Multipoint shape optimization with discrete adjoint method for the design of turbomachine blades.

H. Montanelli

April-October 2013

European Center for Research and Advanced Training in Scientific Computing
42 avenue Coriolis – 31057 Toulouse Cedex 1 – France
Tél : +33 5 61 19 31 31 – Fax : +33 5 61 19 30 00
http://www.cerfacs.fr – e-mail: secretar@cerfacs.fr

Contents

Introduction **7**

1 Shape optimization in aerodynamics **9**
1.1 Setting of the problem . . . 9
1.1.1 Mathematical setting and notations . . . 9
1.1.2 Multipoint optimization . . . 11
1.1.3 Optimization process . . . 11
1.2 Parametrization of geometric objects . . . 12
1.2.1 PARSEC airfoil geometry . . . 12
1.2.2 Free-form deformation . . . 13
1.2.3 Hicks-Henne functions . . . 13
1.2.4 CAD parametrization . . . 15
1.3 Mesh deformation . . . 15
1.4 Numerical resolution of flow equations . . . 16
1.4.1 Navier-Stokes equations . . . 16
1.4.2 Modeling of the turbulence : RANS equations . . . 19
1.4.3 Discretization of the RANS equations with the Finite Volume Method 21
1.5 Discrete adjoint method . . . 25
1.6 Optimization algorithm . . . 26

2 Application for the design of turbomachine blades : minimization of the entropy generation rate around a LS89 blade **29**
2.1 Presentation of the two-dimensional test case . . . 29
2.2 Entropy generation rate minimization without constraint on the mass flow rate and with free leading edge . . . 30
2.2.1 Setting of the single-point optimization problem . . . 30
2.2.2 Setting of the multipoint optimization problem . . . 32
2.2.3 Parametrization and gradient validation with finite differences . . . 33
2.2.4 Results of the single-point optimization . . . 34
2.2.5 Results of the multipoint optimization . . . 37
2.2.6 Comparison of the two multipoint optimizations . . . 45
2.3 Entropy generation rate minimization with constraint on the mass flow rate and with frozen leading edge . . . 46
2.3.1 Setting of the single-point optimization problem . . . 46
2.3.2 Setting of the multipoint optimization problem . . . 46
2.3.3 Parametrization and gradient validation with finite differences . . . 47

2.3.4 Results of the single-point optimization 48
2.3.5 Results of the multipoint optimization 50

Conclusion **59**

Bibliography **61**

Introduction

Numerical shape optimization in aerodynamics is a design process of geometric shapes following a given criteria of aerodynamic performance with both geometric and aerodynamic constraints. There are various methods for shape optimization, including global and local optimization algorithms.

The global optimization algorithms need huge amounts of computational resources since their cost depend exponentially on the number of design variables so they are restricted to a very small number of design variables. Genetic algorithms [32], artificial networks [29] and response surface methods [1] have been used for shape optimization in aerodynamics.

The local algorithms are used for large number of design variables in order to generate sophisticated industrial configurations. Given this large number of variables, classical methods used to compute the gradient of the objective function with respect to design variables (e.g. finite differences, complex variables or linearized methods) are obviously inefficient because they compute the gradient at a cost proportional to the number of design variables. The adjoint method does not have this drawback since it offers a way to calculate the derivatives of an objective function with respect to design variables with low time cost independent of the number of design variables.

The adjoint method was first introduced into fluid mechanics by Pironneau [30] and its first application in aerodynamic design was pioneered by Jameson [19]. Reuther et al. [33] published a few papers about the adjoint method, from drag minimization for transonic flows to noise reduction for supersonic flows. There are two variations of the adjoint method : the continuous adjoint method and the discrete adjoint method. In the continuous adjoint method, the nonlinear flow equations are linearized first with respect to design variables and then an adjoint system is derived from the linearized flow equations, followed by discretization. In the discrete adjoint method, the flow equations are discretized first, followed by the linearization and the adjoint formulation. Details about advantages and drawbacks of these two methods can be found in [25].

In recent years, shape optimization with adjoint method has been widely used in turbomachinery. Wu et al. [46] developed a continuous adjoint solvers for two-dimensional and three-dimensional blade design. Wang et al. [44] successfully employed adjoint aerodynamic optimization design to blades in multistage turbomachinery.

This master thesis focuses on the redesign of the well known two-dimensional LS89 distributor turbine blade of the Von Karman Institute using the discrete adjoint method. The entropy generation rate is considered as the objective function in the design process. The principal interest of this work is the multipoint optimization : a multipoint formulation is presented in order to minimize the entropy generation rate not only a given condition of the turbine characteristic but all over a nominal range around this condition. The turbine blade is parametrized by non-uniform B-splines through a CAD parametrization tool. The mesh is deformed thanks to a mesh deformation module. The flow is modeled by the Reynolds-averaged Navier-Stokes equations with Spalart-Allmaras turbulence model.

Chapter 1 presents the theoretical aspects of the shape optimization. The optimization problem is set and the different modules of the design process presented. Chapter 2 presents the results of single-point and multipoint optimizations performed on the LS89 blade.

Chapter 1

Shape optimization in aerodynamics

Let us begin with the theoretical aspects of the shape optimization.

1.1 Setting of the problem

A shape is described by a few parameters called design variables. Theses variables generate a surface grid and a volume grid built from and around this surface.

1.1.1 Mathematical setting and notations

The notations used for the optimization problem are as follows.

Symbol	Meaning	Comments
$\underline{\boldsymbol{\alpha}}$	Vector of design variables.	
n_α	Size of $\underline{\boldsymbol{\alpha}}$.	For our 2D applications, $n_\alpha \sim 10$.
D_α	Domain of definition of $\underline{\boldsymbol{\alpha}}$.	Bounded.
$\underline{\boldsymbol{S}}(\underline{\boldsymbol{\alpha}})$	Vector of coordinates of the surface grid.	Supposed to be a $\mathcal{C}^1$ function of $\underline{\boldsymbol{\alpha}}$.
n_S	Size of $\underline{\boldsymbol{S}}(\underline{\boldsymbol{\alpha}})$.	For our applications, $n_S \sim 10^2$.
$\underline{\boldsymbol{X}}(\underline{\boldsymbol{\alpha}})$	Vector of coordinates of the volume grid.	Supposed to be a $\mathcal{C}^1$ function of $\underline{\boldsymbol{\alpha}}$.
n_X	Size of $\underline{\boldsymbol{X}}(\underline{\boldsymbol{\alpha}})$.	For our applications, $n_X \sim 10^5$.
$\underline{\boldsymbol{W}}$	Vector of flow variables.	
n_W	Size of $\underline{\boldsymbol{W}}$.	$n_W \approx 5\, n_X$.
$F(\underline{\boldsymbol{\alpha}})$	Objective function.	Nonlinear.

Table 1.1: Notations used for the optimization problem.

The aerodynamic flow around the shape is calculated thanks to the Navier-Stokes equations. We write these equations as :

$$\underline{\boldsymbol{R}}(\underline{\boldsymbol{W}}, \underline{\boldsymbol{X}}) = 0. \tag{1.1}$$

The function $\underline{\boldsymbol{R}}$ is called the explicit numerical residual (see section **1.3**) and is supposed to be a $\mathcal{C}^1$ function from $\mathbb{R}^{n_W} \times \mathbb{R}^{n_X}$ to $\mathbb{R}^{n_W}$. The system (1.1) is a set of n_W nonlinear equations with n_W unknowns. Let us set $\underline{\boldsymbol{W}}$ to $\underline{\boldsymbol{W}}^0$, $\underline{\boldsymbol{\alpha}}$ to $\underline{\boldsymbol{\alpha}}^0$ and so then $\underline{\boldsymbol{X}}$ to $\underline{\boldsymbol{X}}^0 = \underline{\boldsymbol{X}}(\underline{\boldsymbol{\alpha}}^0)$. Under the assumption of the implicit function theorem :

$$\det\left[\frac{\partial \underline{\boldsymbol{R}}}{\partial \underline{\boldsymbol{W}}}(\underline{\boldsymbol{W}}^0, \underline{\boldsymbol{X}}^0)\right] \neq 0 \text{ and } \underline{\boldsymbol{R}} : \mathbb{R}^{n_W} \times \mathbb{R}^{n_X} \to \mathbb{R}^{n_W} \ \mathcal{C}^1, \tag{1.2}$$

the equation (1.1) defines the flow $\underline{\boldsymbol{W}}$ as a function of the grid $\underline{\boldsymbol{X}}$ around $\underline{\boldsymbol{X}}^0$ and, because $\underline{\boldsymbol{X}}$ is a $\mathcal{C}^1$ function of $\underline{\boldsymbol{\alpha}}$, it defines $\underline{\boldsymbol{W}}$ as a function of $\underline{\boldsymbol{\alpha}}$ around $\underline{\boldsymbol{\alpha}}^0$. From now, we assume that (1.2) is always true. (1.1) can be finally rewritten :

$$\underline{\boldsymbol{R}}(\underline{\boldsymbol{W}}(\underline{\boldsymbol{\alpha}}), \underline{\boldsymbol{X}}(\underline{\boldsymbol{\alpha}})) = 0. \tag{1.3}$$

The objective function F can be also rewritten $F(\underline{\boldsymbol{\alpha}}) = F(\underline{\boldsymbol{W}}(\underline{\boldsymbol{\alpha}}), \underline{\boldsymbol{X}}(\underline{\boldsymbol{\alpha}}))$ and is supposed to be a $\mathcal{C}^1$ function of $\underline{\boldsymbol{W}}$ and $\underline{\boldsymbol{X}}$. In aerodynamics it can be the drag, the isentropic ratio or, four our applications, the entropy generation rate we will define later. Our optimization problem is then :

$$\begin{array}{ll} \text{Minimize} & F(\underline{\boldsymbol{W}}(\underline{\boldsymbol{\alpha}}), \underline{\boldsymbol{X}}(\underline{\boldsymbol{\alpha}})), \\ \text{with respect to} & \underline{\boldsymbol{\alpha}}, \\ \text{subject to} & \underline{\boldsymbol{\alpha}} \in D_\alpha. \end{array} \tag{1.4}$$

The problem (1.4) is an optimization problem with bounds on the variables. We use in this thesis the L-BFGS-B algorithm (see section **1.5**). This algorithm requires the gradient of the function F with respect to design variables but does not require the second derivatives : the Hessian matrix is approximated using rank-one updates specified by gradient evaluations. The gradients are calculated with the discrete adjoint method (see section **1.6**).
More generally, the objective function F can depend on physical parameters as the incidence angle, the Mach number or, for our applications, a pressure ratio. The problem (1.4) is then a single-point design problem : the objective function is minimized at a given physical condition. For our applications, we will minimize the entropy generation rate at given pressure ratio Π. We can then write not only $F(\underline{\boldsymbol{\alpha}})$ but $F(\underline{\boldsymbol{\alpha}}, \Pi)$ and (1.4) becomes then :

$$\begin{array}{ll} \text{Minimize} & F(\underline{\boldsymbol{W}}(\underline{\boldsymbol{\alpha}}), \underline{\boldsymbol{X}}(\underline{\boldsymbol{\alpha}}), \Pi), \\ \text{for} & \Pi = \Pi_k \text{ given}, \\ \text{with respect to} & \underline{\boldsymbol{\alpha}}, \\ \text{subject to} & \underline{\boldsymbol{\alpha}} \in D_\alpha. \end{array} \tag{1.5}$$

We will present examples of such problems (1.5) in the second chapter. But what designers are expected is to minimize F over a continuous interval I_Π of conditions. This is called the multipoint optimization and this is the purpose of the next section.

1.1.2 Multipoint optimization

The multipoint optimization is more and more used for the design of aircraft [21]. We will present here only the general aspects and apply it to the design of a turbine blade.
The first step is to identify a discrete set of operating conditions $(\Pi_k)_{k\in[\![1\,;\,n_\Pi]\!]}$ and minimize the function F over this set, hoping that minimizing over this discrete set will minimize the function all over the continuous interval. The question is then : how to choose this set of operating conditions ? Li et al. [23] have shown that if the design variables vector $\underline{\boldsymbol{\alpha}}$ is of size n_α then, for a uniform sampling of I_Π, necessarily $n_\Pi > n_\alpha + 1$. Gallard et al. [15, 16] noticed that this is actually the worst case and in general $n_\Pi \ll n_\alpha$. They have developed an algorithm called GSA (*Gradient Span Analysis*) which selects from a uniform sampling of I_Π of large size N_Π a new sampling of I_Π of lower size n_Π. It is based on modified Gram-Schmidt processes which build a basis of size n_Π of the following set :

$$\mathcal{K}_{\underline{\boldsymbol{\alpha}},\,N_\Pi} = \mathrm{Span}\Big[\underline{\boldsymbol{\nabla}}_{\underline{\boldsymbol{\alpha}}} F(\underline{\boldsymbol{\alpha}},\Pi_k)\Big]_{k\in[\![1\,;\,N_\Pi]\!]}. \tag{1.6}$$

We use this algorithm in this thesis so let us assume that the minimal set of operating conditions $(\Pi_k)_{k\in[\![1\,;\,n_\Pi]\!]}$ has been founded thanks to the algorithm GSA and let us define a new function $\tilde{F}$ by :

$$\tilde{F}(\underline{\boldsymbol{\alpha}}) = \sum_{k=1}^{n_\Pi} \omega_k\, F(\underline{\boldsymbol{\alpha}},\Pi_k), \tag{1.7}$$

with weights $(\omega_k)_{k\in[\![1\,;\,n_\Pi]\!]} \in \mathbb{R}^{n_\Pi}$. We will use two different type of weights in this thesis and will present them with our results in the second chapter. The multipoint optimization problem is finally :

$$\begin{array}{ll} \text{Minimize} & \tilde{F}(\underline{\boldsymbol{W}}(\underline{\boldsymbol{\alpha}}),\underline{\boldsymbol{X}}(\underline{\boldsymbol{\alpha}})),\\ \text{with respect to} & \underline{\boldsymbol{\alpha}},\\ \text{subject to} & \underline{\boldsymbol{\alpha}} \in D_\alpha. \end{array} \tag{1.8}$$

1.1.3 Optimization process

We have worked with the ***AIRBUS*** optimization chain. That is a gradient-based approach using the discrete adjoint state of the Navier-Stokes equations (1.1) within a numerical design optimization suite. The overall optimization process depends on five dedicated modules. The parametrization module ***Padge*** generates the shape via CAD parametrization, using the current design variables. The volume mesh deformation ***VolDef*** updates the mesh thanks to the Improved Integral Method (IIM). The flow and the adjoint state are computed with the industrial CFD solver ***elsA***. The post-processing tool ***Zapp*** computes the objective function. The optimization software ***Dace*** drives the optimization process. The tabular below gives the inputs and outputs of each module.

Module	Inputs	Comments	Outputs
Padge	$\underline{\boldsymbol{\alpha}}$	Calculation of the surface deformation $\underline{\boldsymbol{\delta S}}(\underline{\boldsymbol{\alpha}})$.	$\underline{\boldsymbol{S}}(\underline{\boldsymbol{\alpha}})$, $\underline{\boldsymbol{\delta S}}(\underline{\boldsymbol{\alpha}})$
VolDef	$\underline{\boldsymbol{\delta S}}(\underline{\boldsymbol{\alpha}})$	Calculation of the volume deformation $\underline{\boldsymbol{\delta X}}(\underline{\boldsymbol{\alpha}})$.	$\underline{\boldsymbol{X}}(\underline{\boldsymbol{\alpha}})$
elsA	$\underline{\boldsymbol{X}}(\underline{\boldsymbol{\alpha}})$	Resolution of $\underline{\boldsymbol{R}}(\underline{\boldsymbol{W}}(\underline{\boldsymbol{\alpha}}), \underline{\boldsymbol{X}}(\underline{\boldsymbol{\alpha}})) = 0$.	$\underline{\boldsymbol{W}}(\underline{\boldsymbol{\alpha}})$, $\frac{\partial \underline{\boldsymbol{R}}}{\partial \underline{\boldsymbol{W}}}$, $\frac{\partial \underline{\boldsymbol{R}}}{\partial \underline{\boldsymbol{X}}}$
Zapp	$\underline{\boldsymbol{W}}(\underline{\boldsymbol{\alpha}})$		F, $\frac{\partial F}{\partial \underline{\boldsymbol{W}}}$, $\frac{\partial F}{\partial \underline{\boldsymbol{X}}}$
adjoint ***elsA***	$\frac{\partial F}{\partial \underline{\boldsymbol{W}}}$, $\frac{\partial F}{\partial \underline{\boldsymbol{X}}}$, $\frac{\partial \underline{\boldsymbol{R}}}{\partial \underline{\boldsymbol{W}}}$, $\frac{\partial \underline{\boldsymbol{R}}}{\partial \underline{\boldsymbol{X}}}$	Resolution of $\left(\frac{\partial \underline{\boldsymbol{R}}}{\partial \underline{\boldsymbol{W}}}\right)^{\mathrm{T}} \underline{\boldsymbol{\lambda}} = -\left(\frac{\partial F}{\partial \underline{\boldsymbol{W}}}\right)^{\mathrm{T}}$.	$\frac{dF}{d\underline{\boldsymbol{X}}} = \frac{\partial F}{\partial \underline{\boldsymbol{X}}} + \underline{\boldsymbol{\lambda}}^{\mathrm{T}} \frac{\partial \underline{\boldsymbol{R}}}{\partial \underline{\boldsymbol{X}}}$
adjoint ***VolDef***	$\frac{dF}{d\underline{\boldsymbol{X}}}$	Calculation of $\frac{d\underline{\boldsymbol{X}}}{d\underline{\boldsymbol{S}}}$.	$\frac{dF}{d\underline{\boldsymbol{S}}} = \frac{dF}{d\underline{\boldsymbol{X}}} \frac{d\underline{\boldsymbol{X}}}{d\underline{\boldsymbol{S}}}$
adjoint ***Padge***	$\frac{dF}{d\underline{\boldsymbol{S}}}$	Calculation of $\frac{d\underline{\boldsymbol{S}}}{d\underline{\boldsymbol{\alpha}}}$.	$\frac{dF}{d\underline{\boldsymbol{\alpha}}} = \frac{dF}{d\underline{\boldsymbol{S}}} \frac{d\underline{\boldsymbol{S}}}{d\underline{\boldsymbol{\alpha}}}$

Table 1.2: Inputs and outputs of the different modules of the optimization process.

$\frac{\partial F}{\partial \underline{\boldsymbol{W}}}$, $\frac{\partial F}{\partial \underline{\boldsymbol{X}}}$, $\frac{dF}{d\underline{\boldsymbol{X}}}$, $\frac{dF}{d\underline{\boldsymbol{S}}}$ and $\frac{dF}{d\underline{\boldsymbol{S}}}$ are vectors. $\frac{\partial \underline{\boldsymbol{R}}}{\partial \underline{\boldsymbol{W}}}$, $\frac{\partial \underline{\boldsymbol{R}}}{\partial \underline{\boldsymbol{X}}}$, $\frac{d\underline{\boldsymbol{X}}}{d\underline{\boldsymbol{S}}}$ and $\frac{d\underline{\boldsymbol{S}}}{d\underline{\boldsymbol{\alpha}}}$ are matrices.

We present in the next sections the theoretical aspects behind the modules ***Padge***, ***elsA*** and ***VolDef***.

1.2 Parametrization of geometric objects

The way to parametrize the shape to optimize (a two-dimensional surface of $\mathbb{R}^3$, typically a wing or a blade) is of course a very important aspect of the design-optimization system : the parametrization is given through the vector $\underline{\boldsymbol{\alpha}}$ called as the vector of design variables, or simply design variables. Jameson, pioneer in the area of shape optimization in aerodynamics [19], first used as design variables directly the points coordinates of the shape. It is very easy to implement but it gives huge sizes for $\underline{\boldsymbol{\alpha}}$: if the mesh of the shape is made of n_S points then $n_\alpha = 3 \times n_S$. Moreover with this method there is nothing which provides the surface to be smooth (for us the minimum acceptable regularity is a $\mathcal{C}^2$ regularity to make possible the manufacturing of the wing or the blade) so it is necessary to use a smoother.
We will present here four types of parametrization historically used in aerodynamics, the fourth one is the one we used for this thesis.

1.2.1 PARSEC airfoil geometry

The PARSEC airfoil geometry has been introduced by Sobieczky [39] and is defined by 11 parameters. Theses parameters engender a section of a wing or a blade (so a one-dimensional curve) and have been chosen because they have a big influence on airfoils aerodynamic performance : e.g. leading edge radius, upper and lower crest location including curvature there, trailing edge coordinate and thickness. The Figure 1.1 shows these parameters.

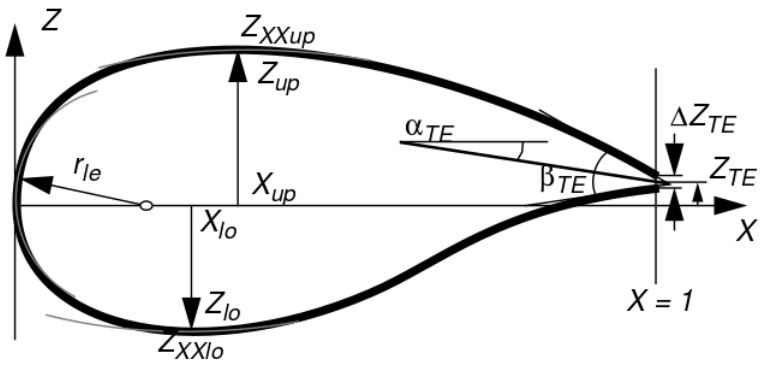

Figure 1.1: The 11 parameters of the PARSEC airfoil geometry.

Main advantages : This method reduces a lot the number of design variables and the PARSEC parameters have a direct physical sense for the designers.

Main drawbacks : The $\mathcal{C}^2$ regularity is not guaranteed and it cannot create localised deformations.

1.2.2 Free-form deformation

The free-form deformation was first described by Soderberg and Parry [40]. The idea is to enclose the wing or the blade (complex geometry) within a cube (basic geometry) and to define then the deformations of the wing via the deformations of the cube. The points of the wing are related to the cube points thanks to the Bernstein polynomials and the design variables α are then the coordinates of the cube points. This parametrization has been used by Samareh [37] and Desideri and Janka [13] for shape optimization in aerodynamics.

Main advantages : With the free-form deformation it is very easy to deform two-dimensional surfaces (even if their geometry is complex), it provides smoothness and it reduces a lot the number of design variables.

Main drawbacks : It is relatively difficult to create low and localized deformations and the parameters don't have a physical sense for the designers.

1.2.3 Hicks-Henne functions

Hicks and Henne have proposed a parametrization for wing or blade sections by adding to an initial configuration linear combinations of localized deformations called Hicks-Henne functions. These functions are defined along the normalized curvilinear abscissa of the blade and depend on two parameters A and B :

$$HH(x) = [\sin(\pi x^{\frac{\ln 0.5}{\ln A}})]^B, \qquad \forall x \in [0\,,1],\ \ \forall A \in]0\,,1[,\ \ \forall B \in [2\,,10]. \tag{1.9}$$

The figure 1.2 shows the influence of the parameters A and B.

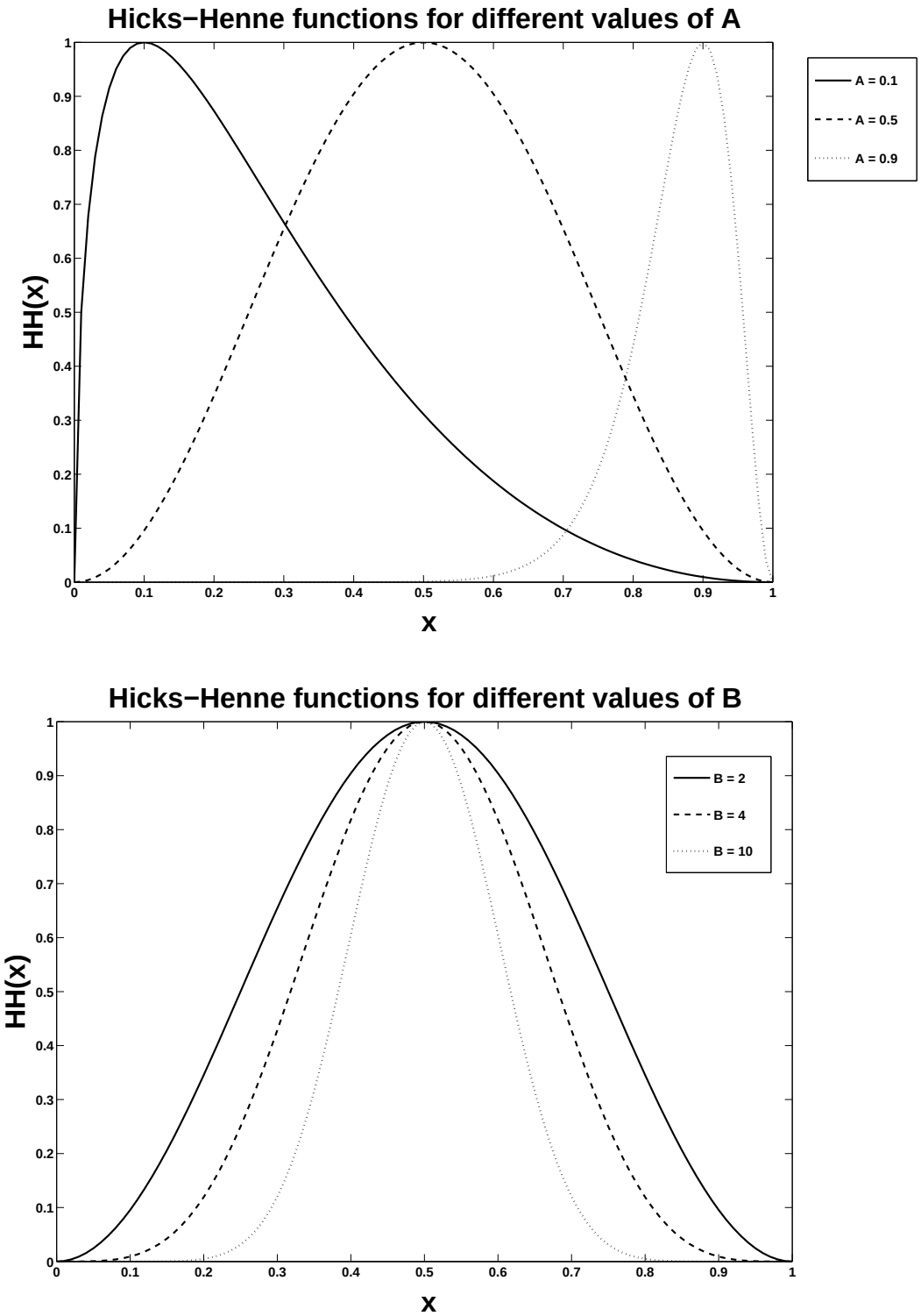

Figure 1.2: Hicks-Henne functions : influence of parameters A and B.

They can be generalized in two dimensions to deform not only a section of the wing but

the wing itself. They have been used by Laurenceau [22] and Reuther et al. [33] for shape optimization in aerodynamics.

Main advantages : It reduces a lot the number of design variables, it provides smooth surfaces and it creates localized deformations.

Main drawbacks : The parameters don't have a physical sense for the designers and it cannot deform the leading edge radius (the derivative of Hicks-Henne functions is equal to zero for $x = 0$).

1.2.4 CAD parametrization

The CAD (*Computer-Aided Design*) parametrization is the solution chosen for the optimization process we have used. It is based on the NURBS (*Non-Uniform Rational Basis Splines*) surfaces and on a PARSEC-like parametrization.
Mathematically the shape $\underline{\boldsymbol{S}}$ is a parametric surface $\underline{\boldsymbol{S}}(u, v)$ with $(u, v) \in [0, 1]^2$, driven by $(m + 1) \times (n + 1)$ control points $\underline{\boldsymbol{P}}_{i,j}$ with weights $\omega_{i,j}$, and defined by :

$$\underline{\boldsymbol{S}}(u, v) = \frac{\sum_{i=0}^{m} \sum_{j=0}^{n} N_{i,p}(u) N_{j,q}(v)\, \omega_{i,j}\, \underline{\boldsymbol{P}}_{i,j}}{\sum_{i=0}^{m} \sum_{j=0}^{n} N_{i,p}(u) N_{j,q}(v)\, \omega_{i,j}}, \quad \forall (u, v) \in [0, 1]^2. \tag{1.10}$$

where the $N_{k,r}$ are the B-spline basis functions.
In practice, we only manipulate concrete parameters like the ones used for the PARSEC parametrization (e.g. thickness, curvature, leading edge radius) and the CAD parametrization tool generates a surface from these parameters. The design variables $\underline{\boldsymbol{\alpha}}$ are therefore not the coordinates of control points $\underline{\boldsymbol{P}}_{i,j}$ and weights $\omega_{i,j}$ but are concrete design parameters.
The CAD parametrization has all the advantages we were looking for : it reduces a lot the number of design variables, it provides smooth surfaces and creates localized deformations, and the designers stay in a known design environment.

The parametrization module ***Padge*** also provides the surface deformation field $\underline{\boldsymbol{\delta S}}$ differentiating (1.10).

1.3 Mesh deformation

Within the optimization process, the volume grid $\underline{\boldsymbol{X}}$ has to be updated because of the surface deformations $\underline{\boldsymbol{\delta S}}$. Two kinds of method exist to update the mesh : automatic mesh generation and mesh deformation.
The automatic mesh generation allows important changes in the shape by generating - if it is necessary - a new topology. But this method is not always differentiable which makes it not well adapted to gradient-based approaches (a gradient-based need the mesh sensitivity with respect to the surface $d\underline{\boldsymbol{X}}/d\underline{\boldsymbol{S}}$, see Table 1.2).
The mesh deformation uses the surface deformations in order to generate a new volume

mesh maintaining the same topology. The module ***VolDef*** uses a distance-based algebraic model called the Improved Integral Method. From a surface deformation field $\underline{\boldsymbol{\delta S}}$, a volume deformation field $\underline{\boldsymbol{\delta X}}$ is generated. More precisely, the volume deformation field $\underline{\boldsymbol{\delta X}}(\underline{\boldsymbol{M}})$ at a given point $\underline{\boldsymbol{M}}$ depends on the surface deformations $\underline{\boldsymbol{\delta S}}(\underline{\boldsymbol{P}})$ of the points $\underline{\boldsymbol{P}}$ (with normals $\underline{\boldsymbol{n}}_P$) of the surface $\underline{\boldsymbol{S}}$ via the formula :

$$\begin{aligned} \underline{\boldsymbol{\delta X}}(\underline{\boldsymbol{M}}) \quad &= \frac{\displaystyle\int_{\underline{\boldsymbol{P}}\in\underline{\boldsymbol{S}}} \Big(\frac{1}{\mathrm{ca}(\underline{\boldsymbol{M}},\underline{\boldsymbol{P}})\,||\underline{\boldsymbol{PM}}||}\Big)^{\alpha} \underline{\boldsymbol{\delta S}}(\underline{\boldsymbol{P}})\, d\underline{\boldsymbol{S}}}{\displaystyle\int_{\underline{\boldsymbol{P}}\in\underline{\boldsymbol{S}}} \Big(\frac{1}{\mathrm{ca}(\underline{\boldsymbol{M}},\underline{\boldsymbol{P}})\,||\underline{\boldsymbol{PM}}||}\Big)^{\alpha} d\underline{\boldsymbol{S}}}, \\ \text{with} \quad \mathrm{ca}(\underline{\boldsymbol{M}},\underline{\boldsymbol{P}}) \quad &= \exp\Big(\frac{3}{2}\Big(1-\frac{\underline{\boldsymbol{n}}_P\underline{\boldsymbol{PM}}}{||\underline{\boldsymbol{PM}}||}\Big)^2\Big). \end{aligned} \tag{1.11}$$

The formula (1.11) is then differentiated to provide the mesh sensitivity with respect to the surface $d\underline{\boldsymbol{X}}/d\underline{\boldsymbol{S}}$.

1.4 Numerical resolution of flow equations

The third step of the optimization process is the simulation of the aerodynamic flow via the resolution of (1.1). Once the shape is defined (section **1.2**) and the mesh deformed (section **1.3**), it is necessary to calculate the aerodynamic flow around the new blade in order to evaluate the new value of the objective function. We use in this work the industrial CFD solver ***elsA*** developed by the French aerospace agency ***ONERA*** and ***CERFACS*** [9, 10, 31].

1.4.1 Navier-Stokes equations

The turbulent aerodynamic flow of a viscous, Newtonian and compressible fluid is governed by the Navier-Stokes equations. These equations are a non-linear system of five PDEs introduced by Navier and Stokes in the 19h century. They are obtained via physical conservation laws of the five unknowns : the density ρ, the momentum $\rho\underline{\boldsymbol{U}} = [\rho u, \rho v, \rho w]^{\mathrm{T}}$ and the total energy ρE of the fluid. $\rho = \rho(\underline{\boldsymbol{X}}, t)$, $\rho\underline{\boldsymbol{U}} = (\rho\underline{\boldsymbol{U}})(\underline{\boldsymbol{X}}, t)$ and $\rho E = (\rho E)(\underline{\boldsymbol{X}}, t)$ are functions of the time $t \in [0,\, T]$ and the space $\underline{\boldsymbol{X}} \in \Omega$ with a bounded domain Ω of $\mathbb{R}^3$.

Conservation laws

$$\left\{\begin{array}{lll} \dfrac{\partial\rho}{\partial t} + \nabla\cdot(\rho\underline{\boldsymbol{U}}) & = & 0, \\ \dfrac{\partial\rho\underline{\boldsymbol{U}}}{\partial t} + \underline{\boldsymbol{\nabla}}\cdot(\rho\underline{\boldsymbol{U}}\otimes\underline{\boldsymbol{U}} + p\underline{\underline{\boldsymbol{I}}} - \underline{\underline{\boldsymbol{\tau}}}) & = & 0, \\ \dfrac{\partial\rho E}{\partial t} + \nabla\cdot(\rho E\underline{\boldsymbol{U}} + p\underline{\boldsymbol{U}} - \underline{\underline{\boldsymbol{\tau}}}\,\underline{\boldsymbol{U}} + \underline{\boldsymbol{q}}) & = & 0, \end{array}\right. \tag{1.12}$$

with initial conditions $\rho(\underline{\boldsymbol{X}}, 0) = \rho_0(\underline{\boldsymbol{X}})$, $(\rho\underline{\boldsymbol{U}})(\underline{\boldsymbol{X}}, 0) = (\rho\underline{\boldsymbol{U}})_0(\underline{\boldsymbol{X}})$, $(\rho E)(\underline{\boldsymbol{X}}, 0) = (\rho E)_0(\underline{\boldsymbol{X}})$, for all $\underline{\boldsymbol{X}}$ in Ω.
Different sorts of boundary conditions can be associated to the system (1.12) and are available

in ***elsA***. For our applications, the boundary $\partial\Omega$ of the domain Ω is divided into five parts and we set $\partial\Omega = \partial\Omega_1 \cup \partial\Omega_2 \cup \partial\Omega_3 \cup \partial\Omega_4 \cup \partial\Omega_b$. We use the injection condition *inj1* for the inlet boundary $\partial\Omega_1$, the constant pressure condition *outpres* for the outlet boundary $\partial\Omega_2$, a periodic condition *periodic* for the lower and upper boundaries $\partial\Omega_3$ and $\partial\Omega_4$, and the adiabatic wall condition *adiawall* for the blade wall $\partial\Omega_b$. We give the explicit formulation of theses conditions in the paragraph **Boundary conditions** below.

State laws

In the system (1.12), the static pressure p has to be defined as a function of the five unknowns ρ, $\rho\underline{\boldsymbol{U}}$ and ρE. Let us recall first that the total energy ρE is the sum of the intern energy ρe and the kinetic energy $1/2\rho\underline{\boldsymbol{U}}^2$ which defines the intern energy ρe as a function of ρ, $\rho\underline{\boldsymbol{U}}$ and ρE :

$$\rho e = \rho E - \frac{1}{2}\frac{(\rho\underline{\boldsymbol{U}})^2}{\rho}. \tag{1.13}$$

In the CFD solver ***elsA***, the fluid is a perfect gas (i) with constant specific heat coefficients (ii) c_P and c_V. Let us note $\gamma = c_P/c_V$. For the air, $\gamma = 1.4$, $c_P = \gamma r/(\gamma - 1)$ and $c_V = r/(\gamma - 1)$ with $r = 287.053$ USI. (ii) leads to :

$$e = c_V T, \tag{1.14}$$

with T the the static temperature of the fluid. Using (1.13), T is then defined as a function of ρ, $\rho\underline{\boldsymbol{U}}$ and ρE :

$$T = \frac{(\gamma - 1)}{r}\frac{1}{\rho}\Big[\rho E - \frac{1}{2}\frac{(\rho\underline{\boldsymbol{U}})^2}{\rho}\Big]. \tag{1.15}$$

(i) leads to $p = r\rho T$ and using (1.15), p is then defined as a function of ρ, $\rho\underline{\boldsymbol{U}}$ and ρE :

$$p = (\gamma - 1)\Big[\rho E - \frac{1}{2}\frac{(\rho\underline{\boldsymbol{U}})^2}{\rho}\Big]. \tag{1.16}$$

Empirical laws

In the system (1.12), the stress tensor $\underline{\underline{\boldsymbol{\tau}}}$ and the heat flux vector $\underline{\boldsymbol{q}}$ have to be modeled. For a Newtonian fluid, $\underline{\underline{\boldsymbol{\tau}}}$ is given by the law :

$$\underline{\underline{\boldsymbol{\tau}}} = \lambda(\nabla \cdot \underline{\boldsymbol{U}})\underline{\underline{\boldsymbol{I}}} + \mu\Big[\underline{\underline{\boldsymbol{\nabla}}}\,\underline{\boldsymbol{U}} + \underline{\underline{\boldsymbol{\nabla}}}\,\underline{\boldsymbol{U}}^{\mathrm{T}}\Big], \tag{1.17}$$

with $(\lambda, \mu) \in \mathbb{R}^* \times \mathbb{R}^*$ the two coefficients of viscosity of the fluid. Classically, using the Stokes hypothesis, $\lambda = -2/3\mu$. The coefficient μ is calculated with the Sutherland law :

$$\mu(T) = \frac{\beta_s\sqrt{T}}{1 + C_s/T}. \tag{1.18}$$

For the air, $\beta_s = 1.457.10^{-6}$ USI and $C_s = 110.4$ USI. The heat flux vector is modeled through the Fourier law :

$$\underline{\boldsymbol{q}} = -k_T(T)\underline{\boldsymbol{\nabla}}T. \tag{1.19}$$

We introduce then the Prandtl number $Pr = c_P\mu/k_T$. For our applications, $Pr = 0.72$ which gives $k_T(T) = c_P\mu(T)/0.72$.

Aerodynamic Relations

We give here a list of variables used for the description of flows. Let us note U the norm of the velocity vector $\underline{\boldsymbol{U}}$ and let us consider a reference state with index ref.

Mach number	$M = \dfrac{U}{a}$
Reynolds number	$Re = \dfrac{\rho_{ref} U_{ref} L_{ref}}{\mu_{ref}}$
enthalpy	$h = e + \dfrac{p}{\rho}$
total enthalpy	$H = E + \dfrac{p}{\rho}$
total pressure	$p_t = p\,(1 + \dfrac{\gamma - 1}{2} M^2)^{\frac{\gamma}{\gamma - 1}}$
total temperature	$T_t = T\,(1 + \dfrac{\gamma - 1}{2} M^2)$
pressure coefficient	$C_p = \dfrac{p - p_{ref}}{1/2 \rho_{ref} U_{ref}^2}$
dynamic viscosity	$\nu = \dfrac{\mu}{\rho}$

Boundary conditions

As we said in the paragraph **Conservation laws**, we use the injection condition *inj1* for the inlet boundary $\partial\Omega_1$, the constant pressure condition *outpres* for the outlet boundary $\partial\Omega_2$, the periodic condition *periodic* for the lower and upper boundaries $\partial\Omega_3$ and $\partial\Omega_4$, and the adiabatic wall condition *adiawall* for the blade wall $\partial\Omega_b$.

Injection condition

For an inlet boundary of a two-dimensional subsonic flow, three conditions have to be given. Let us note $\underline{\boldsymbol{U}} = U \underline{\boldsymbol{d}}$ with $\underline{\boldsymbol{d}}$ a unit vector. The injection condition requires the direction of the flow $\underline{\boldsymbol{d}}$, the total pressure p_t and the total temperature T_t.

$$\left\{ \begin{array}{rcll} \underline{\boldsymbol{d}}(\underline{\boldsymbol{X}}, t) & = & \underline{\boldsymbol{d}}_1\,, & \forall \underline{\boldsymbol{X}} \in \partial\Omega_1,\, \forall t \in [0,\, T], \\ p_t(\underline{\boldsymbol{X}}, t) & = & p_{t1}\,, & \forall \underline{\boldsymbol{X}} \in \partial\Omega_1,\, \forall t \in [0,\, T], \\ T_t(\underline{\boldsymbol{X}}, t) & = & T_{t1}\,, & \forall \underline{\boldsymbol{X}} \in \partial\Omega_1,\, \forall t \in [0,\, T], \end{array} \right. \tag{1.20}$$

where d_1 is a given unit vector and p_{t1} and T_{t1} two given constants.

Constant pressure condition

For a subsonic flow, there is only one condition to be imposed on the outlet boundary. We use here a constant pressure condition :

$$p(\underline{\boldsymbol{X}}, t) \;=\; p_2\,, \quad \forall \underline{\boldsymbol{X}} \in \partial\Omega_2,\, \forall t \in [0,\, T], \tag{1.21}$$

where p_2 is a given constant.

Periodic condition

We use a condition of periodicity between the upper and lower boundaries : the fluxes which comes through the lower boundary $\partial\Omega_3$ are reinjected through the upper boundary $\partial\Omega_4$ and vice versa. The variables $[\rho, \rho\underline{\boldsymbol{U}}^{\mathrm{T}}, \rho E]^{\mathrm{T}}$ and the fluxes are equal on $\partial\Omega_3$ and $\partial\Omega_4$.

Adiabatic wall condition

The blade wall $\partial\Omega_b$ is supposed to be adiabatic which means there is not any heat transfer between the flow and the blade. This can be written as :

$$\left\{ \begin{array}{lcll} \underline{\boldsymbol{\nabla}}\, p(\underline{\boldsymbol{X}}, t) & = & 0\,, & \forall \underline{\boldsymbol{X}} \in \partial\Omega_b,\, \forall t \in [0,\, T], \\ \underline{\boldsymbol{U}}(\underline{\boldsymbol{X}}, t) & = & 0\,, & \forall \underline{\boldsymbol{X}} \in \partial\Omega_b,\, \forall t \in [0,\, T], \\ \underline{\boldsymbol{q}}(\underline{\boldsymbol{X}}, t) & = & 0\,, & \forall \underline{\boldsymbol{X}} \in \partial\Omega_b,\, \forall t \in [0,\, T]. \end{array} \right. \tag{1.22}$$

1.4.2 Modeling of the turbulence : RANS equations

The nonlinear part $\underline{\boldsymbol{\nabla}} \cdot (\rho \underline{\boldsymbol{U}} \otimes \underline{\boldsymbol{U}})$ of the convective flux in the Navier-Stokes equations (1.12) is responsible of the turbulence. When the Reynolds number of the flow is low, the diffusive flux $-\underline{\boldsymbol{\nabla}} \cdot \underline{\underline{\boldsymbol{\tau}}}$ is high enough compared to the convective flux to cancel the non-linearities : the flow is laminar. But from a *certain* Reynolds number (the laminar-turbulent transition is a whole subject and we do not discuss about this here), the convective flux becomes so high that it is the most dominant effect : the flow is turbulent. Turbulent flows appear as an instability of laminar flows. They involve activity over a continuous spectrum of length and time scales, are random in appearance, chaotic, irregular with a high sensibility to the details of boundary and initial conditions. They are three-dimensional, unsteady and have a rotational and dissipative nature. To calculate all the length scales, the required number of points of the grid (obtained via discretization of the domain Ω, see subsection **1.3.3**) is huge. For three-dimensional flows, this number N_{3D} is proportional to $Re^{9/4}$ where Re is the Reynolds number. For our applications, $Re = 264170$ which yields $N_{3D} \sim 10^{12}$. The turbulence is therefore not calculated but modeled via a statistically approach.

Reynolds and Favre averages

A turbulent flow can be described statistically. Reynolds [34] introduced the so-called Reynolds-averaged Navier-Stokes (RANS) equations using a decomposition of a field $g(\underline{\boldsymbol{X}}, t)$ (scalar or

vector) in a mean part $\overline{g}(\underline{\boldsymbol{X}},t)$ and a fluctuation part $g'(\underline{\boldsymbol{X}},t)$:

$$\begin{aligned} &g(\underline{\boldsymbol{X}},t) = \overline{g}(\underline{\boldsymbol{X}},t) + g'(\underline{\boldsymbol{X}},t), \\ &\text{with } \overline{g}(\underline{\boldsymbol{X}},t) = \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} g_k(\underline{\boldsymbol{X}},t) \text{ and } \overline{g'} = 0. \end{aligned} \tag{1.23}$$

This is an average over the realizations g_k of the field. In our applications, we only consider steady turbulent flows which means that the statistical average $\overline{g}$ does not depend on the time. In this case, statistical and time averages can be identified (ergodicity hypothesis, [12]) and $\overline{g}$ can be defined as :

$$\overline{g}(\underline{\boldsymbol{X}},t) = \frac{1}{T}\int_{t_0}^{t_0+T} g(\underline{\boldsymbol{X}},t)\, dt, \tag{1.24}$$

where T is greater than the characteristic period of the fluctuations g'. The Reynolds average is actually only used for incompressible flows. For compressible flows, the Favre average is preferred. The Favre average of a field $g(\underline{\boldsymbol{X}},t)$ is :

$$\tilde{g}(\underline{\boldsymbol{X}},t) = \frac{\overline{\rho(\underline{\boldsymbol{X}},t)g(\underline{\boldsymbol{X}},t)}}{\overline{\rho(\underline{\boldsymbol{X}},t)}}, \tag{1.25}$$

where $\overline{\rho g}$ and $\overline{\rho}$ are Reynolds averages. The field g can then be decomposed as :

$$\begin{aligned} &g(\underline{\boldsymbol{X}},t) = \tilde{g}(\underline{\boldsymbol{X}},t) + g''(\underline{\boldsymbol{X}},t), \\ &\text{with } \overline{\rho g''} = 0 \text{ but } \overline{g''} \neq 0. \end{aligned} \tag{1.26}$$

Reynolds-averaged Navier-Stokes equations

Using the Favre average (1.26) for the Navier-Stokes equations (1.12), we get the RANS equations (for the details of the calculation, see [4]) :

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \nabla\cdot(\rho\underline{\boldsymbol{U}}) = 0, \\ \dfrac{\partial \rho\underline{\boldsymbol{U}}}{\partial t} + \underline{\boldsymbol{\nabla}}\cdot(\rho\underline{\boldsymbol{U}}\otimes\underline{\boldsymbol{U}} + p\underline{\underline{\boldsymbol{I}}} - \underline{\underline{\boldsymbol{\tau}}} - \underline{\underline{\boldsymbol{\tau_t}}}) = 0, \\ \dfrac{\partial \rho(E+k)}{\partial t} + \nabla\cdot(\rho(E+k)\underline{\boldsymbol{U}} + p\underline{\boldsymbol{U}} - (\underline{\underline{\boldsymbol{\tau}}} + \underline{\underline{\boldsymbol{\tau_t}}})\,\underline{\boldsymbol{U}} + \underline{\boldsymbol{q}} + \underline{\boldsymbol{q_t}}) = 0, \end{cases} \tag{1.27}$$

with initial conditions $\rho(\underline{\boldsymbol{X}},0) = \rho_0(\underline{\boldsymbol{X}})$, $(\rho\underline{\boldsymbol{U}})(\underline{\boldsymbol{X}},0) = (\rho\underline{\boldsymbol{U}})_0(\underline{\boldsymbol{X}})$, $(\rho E)(\underline{\boldsymbol{X}},0) = (\rho E)_0(\underline{\boldsymbol{X}})$, for all $\underline{\boldsymbol{X}}$ in Ω. To simplify the equations (1.27) and the initial conditions, the mean flow variables don't have been overlined : ρ and p are Reynolds-averaged, $\underline{\boldsymbol{U}}$ and E are Favre-averaged. The stress tensor $\underline{\underline{\boldsymbol{\tau}}}$ and the heat flux vector $\underline{\boldsymbol{q}}$ have the same expressions as equations (1.17) and (1.19), replacing $\underline{\boldsymbol{U}}$ and T by $\tilde{\underline{\boldsymbol{U}}}$ and $\tilde{T}$. Finally, there are only three new contributions : the Reynolds stress tensor $\underline{\underline{\boldsymbol{\tau_t}}} = -\overline{\rho\underline{\boldsymbol{U}}''\otimes\underline{\boldsymbol{U}}''}$, the turbulent kinetic energy $k = 1/2\overline{\rho\underline{\boldsymbol{U}}''^2}/\overline{\rho}$ and the turbulent heat flux vector $\underline{\boldsymbol{q_t}} = c_P\,\overline{\rho T''\underline{\boldsymbol{U}}''}$.

Boussinesq's hypothesis

To close the system (1.27), the turbulence has to be modeled through the modeling of $\underline{\underline{\boldsymbol{\tau_t}}}$, k and $\underline{\boldsymbol{q_t}}$. Boussinesq introduced in 1877 the concept of turbulent viscosity μ_t. He modeled the effect of the turbulence on the mean flow as an augmentation of the viscosity and defined the Reynolds stress tensor $\underline{\underline{\boldsymbol{\tau_t}}}$ and the turbulent heat flux vector $\underline{\boldsymbol{q_t}}$ as :

$$\begin{aligned} \underline{\underline{\boldsymbol{\tau_t}}} &= -\frac{2}{3}(\rho k + \mu_t \nabla \cdot \underline{\boldsymbol{U}})\underline{\underline{\boldsymbol{I}}} + \mu_t \left[\underline{\underline{\boldsymbol{\nabla}}}\, \underline{\boldsymbol{U}} + \underline{\underline{\boldsymbol{\nabla}}}\, \underline{\boldsymbol{U}}^{\mathrm{T}} \right], \\ \underline{\boldsymbol{q_t}} &= -\frac{c_P \mu_t}{Pr_t} \underline{\boldsymbol{\nabla}} T. \end{aligned} \tag{1.28}$$

Pr_t is the turbulent Prandtl number, constant and equal to 0.9 for our applications.
The Boussinesq's hypothesis simplifies the modeling of the turbulence, reducing the number of unknowns to two : k and μ_t.

Turbulence model

It exists a lot of different models to calculate k and μ_t. These models can be either algebraical models or models using one or two convection equations. Balwin-Lomax [3] and Michel [24] models are algebraical models, the $k-\epsilon$ model of Jones-Launder [20] and the $k-\omega$ model of Wilcox [45] are 2-equation models. We use in this thesis the 1-equation Spalart-Allmaras model [41]. This model uses a convection equation for the modified turbulent dynamic viscosity $\tilde{\nu}_t$, proportional to the turbulent dynamic viscosity ν_t far from the wall. The turbulent kinetic energy k is not modeled : for flows having a moderate level of turbulence, k can be neglected using a well chosen normalization. The effect of the turbulence is therefore only modeled through the turbulent viscosity μ_t.

Initial and boundary conditions

The boundary conditions are the same for the boundaries $\partial\Omega_1$, $\partial\Omega_2$, $\partial\Omega_3$ and $\partial\Omega_4$, and $\partial\Omega_b$: *injection*, *constant pressure*, *periodic* and *adiabatic* conditions respectively. Initial conditions of $[\rho, \rho\underline{\boldsymbol{U}}^{\mathrm{T}}, \rho E]^{\mathrm{T}}$ are also given. There is an initial condition of μ_t and boundary conditions : fixed values on $\partial\Omega_1$ and $\partial\Omega_2$, periodic condition between $\partial\Omega_3$ and $\partial\Omega_4$, and equal to 0 on $\partial\Omega_b$.

1.4.3 Discretization of the RANS equations with the Finite Volume Method

The ***elsA*** software uses the Finite Volume Method (FVM) for the discretization of the RANS equations (1.27). The three-dimensional domain Ω is a multi-block mesh with structured blocks divided into hexahedral cells $\Omega_{i,j,k}$ (a cell of a structured grid is located by three indexes i, j and k). The six faces are denoted by $\Sigma_{i+1/2,j,k}$, $\Sigma_{i-1/2,j,k}$, $\Sigma_{i,j+1/2,k}$, $\Sigma_{i,j-1/2,k}$, $\Sigma_{i,j,k+1/2}$ and $\Sigma_{i,j,k-1/2}$. The unknowns $(\rho, \rho\underline{\boldsymbol{U}}, \rho E)$ are averaged into each cell and stored in the center of the cell (cell-centred method).

Spatial discretization

Let us note :

- $\underline{\boldsymbol{W}} = [\rho, \rho\underline{\boldsymbol{U}}^{\mathrm{T}}, \rho E]^{\mathrm{T}}$ the vector of conservative variables or state vector,
- $\underline{\underline{\boldsymbol{Fc}}}(\underline{\boldsymbol{W}}) = \Big[\rho\underline{\boldsymbol{U}}, (\rho\underline{\boldsymbol{U}} \otimes \underline{\boldsymbol{U}}) + p\underline{\underline{\boldsymbol{I}}}, \rho E\underline{\boldsymbol{U}} + p\underline{\boldsymbol{U}}\Big]^{\mathrm{T}}$ the convective flux matrix,
- $\underline{\underline{\boldsymbol{Fd}}}(\underline{\boldsymbol{W}}, \underline{\underline{\boldsymbol{\nabla}}}\,\underline{\boldsymbol{W}}) = \Big[\begin{pmatrix}0\\0\\0\end{pmatrix}, -(\underline{\underline{\boldsymbol{\tau}}} + \underline{\underline{\boldsymbol{\tau_t}}}), -(\underline{\underline{\boldsymbol{\tau}}} + \underline{\underline{\boldsymbol{\tau_t}}})\,\underline{\boldsymbol{U}} + (\underline{\boldsymbol{q}} + \underline{\boldsymbol{q_t}})\Big]^{\mathrm{T}}$ the diffusive flux matrix,

and $\underline{\underline{\boldsymbol{F}}} = \underline{\underline{\boldsymbol{Fc}}} + \underline{\underline{\boldsymbol{Fd}}}$ the flux matrix. Let us integrate the RANS equations (1.27) in the cell $\Omega_{i,j,k}$ with the notations we have just introduced. We get then the integrated form of the RANS equations :

$$\frac{d}{dt}\int_{\Omega_{i,j,k}} \underline{\boldsymbol{W}}\, d\Omega + \int_{\Sigma_{i,j,k}} \Big[\underline{\underline{\boldsymbol{Fc}}}(\underline{\boldsymbol{W}}) + \underline{\underline{\boldsymbol{Fd}}}(\underline{\boldsymbol{W}}, \underline{\underline{\boldsymbol{\nabla}}}\,\underline{\boldsymbol{W}})\Big]\underline{\boldsymbol{n}}\, d\Sigma = 0, \tag{1.29}$$

with $\Sigma_{i,j,k} = \Sigma_{i+1/2,j,k} \cup \Sigma_{i-1/2,j,k} \cup \Sigma_{i,j+1/2,k} \cup \Sigma_{i,j-1/2,k} \cup \Sigma_{i,j,k+1/2} \cup \Sigma_{i,j,k-1/2}$. To simplify, we forget the indexes (i, j, k) and we get then :

$$\frac{d}{dt}\int_{\Omega} \underline{\boldsymbol{W}}\, d\Omega + \int_{\Sigma} \Big[\underline{\underline{\boldsymbol{Fc}}}(\underline{\boldsymbol{W}}) + \underline{\underline{\boldsymbol{Fd}}}(\underline{\boldsymbol{W}}, \underline{\underline{\boldsymbol{\nabla}}}\,\underline{\boldsymbol{W}})\Big]\underline{\boldsymbol{n}}\, d\Sigma = 0. \tag{1.30}$$

Let us note $\Sigma = \cup_{p=1}^{6}\Sigma_p$ and $V_\Omega = \int_\Omega d\Omega$ the volume of the cell Ω. Let us define :

$$\underline{\boldsymbol{W}}_\Omega = \frac{1}{V_\Omega}\int_\Omega \underline{\boldsymbol{W}}\, d\Omega\,, \tag{1.31}$$

the average value of $\underline{\boldsymbol{W}}$ in the cell Ω and :

$$\underline{\boldsymbol{F}}_{\Sigma_p} = \int_{\Sigma_p} \Big[\underline{\underline{\boldsymbol{Fc}}}(\underline{\boldsymbol{W}}) + \underline{\underline{\boldsymbol{Fd}}}(\underline{\boldsymbol{W}}, \underline{\underline{\boldsymbol{\nabla}}}\,\underline{\boldsymbol{W}})\Big]\underline{\boldsymbol{n}}_p\, d\Sigma\,, \tag{1.32}$$

the flux through the face Σ_p. (1.30) can be rewritten :

$$\frac{d}{dt}(V_\Omega\, \underline{\boldsymbol{W}}_\Omega) = -\sum_{p=1}^{6} \underline{\boldsymbol{F}}_{\Sigma_p}. \tag{1.33}$$

For the discretization of (1.33), we have first to choose a numerical approximation $\underline{\boldsymbol{W}}_\Omega^{app}$ of $\underline{\boldsymbol{W}}_\Omega$: $\underline{\boldsymbol{W}}_\Omega^{app}$ is simply the value of $\underline{\boldsymbol{W}}$ in the center of Ω. We have then to choose a numerical approximation of $\underline{\boldsymbol{F}}_{\Sigma_p}$. (1.33) is one more time rewritten :

$$\frac{d}{dt}(V_\Omega\, \underline{\boldsymbol{W}}_\Omega^{app}) = -\sum_{p=1}^{6} \underline{\underline{\boldsymbol{F}}}^{app}\Big(\underline{\boldsymbol{W}}_\Omega^{app}, \underline{\boldsymbol{W}}_{\Omega_p^1}^{app}, ..., \underline{\boldsymbol{W}}_{\Omega_p^{N_p}}^{app}\Big)\, \underline{\boldsymbol{n}}_{\Sigma_p}\,, \tag{1.34}$$

with $\underline{\boldsymbol{n}}_{\Sigma_p} = \int_{\Sigma_p} \underline{\boldsymbol{n}}_p\, d\Sigma$ and the cells Ω_p^i are the N_p neighbor cells of Ω used for the calculation of the flux on the interface Σ_p. For our applications the grid does not depend on the time and consequently V_Ω neither.

Equation (1.34) becomes finally :

$$\frac{d}{dt}\underline{\boldsymbol{W}}_{\Omega}^{app} = -\frac{1}{V_\Omega}\sum_{p=1}^{6}\underline{\underline{\boldsymbol{F}}}^{app}\Big(\underline{\boldsymbol{W}}_{\Omega}^{app}, \underline{\boldsymbol{W}}_{\Omega_p^1}^{app}, ..., \underline{\boldsymbol{W}}_{\Omega_p^{N_p}}^{app}\Big)\,\underline{\boldsymbol{n}}_{\Sigma_p} = -\underline{\boldsymbol{R}}\Big(\underline{\boldsymbol{W}}_{\Omega}^{app}\Big), \tag{1.35}$$

with $\underline{\boldsymbol{R}}\Big(\underline{\boldsymbol{W}}_{\Omega}^{app}\Big)$ the explicit numerical residual. We have now to choose an approximation flux matrix $\underline{\underline{\boldsymbol{F}}}^{app}$.

Treatment of the convective flux

We use in this thesis the Roe scheme [36] to approximate $\underline{\boldsymbol{Fc}}$. This scheme can be used in a upwind or centred form. The time discretization and the calculation of the gradient of the objective function with respect to design variables require the resolution of a linear system with the matrix $\partial\underline{\boldsymbol{R}}/\partial\underline{\boldsymbol{W}}$. This matrix has a better conditioning when the Roe scheme is upwind [11] so we use here the upwind Roe scheme. Let us note $\underline{\boldsymbol{Fc}}(\underline{\boldsymbol{W}}) = \underline{\underline{\boldsymbol{Fc}}}(\underline{\boldsymbol{W}})\,\underline{\boldsymbol{n}}$ the flux in the direction $\underline{\boldsymbol{n}}$ and $(\underline{\boldsymbol{W}}^l, \underline{\boldsymbol{W}}^r)$ two state vectors. Then the Roe scheme is defined as :

$$\underline{\boldsymbol{F}}_{Roe}(\underline{\boldsymbol{W}}^l, \underline{\boldsymbol{W}}^r) = \frac{1}{2}\Big[\underline{\boldsymbol{Fc}}(\underline{\boldsymbol{W}}^l) + \underline{\boldsymbol{Fc}}(\underline{\boldsymbol{W}}^r)\Big] - \frac{1}{2}|\underline{\underline{\boldsymbol{A}}}|\Big[\underline{\boldsymbol{W}}^r - \underline{\boldsymbol{W}}^l\Big], \tag{1.36}$$

with $|\underline{\underline{\boldsymbol{A}}}| = \underline{\underline{\boldsymbol{M}}}\;\underline{\underline{\boldsymbol{D}}}(|\lambda_A|)\;\underline{\underline{\boldsymbol{M}}}^{-1}$, $\underline{\underline{\boldsymbol{D}}}(|\lambda_A|)$ the diagonal matrix of the absolute values of the eigenvalues of $\underline{\underline{\boldsymbol{A}}}$ and $\underline{\underline{\boldsymbol{A}}} = \underline{\underline{\boldsymbol{A}}}(\underline{\boldsymbol{W}}^l, \underline{\boldsymbol{W}}^r)$ the Roe matrix satisfying the three conditions :

- $\underline{\underline{\boldsymbol{A}}}$ is diagonalizable with real eigenvalues,
- $\underline{\underline{\boldsymbol{A}}}\Big[\underline{\boldsymbol{W}}^r - \underline{\boldsymbol{W}}^l\Big] = \underline{\boldsymbol{Fc}}(\underline{\boldsymbol{W}}^r) - \underline{\boldsymbol{Fc}}(\underline{\boldsymbol{W}}^l)$ and,
- $\underline{\underline{\boldsymbol{A}}}(\underline{\boldsymbol{W}}, \underline{\boldsymbol{W}}) = \boldsymbol{d}_{\underline{\boldsymbol{W}}}\;\underline{\boldsymbol{Fc}}(\underline{\boldsymbol{W}})$.

The Roe flux (1.36) can be non-entropic [18]. Harten [18] introduced a correction to resolve this problem :

$$\underline{\boldsymbol{F}}_{Roe}(\underline{\boldsymbol{W}}^l, \underline{\boldsymbol{W}}^r) = \frac{1}{2}\Big[\underline{\boldsymbol{Fc}}(\underline{\boldsymbol{W}}^l) + \underline{\boldsymbol{Fc}}(\underline{\boldsymbol{W}}^r)\Big] - \frac{1}{2}\Psi(|\underline{\underline{\boldsymbol{A}}}|)\Big[\underline{\boldsymbol{W}}^r - \underline{\boldsymbol{W}}^l\Big], \tag{1.37}$$

with $\Psi(|\underline{\underline{\boldsymbol{A}}}|) = \underline{\underline{\boldsymbol{M}}}\;\underline{\underline{\boldsymbol{D}}}(\Psi(|\lambda_A|))\;\underline{\underline{\boldsymbol{M}}}^{-1}$ and :

$$\Psi(z) = \begin{cases} |z| & \text{if } |z| \geq \sigma, \\ \dfrac{z^2+\sigma^2}{2\sigma} & \text{if } |z| < \sigma, \end{cases} \tag{1.38}$$

for z in $\mathbb{C}$. In (1.38), $\sigma = \epsilon_\sigma(|u| + |v| + |w| + a)$ and ϵ_σ is a parameter ($\epsilon_\sigma = 0.05$ for our applications). The Roe scheme is a first order scheme but can be extended to second order using the MUSCL method of Van Leer [43]. Let us introduce the primitive variables vector $\underline{\boldsymbol{P}} = [\rho, \underline{\boldsymbol{U}}^{\mathrm{T}}, p]^{\mathrm{T}}$ in bijection with the conservative variables vector $\underline{\boldsymbol{W}} = [\rho, \rho\underline{\boldsymbol{U}}^{\mathrm{T}}, \rho E]^{\mathrm{T}}$. $\underline{\boldsymbol{P}}_{\Omega}^{app}$ and $\underline{\boldsymbol{W}}_{\Omega}^{app}$ are the approximations as introduced before. Let us assume that we want to calculate the Roe flux through the face $\Sigma_{i+1/2,j,k}$ (so in the i-direction) : we have to define

two states $\underline{\boldsymbol{W}}^l_{i+1/2,j,k}$ and $\underline{\boldsymbol{W}}^r_{i+1/2,j,k}$ in (1.37), or - it is equivalent - two states $\underline{\boldsymbol{P}}^l_{i+1/2,j,k}$ and $\underline{\boldsymbol{P}}^r_{i+1/2,j,k}$. These two states are defined as :

$$\begin{aligned}\underline{\boldsymbol{P}}^l_{i+1/2,j,k} &= \underline{\boldsymbol{P}}^{app}_{\Omega} + \frac{1}{2}\,\mathrm{slop}_i(i,j,k),\\ \underline{\boldsymbol{P}}^r_{i+1/2,j,k} &= \underline{\boldsymbol{P}}^{app}_{\Omega} - \frac{1}{2}\,\mathrm{slop}_i(i+1,j,k),\end{aligned} \tag{1.39}$$

with slop_i the slope in the i-direction defined as :

$$\mathrm{slop}_i(i,j,k) = \phi(\underline{\boldsymbol{P}}^{app}_{\Omega} - \underline{\boldsymbol{P}}^{app}_{i-1,j,k}, \underline{\boldsymbol{P}}^{app}_{i+1,j,k} - \underline{\boldsymbol{P}}^{app}_{\Omega}), \tag{1.40}$$

and $(\underline{\boldsymbol{P}}^{app}_{i-1,j,k}, \underline{\boldsymbol{P}}^{app}_{i-1,j,k})$ the approximations of the vector $\underline{\boldsymbol{P}}$ in the cells $\Omega_{i-1,j,k}$ and $\Omega_{i+1,j,k}$. ϕ is a slope's limiter and we use the one of Van Albada [42] :

$$\phi_{va}(a,b) = \frac{(b^2+\epsilon)a + (a^2+\epsilon)b}{a^2+b^2+\epsilon} \quad \text{with} \quad 0 < \epsilon \ll 1. \tag{1.41}$$

Finally the approximation convective flux through the face $\Sigma_{i+1/2,j,k}$ is :

$$\underline{\underline{\boldsymbol{Fc}}}^{app}\Big(\underline{\boldsymbol{W}}^{app}_{\Omega}, \underline{\boldsymbol{W}}^{app}_{\Omega_{i-1,j,k}}, \underline{\boldsymbol{W}}^{app}_{\Omega_{i+1,j,k}}\Big)\,\underline{\boldsymbol{n}}_{\Sigma_{i+1/2,j,k}} = \underline{\boldsymbol{F}}_{Roe}\Big(\underline{\boldsymbol{W}}^l_{i+1/2,j,k}, \underline{\boldsymbol{W}}^r_{i+1/2,j,k}\Big). \tag{1.42}$$

Treatment of the diffusive flux

The diffusive flux matrix $\underline{\underline{\boldsymbol{Fd}}}$ involves gradients dependant of the state vector. It involves in particular the gradient of the temperature $\underline{\boldsymbol{\nabla}}T$. Let us define a coordinate system (x,y,z) and set $\underline{\boldsymbol{\nabla}}T = [\partial T/\partial x, \partial T/\partial y, \partial T/\partial z]^{\mathrm{T}}$. The derivatives are supposed to be uniform in a cell Ω (normal $\underline{\boldsymbol{n}} = [n_x, n_y, n_z]$) and we have then via the Green formula for $\partial T/\partial x$:

$$\frac{\partial T}{\partial x} = \frac{1}{V_\Omega}\int_\Omega \frac{\partial T}{\partial x}\,d\Omega = \frac{1}{V_\Omega}\int_\Sigma T n_x\,d\Sigma. \tag{1.43}$$

The formula (1.43) has then to be discretized and the gradients stored in the center of the cell or in the center of the interfaces. See [28] to know more about the different discretization schemes available in ***elsA***.

Discretization of the Spalart-Allmaras model

The convection equation is discretized with a first order Roe scheme. Bompard gives the explicit discretization in his PHD thesis [6].

Time discretization and linearization

Let us recall the equation (1.35) :

$$\frac{d}{dt}\underline{\boldsymbol{W}}^{app}_{\Omega} = -\underline{\boldsymbol{R}}\Big(\underline{\boldsymbol{W}}^{app}_{\Omega}\Big).$$

This equation is then discretized with a first order backward Euler method :

$$\frac{\underline{\boldsymbol{W}}^k_{\Omega} - \underline{\boldsymbol{W}}^{k-1}_{\Omega}}{\Delta t^{k-1}} = -\underline{\boldsymbol{R}}\Big(\underline{\boldsymbol{W}}^k_{\Omega}\Big). \tag{1.44}$$

Two comments on (1.44). First, a implicit scheme is used because it authorises higher CFL numbers and therefore reduces a lot the number of iterations and also the CPU time to reach the convergence and second, we use a first order scheme because we only want to calculate the steady state of (1.35).
The vector $\underline{\boldsymbol{R}}\Big(\underline{\boldsymbol{W}}_\Omega^k\Big)$ is then linearized :

$$\underline{\boldsymbol{R}}\Big(\underline{\boldsymbol{W}}_\Omega^k\Big) = \underline{\boldsymbol{R}}\Big(\underline{\boldsymbol{W}}_\Omega^{k-1}\Big) + \frac{\partial \underline{\boldsymbol{R}}}{\partial \underline{\boldsymbol{W}}}\Big(\underline{\boldsymbol{W}}_\Omega^{k-1}\Big)\Delta\underline{\boldsymbol{W}}_\Omega^k, \tag{1.45}$$

with $\Delta\underline{\boldsymbol{W}}_\Omega^k = \underline{\boldsymbol{W}}_\Omega^k - \underline{\boldsymbol{W}}_\Omega^{k-1}$. (1.44) becomes finally :

$$\Big[\frac{1}{\Delta t^{k-1}}\underline{\underline{\boldsymbol{I}}} + \frac{\partial \underline{\boldsymbol{R}}}{\partial \underline{\boldsymbol{W}}}\Big(\underline{\boldsymbol{W}}_\Omega^{k-1}\Big)\Big]\Delta\underline{\boldsymbol{W}}_\Omega^k = -\underline{\boldsymbol{R}}\Big(\underline{\boldsymbol{W}}_\Omega^{k-1}\Big). \tag{1.46}$$

The dimension of the linear system (1.46) is huge. For our applications, its dimension is about 10^6 and we use a LU-SSOR relaxation method to solve it.

1.5 Discrete adjoint method

We present here the method used for the calculation of the gradient of the objective function with respect to design variables called the discrete adjoint method. Let us recall the Navier-Stokes equations (1.3) :

$$\underline{\boldsymbol{R}}(\underline{\boldsymbol{W}}(\underline{\boldsymbol{\alpha}}), \underline{\boldsymbol{X}}(\underline{\boldsymbol{\alpha}})) = 0.$$

Differentiating these equations we get :

$$\frac{\partial \underline{\boldsymbol{R}}}{\partial \underline{\boldsymbol{W}}}\frac{d\underline{\boldsymbol{W}}}{d\underline{\boldsymbol{\alpha}}} + \frac{\partial \underline{\boldsymbol{R}}}{\partial \underline{\boldsymbol{X}}}\frac{d\underline{\boldsymbol{X}}}{d\underline{\boldsymbol{\alpha}}} = 0, \tag{1.47}$$

which can be rewritten :

$$\frac{d\underline{\boldsymbol{W}}}{d\underline{\boldsymbol{\alpha}}} = -\Big(\frac{\partial \underline{\boldsymbol{R}}}{\partial \underline{\boldsymbol{W}}}\Big)^{-1}\frac{\partial \underline{\boldsymbol{R}}}{\partial \underline{\boldsymbol{X}}}\frac{d\underline{\boldsymbol{X}}}{d\underline{\boldsymbol{\alpha}}}. \tag{1.48}$$

The gradient of F with respect to $\underline{\boldsymbol{\alpha}}$, denoted by $\underline{\boldsymbol{\nabla}}_{\underline{\boldsymbol{\alpha}}}F$ or $dF/d\underline{\boldsymbol{\alpha}}$, has the following expression :

$$\underline{\boldsymbol{\nabla}}_{\underline{\boldsymbol{\alpha}}}F = \frac{\partial F}{\partial \underline{\boldsymbol{X}}}\frac{d\underline{\boldsymbol{X}}}{d\underline{\boldsymbol{\alpha}}} + \frac{\partial F}{\partial \underline{\boldsymbol{W}}}\frac{d\underline{\boldsymbol{W}}}{d\underline{\boldsymbol{\alpha}}}. \tag{1.49}$$

Using (1.48) we obtain :

$$\underline{\boldsymbol{\nabla}}_{\underline{\boldsymbol{\alpha}}}F = \frac{\partial F}{\partial \underline{\boldsymbol{X}}}\frac{d\underline{\boldsymbol{X}}}{d\underline{\boldsymbol{\alpha}}} - \frac{\partial F}{\partial \underline{\boldsymbol{W}}}\Big(\frac{\partial \underline{\boldsymbol{R}}}{\partial \underline{\boldsymbol{W}}}\Big)^{-1}\frac{\partial \underline{\boldsymbol{R}}}{\partial \underline{\boldsymbol{X}}}\frac{d\underline{\boldsymbol{X}}}{d\underline{\boldsymbol{\alpha}}}. \tag{1.50}$$

Assuming there exists a vector $\underline{\boldsymbol{\lambda}}$ such that :

$$\Big(\frac{\partial \underline{\boldsymbol{R}}}{\partial \underline{\boldsymbol{W}}}\Big)^{\mathrm{T}}\underline{\boldsymbol{\lambda}} = -\Big(\frac{\partial F}{\partial \underline{\boldsymbol{W}}}\Big)^{\mathrm{T}}, \tag{1.51}$$

The gradient of F with respect to $\underline{\boldsymbol{\alpha}}$ can be finally rewritten :

$$\underline{\boldsymbol{\nabla}}_{\underline{\boldsymbol{\alpha}}} F = \frac{\partial F}{\partial \underline{\boldsymbol{X}}} \frac{d\underline{\boldsymbol{X}}}{d\underline{\boldsymbol{\alpha}}} + \underline{\boldsymbol{\lambda}}^{\mathrm{T}} \frac{\partial \underline{\boldsymbol{R}}}{\partial \underline{\boldsymbol{X}}} \frac{d\underline{\boldsymbol{X}}}{d\underline{\boldsymbol{\alpha}}}. \tag{1.52}$$

Let us recall that $d\underline{\boldsymbol{X}}/d\underline{\boldsymbol{\alpha}}$ is given by ***Padge***, $\partial\underline{\boldsymbol{R}}/\partial\underline{\boldsymbol{X}}$ by ***elsA***, $\partial F/\partial\underline{\boldsymbol{X}}$ by ***Zapp*** and $\underline{\boldsymbol{\lambda}}$ through the iterative resolution of the equation (1.51). This equation is a linear system of huge size n_W ($\sim 10^5$). The principal interest of this method is that its cost does not depend on the number of design variables and that is why it is used for industrial configurations with hundreds of design variables.
Let us say just one word about the resolution of (1.51). We use the so-called frozen-μ_t [26] approximation. During the mesh deformation, the turbulent viscosity μ_t does not change : its values are frozen. The principal interest of this approximation is that the expression of $\partial\underline{\boldsymbol{R}}/\partial\underline{\boldsymbol{W}}$ does not depend on the choice and the complexity of the turbulence model.

1.6 Optimization algorithm

We present in this section the optimization algorithm used for the resolution of our problem. Let us recall the single-point optimization problem (1.5) :

$$\begin{array}{ll} \text{Minimize} & F(\underline{\boldsymbol{W}}(\underline{\boldsymbol{\alpha}}), \underline{\boldsymbol{X}}(\underline{\boldsymbol{\alpha}}), \Pi), \\ \text{for} & \Pi = \Pi_k \ \text{ given}, \\ \text{with respect to} & \underline{\boldsymbol{\alpha}}, \\ \text{subject to} & \underline{\boldsymbol{\alpha}} \in D_\alpha. \end{array} \tag{1.53}$$

All we will say in this section would be the same for $\tilde{F}$ instead of F for multipoint optimization problems. The L(imited memory)-BFGS-B(ound constrained) algorithm has been chosen to solve (1.5). This algorithm, first described by Byrd et al. [8] is a generalization of the L-BFGS algorithm [27] for optimization problems with bounds. These two algorithms are themselves an extension of the well known BFGS algorithm, which works for unconstrained optimization problems only, described separately in 1970 by Broyden [7], Fletcher [14], Goldfarb [17] and Shanno [38].
The function F is a non-linear function whose gradient with respect to $\underline{\boldsymbol{\alpha}}$ is available. These three algorithms don't require the second derivatives : the Hessian matrix is approximated using rank-one updates specified by gradient evaluations. Let us begin with the BFGS algorithm for unconstrained optimization problems. To simplify the notations we do not write with bold font and do not underline vectors and matrices in this section.

BFGS algorithm

Choose an initial guess α_0, an initial approximate Hessian matrix H_0 and a stopping criterion crt (0)
Evaluate $\nabla F(\alpha_0)$ (1)
Initialize $k \leftarrow 0$
while crt **false do**
  Solve $H_k p_k = -\nabla F(\alpha_k)$ to get p_k (2)
  Perform a line search to get an acceptable stepsize a_k in the direction p_k (3)
  Update $s_k \leftarrow a_k p_k$ and $\alpha_{k+1} \leftarrow \alpha_k + s_k$
  Evaluate $\nabla F(\alpha_{k+1})$ (4)
  Update $y_k \leftarrow \nabla F(\alpha_{k+1}) - \nabla F(\alpha_k)$
  Update $H_{k+1} \leftarrow H_k + \frac{y_k y_k^{\mathrm{T}}}{y_k^{\mathrm{T}} s_k} - \frac{H_k s_k s_k^{\mathrm{T}} H_k}{s_k^{\mathrm{T}} H_k s_k}$ (5)
  Update $k \leftarrow k+1$
end while

The initial approximate Hessian matrix H_0 for (0) has to be a symmetric definite positive matrix (usually a diagonal matrix with positive components). The evaluation of the gradient for (1) and (4) is given trough the resolution of an adjoint equation. The update (5) can be replaced by an update of the inverse of H_{k+1} applying the Shermann-Morrison formula :

$$H_{k+1}^{-1} = \big(I - \frac{s_k y_k^{\mathrm{T}}}{y_k^{\mathrm{T}} s_k}\big) H_k^{-1} \big(I - \frac{y_k s_k^{\mathrm{T}}}{y_k^{\mathrm{T}} s_k}\big) + \frac{s_k s_k^{\mathrm{T}}}{y_k^{\mathrm{T}} s_k}. \tag{1.54}$$

and (2) becomes then simply $p_{k+1} = -H_{k+1}^{-1} \nabla F(\alpha_{k+1})$. See [8] to know more about the line search (3).
The computational cost of one iteration of the algorithm is $\mathcal{O}(n_\alpha^2)$ since the algorithm requires only matrix-vector multiplications. Moreover it is necessary to have $n_\alpha(n_\alpha+1)/2$ storage locations at each iteration since the symmetric matrix H_k (or H_k^{-1}) has to be kept.

This is precisely the interest of the L-BFGS method which never explicitly forms or stores this matrix. Instead it stores information from the past m iterations (with $m \ll n_\alpha$) and uses only this information to implicitly do operations requiring the Hessian (or the inverse Hessian). Let us set $\rho_k = 1/(y_k^{\mathrm{T}} s_k)$ and $v_k = (I - \rho_k y_k s_k^{\mathrm{T}})$. (1.54) becomes then :

$$H_{k+1}^{-1} = v_k^{\mathrm{T}} H_k^{-1} v_k + \rho_k s_k s_k^{\mathrm{T}}. \tag{1.55}$$

Instead of storing H_k^{-1} in order to update H_{k+1}^{-1} (memory cost $n_\alpha(n_\alpha+1)/2$), Nocedal has shown in [27] that it is equivalent to store $(y_i, s_i)_{0 \le i \le k}$ and H_0^{-1} (memory cost $(2(k+1)+1)n_\alpha$ at the kth iteration if H_0^{-1} is diagonal) and update H_{k+1}^{-1} with :

$$
\begin{aligned}
H_{k+1}^{-1} = \quad & v_k^{\mathrm{T}} v_{k-1}^{\mathrm{T}} \dots v_0^{\mathrm{T}} H_0^{-1} v_0 \dots v_{k-1} v_k \\
& + v_k^{\mathrm{T}} \dots v_1^{\mathrm{T}} \rho_0 s_0 s_0^{\mathrm{T}} v_1 \dots v_k \\
& \vdots \\
& + v_k^{\mathrm{T}} v_{k-1}^{\mathrm{T}} \rho_{k-2} s_{k-2} s_{k-2}^{\mathrm{T}} v_{k-1} v_k \\
& + v_k^{\mathrm{T}} \rho_{k-1} s_{k-1} s_{k-1}^{\mathrm{T}} v_k \\
& + \rho_k s_k s_k^{\mathrm{T}}.
\end{aligned}
\tag{1.56}
$$

The idea of Nocedal is then to choose an integer m (with $m \ll n_\alpha$) and to discard the old information storing only the last m (y_i, s_i). The first $m-1$ iterations, L-BFGS and BFGS generate the same search direction : for $k+1 \leq m$ (1.56) is used. But for $k+1 > m$ a special update is used, which uses only the last m (y_i, s_i) and is given by :

$$
\begin{aligned}
H_{k+1}^{-1} = \quad & v_k^{\mathrm{T}} v_{k-1}^{\mathrm{T}} \dots v_{k-m+1}^{\mathrm{T}} H_0^{-1} v_{k-m+1} \dots v_{k-1} v_k \\
& + v_k^{\mathrm{T}} \dots v_{k-m+2}^{\mathrm{T}} \rho_{k-m+1} s_{k-m+1} s_{k-m+1}^{\mathrm{T}} v_{k-m+2} \dots v_k \\
& \vdots \\
& + v_k^{\mathrm{T}} \rho_{k-1} s_{k-1} s_{k-1}^{\mathrm{T}} v_k \\
& + \rho_k s_k s_k^{\mathrm{T}}.
\end{aligned}
\tag{1.57}
$$

Using (1.56) and (1.57), we only need to have $(2m+1)n_\alpha$ storage locations at each iteration : if $m \ll n_\alpha$, $(2m+1)n_\alpha = \mathcal{O}(n_\alpha)$ for L-BFGS which has be compared to $n_\alpha(n_\alpha+1)/2 = \mathcal{O}(n_\alpha^2)$ for BFGS.

The L-BFGS-B algorithm [8] is not presented here. It uses limited BGGS matrices to approximate the Hessian of the objective function and the gradient projection approach [5] to determine the active set of constraints.

Application for the design of turbomachine blades : minimization of the entropy generation rate around a LS89 blade

Our test case is the LS89 blade of a highly loaded transonic turbine distributor. This blade has been designed by the Von Karman Institute in the early 1990s and a lot of experimental studies have been carried out for very different physical conditions [2].

2.1 Presentation of the two-dimensional test case

We have chosen to focus on the condition MUR235 [2] which corresponds to a nominal condition of utilisation. The figure 2.1 shows the mesh used. It is a multi-block mesh with structured blocks.

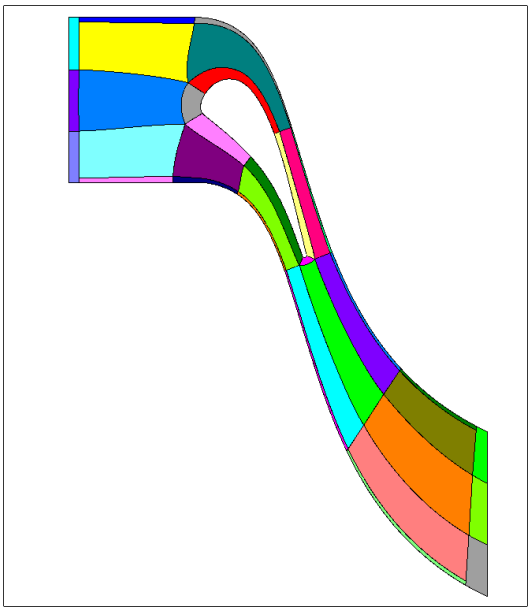

Figure 2.1: Mesh of the two-dimensional LS89 test case : 35 blocks, 153 482 cells, 140 330 nodes.

The mesh has been cut into many blocks in order to perform parallel computing on 16 processors (4 quad-core Intel Xeon Nehalem 2.66 Ghz processors).

The table 2.1 gives a few geometrical parameters describing the LS89 blade.

chord c (mm)	67.647
pitch g/c	0.850
leading edge radius r_{LE}/c	0.061
trailing edge radius r_{TE}/c	0.0105

Table 2.1: Geometrical parameters of the LS89 blade.

The table 2.2 indicates the Reynolds and Mach numbers on the inlet boundary $\partial\Omega_1$.

Re_1	264 170
M_1	0.15

Table 2.2: Reynolds and Mach numbers on the inlet boundary $\partial\Omega_1$.

The table 2.3 gives the boundary conditions on the inlet and outlet boundaries $\partial\Omega_1$ and $\partial\Omega_2$.

Inlet : injection		**Outlet : constant pressure**	
P_{t1} (Pa)	182 704.1	$\Pi = P_{s2}/P_{s1}$	0.583
T_{t1} (K)	413.3		
β_1 (deg.)	0.0		

Table 2.3: Boundary conditions.

The direction of the flow $\underline{\boldsymbol{d}}_1$ is given by the angle of injection β_1.
The MUR235 condition corresponds to a transonic flow. There is a shock wave on the upper surface of the blade and a large turbulent wake from the trailing edge. At this condition the entropy generation rate, calculated via ***elsA*** and ***Zapp*** (with 16 processors and a CPU cost $\sim$ 3840 seconds), is equal to 9.67%. The stopping criteria for ***elsA*** was the number of iterations (1500) of the CFD calculation chosen in order to get numerical residuals on density $\sim 10^{-6}$.

2.2 Entropy generation rate minimization without constraint on the mass flow rate and with free leading edge

We study in this section the entropy generation rate minimization without constraint on the mass flow rate and with free leading edge.

2.2.1 Setting of the single-point optimization problem

The aim of a turbine is to prepare the flow before arriving in the rotative part, speeding up it and deflecting it. The speed up of the flow is generated by a pressure ratio $\Pi = P_{s2}/P_{s1} < 1$ on the outlet boundary $\partial\Omega_2$. This pressure ratio Π characterizes the flow : a given pressure ratio defines a specific condition. The nominal condition MUR235 corresponds to $\Pi = 0.583$ and from now we note $\Pi = \Pi_{nom}$.
In the case of a flow without entropy generation or, and it is equivalent, without loss of total

pressure, the Mach number on $\partial\Omega_2$ is equal to the maximum Mach number which can be generated by Π, called the isentropic Mach number $M_{2,is}$. We have in this case the relation :

$$\frac{P_{s2}}{P_{s1}} = \frac{(1+\frac{\gamma-1}{2}M_1^2)^{\frac{\gamma}{\gamma-1}}}{(1+\frac{\gamma-1}{2}M_{2,is}^2)^{\frac{\gamma}{\gamma-1}}}. \tag{2.1}$$

The relation (2.1) shows that a given pressure ratio Π corresponds to a given $M_{2,is}$ and vice versa : the condition MUR235 corresponds to $M_{2,is} = 0.927$.
The goal is then, from a given ratio Π, to obtain the highest Mach number on $\partial\Omega_2$. A way to maximize M_2 is to minimize the entropy generation rate. It is classically used for the shape optimization for turbine blades [23, 44]. The entropy generation rate is characterized, for two-dimensional distributor turbine flows, by the ratio P_{t2}/P_{t1}. More precisely, minimizing the entropy generation rate is equivalent to maximizing the ratio P_{t2}/P_{t1} so the objective function used is :

$$F(\underline{\boldsymbol{\alpha}}) = 1 - \frac{P_{t2}(\underline{\boldsymbol{\alpha}})}{P_{t1}}, \tag{2.2}$$

and the single-point optimization problem at the condition MUR235 is :

$$\begin{array}{ll} \text{Minimize} & 1 - \dfrac{P_{t2}(\underline{\boldsymbol{\alpha}}, \Pi)}{P_{t1}}, \\ \text{for} & \Pi = \Pi_{nom}, \\ \text{with respect to} & \underline{\boldsymbol{\alpha}}, \\ \text{subject to} & \underline{\boldsymbol{\alpha}} \in D_\alpha. \end{array} \tag{2.3}$$

Let us recall that $\underline{\boldsymbol{\alpha}}$ is the design variables vector of size n_α and that we use a CAD parametrization. For our applications $n_\alpha = 18$ (see section **2.2.3**). P_{t2} is calculated as follows :

$$P_{t2} = \frac{1}{Q_2} \iint_{\underline{\boldsymbol{X}} \in \partial\Omega_2} P_t(\underline{\boldsymbol{X}})\, \rho(\underline{\boldsymbol{X}})\, \underline{\boldsymbol{U}}(\underline{\boldsymbol{X}}) \cdot \underline{\boldsymbol{n}}_2(\underline{\boldsymbol{X}})\, d\Sigma, \tag{2.4}$$

with Q the mass flow rate defined by :

$$Q = \iint_{\underline{\boldsymbol{X}} \in \partial\Omega_2} \rho(\underline{\boldsymbol{X}})\, \underline{\boldsymbol{U}}(\underline{\boldsymbol{X}}) \cdot \underline{\boldsymbol{n}}_2(\underline{\boldsymbol{X}})\, d\Sigma. \tag{2.5}$$

Equation (2.8) is a mass flow rate average of P_t on $\partial\Omega_2$. This kind of average is classically used for total variables [35]. Equations (2.8) and (2.9) are calculated by the post-processing tool ***Zapp***.
The goal of the optimization is to reduce the shock intensity and the wake width in order to minimize the entropy generation rate. We present in the section **2.2.4** the results of the single-point optimization.

2.2.2 Setting of the multipoint optimization problem

As we explained in the section **1.1.2**, it can be interesting to minimize F not only for the nominal condition Π_{nom} but over a continuous interval of conditions I_Π around $\Pi_{nom} = 0.583$. We choose $\mathrm{I}_\Pi = [0.506\,, 0.633]$, which corresponds to $M_{2,\,is} = [0.85\,, 1.05]$, and we sample it uniformly with $N_\Pi = 21 > n_\alpha = 18$. We perform the GSA algorithm and we obtain five different conditions. The table 2.12 shows these conditions and the corresponding isentropic Mach numbers.

Conditions	**C01**	**C05**	**C15**	**C17**	**C21**
Pressure ratio Π	Π_{01}=0.633	Π_{05}=0.607	Π_{15}=0.543	Π_{17}=0.530	Π_{21}=0.506
Isentropic Mach number $M_{2,is}$	0.85	0.89	0.99	1.01	1.05

Table 2.4: Pressure ratios selected by GSA and corresponding isentropic Mach numbers.

The algorithm GSA requires calculations of gradient $\underline{\boldsymbol{\nabla}}_{\underline{\boldsymbol{\alpha}}} F(\underline{\boldsymbol{\alpha}}, \Pi_k)$ for the 21 conditions Π_k. Each gradient calculation uses all the modules of the optimization process (from ***Padge*** to adjoint ***Padge***, see table 1.2 section **1.1.3**) and is performed on 16 processors. The CPU cost of one iteration of the optimization process is about 7200 seconds.
We can now define $\tilde{F}$:

$$\tilde{F}(\underline{\boldsymbol{\alpha}}) = w_{01}\, F(\underline{\boldsymbol{\alpha}}, \Pi_{01}) + w_{05}\, F(\underline{\boldsymbol{\alpha}}, \Pi_{05}) + w_{15}\, F(\underline{\boldsymbol{\alpha}}, \Pi_{15}) + w_{17}\, F(\underline{\boldsymbol{\alpha}}, \Pi_{17}) + w_{21}\, F(\underline{\boldsymbol{\alpha}}, \Pi_{21}).$$

We present in the section **2.2.5** the results of the multipoint optimization with two different types of weights.

2.2.3 Parametrization and gradient validation with finite differences

The LS89 blade has been parametrized with 18 design variables. The trailing edge of the blade has been frozen in order to keep the same direction of the wake (as we said in section **2.1.2** the turbine has to prepare the flow before arriving in the rotative part and one of the important criteria is the flow deflection). The thicknesses have been also frozen for structural constraints.
We have compared the values of the gradient $\underline{\boldsymbol{\nabla}}_{\underline{\boldsymbol{\alpha}}} F$ given by the discrete adjoint method with values given by finite differences (using a second order scheme). The number of iterations (800) of the iterative resolution of the adjoint equation (1.51) has been chosen in order to provide low errors : the average error is equal to 1.86%. The figure 2.2 shows the error with respect to design variables. We have actually computed $\underline{\boldsymbol{\nabla}}_{\underline{\boldsymbol{\alpha}}}(1000\, F)$ because the derivatives have a relatively low level and a few modules of the optimization process work only with single precision.

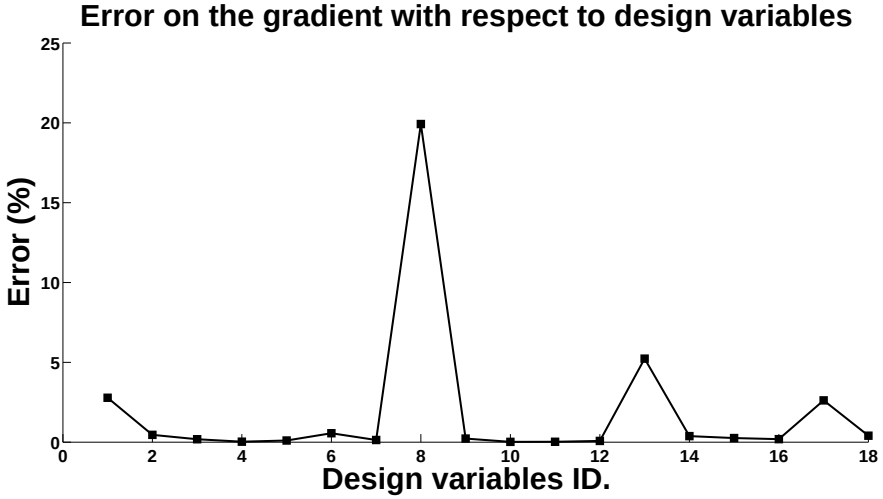

Figure 2.2: Values of the gradient $\underline{\boldsymbol{\nabla}}_{\underline{\boldsymbol{\alpha}}} F$ given by the discrete adjoint method compared with values given by finite differences.

The design variables #8 and #13 have the highest errors : 19.92% and 5.2%. These two variables represent curvatures of the blade at two different positions. The result is moreover very satisfying because the average error is very low.

2.2.4 Results of the single-point optimization

We present in this section the results of the single-point optimization obtained with the LBFGSB algorithm. The figure 2.3 shows the optimization history. The table 2.5 synthesizes the results.

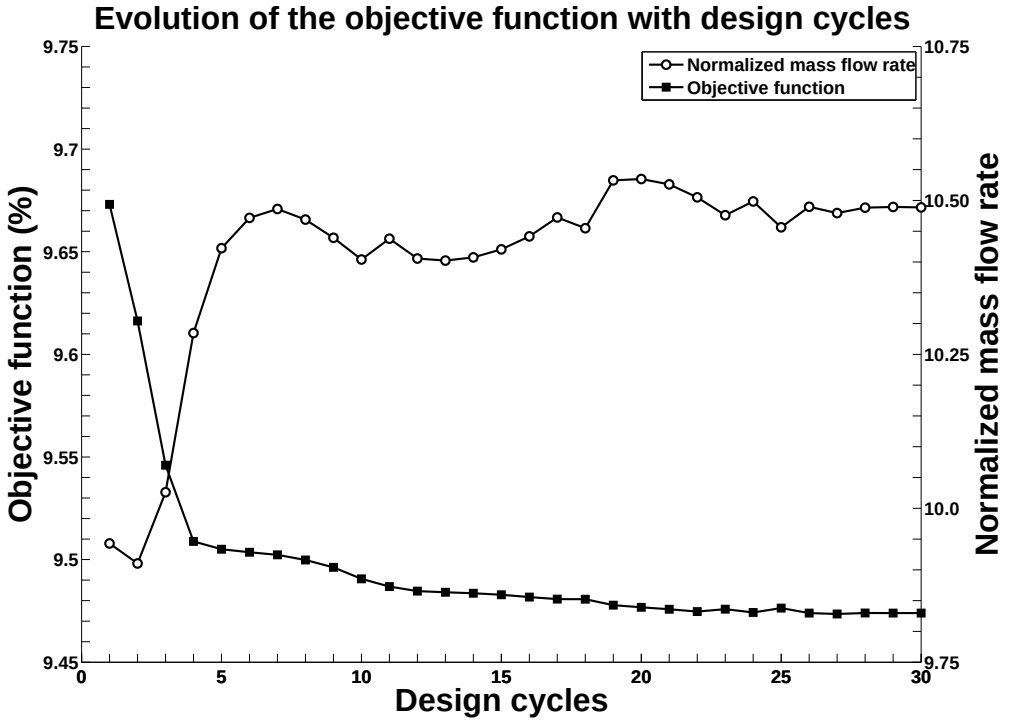

Figure 2.3: Evolution of the objective function with design cycles and correlative augmentation of the mass flow rate.

	Initial value	**Final value**	**Variation (%)**
Entropy generation rate (%)	9.67	9.47	- 2.07
Normalized mass flow rate	9.94	10.49	+ 5.53

Table 2.5: Inital and final values of the entropy generation rate and the mass flow rate.

A few comments about the figure 2.3. The calculation has been performed on 16 parallel processors. The convergence has been reached after 30 iterations of the optimization process and for a CPU cost equal to about 216000 seconds (CPU cost $\sim$ 7200 seconds per iteration, as said in section **2.1.3**). The entropy generation has been successfully reduced of 2.07%. The diminution of total pressure loss has correlatively increased the mass flow rate. In other words, a part of the energy which was lost through the shock wave and inside the turbulent

wake has been given back to the flow, increasing the mass flow rate through $\partial\Omega_2$. The figure 2.4 compares the initial and final blade geometries.

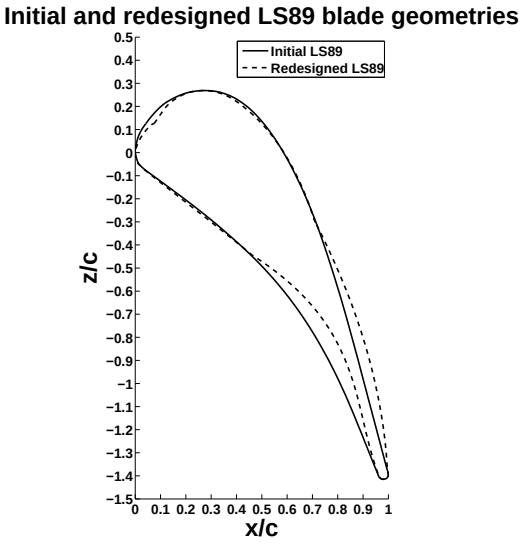

Figure 2.4: Comparison of initial and final LS89 blade geometries.

The redesigned LS89 blade best guides the flow delaying the flow separation and consequently the width of the trailing edge wake. The redesigned geometry is not $\mathcal{C}^2$ at the leading edge which suggests that the parametrization has to be improved (the leading edge is parametrized by one radius of curvature and it seems two or more could be a better way to characterize it). Figures 2.5 and 2.6 compare the characteristics of the initial and the redesigned LS89 blades.

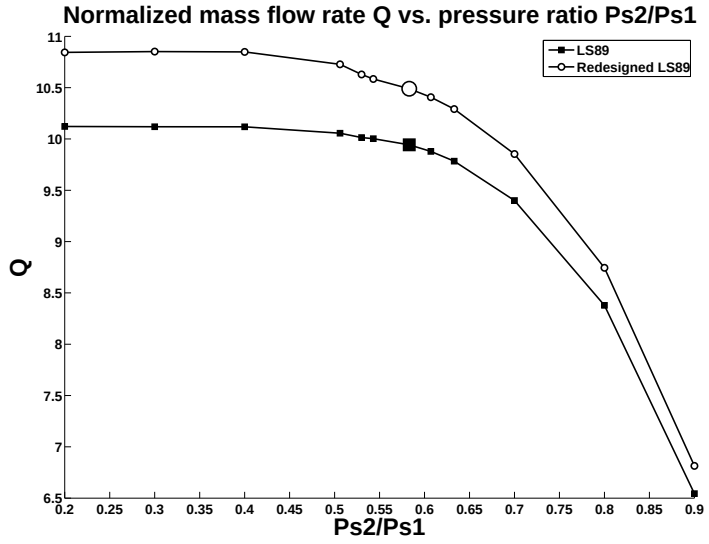

Figure 2.5: Comparison of the characteristics of the initial and the redesigned LS89 blades. The nominal condition is represented with larger markers.

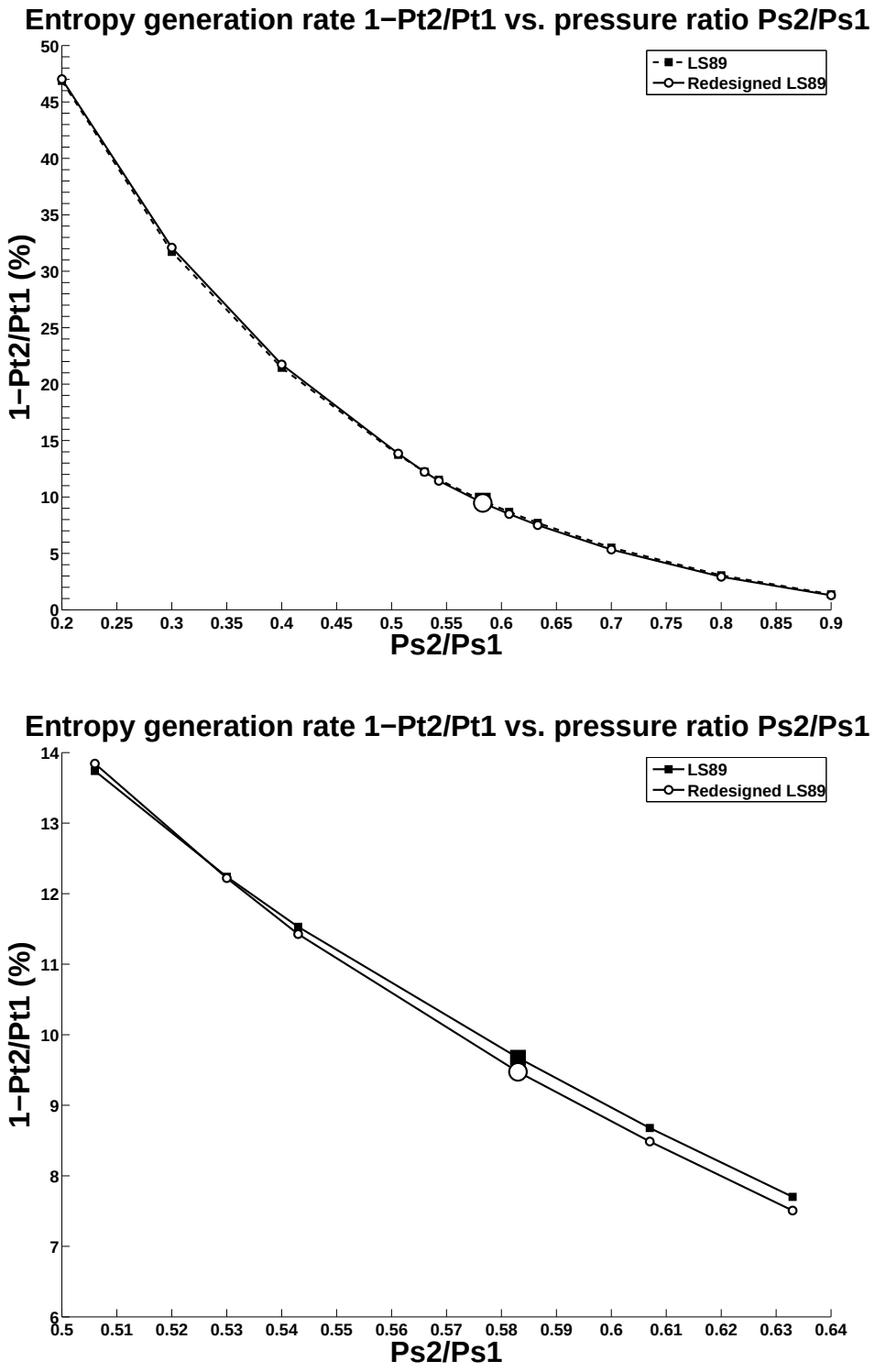

Figure 2.6: Comparison of the characteristics of the initial and the redesigned LS89 blades. The nominal condition is represented with larger markers.

The figure 2.5 shows that the mass flow rate has been increased all over the characteristic. For a given geometry, increasing the mass flow rate raises the entropy generation rate. The goal of the optimization is then to design shapes which generate higher mass flow rates without increasing the entropy generation rate : the figure 2.6 indicates the entropy generation rate has been reduced for a few conditions but also increased for others. The table 2.6 synthesizes the results on the interval of pressure ratio $[0.506\,, 0.633]$.

Condition	**C21**	**C17**	**C15**	***nominal***	**C05**	**C01**
Pressure ratio	0.506	0.530	0.543	0.583	0.607	0.633
Variation of entropy generation rate (%)	+ 0.76	- 0.15	- 0.90	- 2.07	- 2.21	- 2.51

Table 2.6: Variation of entropy generation rate with respect to pressure ratio.

The objective function has been reduced for conditions **C01**, **C05**, **C15**, **C17** and for the nominal condition, but has been increased for condition **C21**. This is precisely what designers call a poor design and a way to obtain better designs is given by the multipoint optimization.

2.2.5 Results of the multipoint optimization

We present in this section the results of the multipoint optimization for two types of weights. We present first a multipoint optimization with unit weights and present then a multipoint optimization with non-unit weights. We compute the calculations on 16 parallel processors. The cost of one iteration of the optimization process is here much higher because one iteration of the multipoint optimization corresponds to five iterations of the single-point optimization process since the evaluation of the multipoint objective function requires five single-point objective function evaluations.

Unit weights

In this section we use unit weights so the objective function $\tilde{F}$ can be written as follows :

$$\tilde{F}(\underline{\boldsymbol{\alpha}}) = F(\underline{\boldsymbol{\alpha}}, \Pi_{01}) + F(\underline{\boldsymbol{\alpha}}, \Pi_{05}) + F(\underline{\boldsymbol{\alpha}}, \Pi_{15}) + F(\underline{\boldsymbol{\alpha}}, \Pi_{17}) + F(\underline{\boldsymbol{\alpha}}, \Pi_{21}).$$

The evaluation of each $F(\underline{\boldsymbol{\alpha}}, \Pi_k)$ is performed on 16 parallel processors so then the optimization process requires here $16 \times 5 = 80$ processors. The figure 2.7 shows the optimization history and the table 2.7 synthesizes the results.

Condition	**C21**	**C17**	**C15**	***nominal***	**C05**	**C01**
Pressure ratio	0.506	0.530	0.543	0.583	0.607	0.633
Variation of entropy generation rate (%)	- 2.57	- 1.66	- 1.11	- 0.81	- 0.92	- 1.13

Table 2.7: Variation of entropy generation rate with respect to pressure ratio.

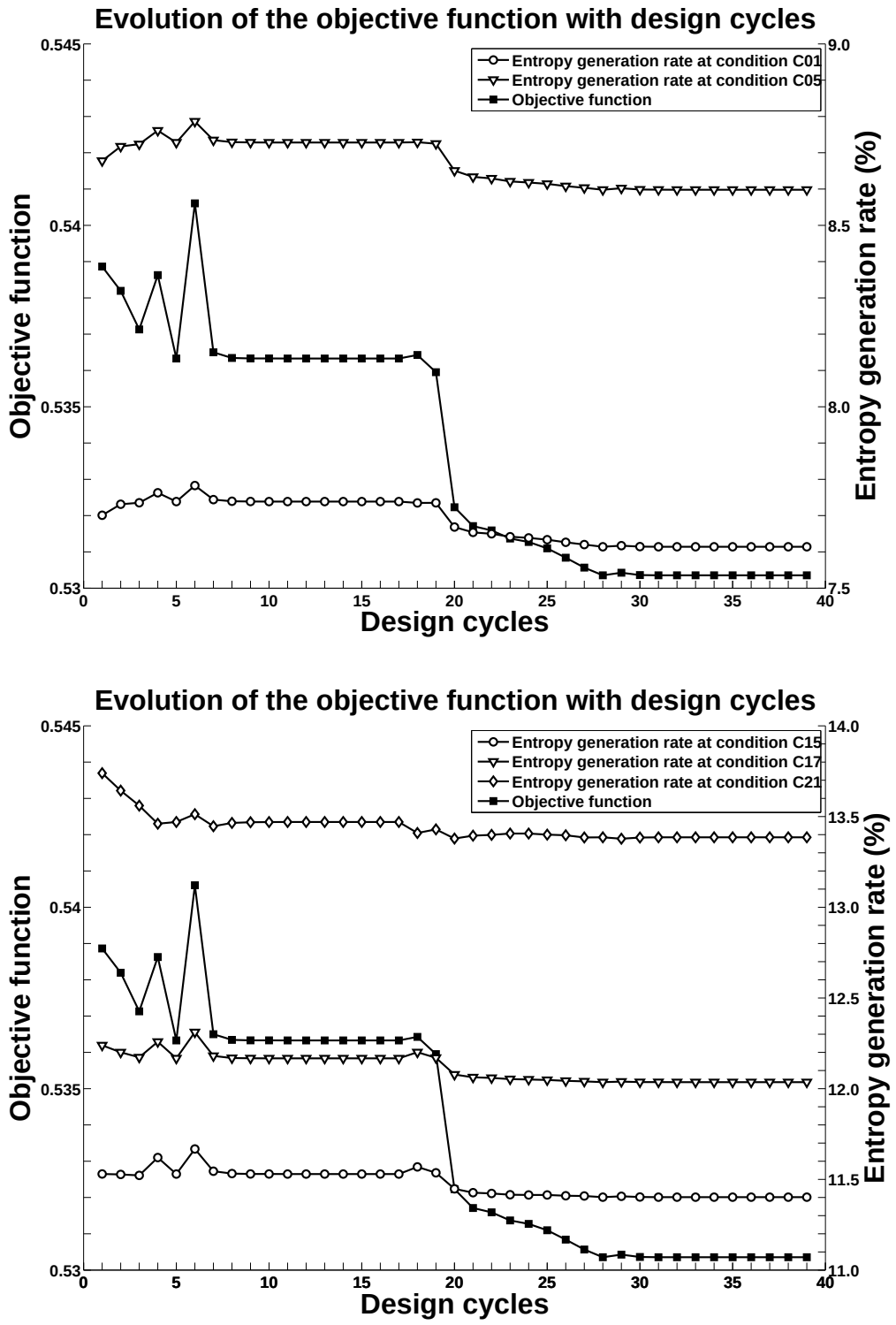

Figure 2.7: Evolution of the objective function with design cycles and correlative reduction of the entropy generation rate at the five selected conditions.

The figure 2.7 and the table 2.7 show that the entropy generation rate has been successfully reduced for all the conditions and even all over the characteristic as shown on the figure 2.8. The convergence has been reached after 40 iterations of the optimization process and for a CPU cost equal to about 1440000 seconds (CPU cost $\sim 7200 \times 5 = 36000$ seconds per iteration, as said in section **2.2.2**). The figure 2.9 shows that the given back energy has been transmitted to the flow through an augmentation of the mass flow rate.

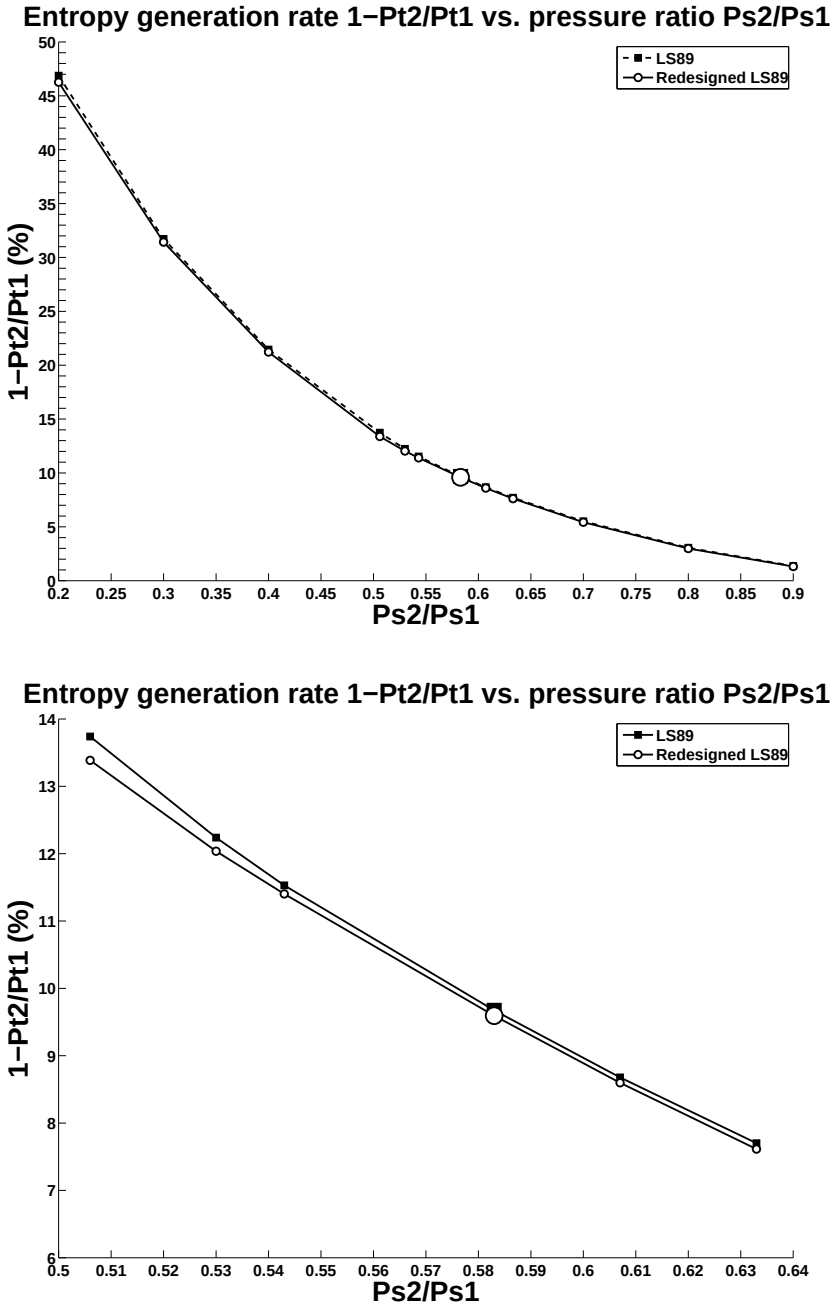

Figure 2.8: Comparison of the characteristics of the initial and the redesigned LS89 blades. The nominal condition is represented with larger markers.

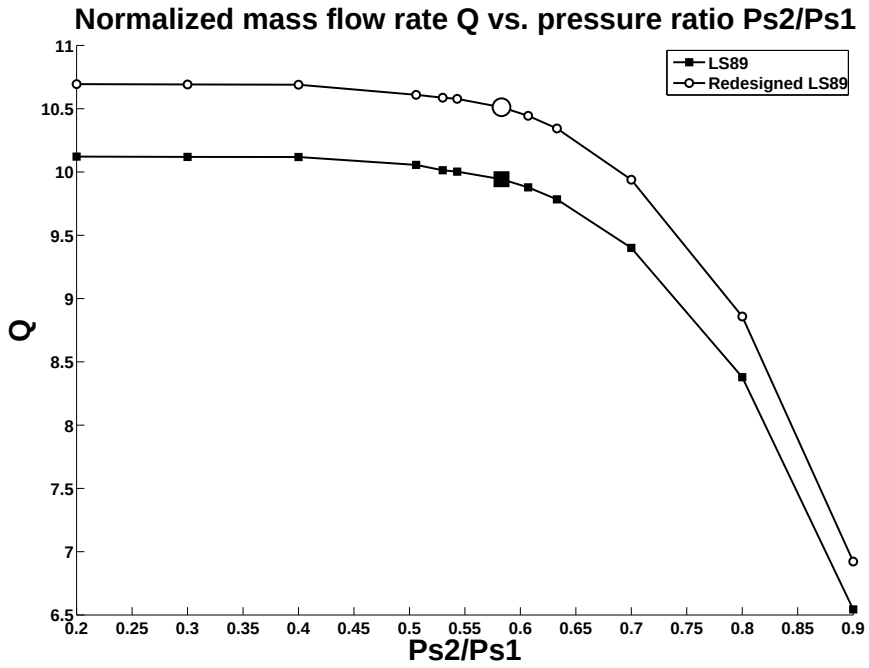

Figure 2.9: Comparison of the characteristics of the initial and the redesigned LS89 blades. The nominal condition is represented with larger markers.

And finally, the figure 2.10 compares the initial and final LS89 blade geometries.

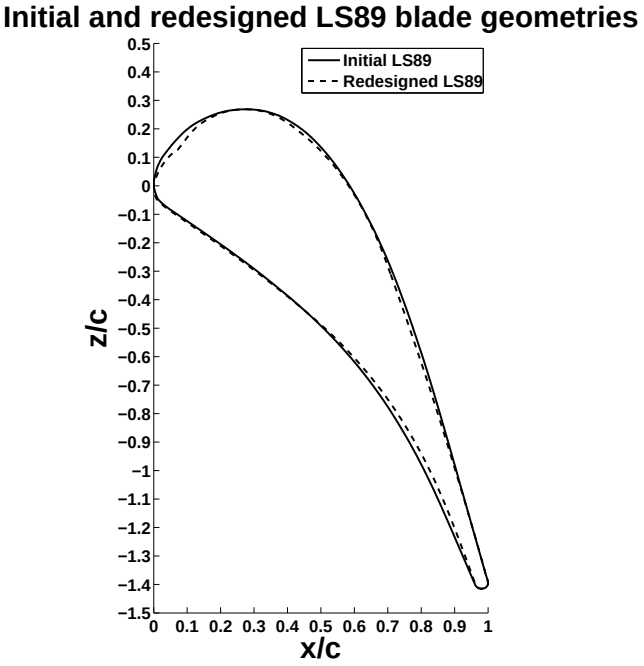

Figure 2.10: Comparison of initial and final LS89 blade geometries.

The redesigned geometry is one more time not $\mathcal{C}^2$ at the leading edge which suggests that the parametrization we used is not very well adapted for this blade.

Non-unit weights

We present in this section results obtained for a multipoint optimization with non-unit weights. The aim is to force the algorithm to target the Pareto front. Let us recall the expression of $\tilde{F}$:

$$\tilde{F}(\underline{\boldsymbol{\alpha}}) = w_{01}\, F(\underline{\boldsymbol{\alpha}}, \Pi_{01}) + w_{05}\, F(\underline{\boldsymbol{\alpha}}, \Pi_{05}) + w_{15}\, F(\underline{\boldsymbol{\alpha}}, \Pi_{15}) + w_{17}\, F(\underline{\boldsymbol{\alpha}}, \Pi_{17}) + w_{21}\, F(\underline{\boldsymbol{\alpha}}, \Pi_{21}).$$

If one of the functions, let us say $F(\underline{\boldsymbol{\alpha}}, \Pi_{01})$, can be reduced a lot then the algorithm could just reduce this function and increase the others, just because the descent direction taken to reduce $F(\underline{\boldsymbol{\alpha}}, \Pi_{01})$ is the steepest one. The idea is then to perform a single-point optimization for each condition (with variation of entropy generation rate Δegr) and choose the weights equal to the absolute values of $1/\Delta egr$. The table 2.8 gives the variations of entropy generation rate for single-point optimizations and the associated weights.

Condition	**C21**	**C17**	**C15**	**C05**	**C01**
Variations of entropy generation rate for a single-point optimization Δegr **(%)**	- 2.24	- 1.53	- 1.47	- 2.08	- 2.63
Associated weights $1/\Delta egr$	44.5	65.4	68.0	48.1	38.0

Table 2.8: Variations of entropy generation rate for single-point optimizations and associated weights for the multipoint optimization.

In practice, the variations Δegr could be estimated by the designers so the weights are not necessary calculated with a single-point optimization.
We have chosen actually weights equal to $0.1/\Delta egr$ in order to obtain reasonable values for the objective function (let us recall that we compute $1000\, F(\underline{\boldsymbol{\alpha}}, \Pi)$ and not $F(\underline{\boldsymbol{\alpha}}, \Pi)$ for the gradient accuracy). The objective function $\tilde{F}$ is then :

$$\tilde{F}(\underline{\boldsymbol{\alpha}}) = 3.08\, F(\underline{\boldsymbol{\alpha}}, \Pi_{01}) + 4.81\, F(\underline{\boldsymbol{\alpha}}, \Pi_{05}) + 6.80\, F(\underline{\boldsymbol{\alpha}}, \Pi_{15}) + 6.54\, F(\underline{\boldsymbol{\alpha}}, \Pi_{17}) + 4.45\, F(\underline{\boldsymbol{\alpha}}, \Pi_{21}).$$

The evaluation of each $F(\underline{\boldsymbol{\alpha}}, \Pi_k)$ is performed on 16 parallel processors so then the optimization process requires here $16 \times 5 = 80$ processors. The figure 2.11 shows the optimization history and the table 2.9 synthesizes the results.

Condition	**C21**	**C17**	**C15**	***nominal***	**C05**	**C01**
Pressure ratio	0.506	0.530	0.543	0.583	0.607	0.633
Variation of entropy generation rate (%)	- 2.35	- 1.58	- 1.06	- 0.71	- 0.77	- 0.90

Table 2.9: Variation of entropy generation rate with respect to pressure ratio.

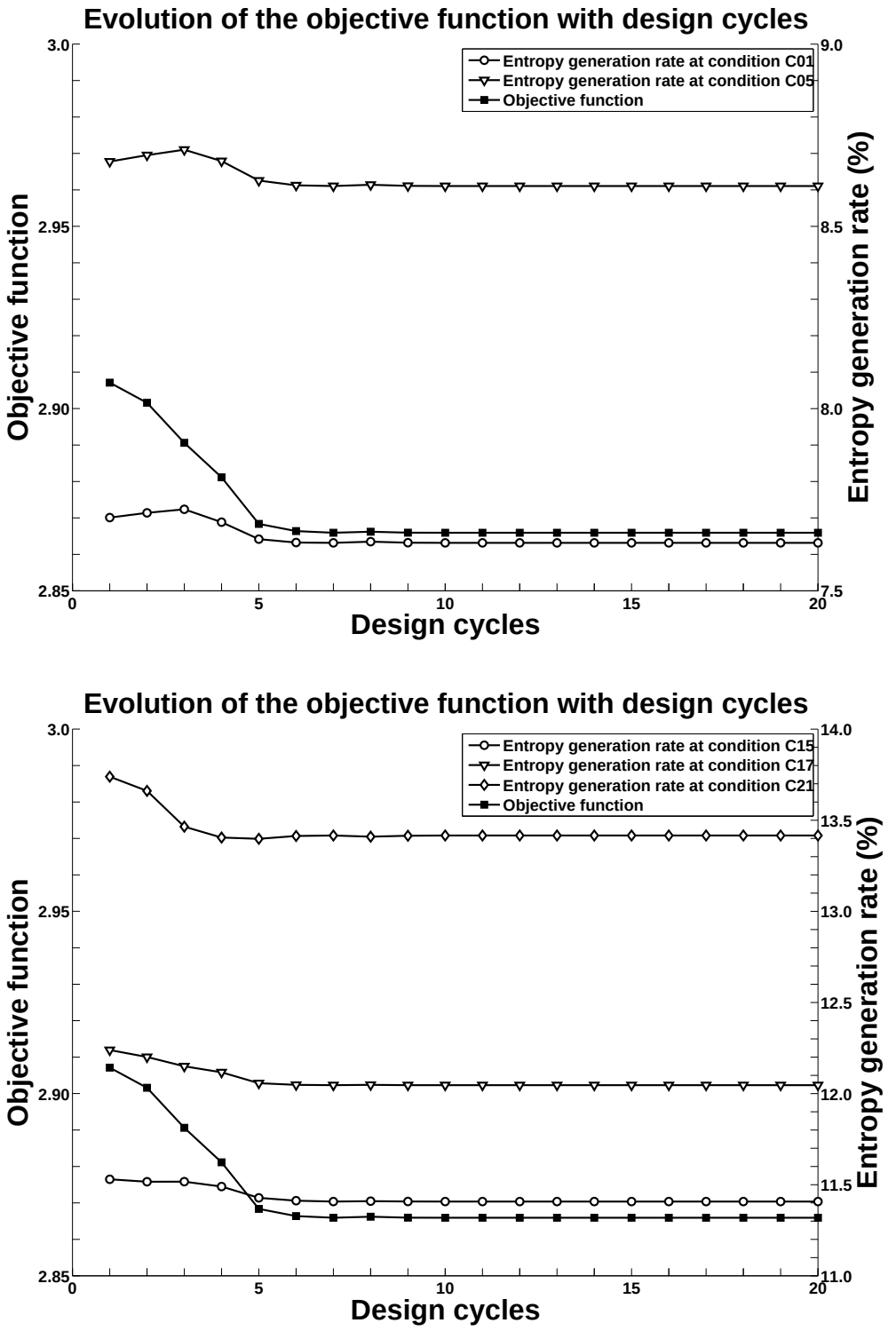

Figure 2.11: Evolution of the objective function with design cycles and correlative reduction of the entropy generation rate at the five selected conditions.

The figure 2.11 and the table 2.9 shows that the entropy generation rate has been one more time successfully reduced for all the conditions and even all over the characteristic as shown on the figure 2.12. The convergence has been reached after 10 iterations of the optimization process and for a CPU cost equal to about 360000 seconds The figure 2.13 shows that this reduction is correlated to an augmentation of mass flow rate.

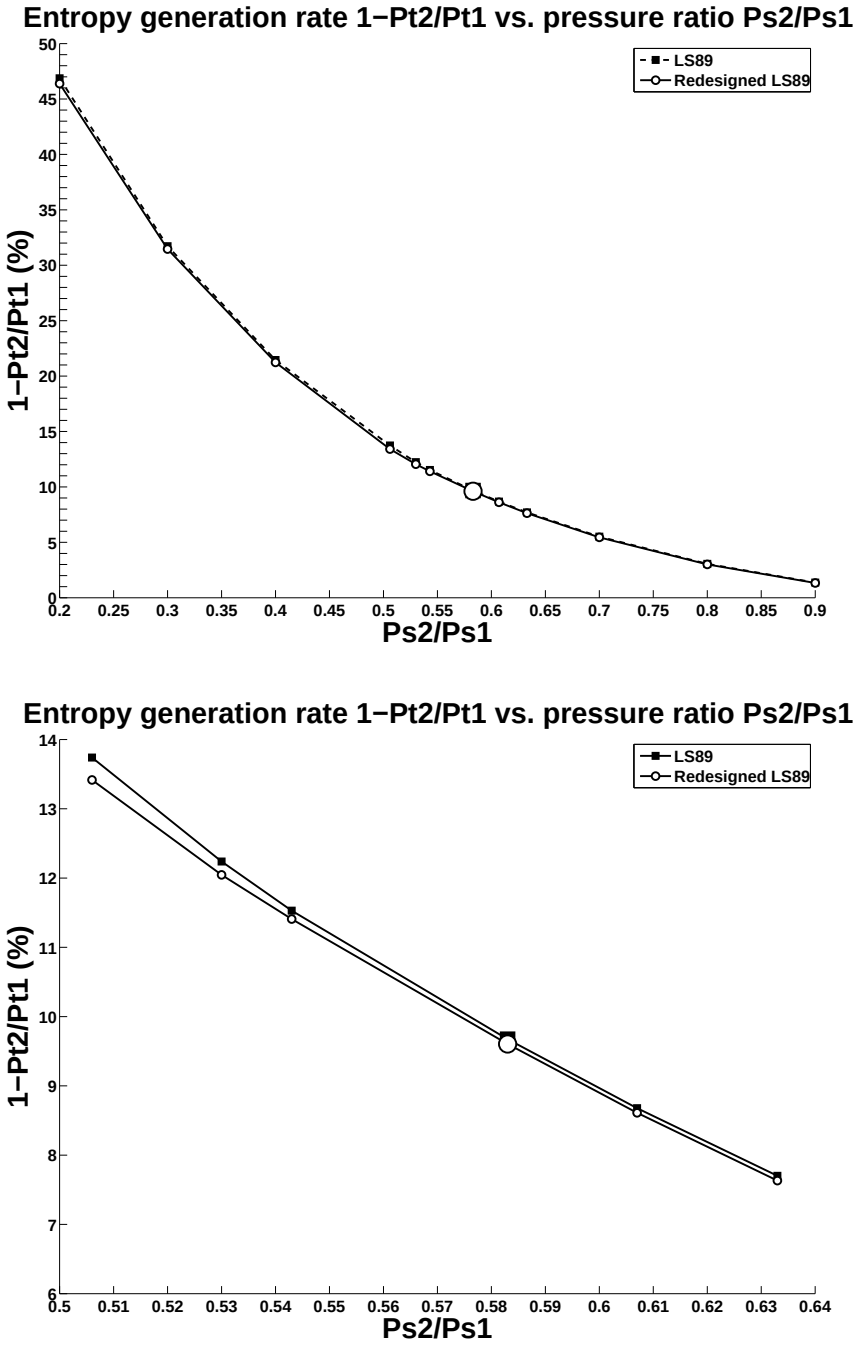

Figure 2.12: Comparison of the characteristics of the initial and the redesigned LS89 blades. The nominal condition is represented with larger markers.

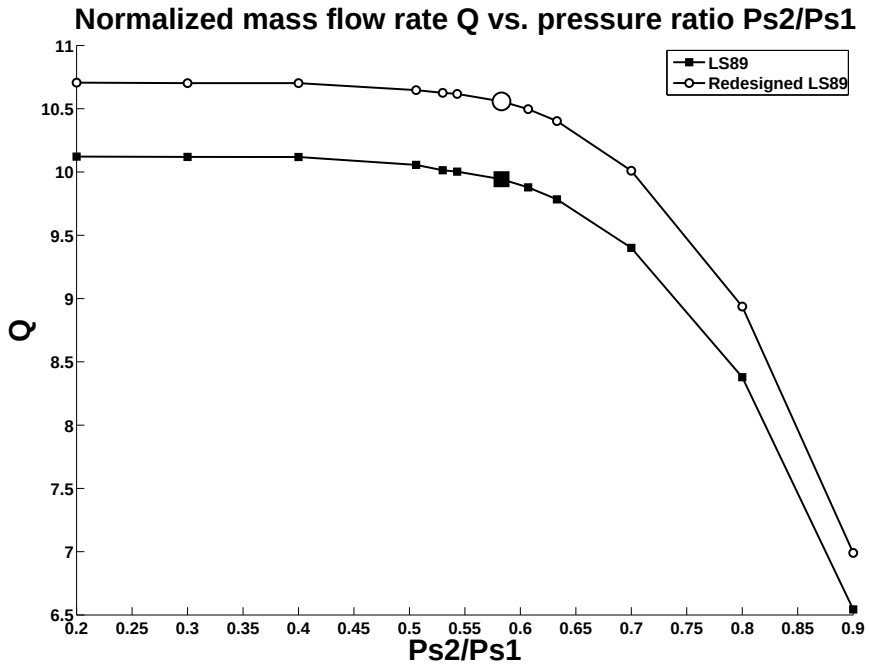

Figure 2.13: Comparison of the characteristics of the initial and the redesigned LS89 blades. The nominal condition is represented with larger markers.

The figure 2.14 compares the initial and final LS89 blade geometries.

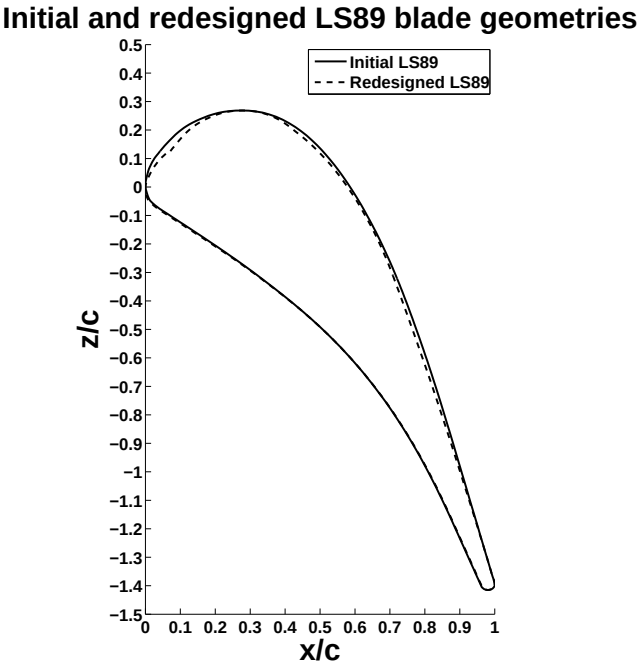

Figure 2.14: Comparison of initial and final LS89 blade geometries.

2.2.6 Comparison of the two multipoint optimizations

The two multipoint optimizations provide satisfying results (table 2.10).

Condition	**C21**	**C17**	**C15**	***nominal***	**C05**	**C01**
Pressure ratio	0.506	0.530	0.543	0.583	0.607	0.633
Variation of entropy generation rate (%) with						
unit weights	- 2.57	- 1.66	- 1.11	- 0.81	- 0.92	- 1.13
non-unit weights	- 2.35	- 1.58	- 1.06	- 0.71	- 0.77	- 0.90

Table 2.10: Comparison of variations of entropy generation rate for the two multipoint optimizations.

The multipoint optimization with unit weights gives lightly better results but for a CPU cost much higher (table 2.11).

	Unit weights	**Non-unit weights**
Number of iterations of the optimization process	40	10
CPU cost (seconds)	1440000	360000

Table 2.11: Comparison of CPU cost to reach the convergence.

The figure 2.15 compares the two geometries.

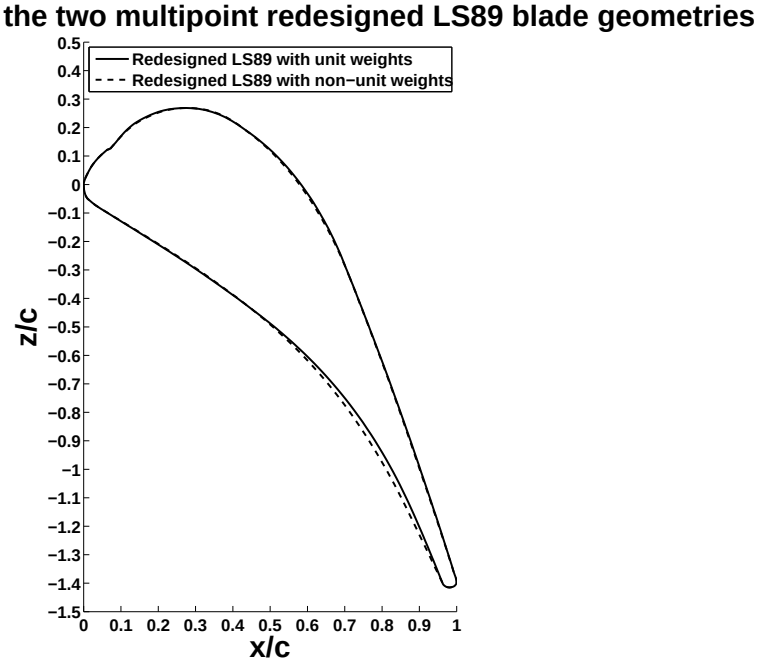

Figure 2.15: Comparison of initial and final LS89 blade geometries.

2.3 Entropy generation rate minimization with constraint on the mass flow rate and with frozen leading edge

The redesigned LS89 of last section is not $\mathcal{C}^2$. We have worked then on a new parametrization with 25 design variables and a frozen leading edge. We have performed a single-point optimization with it and the results are very different : the entropy generation decreased of 0.8% but the mass flow rate also decreased of 2.28%. To avoid the entropy decrease caused by mass flow rate reduction we choose then to minimize the entropy generation rate with constraint on the mass flow rate and with frozen leading edge.

2.3.1 Setting of the single-point optimization problem

The objective function is then :

$$F(\underline{\boldsymbol{\alpha}}, \Pi) = \frac{S(\underline{\boldsymbol{\alpha}}, \Pi)}{S_0(\Pi)} + \sigma \left(\frac{Q(\underline{\boldsymbol{\alpha}}, \Pi)}{Q_0(\Pi)} - 1 \right)^2, \tag{2.6}$$

where $S(\underline{\boldsymbol{\alpha}}, \Pi)$ is the outlet surface averaged entropy generation rate :

$$S(\underline{\boldsymbol{\alpha}}, \Pi) = 1 - \frac{P_{t2}(\underline{\boldsymbol{\alpha}}, \Pi)}{P_{t1}}, \tag{2.7}$$

with :

$$P_{t2}(\underline{\boldsymbol{\alpha}}, \Pi) = \frac{1}{S_2} \iint_{\underline{\boldsymbol{X}} \in \partial\Omega_2} P_t(\underline{\boldsymbol{X}}) \, d\Sigma, \tag{2.8}$$

and $Q(\underline{\boldsymbol{\alpha}}, \Pi)$ the mass flow rate :

$$Q = \iint_{\underline{\boldsymbol{X}} \in \partial\Omega_2} \rho(\underline{\boldsymbol{X}}) \, \underline{\boldsymbol{U}}(\underline{\boldsymbol{X}}) \cdot \underline{\boldsymbol{n}}_2(\underline{\boldsymbol{X}}) \, d\Sigma. \tag{2.9}$$

The index 0 corresponds of values for the initial LS89 blade. We present in the section **2.3.4** the results of the single-point optimization.

2.3.2 Setting of the multipoint optimization problem

As we explained in the section **1.1.2**, it can be interesting to minimize F not only for the nominal condition Π_{nom} but over a continuous interval of conditions I_Π around $\Pi_{nom} = 0.583$. We choose here a larger set $\mathrm{I}_\Pi = [0.476\,, 0.732]$, which corresponds to $M_{2,\,is} = [0.7\,, 1.1]$, and we sample it uniformly. We perform the GSA algorithm and we obtain five different conditions. The table 2.12 shows these conditions and the corresponding isentropic Mach numbers.

Conditions	**C01**	**C05**	**C14**	**C17**	**C21**
Pressure ratio Π	Π_{01}=0.732	Π_{05}=0.680	Π_{15}=0.562	Π_{17}=0.524	Π_{21}=0.476
Isentropic Mach number $M_{2,is}$	0.7	0.78	0.96	1.02	1.1

Table 2.12: Pressure ratios selected by GSA and corresponding isentropic Mach numbers.

The algorithm GSA requires calculations of gradient $\underline{\boldsymbol{\nabla}}_{\underline{\boldsymbol{\alpha}}} F(\underline{\boldsymbol{\alpha}}, \Pi_k)$ for the 21 conditions Π_k. Each gradient calculation uses all the modules of the optimization process (from ***Padge*** to adjoint ***Padge***, see table 1.2 section **1.1.3**) and is performed on 16 processors. The CPU cost of one iteration of the optimization process is about 7200 seconds.
We can now define $\tilde{F}$:

$$\tilde{F}(\underline{\boldsymbol{\alpha}}) = w_{01}\, F(\underline{\boldsymbol{\alpha}}, \Pi_{01}) + w_{05}\, F(\underline{\boldsymbol{\alpha}}, \Pi_{05}) + w_{14}\, F(\underline{\boldsymbol{\alpha}}, \Pi_{14}) + w_{17}\, F(\underline{\boldsymbol{\alpha}}, \Pi_{17}) + w_{21}\, F(\underline{\boldsymbol{\alpha}}, \Pi_{21}).$$

We present in the section **2.3.5** the results of the multipoint optimization with two different types of weights.

2.3.3 Parametrization and gradient validation with finite differences

The LS89 blade has been parametrized with 25 design variables. The trailing edge of the blade has been frozen in order to keep the same direction of the wake (as we said in section **2.2.1** the turbine has to prepare the flow before arriving in the rotative part and one of the important criteria is the flow deflection). The thicknesses have been also frozen for structural constraints. The leading edge has been frozen to provide $\mathcal{C}^2$ geometries.
We have compared the values of the gradient $\underline{\boldsymbol{\nabla}}_{\underline{\boldsymbol{\alpha}}} F$ given by the discrete adjoint method with values given by finite differences (using a second order scheme). The number of iterations (800) of the iterative resolution of the adjoint equation (1.51) has been chosen in order to provide low errors : the average error is equal to 1.65%. The figure 2.2 shows the error with respect to design variables. We have actually computed $\underline{\boldsymbol{\nabla}}_{\underline{\boldsymbol{\alpha}}}(100\, F)$ because the derivatives have a relatively low level and a few modules of the optimization process work only with single precision.

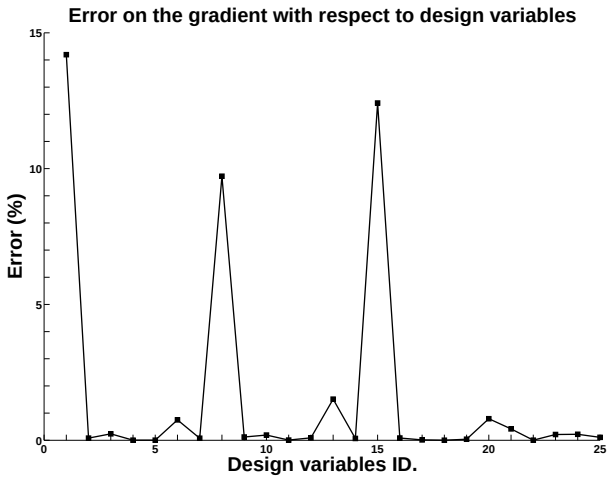

Figure 2.16: Values of the gradient $\underline{\boldsymbol{\nabla}}_{\underline{\boldsymbol{\alpha}}} F$ given by the discrete adjoint method compared with values given by finite differences.

The design variables #1, #9 and #15 have the highest errors : 14.19%, 9.72% and 12.41%. These three variables represent curvatures of the blade at three different positions. The result is moreover very satisfying because the average error is very low.

2.3.4 Results of the single-point optimization

We present in this section the results of the single-point optimization obtained with the LBFGSB algorithm. The figure 2.17 shows the optimization history. The table 2.13 synthesizes the results.

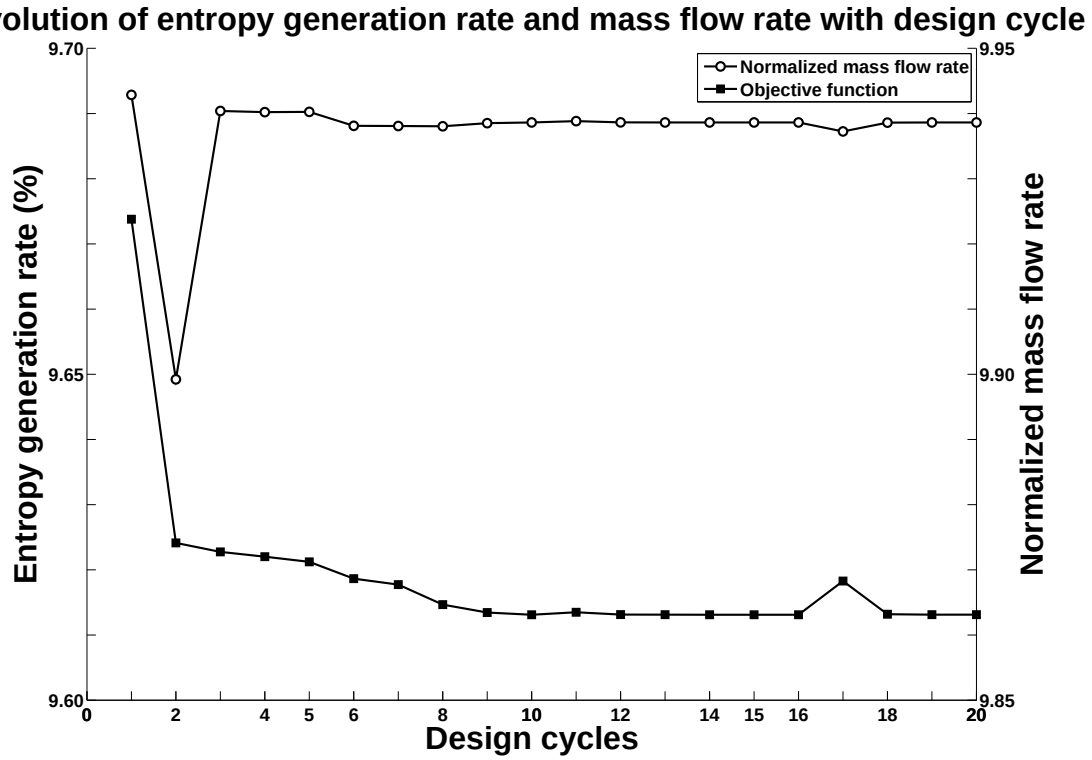

Figure 2.17: Evolution of the entropy generation rate with design cycles and correlative evolution of the constrained mass flow rate.

	Initial value	**Final value**	**Variation (%)**
Entropy generation rate (%)	9.674	9.613	- 0.627
Normalized mass flow rate	9.943	9.939	- 0.043

Table 2.13: Inital and final values of the entropy generation rate and the mass flow rate.

A few comments about the figure 2.17. The calculation has been performed on 16 parallel processors. The convergence has been reached after 20 iterations of the optimization process and for a CPU cost equal to about 144000 seconds (CPU cost $\sim$ 7200 seconds per iteration, as said in section **2.3.2**). The entropy generation has been successfully reduced of 0.627%. The mass flow rate fluctuation is very low and more than acceptable : Wang and his collaborators

[44] estimate that the maximum acceptable fluctuation is ± 0.5%. The figure 2.18 compares the initial and final blades geometries.

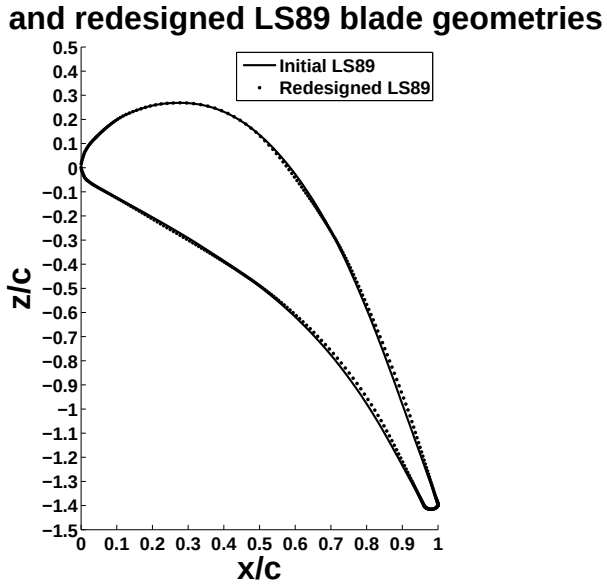

Figure 2.18: Comparison of initial and final LS89 blade geometries.

Figure 2.19 shows the variation of entropy generation rate Δegr with respect to Ps2/Ps1.

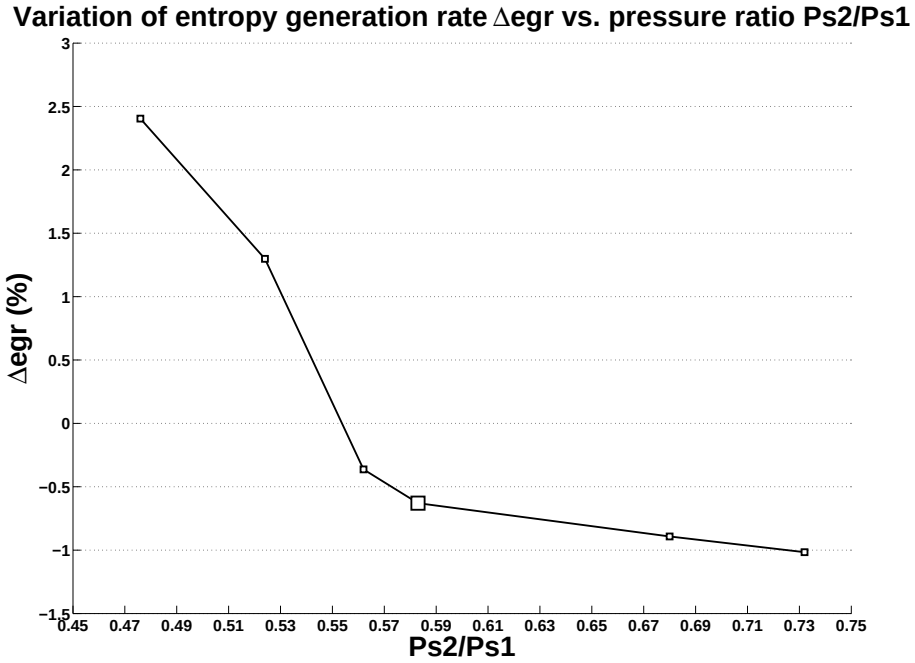

Figure 2.19: Variation of entropy generation rate Δegr with respect to the pressure ratio Ps2/Ps1. The nominal condition is represented with larger markers.

Figure 2.20 compares the mass flow rate of the initial and the redesigned LS89 blades for the pressure ratios selected by the GSA algorithm.

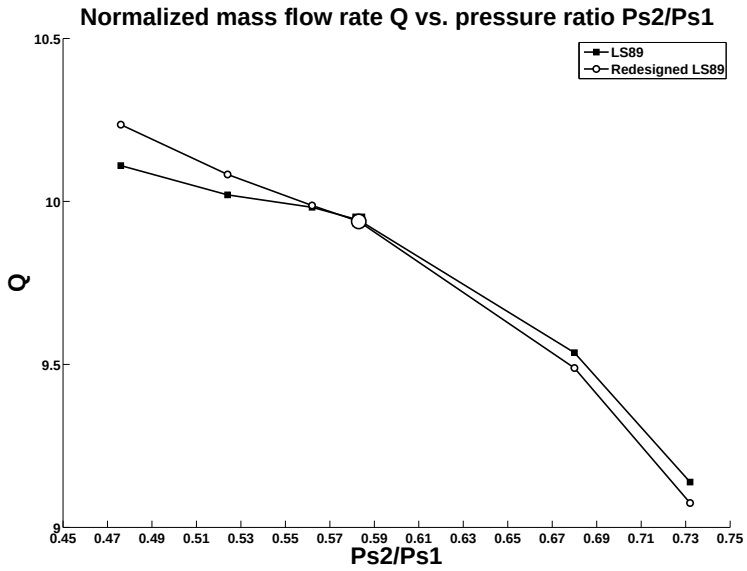

Figure 2.20: Comparison of the mass flow rate of the initial and the redesigned LS89 blades with respect to the pressure ratio Ps2/Ps1. The nominal condition is represented with larger markers.

The table 2.14 synthesizes the results on the interval of pressure ratio $[0.476\,, 0.732]$.

Condition	**C21**	**C17**	**C14**	***nominal***	**C05**	**C01**
Pressure ratio	0.476	0.524	0.562	0.583	0.680	0.732
Δegr (%)	+ 2.414	+ 1.326	- 0.354	- 0.627	- 0.899	- 1.012
ΔQ (%)	+ 1.238	+ 0.617	+ 0.052	- 0.043	- 0.493	- 0.708

Table 2.14: Variation of entropy generation rate and mass flow rate with respect to pressure ratio.

The entropy generation rate has been reduced for conditions **C01**, **C05**, ***nominal*** and **C14** but increased for conditions **C17** and **C21**. Moreover the mass flow rate fluctuation is not acceptable for conditions **C01**, **C17** and **C21**. This is precisely what designers call a poor design and a way to obtain better designs is given by the multipoint optimization.

2.3.5 Results of the multipoint optimization

We present in this section the results of the multipoint optimization for two types of weights. We present first a multipoint optimization with unit weights and present then a multipoint optimization with non-unit weights. We compute the calculations on 16 parallel processors. The cost of one iteration of the optimization process is here much higher because one iteration of the multipoint optimization corresponds to five iterations of the single-point optimization

process since the evaluation of the multipoint objective function requires five single-point objective function evaluations.

Unit weights

In this section we use unit weights so the objective function $\tilde{F}$ can be written as follows :

$$\tilde{F}(\underline{\boldsymbol{\alpha}}) = F(\underline{\boldsymbol{\alpha}}, \Pi_{01}) + F(\underline{\boldsymbol{\alpha}}, \Pi_{05}) + F(\underline{\boldsymbol{\alpha}}, \Pi_{14}) + F(\underline{\boldsymbol{\alpha}}, \Pi_{17}) + F(\underline{\boldsymbol{\alpha}}, \Pi_{21}).$$

The figure 2.21 shows the optimization history and compares the initial and final LS89 blade geometries.

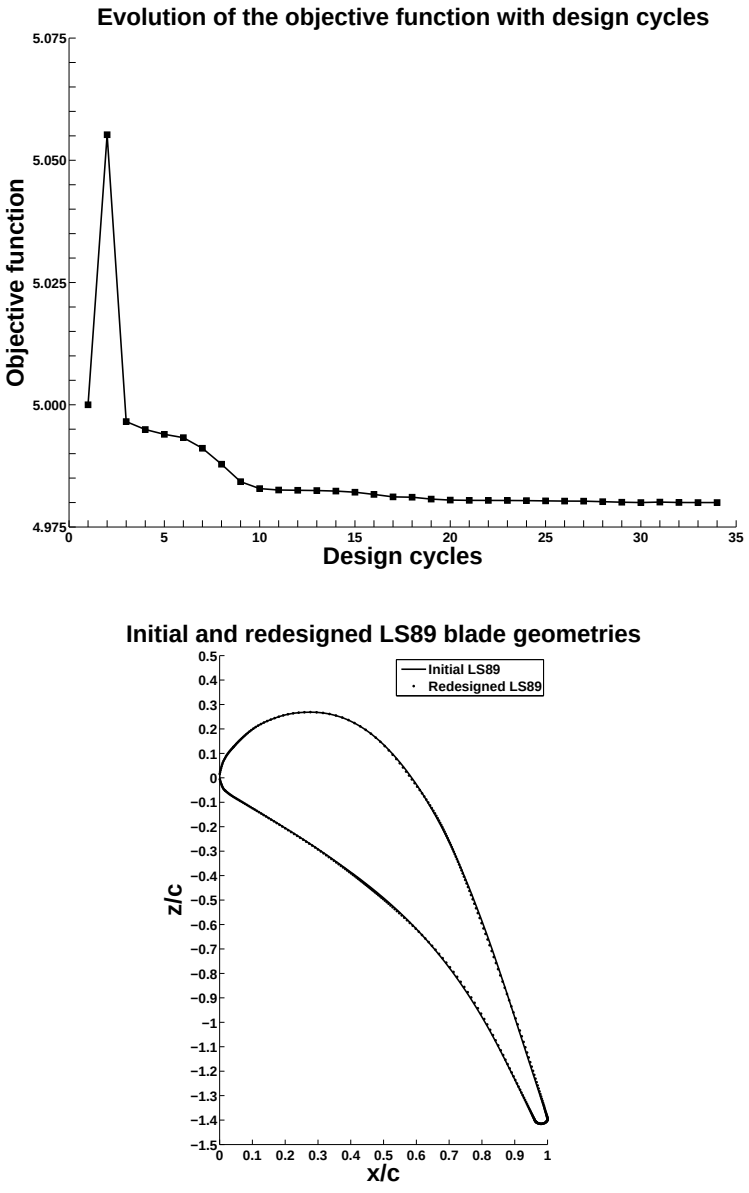

Figure 2.21: Evolution of the objective function with design cycles and comparison of initial and final LS89 blade geometries.

Figure 2.22 shows the variation of entropy generation rate Δegr with respect to Ps2/Ps1.

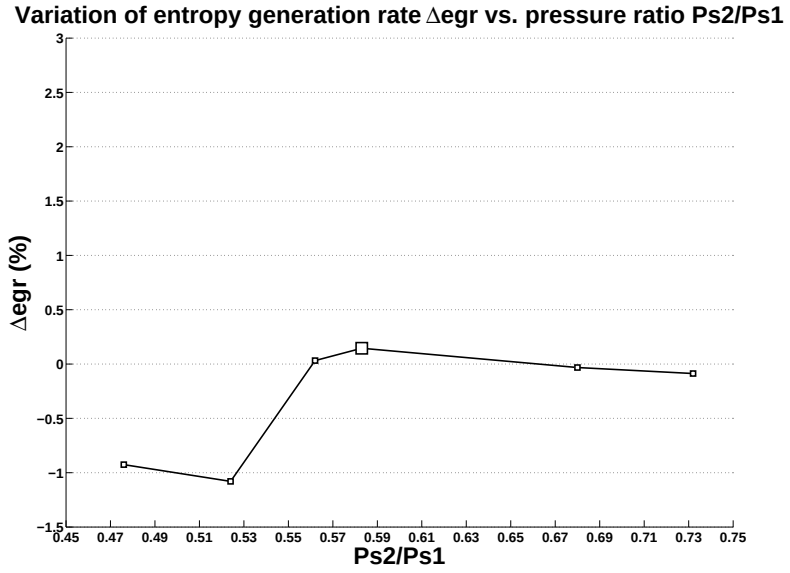

Figure 2.22: Variation of entropy generation rate Δegr with respect to the pressure ratio Ps2/Ps1. The nominal condition is represented with larger markers.

Figure 2.23 compares the mass flow rate of the initial and the redesigned LS89 blades for the pressure ratios selected by the GSA algorithm.

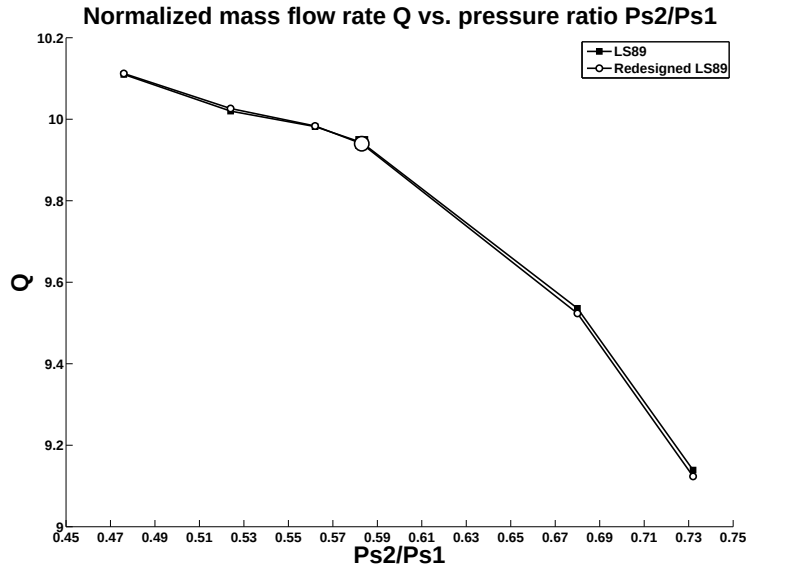

Figure 2.23: Comparison of the mass flow rate of the initial and the redesigned LS89 blades with respect to the pressure ratio Ps2/Ps1. The nominal condition is represented with larger markers.

The table 2.15 synthesizes the results on the interval of pressure ratio [0.476 , 0.732].

Condition	**C21**	**C17**	**C14**	***nominal***	**C05**	**C01**
Pressure ratio	0.476	0.524	0.562	0.583	0.680	0.732
Δegr (%)	- 0.917	- 1.053	+ 0.040	+ 0.148	- 0.039	- 0.083
ΔQ (%)	+ 0.021	+ 0.054	+ 0.012	- 0.030	- 0.131	- 0.176

Table 2.15: Variation of entropy generation rate and mass flow rate with respect to pressure ratio.

The entropy generation rate has been reduced for conditions **C01**, **C05**, **C17**, and **C21** but increased for the nominal condition and condition **C14**. The mass flow rate fluctuations are acceptable. This is not a good optimization since the entropy generation increased for a few conditions and in particularly for the nominal one. Using non-unit weights seems then the only way to get better results.

Non-unit weights

In this section we use non-unit weights equal to $1/\Delta egr$. The figure 2.24 shows the optimization history and compares the initial and final LS89 blade geometries.

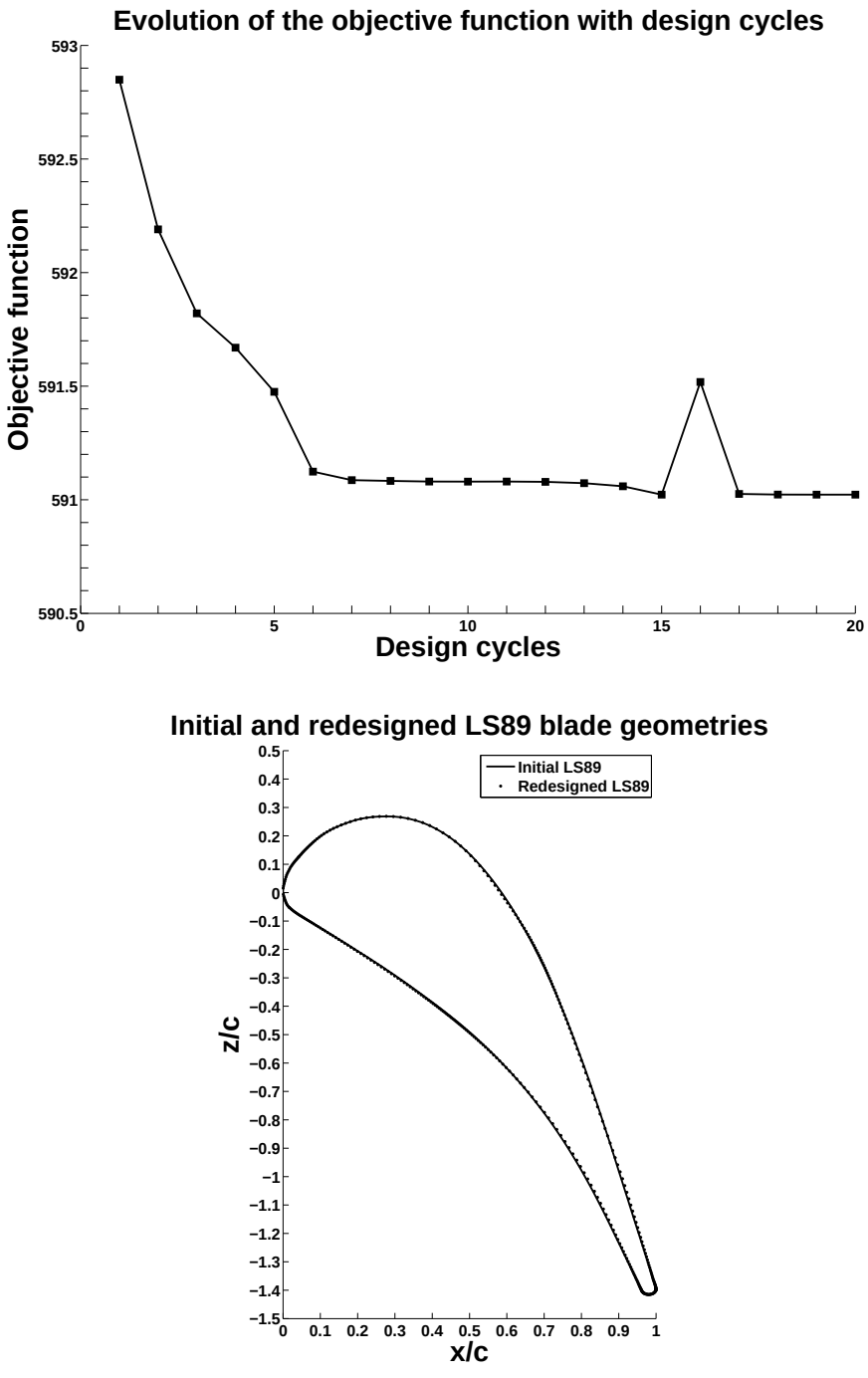

Figure 2.24: Evolution of the objective function and comparison of geometries.

Figure 2.25 shows the variation of entropy generation rate Δegr with respect to Ps2/Ps1.

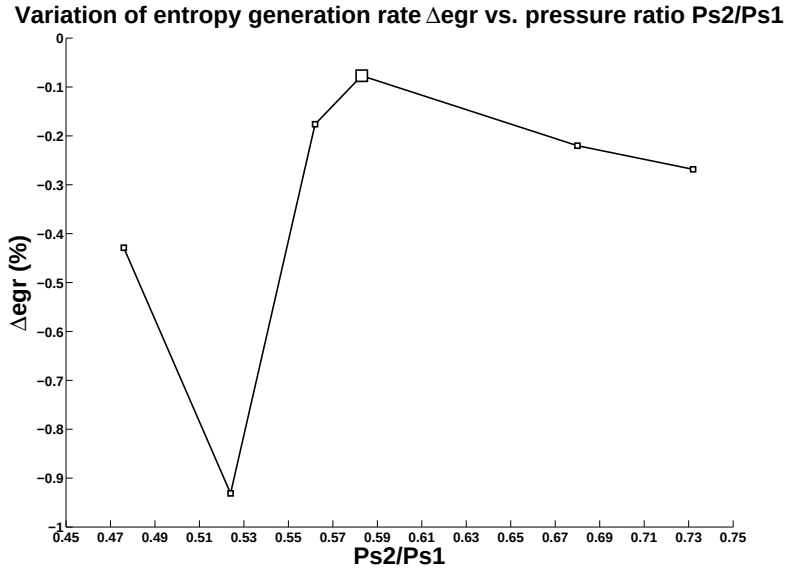

Figure 2.25: Variation of entropy generation rate Δegr with respect to the pressure ratio Ps2/Ps1. The nominal condition is represented with larger markers.

Figure 2.26 compares the mass flow rate of the initial and the redesigned LS89 blades for the pressure ratios selected by the GSA algorithm.

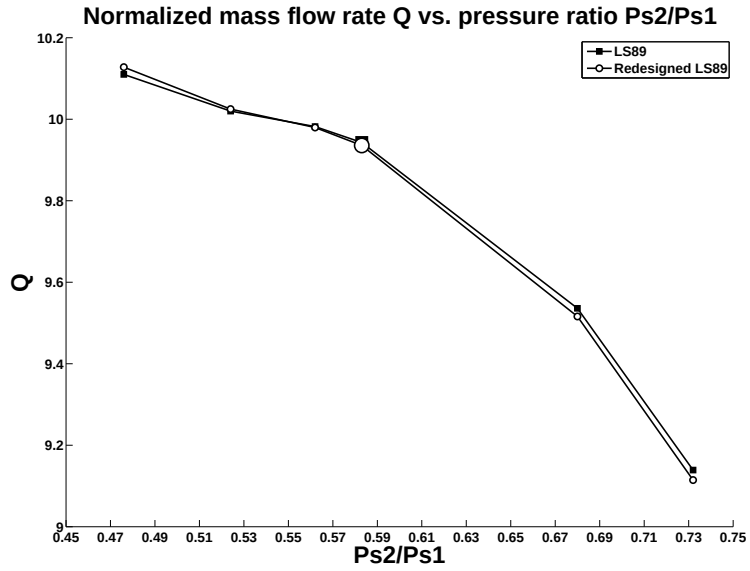

Figure 2.26: Comparison of the mass flow rate of the initial and the redesigned LS89 blades with respect to the pressure ratio Ps2/Ps1. The nominal condition is represented with larger markers.

The table 2.16 synthesizes the results on the interval of pressure ratio $[0.476\,,0.732]$.

Condition	**C21**	**C17**	**C14**	***nominal***	**C05**	**C01**
Pressure ratio	0.476	0.524	0.562	0.583	0.680	0.732
Δegr (%)	- 0.420	- 0.904	- 0.168	- 0.075	- 0.227	- 0.264
ΔQ (%)	+ 0.175	+ 0.040	- 0.026	- 0.072	- 0.210	- 0.273

Table 2.16: Variation of entropy generation rate and mass flow rate with respect to pressure ratio.

The entropy generation rate has been successfully reduced for all conditions. The mass flow rate fluctuations are acceptable. This is a good optimization.

Conclusion

In the present master thesis a numerical aerodynamic design technique for 2D turbine blades, based on a multipoint shape optimization with discrete adjoint method, has been developed and validated in an environment involving industrial numerical codes.

I worked first on a bibliography of both applied mathematics (optimization, Finite Volume Method, parametrization with non-uniform B-splines) and aerodynamics (turbine characteristics, turbulence models). I focused then on the choice of the test case flow characteristics (boundary conditions, Reynolds and Mach numbers, turbulence model) and on the sensitivity of the flow variables with respect to these characteristics. I have also analysed the sensitivity of the flow variables with respect to the numerical parameters of the CFD solver (number of iterations, CFL number) and to the turbulence model. Once the flow calculation was correctly set I have worked on the parametrization of the blade. A few different parametrizations have been tested from 18 to 53 variables, with and without frozen variables. One of the most important part of this work was the setting of the optimization problem. A few objective functions have been tested (aerodynamic drag, surface averaged pressure ratio, mass flow rate averaged pressure ratio) and with or without constraints formulations have been tried. The gradient of the objective function with respect to the design variables has been validated with finite differences and the choice of the optimization algorithm has been then driven by the setting of the optimization problem.

The single-point optimization has engendered a poor design, decreasing the objective function at the targeted condition but increasing it for some others around. The results of the multipoint optimization have been on the other hand very satisfying. The redesigned blades have lower entropy generation rate for all the conditions.

This method has now to be adapted to more complex configurations (e.g. 3D rotor turbine configurations). The discrete adjoint method does not have been validated at ***CERFACS*** for such rotative configurations. The entropy generation rate could be replace by the isentropic ratio. It would be also interesting to optimize with a larger set of conditions I_{Π}. An other possible work would be to perform the GSA algorithm at each iteration of the optimization process in order to see if the dimension of the $\mathcal{K}_{\underline{\boldsymbol{\alpha}}, N_{\Pi}}$ spaces varies during the optimization process.

Bibliography

[1] C.S. Ahn and K.Y. Kim. Aerodynamic design optimization of a compressor rotor with Navier-Stokes analysis. In Proceedings of the Institution of Mechanical Engineers, volume 217, pages 179–183, 2003.

[2] T. Arts, M. Lambert De Rouvroit, and A. W. Rutherford. Aero-thermal investigation of a highly loaded transonic turbine guide vane cascade. Technical Report 174, Von Karman Institute, 1990.

[3] B.S. Baldwin and H. Lomax. Thin layer approximation and algebraic model for separated turbulent flows. AIAA Paper, 1978-257, 1978.

[4] A. Barthet. Amélioration de la prévision des coefficients aérodynamiques autour de configurations portantes par la méthode adjointe. PhD thesis, Institut National Polytechnique de Toulouse, 2007.

[5] D.P. Bertsekas. Projected Newton methods for optimization problems with simple constraints. SIAM Journal of Control and Optimization, 20:221–246, 1982.

[6] M. Bompard. Modèles de substitution pour l'optimisation globale de forme en aérodynamique et méthode locale sans paramétrisation. PhD thesis, Université de Nice-Sophia Antipolis, 2011.

[7] C. G. Broyden. The convergence of a class of double-rank minimization algorithms. Journal of the Institute of Mathematics and Its Applications, 6:76–90, 1970.

[8] R.H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. Technical Report NAM-08, Northwestern University, 1994.

[9] L. Cambier and M. Gazaix. elsA : An efficient object-oriented solution to CFD complexity. AIAA Paper, 2002-0108, 2002.

[10] L. Cambier, M. Gazaix, S. Heib, S. Plot, P. Poinot, J.P. Veuillot, J.F. Boussuge, and M. Montagnac. CFD Platforms and Coupling : An Overview of the Multi-Purpose elsA Flow Solver. Aerospace Lab, Issue 2, March 2011.

[11] M. Cormery. De l'optimisation aérodynamique vers l'optimisation multidisciplinaire dans un contexte industriel. PhD thesis, Université Paul Sabatier de Toulouse, 2000.

[12] J. Cousteix. Turbulence et couche limite. Cépaduès, 1989.

[13] J.A. Desideri and A. Janka. Multilevel Shape Optimization for Aerodynamic Optimization. In European Congress on Computational Methods in Applied Sciences and Engineering in Jyvaskyla (Finland), 2004.

[14] R. Fletcher. A new approach to variable metric algorithms. Computer Journal, 13:317–322, 1970.

[15] F. Gallard, M. Meaux, M. Montagnac, and B. Mohammadi. Aerodynamic aircraft design for mission performance by multipoint optimization. In 21st AIAA Computational Fluid Dynamics Conference, pages –. American Institute of Aeronautics and Astronautics, June 2013. Design Optimization Techniques I.

[16] F. Gallard, M. Montagnac, B. Mohammadi, and M. Meaux. Robust parametric shape design by multipoint optimization. Technical Report TR-CFD-12-33, CERFACS, 2012.

[17] D. Goldfarb. A family of variable metric updates derived by variational means. Mathematics of Computation, 24:23–26, 1970.

[18] A. Harten and J.M. Hyman. Self-adjusting grid methods for one-dimensional hyperbolic conservation laws. Journal of Computational Physics, 50:235–269, 1983.

[19] A. Jameson. Aerodynamic Design via Control Theory. Journal of Scientifing Computing, 3(3):233–260, 1988.

[20] W.P. Jones and Launder B.E. The prediction of laminarization with a two-equation model of turbulence. International Journal of Heat and Mass Transfer, 15:301–314, 1972.

[21] Gaetan K. W. Kenway and Joaquim R. R. A. Martins. Multi-point High-fidelity Aerostructural Optimization of a Transport Aircraft configuration. Journal of Aircraft, 2013.

[22] J. Laurenceau. Surface de Réponse par Krigeage pour l'Optimisation de formes Aérodynamiques. PhD thesis, INP Toulouse, 2008.

[23] W. Li, L. Huyse, and Padula S. Robust airfoil optimization to achieve consistent drag reduction over a Mach range. Structural and Multidisciplinary Optimization, 24(1):28–50, 2002.

[24] M. Michel, C. Quémard, and R. Durant. Application d'un schéma de longueur de mélange à l'étude des couches limites turbulentes d'équilibre. Technical Report 154, ONERA, 1969.

[25] S.K. Nadarajah and A. Jameson. A comparaison of the continuous and discrete adjoint approach in automatic aerodynamic optimization. AIAA Paper, 00-0667, 2000.

[26] E.J. Nielsen and W.K. Anderson. Aerodynamic design optimization on unstructured meshes using the Navier-Stokes equations. AIAA Journal, 37:1411–1419, 1999.

[27] J. Nocedal. Updating quasi-newton matrices with limited storage. Mathematics of Computation, 35(151):773–782, 1980.

[28] C. Pham. Linéarisation du flux visqueux des équations de Navier-Stokes et de modèles de turbulence pour l'optimisation aérodynamique en turbomachines. PhD thesis, Ecole Nationale Supérieure d'Arts et Métiers, 2006.

[29] S. Pierret and R.A. Van den Braembussche. Turbomachinery blade design using a Navier-Stokes solver and artificial neural network. Journal of Turbomachinery, 121:326–332, 1999.

[30] O. Pironneau. On optimum profiles in Stokes flows. Journal of Fluid Mechanics, 59(01):117–128, 1973.

[31] G. Puigt, M. Gazaix, M. Montagnac, M.C. Le Pape, M. De la Llave Plata, C. Marmignon, J.F. Boussuge, and C. Couaillier. Development of a new hybrid compressible solver inside the CFD elsA software. In Proceedings of the 20th AIAA Computational Fluid Dynamics Conference, 2011.

[32] D. Quagliarella and A.D. Cioppa. Genetic algorithms applied to the aerodynamic design of transonic airfoils. Journal of Aircraft, 32(4):889–891, 1995.

[33] J. Reuther, A. Jameson, J. Farmer, L. Martinelli, and D. Saunders. Aerodynamic shape optimization of complex aircraft configurations via an adjoint formulation. In Proceedings of the 34th AIAA Aerospace Sciences Meeting and Exhibit in Reno (USA), 1996.

[34] O. Reynolds. An experimental investigation of the circumstances which determine whether the motion of water shall be direct and sinous, and of the law of resistance in parallel channels. Proceedings of the Royal Society of London, 35:84–99, 1883.

[35] N. Rochuon. Analyse de l'écoulement tridimensionnel et instationnaire dans un compresseur centrifuge avec fort taux de pression. PhD thesis, Ecole centrale de Lyon, 2007.

[36] P.L. Roe. Approximate Riemann solvers, parameter vectors and difference schemes. Journal of Computational Physics, 43:357–372, 1981.

[37] J.A. Samareh. Novel Multidisciplinary Shape Parametrization Approach. Journal of Aircraft, 38:1015–1024, 2001.

[38] D.F. Shanno. Conditioning of quasi-newton methods for function minimization. Mathematics of Computation, 24:647–656, 1970.

[39] H. Sobieczky. Parametric Airfoils and Wings. Notes on Numerical Fluid Mechanics, 68:71–88, 1998.

[40] T.W. Soderberg and S.R. Perry. Free-form Deformation of polygonal data. In Proceedings of the International Electronic Image Week in Nice (France), pages 633–639, 1986.

[41] P.R. Spalart and S.R. Allmaras. A one-equation turbulence model for aerodynamic flows. AIAA Paper, 1992-0439, 1992.

[42] G.D. Van Albada, B. Van Leer, and W.W. Roberts. A comparative study of computational methods in cosmic gas dynamics. Astronomy and Astrophysics, 108:76–84, 1982.

[43] B. Van Leer. Towards the ultimate conservative difference schemes. Journal of Computational Physics, 32:101–136, 1979.

[44] D.X. Wang and L. He. Adjoint aerodynamic design optimization for blades in multistage Turbomachinery. Journal of Turbomachinery, 132, 2010.

[45] D.C. Wilcox. A two-equation turbulence model for wall-bounded and free-shear flows. AIAA Paper, 1993-2905, 1993.

[46] H. Wu, F. Liu, and H. Tsai. Aerodynamic design of turbine blades using an adjoint equation method. AIAA Paper, 05-1006, 2005.