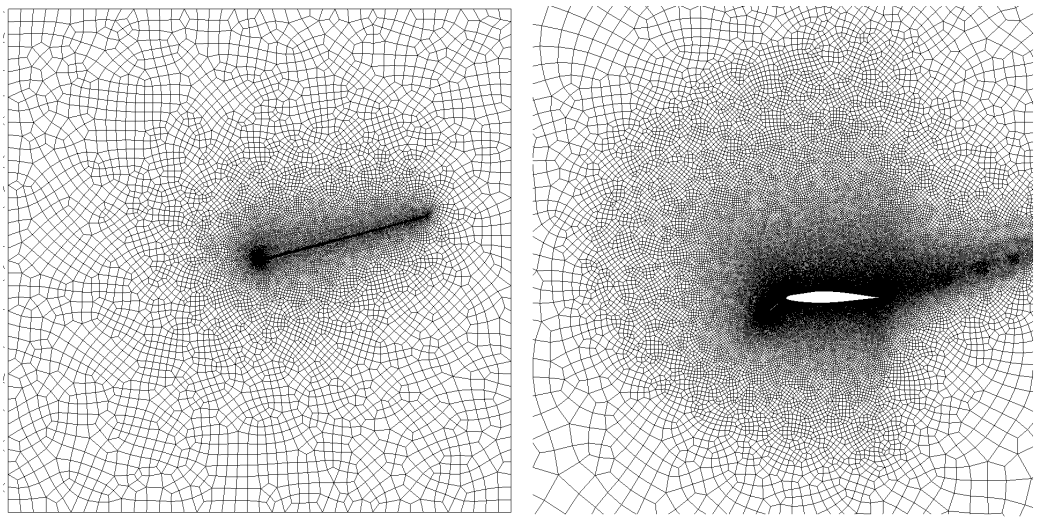


Figure 5.8: Mesh used for the fine evaluation; domain overview (left), zoom near airfoil (right)

has almost 500000 faces and it is able to predict the lift coefficient of the test configuration up to the third decimal; the maximum element size over the airfoil has size of $0.003c$ while on the far-field element of length $15c$ are used and the wake region is refined with element of length $0.005c$ (fig. 5.8).

5.5 Coarse optimization

The coarse optimization is carried out using $fmincon\ functopn$ available in the *MATLAB* Optimization Toolbox. The optimizer algorithm is set to be *interior-point* [32], a gradient-based method that transforms a nonlinear constrained problem, with both equality and inequality constrain, in a series

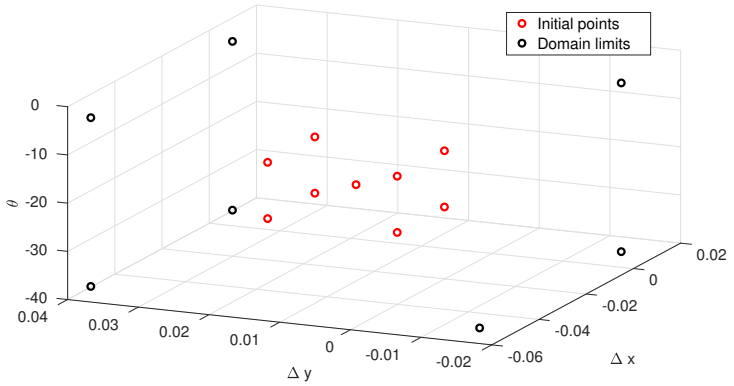


Figure 5.9: Initial points for the first coarse optmization

of equality constrained problems (eq. 5.14).

$$\begin{gathered}
\mathbf{x}^* = arg\,min\,\mathbf{f}(\mathbf{x}) := \{\mathbf{x_l} < \mathbf{x} < \mathbf{x_u} \,|\, \mathbf{g}(\mathbf{x}) < \mathbf{0} \,|\, \mathbf{h}(\mathbf{x}) = \mathbf{0}\}; \\
\Downarrow \\
\mathbf{x}^*,\mathbf{s}^* = arg\,min\,\mathbf{f}_\mu(\mathbf{x},\mathbf{s}) = arg\,min\left(\mathbf{f}(\mathbf{x}) - \mu\sum_i^{max} ln(s_i)\right) \\
\mathbf{x}^*,\mathbf{s}^* := \{\mathbf{x_l} < \mathbf{x} < \mathbf{x_u} \,|\, \mathbf{g}(\mathbf{x}) + \mathbf{s} = \mathbf{0} \,|\, \mathbf{h}(\mathbf{x}) = \mathbf{0}\};
\end{gathered} \tag{5.14}$$

Where $\mathbf{s}$ is a positive vector composed by slack variables; it has the same dimension of the enforced nonlinear inequalities: as μ decreases to zero the minimum of $\mathbf{f}_\mu$ approaches to the minimum of $\mathbf{f}$. The *MATLAB* algorithm first attempts to solve the approximated equation constrained through Lagrangian multiplier using a gradient method (Newton step); alternatively a conjugate gradient step is performed.

As gradient-based method, the convergence point depends on the initial guess and the stopping criteria can be satisfied at a local minimum; to reduce the probability of this occurrence multiple, starting point are set for the optimization. Nine factorial distributed points are set (fig. 5.9) for the fist *OMM* iteration. The optimal point from the last manifold mapping optimization and the best point recorded in the optimization are then added to the array for the following iterations. The derivatives are approximated by the solver through finite forward difference. As for the supersonic optimization the fitness and constrain vectors are computed in the same function and stored. Due the the deterministic nature of the algorithm it is possible to save further time keeping the values from the previous coarse optimizations, to be effective the value must be saved before applying the corrector operator. The

Starting point	10 (9 for first iteration)
Objective function tolerance	1e-4
Step tolerance	1e-4
Max evaluation	1000
Max iteration	100
Hessian approximation	BFGS
Subproblem setting	idl

Table 5.4: *fmincon* function main parameters

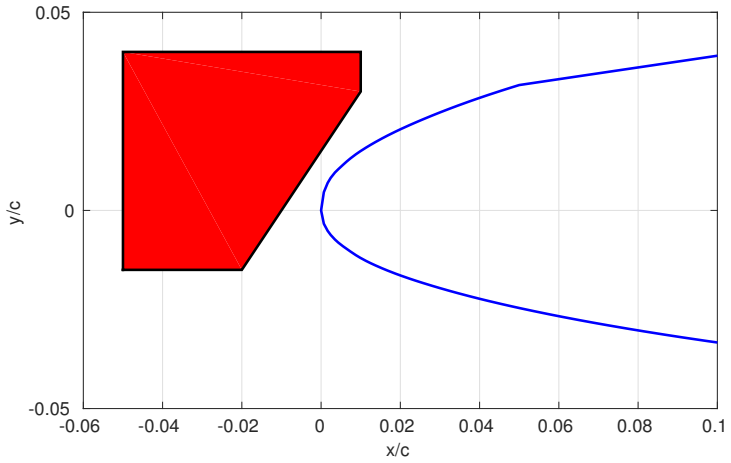


Figure 5.10: Suitable position of the slat's trailing edge

main parameters of scheme are reported in the tab 5.4.

5.5.1 Optimization constrains

Linear constrains

The coordinates of the slat's trailing edge are enclosed in a polygon defined by a linear inequality constrain and bounds for the input parameters (fig. 5.15). First two element of the input vector are normalized respect to the unit chord,the third one is expressed in degree.

$$\begin{aligned} \mathbf{x_{min}} &= [-0.05c, -0.015c, -40]; \\ \mathbf{x_{max}} &= [\ 0.01c,\ 0.015c,\ \ -5]; \\ \mathbf{A_{ineq}} &= \begin{bmatrix} 3/2 & -1 \end{bmatrix}; \\ \mathbf{b_{ineq}} &= -0.015/c; \\ \text{such that } \mathbf{A_{ineq}x} &< \mathbf{b_{ineq}}; \end{aligned} \tag{5.15}$$

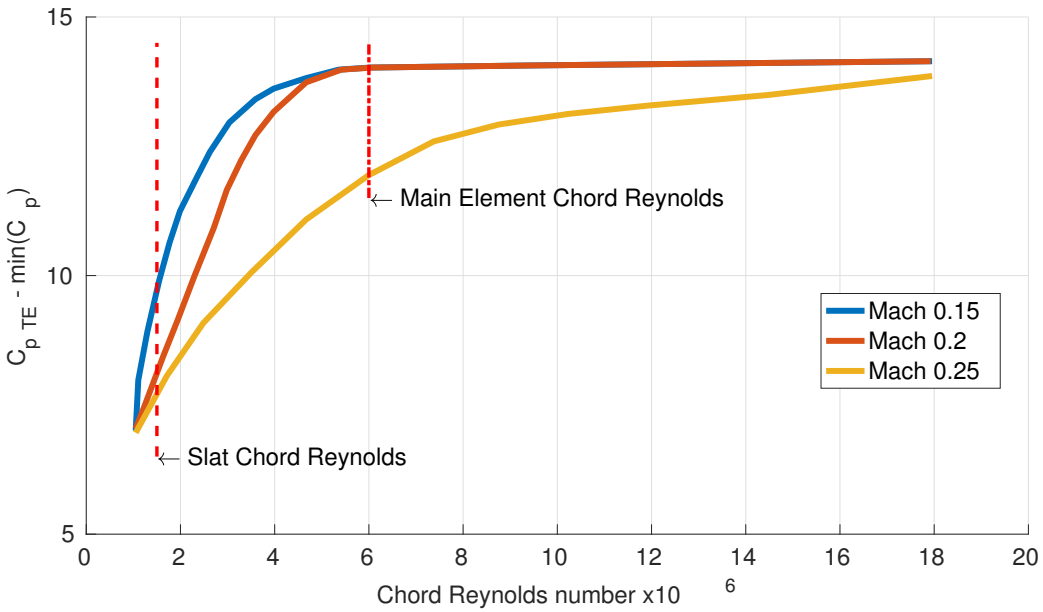


Figure 5.11: Pressure difference rule; maximum lift criteria for different Mach and Reynolds

The resulting box is showed in figure 5.10

Nonlinear inequality constrain

Valarezo et al provided an hybrid criteria that couples panel method computation and physical criterion to predict maximum lift performance. The method has been conceived to overcome the drawbacks of the pressure coefficient peak rule proposed by *Smith* [33] that:

- is independent from Reynolds and Mach number
- is not applicable to airfoil with leading edge devices, where has been observed pressure peaks much greater than the Smith prediction

It was noted from *Valarezo* that even if the pressure peak widely varied depending on free-stream condition, there was a correlation between the pressure difference between the suction peak and trailing edge value and the given Mach and Reynolds number, the relation obtainer is reported in figure 5.11.

The critical pressure difference value (PDV) at which is reach maximum lift is 14 for the main airfoil and 10 for the slat; this values are obtained considering the Reynolds number associate with the element's chord. The hypothesis of incompressible flow are not contemplated by the provided chart

5.11 because it is obtained from experimental tests. Mach 0.15 curve is take as reference instead.

For the slatted-airfoil configuration the *HSPM* provides the pressure distribution over the airfoils in form of an array of size $2 \times n$ where n is the number of panels. The obtained distribution is then corrected with the operator $\tilde{\mathbf{S}}_\mathbf{g}$ before the execution of function Δ. Δ performs the extraction of the array's portion relative to the considered surface and then proceed to subtract the trailing edge value to the pressure peak (eq. 5.17).

$$\mathbf{g}(\mathbf{x}) = \begin{bmatrix} -PDV_{main} + \Delta_{main}(\tilde{\mathbf{S}}_\mathbf{g}(\mathbf{C_p}|_{HS}(\mathbf{x})) \\ -PDV_{slat} + \Delta_{slat} \ \ (\tilde{\mathbf{S}}_\mathbf{g}(\mathbf{C_p}|_{HS}(\mathbf{x})) \end{bmatrix} < 0 \tag{5.16}$$

$$\begin{aligned} \Delta_{main}(\mathbf{C_p}) &= \mathbf{C_p}(1) - min(\mathbf{C_p}(1:n)); \\ \Delta_{slat}\ (\mathbf{C_p}) &= \mathbf{C_p}(n+1) - min(\mathbf{C_p}(n+1:end)); \end{aligned} \tag{5.17}$$

Nonlinear equality constrains

No nonlinear equality constrain have been set for this optimization.

5.5.2 Optimization objective function

The objective function is different for the two conditions, but in either case it is derived form the aerodynamic coefficient obtained from $HSPM$ and then corrected using $\tilde{\mathbf{S}}$ operator.

$$\mathbf{C_{cor}} = \begin{bmatrix} C_l(\mathbf{x})|_{cor} \\ C_d(\mathbf{x})|_{cor} \end{bmatrix} = \tilde{\mathbf{S}} \left(\begin{bmatrix} C_l(\mathbf{x})|_{HSPM} \\ C_d(\mathbf{x})|_{HSPM} \end{bmatrix} \right) \tag{5.18}$$

The actual value to be minimized is:

$$\begin{aligned} c(\mathbf{x})|_{max\ C_l} &= -C_l(\mathbf{x})_{cor}; \\ c(\mathbf{x})|_{max\ TO} &= -\frac{C_l(\mathbf{x})|^3_{cor}}{C_d(\mathbf{x})|^2_{cor}}; \end{aligned} \tag{5.19}$$

5.6 Optimization workflow

The work-flow used for the second optimization has been reported in figure 5.12. Differently from the supersonic case there is a chance of an unstable fine solution. This can occur if particular regions of the input vector domain are reached due the corrector operator; in order to prevent the optimizer to accept such configuration and so wrongfully compute the correction for the following iteration, an internal loop has been set to reduce the acceptable *PDV* until the *CFD* results are stable. For the $i - esim$ *OMM* iteration the expensive *CFD* computation can be executed up to seven times, each time the coarse optimization is carried out with a more restrictive *PDV* value. Once the stable solution is obtained, the optimization loop regularly proceeds with the corrector computation. For the next *OMM* iteration, the coarse optimization will again try to enforce the *Valarezo PDV*; in the occurrence of a stable solution with *PDV* greater than the theorized one the *PDV* will be increased for the iteration to follow.

5.7 *OMM* optimization

In the following sections the results of the optimizations carried out for the two cases will be reported.

5.7.1 Max lift design condition

The max lift design optimization terminated because of the reaching of fine evaluation limit; nevertheless a significant improvement respect the *HSPM* optimum was fund. The obtained configurations are reported in figure 5.13; as the optimization did not reach convergence, the initial *HSPM* optimum, the *CFD* best result and the last recorded position are reported. The resulting flow field is showed in figures 5.20 5.21. Comparing the coarse optimum and the fine best result (tab. 5.5) shows how lift coefficient improved by almost 5%; the optimization did not constrain neither the airfoil efficiency nor its drag coefficient; as result, the latter increased by 75% due to the massive wake caused by the slat.

The optimization lasted 9 *OMM* iterations; the fine evaluation limit was reached due to the internal loop that reinforced the *PDV* inequality constrain (figure 5.14); in the eighth *OMM* iteration four additional coarse optimizations, followed by as much fine evaluation, were required to obtain a stable *CFD* solution. On the ninth one a total of 3 internal loops had to be completed; the enforcement of a stable solution in the last two iterations required more fine evaluation than all the previews combined (tab. 5.6).

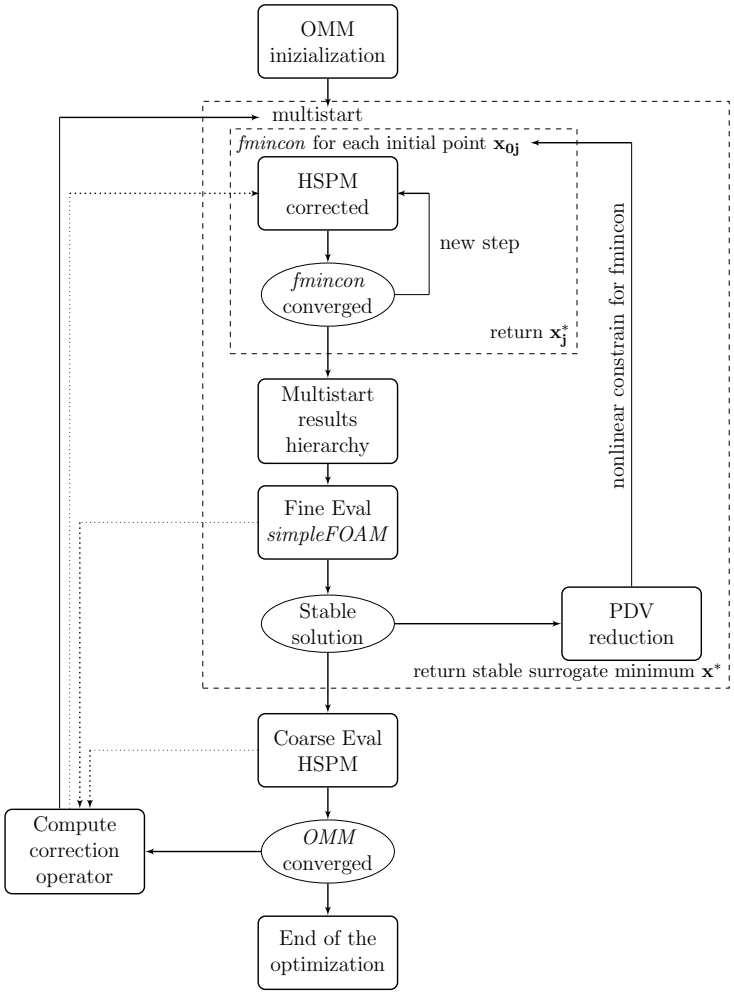


Figure 5.12: High lift configuration workflow

To prevent an excessive running time of the optimization, the break criterion was set to on the number of fine evaluations instead to the *OMM* iteration.

The objective value, showed in figure 5.15, records its minimum (maximum lift coefficient) on the seventh iteration; the maximum agreement between the fine evaluation and the corrected coarse one occurred on the sixth iteration.

The input vector components (fig 5.16) and its norm (fig 5.17) showed sign of convergence between iteration 4 and 6, but the difference in the norm (one of the convergence criteria) increased by a factor of 10 in the iteration 7.

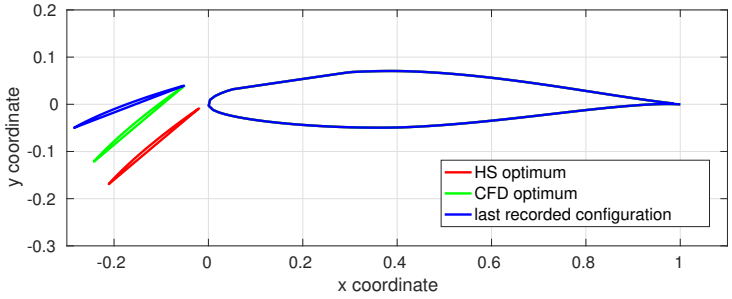


Figure 5.13: Slat positions resulting from the lift coefficient optimization

-	$\Delta x/c$	$\Delta y/c$	θ	$C_l\|_{CFD}$	$C_d\|_{CFD}$
HSPM optimum	-0.0186	-0.0076	-39.992	1.8099	0.0219
CFD optimum	-0.0500	0.0400	-20.955	1.9025	0.0917
% difference	-	-	-	+4.8688	+76.0833

Table 5.5: Max lift: Comparison between *HSPM* and *CFD* optimums

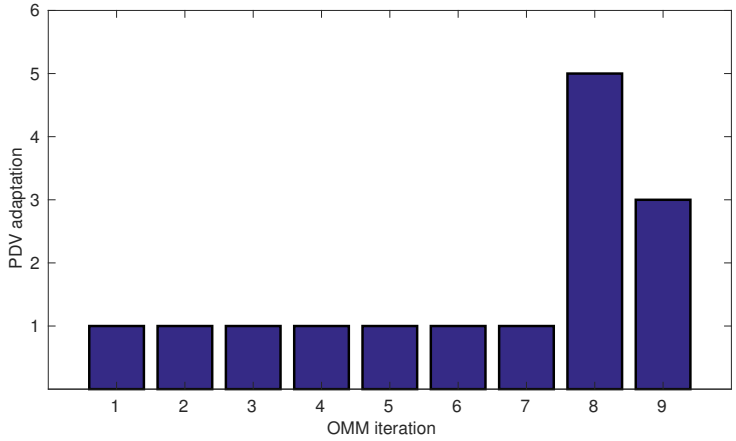


Figure 5.14: Max lift: Fine evaluation per *OMM* iteration due to *PDV* reduction loop

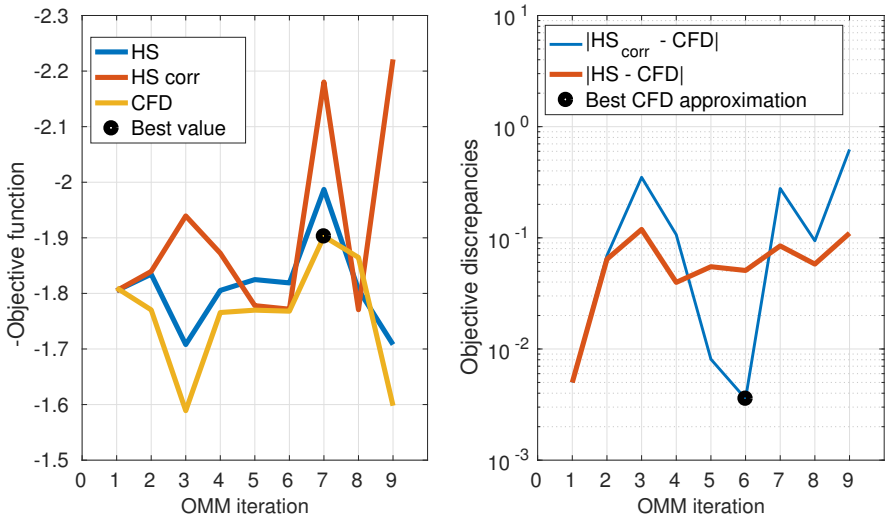


Figure 5.15: Max lift: Objective function evolution (left), error between models (right)

Total time of *OMM*	~ 43.7 [h]
Total time of coarse optimization	~ 8.3 [h]
Mean number of evaluation per coarse optimization[1]	~ 1290 ($\sim 117 \cdot 11$)
Total time of *CFD* computation	~ 35.4 [h]
Time of *CFD* stable computation[2]	~ 15.3 ($\sim 1.79 \cdot 9$) [h]
Time of *CFD* rejected computation[3]	~ 20.0 ($\sim 3.3 \cdot 6$) [h]

Table 5.6: Max lift: *OMM* recap

The nonlinear inequality progressions are reported in figure 5.18 and 5.19; in case of *PDV* correction (iteration 8 and 9), the value extracted are the ones from the last internal iteration. The *CFD* values for the slat are always satisfied while the main airfoil appears to exceed the constrain in the last two iterations. This is caused by the *PDV* correction that operates on both elements in the same way: the reduction needed to obtain stable solution on the slat is much greater than the one required for the main element causing the constrain not to be reachable on the latter.

[1]Every optimization is composed by 11 starting points that require 117 function evaluations on average

[2]A total of 9 *CFD* evaluations converged to a stable solution, their average computation time is 1.7 hours

[3]A total of 6 *CFD* evaluations did not converged to a stable solution, their average computation time is 3.6 hours

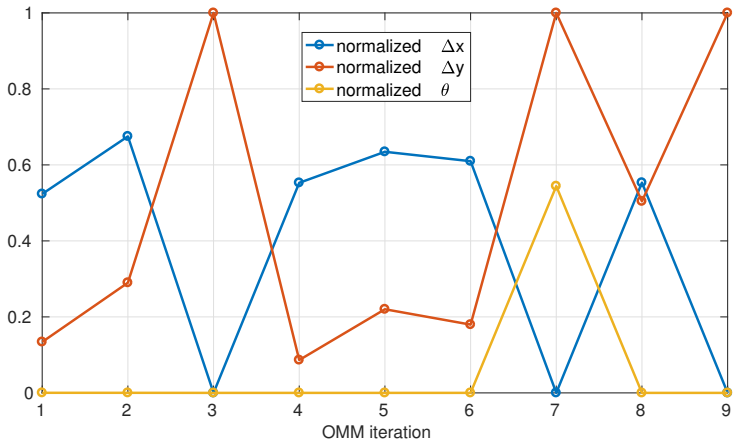


Figure 5.16: Max lift: Evolution of the input vector normalized components

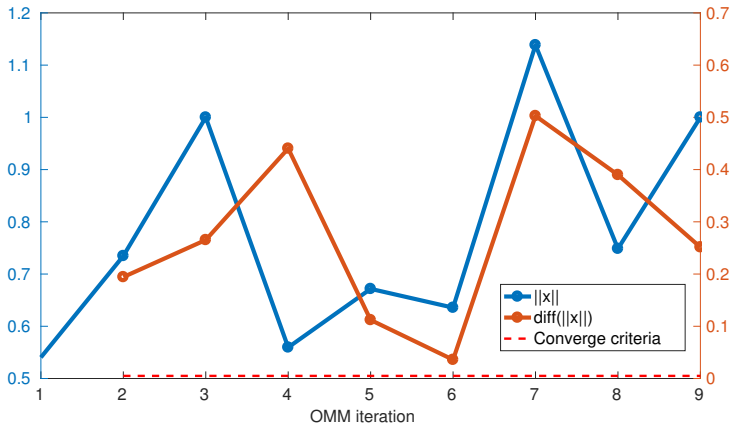


Figure 5.17: Max lift: Input vector norm evolution (left axis), difference between two iteration (right axis)

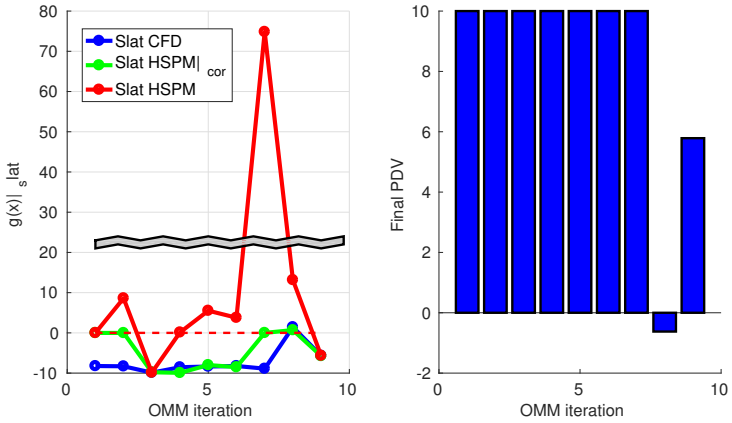


Figure 5.18: Max lift: Nonlinear inequality constrain for the slat (left); reference PDV to obtain stable solution (right)

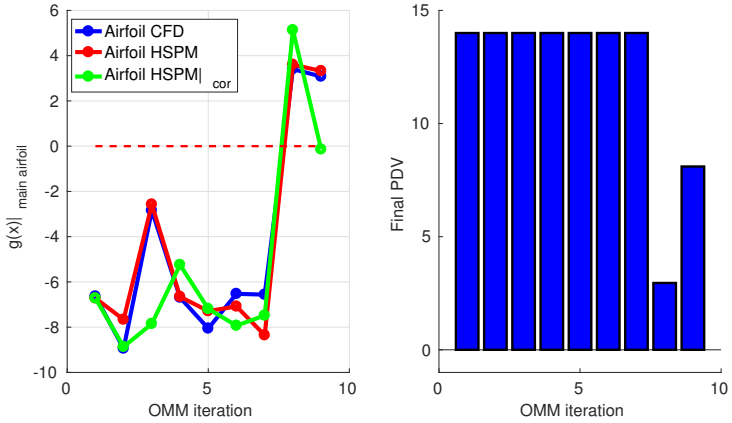


Figure 5.19: Max lift: Nonlinear inequality constrain for the airfoil (left); reference PDV to obtain stable solution (right)

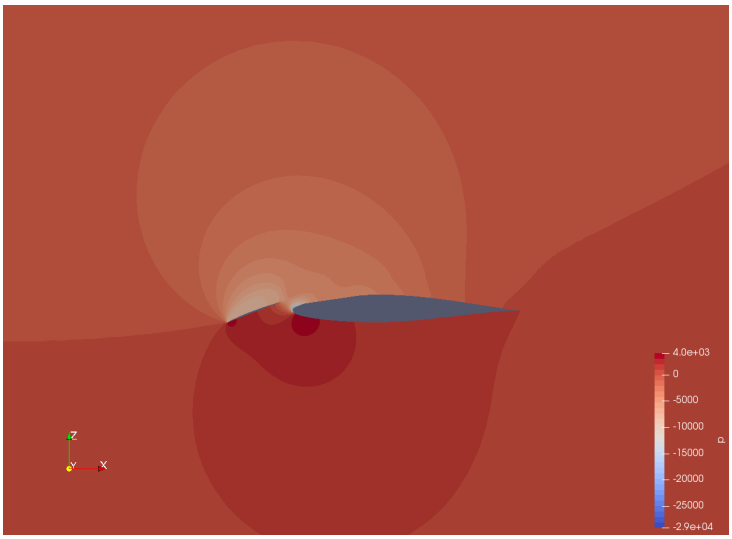


Figure 5.20: Max lift: Pressure contour

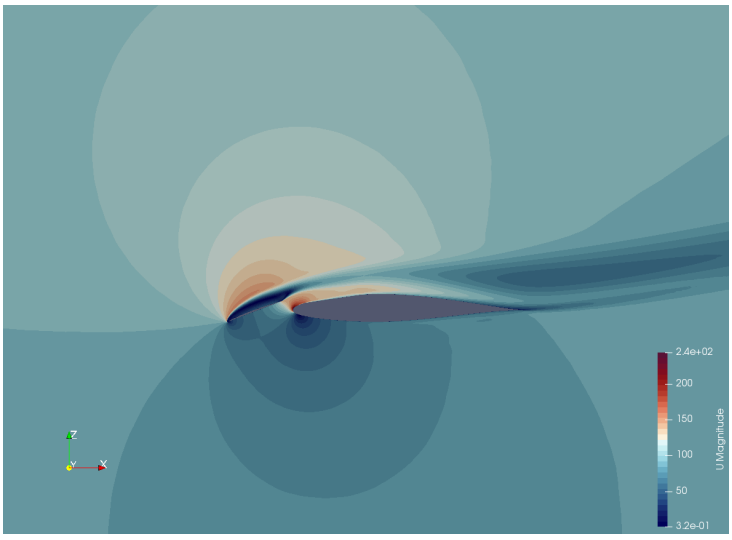


Figure 5.21: Max lift: Velocity contour

Comparison with the original airfoil

The *CFD* optimum has been tested at different angles of attack; its $C_l - \alpha$ curve is compared to the slatless airfoil one; the results are showed in figure 5.22 and tab 5.7. As for the supersonic case all coefficients are referred to the unit chord.

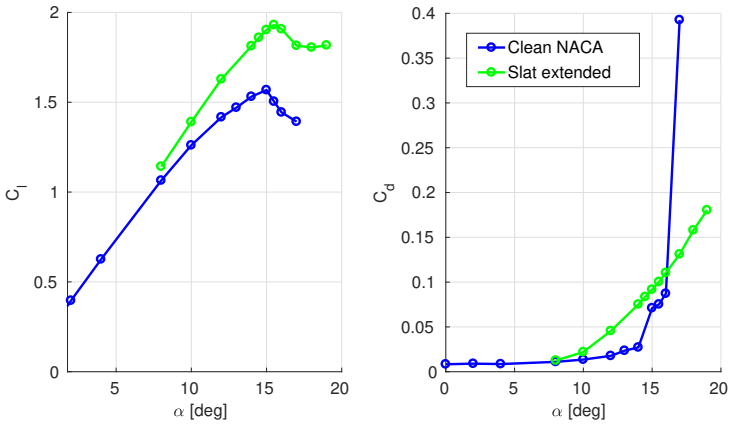


Figure 5.22: $C_l - \alpha$ comparison between slatted and clean airfoil

α	C_l	C_d	C_l	C_d
0.0	0.137	-	0.008	-
2.0	0.396	-	0.009	-
4.0	0.626	-	0.009	-
8.0	1.065	1.1420	0.011	0.013
10.0	1.261	1.3902	0.014	0.022
12.0	1.417	1.6282	0.018	0.046
14.0	1.531	1.8127	0.027	0.075
15.0	1.568	1.9025	0.071	0.092
15.5	1.504	1.9302	0.075	0.100
16.0	1.444	1.9070	0.087	0.110
17.0	1.392	1.8150	0.393	0.131
18.0	-	1.8050	-	0.158
19.0	-	1.8170	-	0.180

Table 5.7: Comparison with the clean airfoil

The lift coefficient, due to the increase of lifting surface, greatly improves, but the computed stall angle remains virtually the same. This can be due to the unusual shape of the slat that almost resembles a flat plate.

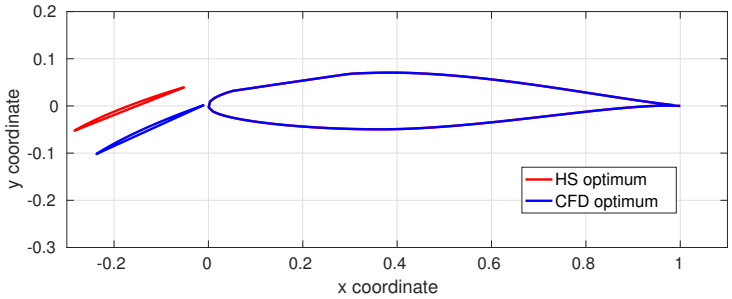


Figure 5.23: Slat positions resulting from the climb index optimization

-	$\Delta x/c$	$\Delta y/c$	θ	$C_l\|_{CFD}$	$C_d\|_{CFD}$	climb index
Clean airfoil	-	-	-	1.064	0.0110	9957.2
HSPM optimum	-0.0500	0.0400	-21.57	1.132	0.0121	9869.4
CFD optimum	-0.0085	0.0022	-24.49	1.131	0.0119	10219.2
% difference						
CFD-Clean airfoil	-	-	-	+5.923	+7.563	+2.56
CFD-HSPM	-	-	-	-0.083	-1.883	+3.42

Table 5.8: Max climb index: Comparison between *HSPM* and *CFD* optimums

5.7.2 Max climb index design condition

The optimization converged in five iterations since the change in the norm of the input vector was lower than the chosen tolerance. The configuration obtained at the end of the optimization is reported in figure 5.23; the flow solution are showed in in figures 5.32 5.33.

The objective function ($-C_l^3/C_d^2$) is showed in figures 5.24. Due to its order of magnitude, a semilogarithmic plot has been reported. Both the best climb index and the maximum agreement are obtained on the last iteration. The evolution of the aerodynamic coefficients and their errors are reported in figure 5.25 5.26. The massive error committed by the panel method, that underestimate drag due to the lack of viscous effect, is gradually corrected and became lower than $1e-4$ by the last iteration. The overall improvement has been obtained by the reduction of the drag coefficient by almost 2% while the lift coefficient was 0.08% lower than the low-fidelity optimum.

The convergence of the input vector components' are shown in figures 5.27 5.28. The error decreases till reach convergence from iteration four; differently from the high lift condition, none of the components is near its respective bounds.

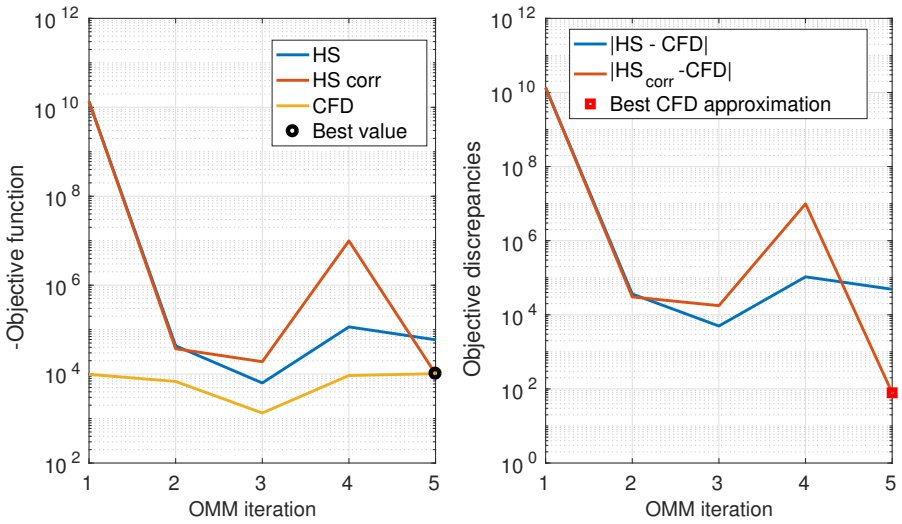


Figure 5.24: Max climb index: Objective function evolution (left), error between models (right)

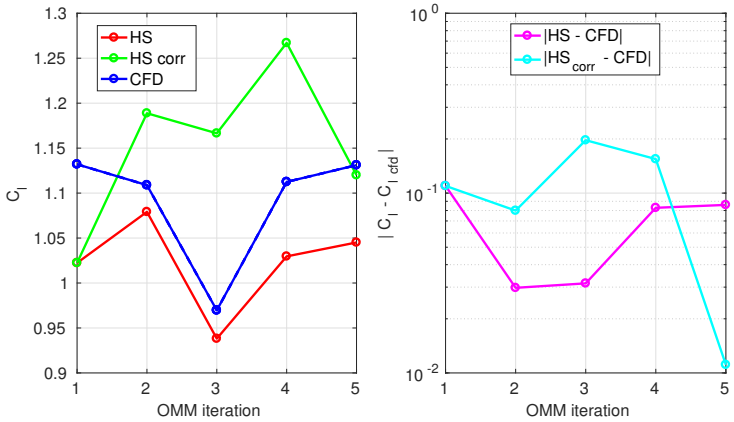


Figure 5.25: Max climb index: C_l evolution (left); error between models (right)

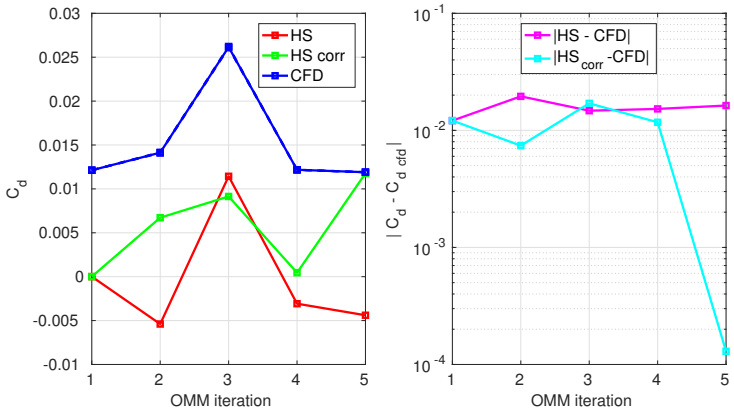


Figure 5.26: Max climb index: C_d evolution (left); error between models (right)

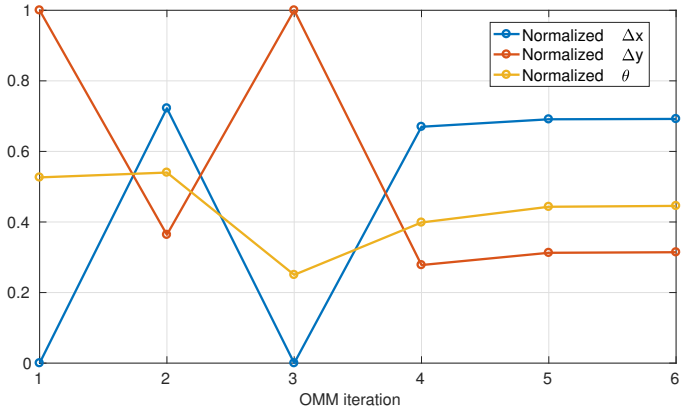


Figure 5.27: Max climb index: Evolution of the input vector normalized components

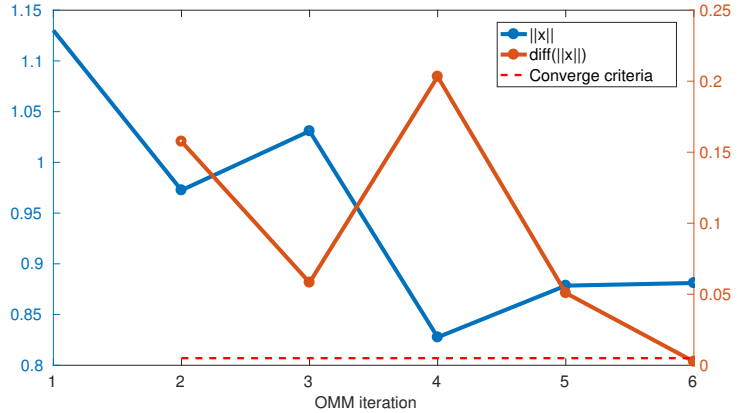


Figure 5.28: Max climb index: Input vector norm evolution (left axis), difference between two iteration (right axis)

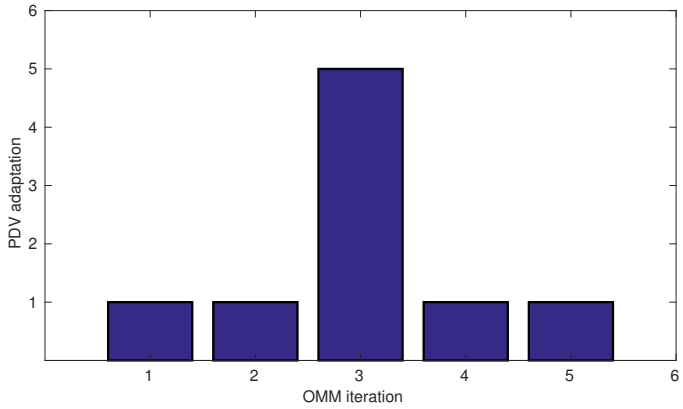


Figure 5.29: Max climb index: Fine evaluation per *OMM* iteration due to *PDV* reduction loop

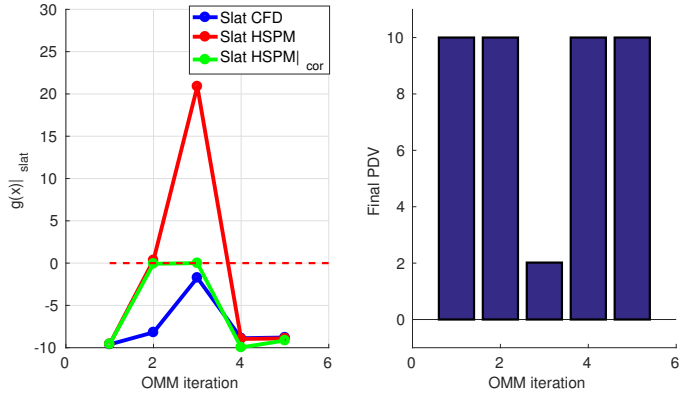


Figure 5.30: Max climb index: Nonlinear inequality constrain for the slat (left); reference *PDV* to obtain stable solution (right)

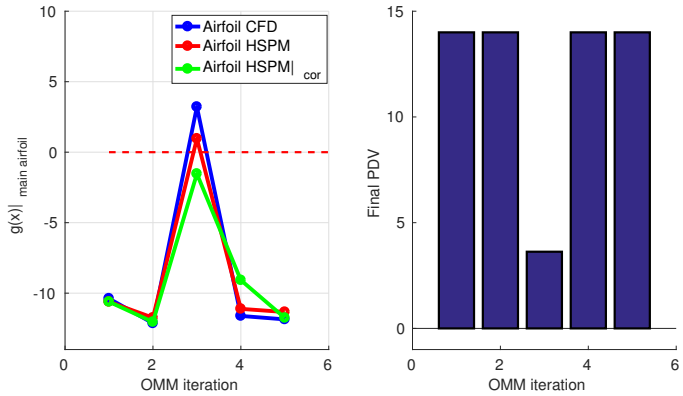


Figure 5.31: Max climb index: Nonlinear inequality constrain for the airfoil (left); reference *PDV* to obtain stable solution (right)

Total time of *OMM*	~ 35.1 [h]
Total time of coarse optimization	~ 4.5 [h]
Mean number of evaluation per coarse optimization[4]	~ 528 ($\sim 48 \cdot 11$)
Total time of *CFD* computation	~ 30.5 [h]
Time of *CFD* stable computation[5]	~ 8.5 ($\sim 1.7 \cdot 5$) [h]
Time of *CFD* rejected computation[6]	~ 21.9 ($\sim 5.4 \cdot 4$) [h]

Table 5.9: Max Climb index: *OMM* recap; optimization carried out on a dual core *Intel Core i7-7500*

Compared to the high lift condition, an instable solution is made less likely to occur due to the lower angle of attack. On iteration 3 (fig.5.29), such phenomenon was caused by an over deflected slat position obtained from the corrected coarse optimization. Two more evaluations were needed to find a stable converged solution; as for the high lift case, the *PDV* reduction for both slat and airfoil resulted in an over restrictive constrain that is not formally respect by the airfoil (figures 5.30 5.31) but allows a stable solution.

As for the previews case, the majority of the running time has been used to perform the un-converged computations (tab. 5.9); an improvement in the efficiency of the *PDV* correction together with a different *CFD* setup may be a topic for future developments in order to reduce the overall computational time.

[4]Every optimization is composed by 11 starting points that require 48 function evaluations on average

[5]A total of 5 *CFD* evaluations converged to a stable solution, their average computation time is 1.7 hours

[6]A total of 4 *CFD* evaluations did not converged to a stable solution, their average computation time is 5.4 hours

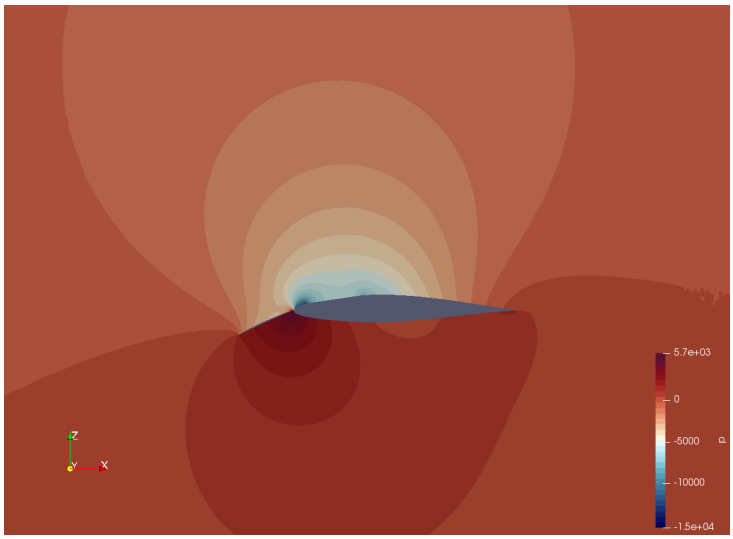


Figure 5.32: Max lift: Pressure contour

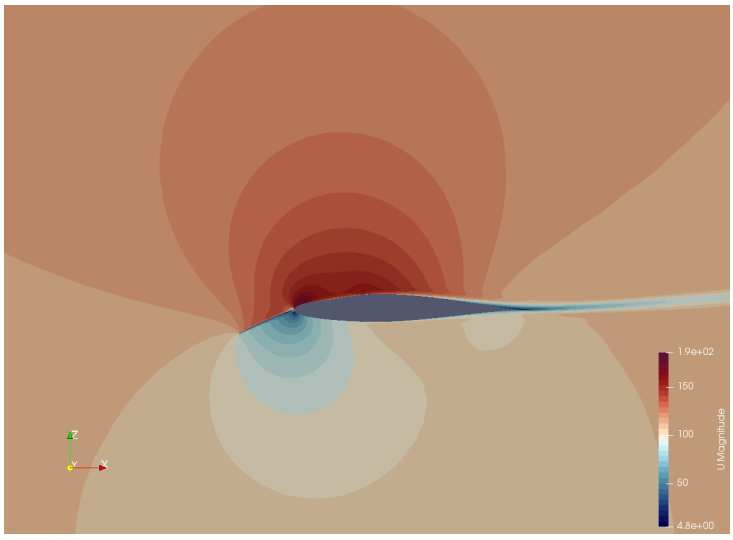


Figure 5.33: Max lift: Velocity contour

Chapter 6

Conclusion and future developments

6.1 Conclusion

This work provided a preliminary assessment of the aerodynamic performance given by the implementation of leading edge surfaces. In the prospective of a two-optimum regime *SST*, this fist test provided encouraging results for the *2D* wing section; the drag obtained by the optimized design was predictably reduced respect the baseline airfoil ($\sim -50\%$). Its value resulted comparable with a supersonic airfoil of the same thickness (appendix A). The first optimization proved that *SET* provided a valid surrogate model since the coarse model solution was improved by only a 0.6% from the first iteration. The discrepancies between fine and coarse model were reduced under evaluation precision after three *OMM* iterations. The lift constrain did not improved over the optimization but its error was already negligible; the thickness constrain was not been reached on the lower slat but it was improved over the course of the optimization. The feasible space was more demanding then expected; from a rough esteem base on the coarse optimization time, only $\sim 20\%$ of the created individual was fully evaluated.

The high lift configuration resulting from the utilization of the upper surface provided positive results; in both conditions, even if in one of the optimization was terminated due to the reaching of maximum iterations, the objective function was improved from the baseline airfoil. A considerate improvement was registered also from the surrogate optimum; the combined application of *Hess-Smith panel method* (HSPM), *Valarezo Pressure Difference Value* (PDV) as surrogate model showed encouraging results. It was

demonstrated that overcorrection of the coarse model, that can result in instable solution or stalled configuration, could be prevented with an internal loop acting on the PDV value. The lift coefficient, in the maximum lift case, consistently improved for all the tested angles of attack; even thought the sharp leading edge of the surface, the stall angle slightly improved.

6.2 Future developments

The undertaken approach of varying the wing geometry could potentially be more effective than the variable sweep configuration since it directly affects the leading edge sharpness. In order to be able to make a more precise statement, the *3D* effects, and thus the full airplane geometry, should be taken under consideration. Even then, the structural and cinematic constrain could still make this solution inefficient or even impossible.
The surrogate approach proved its value so, in a eventual development of this work, it is reasonable to keep relying on it. The *multi-resolution* approach, that generally guaranties best convergence results, was not used due to the fact that the genetic algorithm needs a great number of evaluations to reach the optimum; if the geometrical parametrization was rethought to be less demanding, then the optimizer could be chosen among a wider range of algorithms and *multi-resolution* could be performed. Viscous effect should also be considered in order to be able to correctly evaluate the boundary layer interaction with the shocks and geometrical discontinuities. A final consideration on the supersonic cruise configuration, for a practical implementation, could concern the pitching moment. The *SST*s operated in the past needed a system to contrast the shift of the aerodynamic center position between subsonic and supersonic flight; the *Concorde*, for example, used to move fuel between tanks to trim the plane. Minimize the discrepancy between subsonic and supersonic pitching moment could theoretically improve the range of the plane since a more efficient use of the fuel tanks can be archive.
For the high lift configuration, work could be done to improve the capability prediction of the surrogate approach. Implementing an integral boundary layer formulation to the panel method would reduce the gap between models. PDV proved to be a good constrain criteria for the coarse optimization, but it showed lack of robustness with the corrector operator. The loop implemented to obtain stable solution was effective but could be improved to be more efficient. Beside this computational aspects, the physics of the simulation is still a preliminary approximation of the real environment in which such device would operate; compressibility and tridimensional effects are neglected, but, most importantly, no structural nor actuation aspects are considered.

Appendix A

Supersonic airfoil comparison

The modified airfoil as expected provides a great improvement compared to the original $NACA$ 64212 in supersonic flight. The aim of this chapter is to compare the design obtained by the first optimization with an airfoil specifically designed for supersonic flight.

Figure A.1: *Lockheed F-104 Starfighter* wing (the airfoil obtained through image processing is showed in green) [1]

The tip airfoil of the bi-sonic interceptor *Lockheed F-104 Starfighter* has been chosen due its almost to straight wing design; the airfoil is biconvex and extremely thin (thickness of 3.36%). Its coordinates, which are not found in literature, are obtained thought image processing (fig. A.1) of a section photo. The computations are made over a range of Mach numbers between 1.3 and 2. The solver and the mesh settings are the same used to compare the *NACA 64212* and the slatted airfoil. Lift constrain is again enforced through the angle of attack.

[1]Photo credit: *Bill Spidle*: `https://www.flickr.com/photos/22565451@N02/24995126057/`

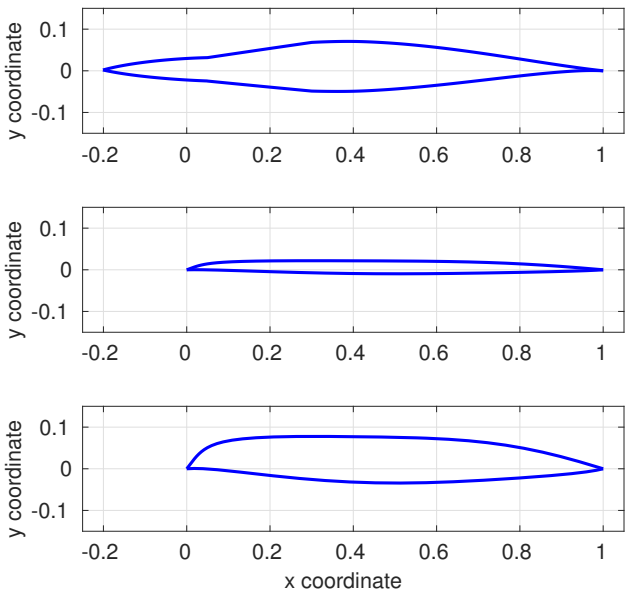


Figure A.2: Slatted airfoil (top) compared to *Lockheed F-104 Starfighter* airfoil (middle) and the stretched 12% thickness (bottom)

Three sets of data have been computed from the *Lockheed* airfoil:

- Unit chord airfoil: the normalized airfoil.
- 0.12 thickness (scaled): the normalized airfoil has been scaled to obtain the maximum thickness of 0.12; the chord, in this configuration, is 3.57.
- 0.12 thickness (stretched): the normalized airfoil has been stretched along *y direction* to obtain a thickness of 0.12; the chord is still unitary, as consequence of the stretching the leading edge angle has increased.

The drag per unit span is the quantity to be evaluated; the values are reported in figure A.3 and table A.1, where $\Delta\%$ represents the percentage difference compared to the optimized slatted configuration. The unit chord airfoil as expected has much lower drag, but it is also impractical in terms of fuel tanks allocation. The other two cases, taught to be comparable with the modified airfoil internal volume, show different trends: in the scaled, case the drag is significantly reduced for the lower Mach numbers and it become comparable in the design point. The third case, due to the greater value of the leading edge angle, shows a detached shock for all the Mach tested and thus the drag values are much higher than the modified design.

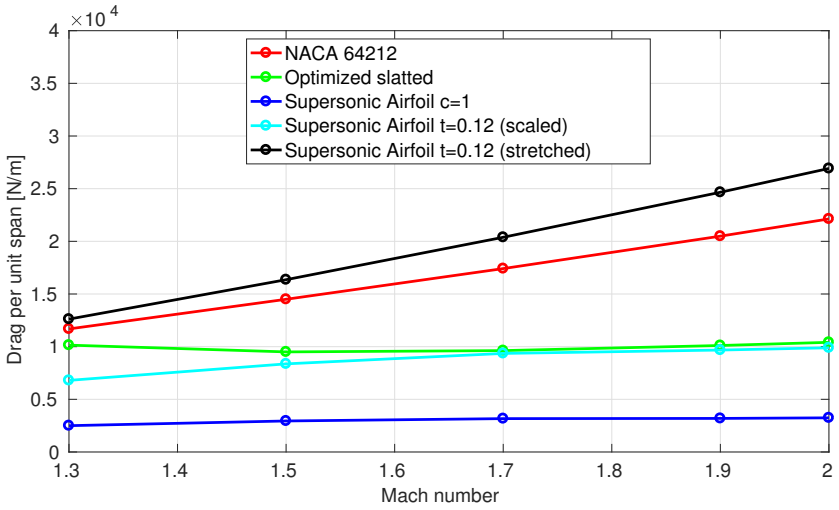


Figure A.3: Drag per unit span comparison

Mach	NACA	Slatted	Supersonic c=1		Supersonic Scaled		Supersonic Stretched	
-	N/m	N/m	N/m	Δ%	Drag/m	Δ%	Drag/m	Δ%
1.3	11684.5	10145.0	2498.6	-75.4	6798.3	-33.0	12614.4	+24.3
1.5	14497.8	9502.6	2944.6	-69.0	8366.2	-12.0	16358.5	+72.1
1.7	17412.5	9625.9	3167.4	-67.1	9357.9	-2.8	20384.9	+111.7
1.9	20491.2	10110.4	3187.7	-68.5	9678.0	-4.3	24659.6	+143.9
2.0	22129.2	10411.0	3242.1	-68.9	9893.5	-5.0	26918.1	+158.5

Table A.1: Drag per unit span comparison

Appendix B

HSPM alghoritm validation

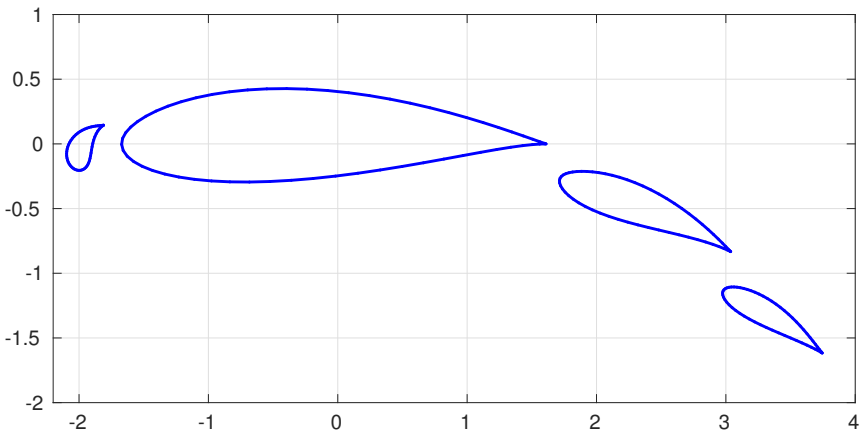


Figure B.1: *Suddhoo-Hall* four-element configuration

In order to verify the correct implementation of the method, a comparison with the analytic results obtained by *Suddhoo et al* [31] has been made. The analytical pressure distribution is obtained by transforming a set of cycles using a succession of the *Karman-Trefftz* mapping. Among the geometries available in the article, the four airfoil configuration has been chosen (figure B.1). The article gives the pressure coefficient and the coordinates of the geometries for 61 control points for each airfoil. Before proceeding with the computation, each geometry has been interpolated over 100 cosine-distributed points in order to obtain a more refined geometry.

The error assessment shown that, in this configuration, the mean difference between the analytic solution and the *HSPM* result is of the order of

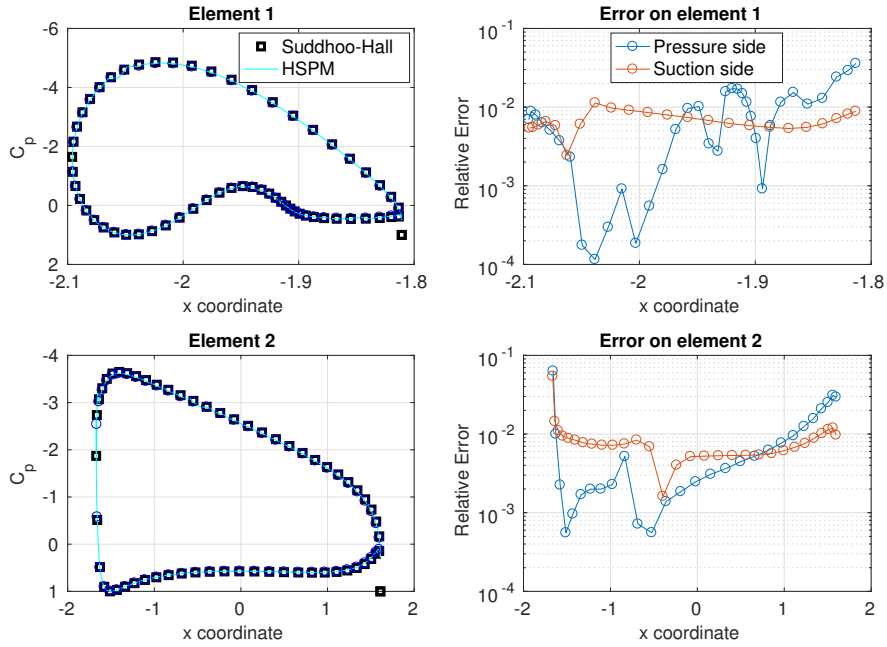


Figure B.2: *HSPM* and analytic results comparison for element 1 and 2

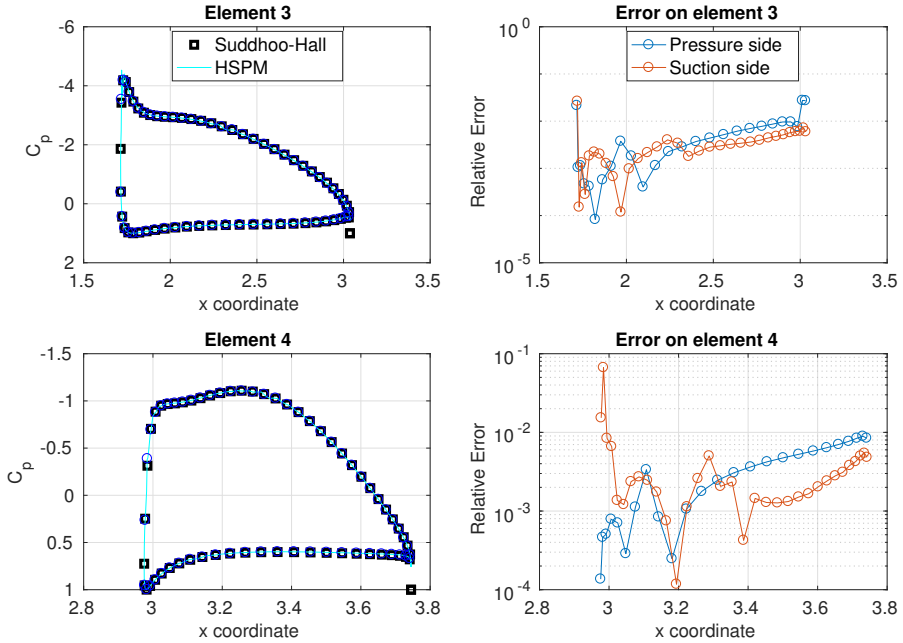


Figure B.3: *HSPM* and analytic results comparison for element 3 and 4

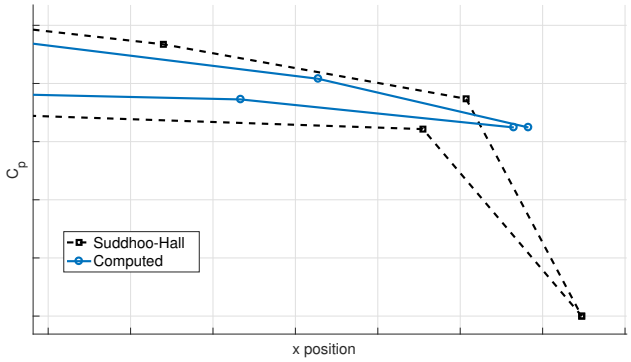


Figure B.4: Trailing edge peak error

$1e-2$. The method registered the greatest error where the pressure gradient is higher, this can be observed on the leading edge of every airfoil (figures B.2 B.3). The analytic pressure coefficient at the trailing edge cannot be correctly computed (fig. B.4) due to the limited number of panels.

The overall results can be considered satisfying, the method is correctly implemented and the pressure coefficient is computed with enough precision.

Appendix C

Pressure distribution correction test

Before the actual high lift optimization a simpler, much cheaper problem has been set in order to test the *HSPM* capability as *low-fidelity* model; to save time and computational resources *Xfoil* [34] has been used as high-fidelity model. *Xfoil* is a linear-vorticity panel method that uses source distribution both on the airfoil and on the wake to solve the potential flow equation; an integral method can be applied to treat both laminar and turbulent boundary layer.

The choice of this solver is based on the fact that it is able to add physical component to the problem, respect the simpler *HSPM*, remaining computational cheap. As inexpensive solver (especially compared to *CFD*) it serves perfectly to be use as benchmark fine model since it is possible to rapidly explore the entire space of solution and asses the global convergence of the *OMM*.

The case aim to test how the *OMM* perform to correct the pressure distribution over the airfoil. The airfoil is a *NACA 0012* at Reynolds $9 \cdot 10^6$, the objective is to find the angle of attack such that the *Valarezo* condition is met. Both the objective function (error respect PDV) and the nonlinear inequality constrain are obtained from the pressure distribution array computed by the same algorithm that will be used in the later optimization; the only variable of this problem is the angle of attack α. The coarse optimization is carried out using *fmincon*; a nonlinear constrain is implemented such that the pressure difference shall not be greater than the *Valarezo* value. At the end of the coarse optimization *Xfoil* is operated in *viscid* mode to compute the better approximated pressure difference. As the *OMM* progresses, the pressure coefficient on each panel will be corrected using the operator $\tilde{\mathbf{S}}$ (fig. C.1). At the end of each *OMM* iteration the pressure distribution is

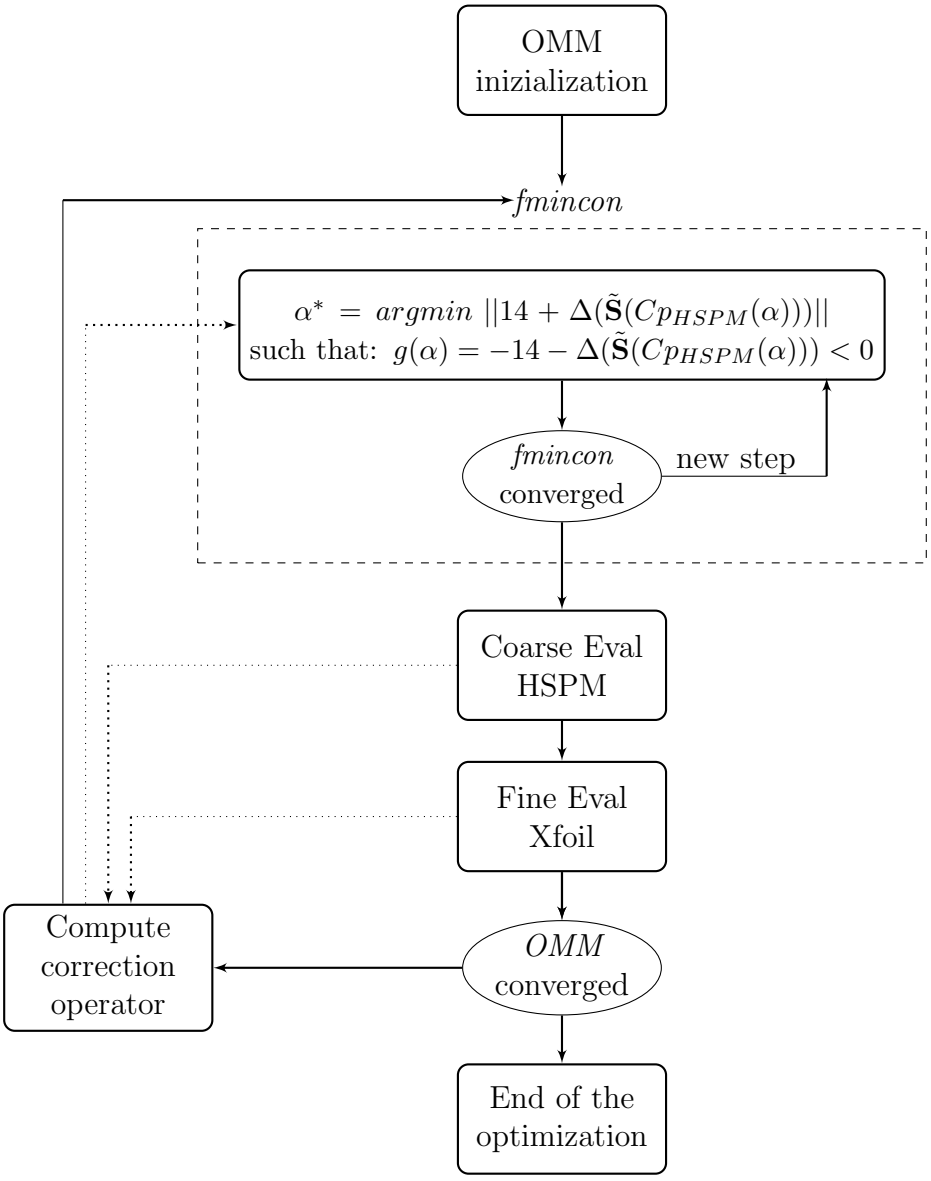


Figure C.1: Test case workflow

computed over the domain for both the corrected *Hess-Smith* and *Xfoil* (fig. C.3) to better quantify the effect of the correction.

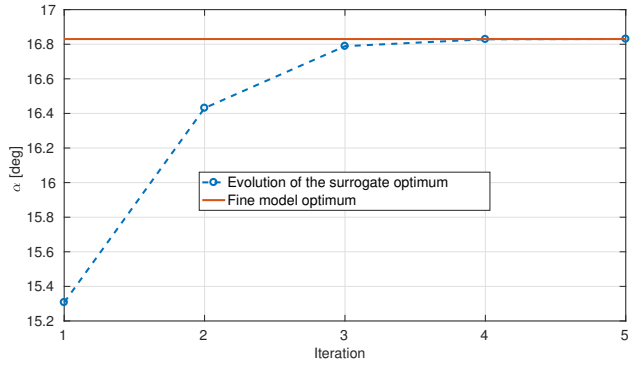


Figure C.2: Alpha convergence history

-	Iter 1	Iter 2	Iter 3	Iter 4	Iter 4	XFoil
α	15.307	16.431	16.788	16.828	16.830	16.831

Table C.1: Convergence history

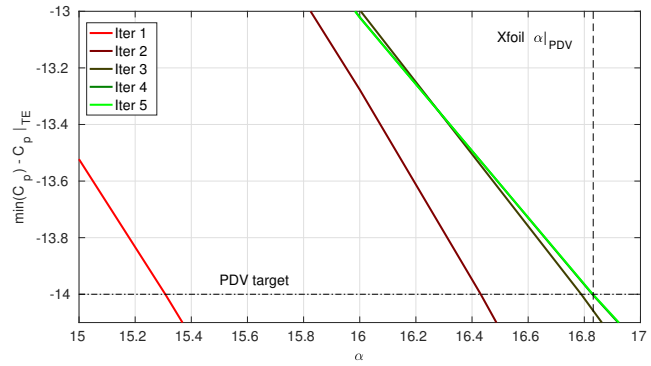


Figure C.3: Envelope of the *PDV* for each *OMM* iteration

The optimization is converged in five iteration since the change in the mean error over the panels satisfies the convergence condition. The target angle of attack has been successfully located (fig. C.2); the correction of the pressure distribution proved to be valid since the error on the last iteration is lower than 10^{-5} (fig. C.4).

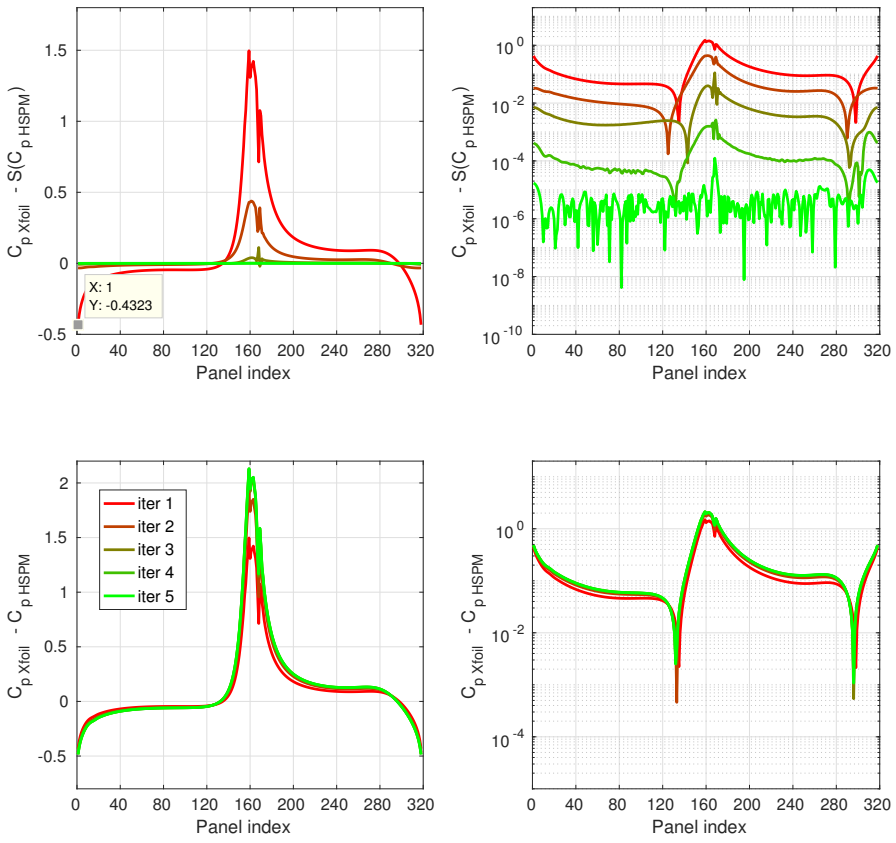


Figure C.4: Evolution of the error on each panel; the numeration starts form the trailing edge and proceed clockwise

Ringraziamenti

Desidero ringraziare il prof. Maurizio Boffadossi, relatore di questa tesi, per i preziosi suggerimenti che ha saputo fornirmi, dimostrandomi sempre grande pazienza e disponibilità durante la stesura dell'elaborato.

Una citazione doverosa va agli amici e alla mia famiglia che, nella quotidianeità, sono stati fonte di motivazione, sostegno ed incoraggiamento fondamentali per il raggiungimento di questo traguardo.

Un grazie particolare a mio nonno, che con la sua costante vicinanza, mi ha sempre stimolato a dare il meglio per perseguire i miei obiettivi.

Bibliography

[1] JW Pawlowski, DH Graham, CH Boccadoro, Peter G Coen, and Domenic J Maglieri. Origins and Overview of the Shaped Sonic Boom Demonstration Program. *AIAA paper*, (January):1–14, 2005.

[2] Yicheng Sun and Howard Smith. Review and prospect of supersonic business jet design, apr 2017.

[3] Francis S. Johnson. Ozone and SSTs. *Biological Conservation*, 4(3):220–222, apr 1972.

[4] Katherine Brown. NASA Completes Milestone Toward Quieter Supersonic X-Plane. 2017.

[5] Vladimir Karnazov. TsAGI Plans ICAO Chapter 14-compliant SSBJ | Business Aviation News: Aviation International News, 2017.

[6] Slawomir Koziel, David Echeverría Ciaurri, and Leifur Leifsson. Surrogate-Based Methods. pages 33–59. Springer, Berlin, Heidelberg, 2011.

[7] John W. Bandler, Qingsha S. Cheng, Sameh A. Dakroury, Ahmed S. Mohamed, Mohamed H. Bakr, Kaj Madsen, and Jacob Søndergaard. Space mapping: The state of the art. *IEEE Transactions on Microwave Theory and Techniques*, 52(1 II):337–361, jan 2004.

[8] D Echeverría and P W Hemker. Manifold mapping: a two-level optimization technique. *Comput Visual Sci*, 11:193–206, 2008.

[9] S. B. Pope. *Turbulent flows*. Cambridge University Press, 2000.

[10] F R Menter, M Kuntz, and R Langtry. Ten Years of Industrial Experience with the SST Turbulence Model. 2003.

[11] B.E. Launder and D.B. Spalding. The numerical computation of turbulent flows. *Computer Methods in Applied Mechanics and Engineering*, 3(2):269–289, mar 1974.

[12] Mathworks Inc. MATLAB 2016b, 2016.

[13] Thomas D. Economon, Francisco Palacios, Sean R. Copeland, Trent W. Lukaczyk, and Juan J. Alonso. SU2: An Open-Source Suite for Multiphysics Simulation and Design. *AIAA Journal*, 54(3):828–846, mar 2016.

[14] Su2devsociety.org. SU2 v5, 2017.

[15] The OpenFOAM Fondation. OpenFOAMv5.0, 2017.

[16] S. V. Patankar and D. B. Spalding. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *International Journal of Heat and Mass Transfer*, 15(10):1787–1806, oct 1972.

[17] Geuzaine Christophe and Remacle Jean-François. Gmsh 3.06, 2017.

[18] Charles L Ladson. Effects of Independent Variation of Mach and Reynolds Numbers on the Low-Speed Aerodynamic Characteristics of the NACA 0012 Airfoil Section. *NACA Technical Memorandum 4074*, 1998.

[19] Jie Ren, Leifur T. Leifsson, Slawomir Koziel, and Yonatan Tesfahunegn. Multi-Fidelity Aerodynamic Shape Optimization Using Manifold Mapping. In *57th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Reston, Virginia, jan 2016. American Institute of Aeronautics and Astronautics.

[20] Jacob Siegler, Jie Ren, Leifur Leifsson, Slawomir Koziel, and Adrian Bekasiewicz. Supersonic Airfoil Shape Optimization by Variable-fidelity Models and Manifold Mapping. *Procedia Computer Science*, 80:1103–1113, jan 2016.

[21] Kalyanmoy Deb. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2-4):311–338, 2000.

[22] Kusum Deep, Krishna Pratap Singh, M. L. Kansal, and C. Mohan. A real coded genetic algorithm for solving integer and mixed integer optimization problems. *Applied Mathematics and Computation*, 212(2):505–518, jun 2009.

[23] Roxana M Greenman. Two-Dimensional High-Lift Aerodynamic Optimization Using Neural Networks. *Optimization*, (June), jun 1998.

[24] Masahiro Kanazaki, Kentaro Tanaka, Shinkyu Jeong, and Kazuomi Yamamoto. Design Exploration Of High-Lift Airfoil Using Kriging Model and Data Mining Technique. *Eur. Conf. Computational Fluid Dynamics (ECCOMAS CFD 2006), Egmond aan Zee, The Netherlands*, pages 1–16, 2006.

[25] Leifur Leifsson, Elvar Hermannsson, and Slawomir Koziel. Optimal shape design of multi-element trawl-doors using local surrogate models. *Journal of Computational Science*, 10:55–62, sep 2015.

[26] Ingi M. Jonsson, Leifur Leifsson, Slawomir Koziel, Yonatan A. Tesfahunegn, and Adrian Bekasiewicz. Shape optimization of trawl-doors using variable-fidelity models and space mapping. In *Procedia Computer Science*, volume 51, pages 905–913. Elsevier, jan 2015.

[27] Jochen Wild, Joel Brezillon, Olivier Amoignon, Jürgen Quest, Frederic Moens, and Domenico Quagliarella. Advanced Design by Numerical Methods and Wind Tunnel Verification within the European High-Lift Program. *Journal of Aircraft*, 46(1):157–167, 2009.

[28] Ernesto Benini, Rita Ponza, and Andrea Massaro. High-Lift Multi-Element Airfoil Shape and Setting Optimization Using Multi-Objective Evolutionary Algorithms. *Journal of Aircraft*, 48(2):683–696, mar 2011.

[29] Walter Valarezo and Vincent Chin. Method for the Prediction of Wing Maximum Lift. 31(6):1419–1421, 1994.

[30] Joseph Katz and Allen Plotkin. Low Speed Aerodynamics. *McGrawhill Inc*, (February 2013):351, 1991.

[31] A. Suddhoo and I. M. Hall. Test Case for the Plane Potential Flow Past Multi-Element Aerofoils. *Aeronaut. J.*, 89(890):403–414, 1985.

[32] Richard H. Byrd, Jean Charles Gilbert, and Jorge Nocedal. A trust region method based on interior point techniques for nonlinear programming. *Mathematical Programming*, 89(1):149–185, nov 2000.

[33] A. M.O. Smith. High-Lift Aerodynamics. *Journal of Aircraft*, 12(6):501–530, jun 1975.

[34] Mark Drela. XFOIL 6.99, 2013.