INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Shape optimisation in unsteady flows

E. Laporte, P. Le Tallec

No 3693

Mai 1999

THÈME 4

ISSN 0249-6399

Shape optimisation in unsteady flows

E. Laporte, P. Le Tallec

Thème 4 — Simulation et optimisation
de systèmes complexes
Projet M3N

Rapport de recherche n ° 3693 — Mai 1999 — 34 pages

Abstract: In aerodynamics, most of shape optimisation methods are gradient based methods. But cost functions are often very sophisticated and gradients are difficult to compute. We study herein numerical technics for calculating gradients of such cost functions for unsteady flows. One of the main problems is the non-differentiability of the mesh generation step. We propose here a way of solving this problem using automatic differentiation and mesh adaption

Key-words: aerodynamic design, shape optimisation, unsteady flows, automatic differentiation, mesh adaption

(Résumé : tsvp)

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
Téléphone : 01 39 63 55 11 - International : +33 1 39 63 55 11
Télécopie : (33) 01 39 63 53 30 - International : +33 1 39 63 53 30

Optimisation de forme pour écoulements instationnaires

Résumé : En aérodynamique, la plupart des méthodes d'optimisation sont des méthodes de gradients. Mais les fonctions de coût sont souvent très sophistiquées, et leurs gradients difficiles à calculer. Nous étudions ici des techniques numériques pour le calcul de gradients de fonctions de coûts calculées sur des écoulements instationnaires. L'un des principaux problèmes est l'étape de génération du maillage qui est une partie non différentiable de l'algorithme. Nous proposons une approche de résolution des différents problèmes basée sur la différentiation automatique et l'adaptation de maillage.

Mots-clé : Aérodynamique, optimisation de formes, écoulements instationnaires, différentiation automatique, adaptation de maillage

Introduction

This paper is concerned with the study of shape gradients properties in order to set up mesh independent strategies of optimum design. It also considers the calculation of the gradient of cost functions computed on unsteady flows. We want to show that mesh adaption or other techniques which change mesh topology can be used in good agreement with optimum design, and that cost functions gradients can be computed with enough accuracy to handle difficult problems, such as unstady/aeroelastic problems.

We present our problem's formulation for both steady and unsteady flows. Then we present our gradient based optimisation strategy. We also describe the computation of the gradient with respect to a mesh, and the way we handle topological changes in the mesh during the optimisation, by using arbitrary smooth lifting operator. We show how we can derive numerical schemes to solve adjoint equations from the schemes used to solve direct problems, either steady or unsteady, with the help of automatic differentiation.

All these strategies are validated by numerical results, obtained on steady problems (adjoint states and mesh adaption) and unsteady problems (including an aeroelatic problem).

1 Problem's formulation

1.1 A steady problem

A generic problem in aerodynamics is to minimise the drag of an airfoil with respect to its shape γ under some given constraints (cf. fig. 1). The constraints can be either geometrical (volume, ...) or aerodynamic (lift, ...) constraints. The flow around the airfoil is charac-

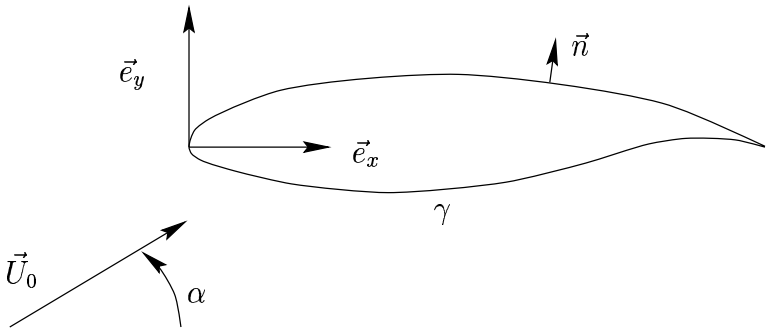


Figure 1: Airfoil in an unbounded domain ($\vec{U}_0$ = velocity in the free stream)

terised by the state variables $W_\gamma(t, \vec{x})$ defined for all $\vec{x}$ outside γ and satisfying the state

equation

$$\frac{\partial W_\gamma}{\partial t} = E\left(t, \gamma, W_\gamma\right), \tag{1}$$

where E can be the Euler or the Navier-Stokes equations. For periodic problems of period T, we add the following conditions:

$$W_\gamma\left(0, \vec{x}\right) = W_\gamma\left(T, \vec{x}\right) \quad \forall \vec{x},$$

and we use cost functions of the form

$$j\left(\gamma\right) = \int_0^T J\left(\gamma, W_\gamma\left(t\right)\right). \tag{2}$$

For example, J can represent the drag around the body γ in a flow characterised by the state W_γ. In steady cases, the state equation can be written as

$$E\left(\gamma, W_\gamma\right) = 0, \tag{3}$$

and the cost function reduces to

$$j\left(\gamma\right) = J\left(\gamma, W_\gamma\right). \tag{4}$$

The constraints can be mechanical functions like the drag, which can be written as (2) or (4), or functions like the volume, which is a function of γ only.

As seen later, the shape γ will be defined by a set of parameters

$$\alpha = \left(\alpha_i\right)_{i=1,\dots,p} \in \mathbb{R}^p,$$

with $p \in \mathbb{N}$.

1.2 An aeroelastic periodic problem

A major problem in turbine blades design is aerostability. We propose here to optimize this stability by controlling the energy exchanged between the fluid and the structure during the eigenmotion of the structure.

For this purpose, let us consider the turbine blade described on figure 2. We first compute its vibrational modes by solving the classical structural eigenvalue and eigenmode problem on Ω_{int} : find $\vec{u} \in \left(H^1\left(\Omega_{int}\right)\right)^2$ and $\lambda \in \mathbb{R}$ such as

$$a\left(\vec{u}, \vec{v}\right) = \lambda\left(\vec{u}, \vec{v}\right) \quad \forall \vec{v} \in \left(H^1\left(\Omega_{int}\right)\right)^2, \tag{5}$$

where a is the linear elasticity operator. For isotropic materials, this operator reduces to

$$a\left(\vec{u}, \vec{v}\right) = \lambda_L \int_{\Omega_{int}} \left(\operatorname{div} \vec{u}\right)\left(\operatorname{div} \vec{v}\right) d\Omega_{int} + 2\mu_L \sum_{i,j=1}^{2} \int_{\Omega_{int}} \epsilon_{ij}\left(\vec{u}\right) \epsilon_{ij}\left(\vec{v}\right) d\Omega_{int},$$

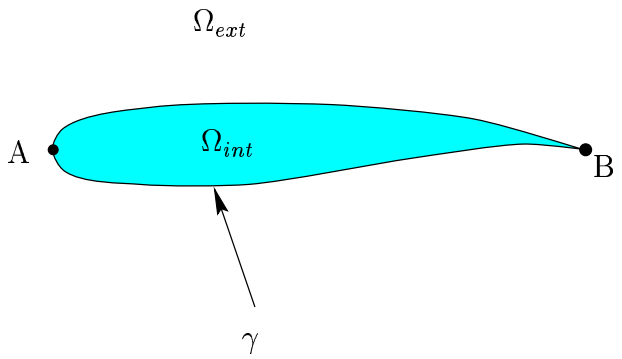


Figure 2: Ω_{int} and Ω_{ext}

with λ_L and μ_L denoting the Lamé coefficients, and

$$\epsilon_{ij}(\vec{v}) = \frac{1}{2}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right)$$

the linearized strain tensor. Moreover, $(\vec{u}, \vec{v})$ represents the mass operator, which is typically of the form

$$(\vec{u}, \vec{v}) = \int_{\Omega_{int}} \rho \vec{u}.\vec{v}\, d\Omega_{int}.$$

If we add a condition such as $\vec{u} = \vec{0}$ on a part of the boundary of $\hat{\gamma}$, this structural problem is well posed.

We denote λ_1 the smallest eigenvalue and $\vec{u}_1$ the corresponding eigenmode. Then we compute the periodic flow $W(t)$ around the blade when the boundary of the airfoil vibrates on its fundamental mode according to $\vec{u}\sin(\omega t)$ (cf. fig. 3), where

$$\omega = \omega_0\sqrt{\frac{\lambda_1}{\lambda_0}}, \quad T = \frac{2\pi}{\omega},$$

and ω_0 and λ_0 are non-dimensional constants.

The relevant cost function to optimize the stability is then of the form

$$\frac{1}{T(\gamma)}\int_0^{T(\gamma)} J(\gamma, W(t))\, dt, \tag{6}$$

where the period T depends here on λ_1 and hence on the shape γ. To improve the stability of the airfoil, we choose the energy transferred from the fluid to the airfoil as the cost function $J(\gamma, W(t))$ to be minimized. The mean value of this function should be as negative as

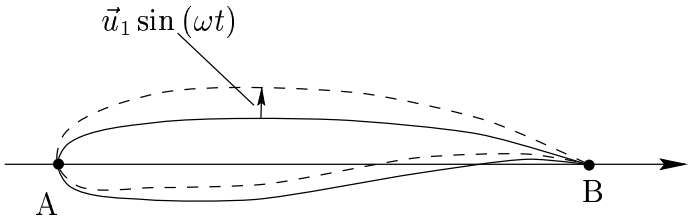


Figure 3: Deformation of the airfoil

possible. By construction, J is given by

$$J(\gamma, W(t)) = -\int_{\sigma(t)} p(t)\,\vec{n}(t).\vec{V}(t)\; d\sigma(t), \tag{7}$$

where we note:

- $\vec{d}(t,.) = \vec{u}_1(.)\sin(\omega t)$ the deformation applied to the airfoil.
- $\sigma(t,.) = \gamma(.) + \vec{d}(t,.)$ the airfoil position at time t;
- $\vec{V}(t,.) = \frac{\partial}{\partial t}\vec{d}(t,.) = \omega\vec{u}_1(.)\cos(\omega t)$ the speed of airfoil boundary;
- $\vec{n}(t,.)$ the unit outwards normal vector;
- $p(t,.)$ the pressure applied on the airfoil $\sigma(t,.)$;

2 Gradient based optimisation strategy

2.1 Discretisation

The numerical computation of the gradient of the functions (2) or (4) can be achieved in a rather flexible and efficient way by describing the computational domain $\Omega = \Omega_{ext} \cup \Omega_{int}$ through its computational grid, whose evolution can be easily governed by a finite number of control parameters $\alpha = (\alpha_i)_{i=1,\dots,p} \in \mathbb{R}^p$. The choice of these control parameters may be problem and solver dependent, but once done, the optimisation problem can be fully described through the numerical solvers to be used during its solution. The full description now consists of

1. a reference configuration $\Omega^0 \in \Omega_{adm}$, and the corresponding contour γ^0;
2. a map $\gamma(\alpha, \gamma^0)$ defining the shape γ of the contour from its initial shape and from the normal motion $(\alpha_i)_{i=1,p}$ of carefully chosen control points (cf. for example § 3.3);

3. a computational grid $\{X(\alpha) = (x_i)_{i=1,ns}, NU = (NU_{i,l})_{i=1,ndloc,l=1,nt}\}$ defined on the configuration Ω delimited by γ describing the position $X \in \mathbb{R}^3$ of the ns grid nodes, and giving for each cell $l = 1$ to NT the set $NU_{.,l}$ of its $ndloc$ vertices. In this construction, the shape of the object is entirely characterised by its computational grid, and the parameters α controlling the shape are in fact only used to control the grid deformation;

4. the discretisation scheme

$$E_h(X, W) = 0 \text{ in } \mathbb{R}^q$$

used for approximating the state equation $E = 0$ on the computational grid X, and for defining the approximate discrete solution $W \in \mathbb{R}^q$ of the state equation. For example, after finite element discretisation, the structural eigenproblem of our aeroelastic example takes the form

$$\text{Find } \lambda_1, \vec{u}_{1,h} = \sum_{i=1}^{3\times ns} \vec{V}^i \varphi_i \text{ such that}$$

$$K(X_{int})\vec{V}_1 = \lambda_1 M(X_{int})\vec{V}_1,$$

$$< \vec{V}_1, M(X_{int})\vec{V}_1 >= 1,$$

with

$$K(X_{int})_{ij} = \int_{\Omega_{int}} E\varepsilon(\varphi_i) : \varepsilon(\varphi_j),$$
$$M(X_{int})_{ij} = \int_{\Omega_{int}} \rho\varphi_i \cdot \varphi_j.$$

Above, $\varphi_{3(l-1)+k}$ denotes the piecewise polynomial shape function taking the value 1 in the direction $\vec{e}_k$ at node l and with zero values at all other degrees of freedom.

With this description, the design constraints $g(\gamma) \leq 0$ can be also expressed as a functions of the present position of the grid points X and can be reduced to the generic form

$$g_{h_i}(\alpha, X) \leq 0, \forall 1 \leq i \leq m.$$

Altogether, the shape opitmisation problem is then reduced to the approximate problem

$$(P_h) \min_{\alpha \in \mathbb{R}^p, g_h(\alpha, X) \leq 0} j_h(\alpha) = J(X(\alpha), W_h(\alpha)),$$

where $W_h(\alpha)$ denotes the solution of the discrete state equation

$$(E_h) \begin{cases} E_h(X, W_h) = 0 \text{ in } \mathbb{R}^q, \\ X = X(\alpha). \end{cases}$$

This new formulation is convenient for two main reasons:

- it has the form of a classical finite dimensional constrained optimisation problem and therefore can be approached by standard techniques of mathematical programming;
- all functions appearing in this formulation can be explicitly calculated on a computer.

On the other hand, this formulation ignores the natural topology of the problem under study and is dependent on the quality of the discretisation strategy which is used. This can be overcome by using adaptive discretisation techniques, as done in several recent research developments [6]. The only generic difficulty consists in the construction of the map $\alpha \mapsto X(\alpha)$ describing the evolution of the grid as a function of the control parameters α. This construction is described for example in [6]and in § 2.3.

2.2 Basic minimisation strategy

The above finite dimensional optimisation problem can now be solved by any standard gradient based algorithm. The corresponding flowchart is:

1. let $\alpha_n = (\alpha_{n,i})_{i=1,\dots,p} \in \mathbb{R}^p$ be the current value of the design parameters;
2. compute the gradient $\frac{dj_h}{dX}$ of the cost function with respect to the coordinates using for example Lagrangian techniques:

$$\frac{dj_h}{dX} = \frac{\partial L}{\partial X}\left(X, \mathcal{W}, \bar{\mathcal{W}}\right),$$

   where $\mathcal{W}$ denotes the full vector of the state variables of the problem (aerodynamic and mechanic), $\bar{\mathcal{W}}$ the vector of the adjoint variables and L the Lagrangian of the optimisation problem;
3. compute the gradient $\frac{dj_h}{d\alpha} = \frac{dj_h}{dX}\frac{dX}{d\alpha}$ of the cost function with respect to the parameters;
4. update α by $\alpha_{n+1} = \alpha_n - S_n^{-1}\frac{dj_h}{d\alpha}$.

Above S_n is any convenient definite positive approximation of the Hessian of j (cf. for example [5]).

2.3 Mesh generation

Generating a mesh of Ω is usually based on the existence of a preexisting mesh $X^0 = \left(\left(x_i^0\right), NU^0\right)$ of the underlying reference configuration. The map γ defines then the motion of the boundary grid points $(x_i)_{i\in\gamma}$.

Two strategies can then be followed: the first is an explicit lifting propagating the motion of the boundary nodes to the inside nodes through an adequate averaging operator. For example, if we write

- δ_k the motion of the boundary node x_k^0;

- w_k given coefficents for each $x_k^0 \in \gamma^0$;
- $\beta \geq 2$ an arbitrary coefficient;
- for each $x_i^0 \notin \gamma^0$,

$$c_{ki} = \frac{1}{|x_k^0 - x_i^0|^\beta},$$

and

$$c_i = \sum_{x_k^0 \in \gamma^0} w_k c_{ki},$$

then the operator $\mathcal{R}_h$ defined for each $x_i^0 \notin \gamma^0$ by

$$\mathcal{R}_h\left(x_i^0, \delta\right) = \frac{1}{c_i} \sum_{x_k^0 \in \gamma^0} w_k c_{ki} \delta_k$$

can be used for this purpose. This operator is robust but time consuming (computational time proportional to the total number of nodes times the number of boundary nodes). Another lifting operator, less time consuming but less robust, is given by:

1. the motion of all the inside nodes is set to zero;
2. for each element, compute the average motion of the nodes of the element;
3. for each inside node, compute the average motion of the elements which the node belongs to;
4. iterate step (ii) and (iii).

The second strategy to generate a mesh of Ω is adaptative remeshing where nodes are added or removed, and edges are swapped in order to fulfill an accuracy requirement defined simultaneously on the state equation and on the adjoint equation (cf. for example [4]).

3 Control parameters and mesh strategy

3.1 Theoretical result

The difficult part in the above algorithm is to obtain the gradient $\frac{dj}{d\alpha}$ from the gradient computed with respect to the mesh coordinates. Several techniques already exist. The most popular one was described in the previous section and consists in using a smooth deformation operator which computes a deformation field on the mesh from control parameters or from a given deformation on the boundary. We have then a completely differentiable function

$X = X(\alpha)$. The main drawback of this technique is that no topological change in the mesh is possible. If the mesh is poor at the beginning of the optimisation process, then it will stay poor during all the process.

We propose here to use the fact that, at the analytical level, the gradient with respect to a shape does not depend on what happens inside the domain where we solve our PDE [10], e.g. if $\theta_1 = \theta_2$ on $\partial\Omega$, then

$$\left\langle \frac{dj}{d\Omega}, \theta_1 \right\rangle = \left\langle \frac{dj}{d\Omega}, \theta_2 \right\rangle .$$

This is no more true on the discrete level (that is when working with cost j_h), but we can prove that under some assumptions, this property remains valid at the discrete level, at least at the limit where the discretisation step h goes to zero. Let us take an example: suppose that we have constructed a mesh X with a discretisation step h, a cost function $j_h(X) = J(X, W(X))$, where $W(X)$ is the solution of a classical Dirichlet problem with a finite element method. It has been proved [7] that, for all admissible θ,

$$\left\langle \frac{dj_h}{d\Omega}, \theta \right\rangle = \left\langle \frac{dj}{d\Omega}, \theta \right\rangle + \underset{h\to 0}{\mathrm{O}}(h) .$$

Suppose also that the shape γ defining the boundary of the open set Ω_γ, to which the mesh X is associated, is parametrised by a set of parameters α

$$\alpha = (\alpha_i)_{i=1,\dots,p} \in \mathbb{R}^p ,$$

with $p \in \mathbb{N}$, and that $\alpha \mapsto \gamma = \gamma(\alpha)$ is differentiable.

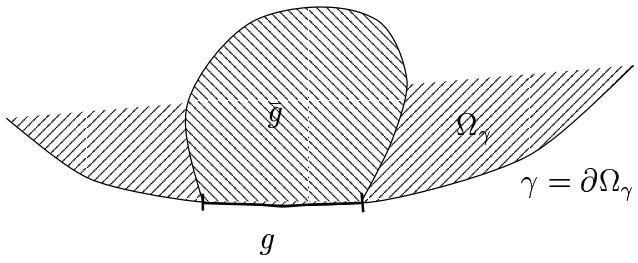


Figure 4: Lifting of a function

Furthermore, we choose a given explicit lifting operator such as those described earlier[1], e.g. a function $\mathcal{R} : H^{\frac{1}{2}}(\partial\Omega_\gamma) \mapsto H^1(\Omega_\gamma)$, that constructs from a function $g \in H^{\frac{1}{2}}(\partial\Omega_\gamma)$ a

[1] Note that we use here the same tool for two different purposes: lifting a function from its definition on a boundary and moving a mesh from a given motion on its boundary. We underline here the fact that these two actions are completely different, even if in certain cases, the displacement of the mesh can be included in the cost function, and the lifting can be viewed as the adjoint operator of the displacement.

function $\bar{g} = \mathcal{R}(g) \in H^1(\Omega_\gamma)$ such that

$$\bar{g} = g \quad on \quad \partial\Omega_\gamma.$$

Then we can show that, if Ω_γ is convex, then, for $i = 1, \ldots, p$, we have

$$\left\langle \frac{dj_h}{dX}, \mathcal{R}\left(\frac{\partial\gamma}{\partial\alpha_i}\right) \right\rangle = \frac{\partial\tilde{j}}{\partial\alpha_i} + O_{h\to 0}(h),$$

where $\tilde{j}$ is the analytical cost function $\alpha \in \mathbb{R}^p \mapsto \tilde{j}(\alpha) = j\left(\Omega_{\gamma(\alpha)}\right) \in \mathbb{R}$. This means that we are no longer dependent on the choice of $\mathcal{R}$. Thanks to this property, we do not care any more about the non-differentiability of the mesh generator and we can nevertheless get a good approximation of the gradient in Ω_γ, by setting

$$\frac{\partial j}{\partial\alpha_i} = \left\langle \frac{dj}{dX}, \mathcal{R}\left(\frac{\partial\gamma}{\partial\alpha_i}\right) \right\rangle. \tag{8}$$

We can then re-mesh or adapt the mesh when needed, or still use a smooth deformation. The figure 5 shows the global gradient computation algorithm based on this remeshing strategy.

3.2 Numerical experiments for the periodic heat equation

We want here to validate the computation of the gradient of a periodic cost function. In this example, the state equation is the T-periodic heat equation *inside* a bounded open set Ω of boundary $\gamma = \partial\Omega$: find u_Ω such that

$$\begin{cases} \frac{\partial u_\Omega}{\partial t} - \Delta u_\Omega = f(t) & \text{in } \Omega, \\ u_\Omega(t,.) = 0 & \text{on } \gamma = \partial\Omega \quad \forall t \in]0,T[, \\ u_\Omega(0,.) = u_\Omega(T,.) & \text{in } \Omega. \end{cases}$$

The analytical cost function is given by

$$j(\Omega) = \int_0^T \int_\Omega u_\Omega^2 \, d\Omega \, dt.$$

We can show [3, 9, 10, 14] that if $\partial\Omega$ is locally lipschitz, then j is differentiable, and if u_Ω is smooth enough (at least in $H^2(\Omega)$), then

$$\left\langle \frac{dj}{d\Omega}, \theta \right\rangle = \int_0^T \int_{\partial\Omega} \frac{\partial u}{\partial n}\frac{\partial v}{\partial n}(\theta.n) \, d\gamma \, dt,$$

where $\theta \in \left(W^{1,\infty}(\Omega)\right)^2$, and v is the solution of the adjoint state equation:

$$\begin{cases} -\frac{\partial v_\Omega}{\partial t} - \Delta v_\Omega = 2u(t) & \text{in } \Omega, \\ v_\Omega(t,.) = 0 & \text{on } \gamma = \partial\Omega \quad \forall t \in]0,T[, \\ v_\Omega(0,.) = v_\Omega(T,.) & \text{in } \Omega. \end{cases}$$

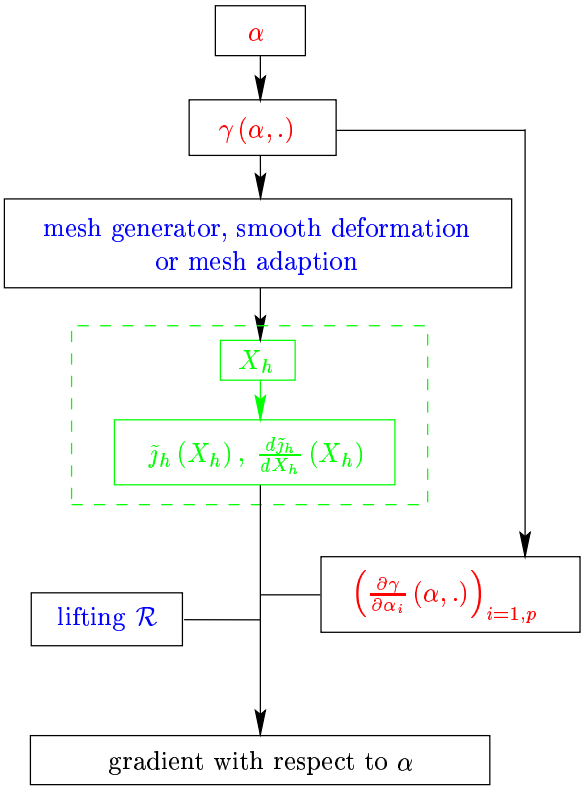


Figure 5: Global gradient computation algorithm

We can then compare the analytical gradient $\frac{dj}{d\Omega}$ and the discrete gradient $\frac{dj_h}{dX}$: for a mesh described by $X = (x_i)_i$ with $x_i = \left(x_i^1, x_i^2\right)$, if we introduce the continuous piecewise linear function φ_i such that $\varphi_i(x_i) = 1$ and $\varphi_i(x_j) = 0$ for $j \neq i$, and

$$\xi_i^1 = \begin{pmatrix} \varphi_i \\ 0 \end{pmatrix}, \quad \xi_i^2 = \begin{pmatrix} 0 \\ \varphi_i \end{pmatrix},$$

then we have

$$\frac{\partial j_h}{\partial x_i^j} = \left\langle \frac{dj}{d\Omega}, \xi_i^j \right\rangle \quad \forall i, j.$$

On figure 6, we compared the numerical gradient and the analytical gradient for a periodic problem of period $T = 1$, on the square $\Omega =]0,1[\times]0,1[$, with

$$f\left(t, x^1, x^2\right) = \left[\frac{\omega^2}{2} + 2\pi^4\right] \sin(\omega t) \sin\left(\pi x^1\right) \sin\left(\pi x^2\right).$$

The size of the discretisation step h is $\frac{1}{20}$ on the straight edges. We can then show that, for a vertex $x_i = \left(x_i^1, x_i^2\right)$ on $\{0\} \times]0,1[$, we have for example

$$\left\langle \frac{dj}{d\Omega}, \xi_i \right\rangle = \begin{pmatrix} -\frac{h}{4}\pi^4 + \frac{\pi^2}{16h}\left[2\cos\left(2\pi x_i^2\right) - \cos\left(2\pi\left(x_i^2 + h\right)\right) - \cos\left(2\pi\left(x_i^2 - h\right)\right)\right] \\ 0 \end{pmatrix}$$

The values of the whole analytical gradient are described by the figure on the right in figure 6.

The discrete gradient on the left part of figure 6 was computed using the method described in 4.3.

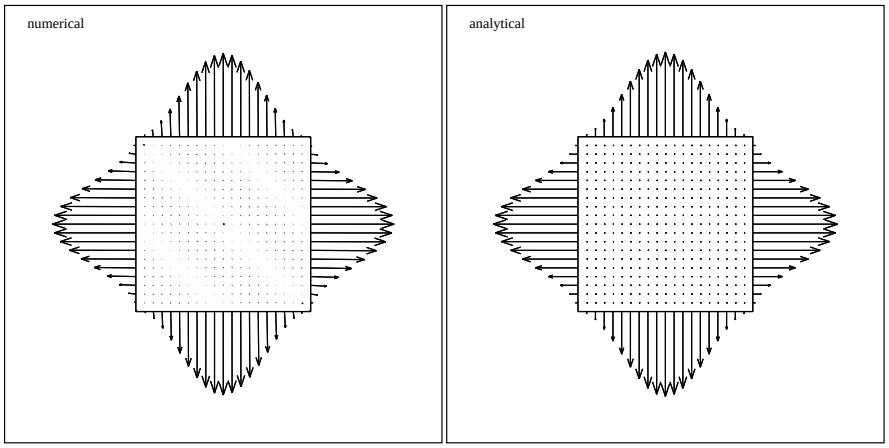


Figure 6: Comparison between numerical (left) and analytical (right) gradients

3.3 Numerical experiments for the problem of the Laplacian

We want to evaluate the effects of mesh adaption on the speed of convergence on our optimisation algorithm. In this example, we solve the Dirichlet problem for a Poisson problem set *inside* a bounded open set (which looks like an airfoil). The cost function is

$$j(\gamma) = \int_{\Omega_\gamma} \left(u_\gamma - u_0\right)^2,$$

with u_γ solution of the Dirichlet problem

$$\begin{cases} u_\gamma \in H_0^1\left(\Omega_\gamma\right), \\ -\Delta u_\gamma = f \text{ on } \Omega_\gamma, \end{cases}$$

where Ω_0 is a given open set, γ_0 is the boundary $\partial\Omega_0$ of Ω_0 and u_0 is defined by

$$\begin{cases} u_0 \in H_0^1(\Omega_0), \\ -\Delta u_0 = f \text{ on } \Omega_0, \\ u_0 = 0 \text{ outside } \Omega_0. \end{cases}$$

In this problem, the shape γ_0 is the target solution of the optimisation problem (cf. fig. 7).

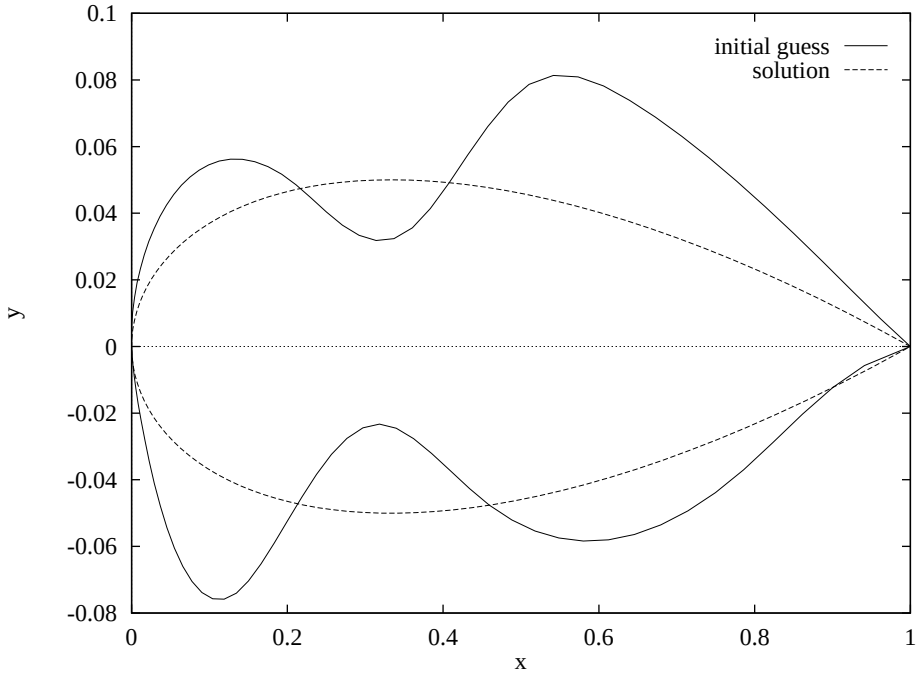


Figure 7: Initial guess and target shape (solution) γ_0

The parametrisation is defined by $\gamma(\alpha, t) = \gamma_0(t) + \psi(\alpha, t)\, n_0(t)$, $t \in [0, 2[$, where n_0 is the normal to the fixed shape γ_0, and ψ is a parametrised scalar function. We choose p real numbers $t_1, \ldots, t_p$ such that

$$t_1 = 0 < t_2 < \ldots < t_{p-1} < t_p = 2,$$

and $t \mapsto \psi(\alpha, t)$ is the piecewise polynomial function, continous with continuous first and second derivatives, such that

$$\begin{cases} \psi(\alpha, t_{i+1}) = \alpha_i, \quad i = 1, \ldots, p, \\ \frac{\partial \psi}{\partial t}(\alpha, t_1) = \alpha_{p-1} \\ \frac{\partial \psi}{\partial t}(\alpha, t_p) = \alpha_p. \end{cases}$$

It has been shown that this kind of parametrisation is natural for shape optimisation [1, 11]. In the example treated below, we use $p = 16$ shape parameters, and γ_0, u_0 and f are given by

$$\gamma_0(t) = \begin{pmatrix} (t-1)^2 \\ 0,13t(t-1)(t-2) \end{pmatrix}, \quad t \in [0, 2[,$$

$$u_0(x, y) = y^2 - 0,0169x(x-1)^2,$$

and

$$f(x, y) = 1,014 \times x - 2,676.$$

We use an interior point algorithm [5], with box constraints on the shape parameter α and with a constraint on the volume. Indeed, without any constraint, the algorithm can converge in some case to the trivial solution of a void open set.

On figure 5, we see that we can choose between:

- moving smoothly the mesh of the initial shape;
- creating a new mesh for each new shape with a standard mesh generator;
- creating a new mesh for each new shape with mesh adaption,

and we can also choose the lifting $\mathcal{R}$ used in the calculation (8). For this latter, we will simply use the two lifting operators seen in § 2.3. In the present experiment, they are numbered 2 for the explicit lifting and 3 for the iterative one. The lifting 1 that we will use simply consists in considering that the support of the lifting is included between the boundary and the first layer of mesh nodes around the boundary, and the resulting lifting has a value 0 on any node of the mesh which is not on the boundary. This can be a reasonnable hypothesis on the continuous level, but not on the discrete level in matter of convergence. Nevertheless, we will see that choosing this lifting does not really affect the results.

The figures 8 and 9 show the convergence of the value of the cost function and the square of the Lagrangian norm towards 0 for various liftings $\mathcal{R}$ (1, 2 and 3) and when a new mesh is created for each new shape, either by a standard mesh generator (M) or by mesh adaption (A). This gives six curves for each variable (cost function value or Lagrangian norm) whose names are M1, M2, M3, A1, A2 and A3. The shape optimisation with a smooth deformation

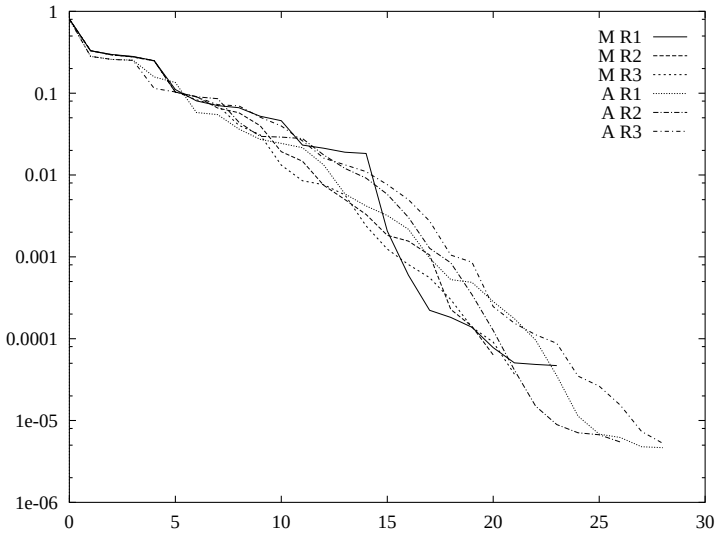


Figure 8: Cost function values with respect to the number of optimisation iterations

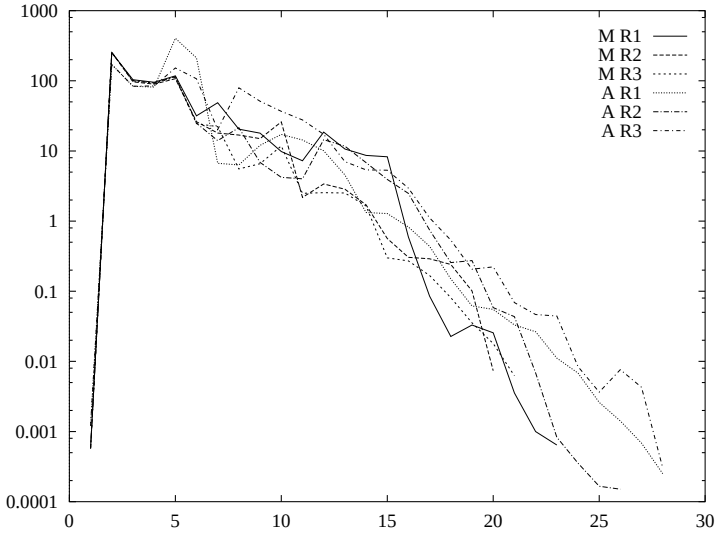


Figure 9: Square of the Lagrangian norm with respect to the number of optimisation iterations

of the initial mesh was excluded: it always led to mesh tangling because of our choice of initial guess (cf. fig. 7). This proves the necessity of robust gradient computations!

We can see that the convergence properties of the interior point algorithm are not really affected by our different choices of strategies. Nevertheless, in some cases, the line search (done by the minimisation algorithm) becomes delicate especially when the instabilities due to the discretisation are greater than the variations of the analytical function. To avoid this problem, new strategies can be used [12].

4 Computing the gradient of an aerodynamical cost function with respect to a mesh

4.1 Automatic Differentiation

Automatic Differentiation is a tool which takes the Fortran subroutine computing a function and gives a Fortran subroutine computing the derivatives of this function (cf. table 1). We use here the automatic differentiator Odyssée which has already been successfully used to compute derivates of large functions [2]. We mostly use the *reverse mode* (last column of

math. func.	$y = f(x)$	$dxdy = \frac{\partial f}{\partial x}$	$xccl = \left\langle \left(\frac{\partial f}{\partial x}(x)\right)^*, yccl \right\rangle$
Routines	`f(x,y)`	`fd(x,y,dydx)`	`fcl(x,y,xccl,yccl)`
In. var.	`x`	`x`	`x,yccl`
Out. var.	`y`	`y,dydx`	`xccl`
Diff. mode		direct	reverse
	Odyssée in.	Odyssée output	

Table 1: Odyssée 's input-output

table 1), which is adapted to compute gradients of scalar functions depending on a very large number of input variables.

4.2 The fluid solver

To solve the state equations (1) or (3) and to compute the corresponding cost function (2) or (4), we use the `NSC2KE` software [8]. This software integrates the flow equations using an explicit time scheme and a finite volume-Galerkin upwind technique using a Roe Riemann solver for the convective part of the equations. The viscous terms (when needed) are treated using a standard Galerkin technique.

After discretisation, the state equations (without changing notation) become,

$$E(X, W) = 0 \tag{9}$$

for the steady case (3) and

$$-P(X).\left(W^{k+1} - W^k\right) + \Delta t\, E\left(X, W^k\right) = 0 \tag{10}$$

for the unsteady case (1). In both cases, X is the vector of the mesh vertices which replaces γ as a description of the shape within the software, and $P(X)$ is a diagonal matrix whose every diagonal coefficient $P(X)_{i,i}$ is the cell area of the vertex i. In the steady case, we use local time stepping (Δt_i) as a preconditionner, and solve (9) by a time marching algorithm of the form

$$-P(X).\left(W^{k+1}-W^k\right)+\Delta t_i\, E\left(X,W^k\right)=0.$$

4.3 Gradient calculation

We now proceed to compute the gradient of the cost function with respect to mesh coordinates.

The steady case

Let us first study the steady case. The Lagrangian corresponding to our constrained minimisation problem writes

$$L(X,W,p)=J(X,W)+p.E(X,W)$$

where p is the adjoint variable associated to W. After differentiation, the gradient of the discrete cost function j_h becomes

$$\frac{dj_h}{dX}=\frac{\partial L}{\partial X}=\frac{\partial J}{\partial X}+p.\frac{\partial E}{\partial X},$$

where p satisfies the *adjoint equation*

$$\frac{\partial L}{\partial W}=\frac{\partial J}{\partial W}+p.\frac{\partial E}{\partial W}=0. \tag{11}$$

To solve (11), we can use the fact that the explicit time scheme (10) converges because the spectral radius of $(P(X))^{-1}.\frac{\partial E}{\partial W}$ is less than one. Then this also holds for the transpose operator $\left(\frac{\partial E}{\partial W}\right)^*.(P(X))^{-1}$, and since $P(X)$ is symmetric, this ensures that the adjoint scheme

$$P(X).\left(p^{k+1}-p^k\right)=\Delta t_i\left(\left(\frac{\partial E}{\partial W}\right)^*.p^k+\frac{\partial J}{\partial W}\right) \tag{12}$$

converges. With such a method, we avoid computing huge matrices like $\frac{dW}{dX}$, and we are guaranteed to converge with the same number of iterations as needed for the resolution of the steady state equation with the scheme (10).

We can see on figure 10 the convergence history, i.e. the value of the residual with respect to the number of iterations, for the initial scheme (10) which solves the state equation, and for the new scheme (12) which solves the adjoint equation. The figure displays two convergence histories for the scheme (12): in the first case, we compute the adjoint variables when the cost function is the drag, in the second case, the cost function is the lift.

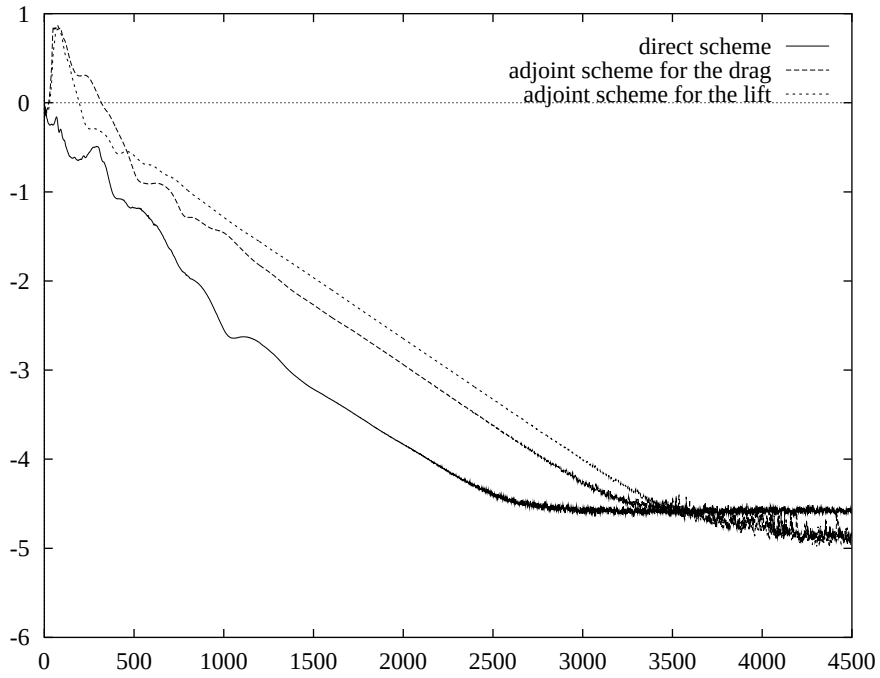


Figure 10: Convergence histories of the direct scheme (10) and of the adjoint scheme (12).

The periodic case

The periodic problem is more difficult. The Lagrangian now writes

$$\begin{aligned} L(X,W,p) &= \int_0^T J(X,W)\,dt + \int_0^T p(t).\left(-P(X).\frac{\partial W}{\partial t} + E(t,X,W)\right)dt \\ &\quad + p(0).P(X).(W(T)-W(0)) \\ &= \int_0^T J(X,W)\,dt + \int_0^T \frac{\partial p}{\partial t}.P(X).W(t) + p(t)\,E(t,X,W)\,dt \\ &\quad + (p(0)-p(T)).P(X).W(T). \end{aligned}$$

In this case, the gradient becomes

$$\frac{dj_h}{dX} = \int_0^T \frac{\partial J}{\partial X}\,dt + \int_0^T \frac{\partial p}{\partial t}.\frac{dP}{dX}.W(t) + p(t)\,\frac{\partial E}{\partial X}\,dt, \tag{13}$$

where p is solution of the periodic adjoint equation defined in a weak form by

$$\int_0^T \frac{\partial J}{\partial W} U(t)\, dt \quad + \quad \int_0^T \frac{\partial p}{\partial t}.P(X).U(t) + p(t)\frac{\partial E}{\partial W} U(t)\, dt \\ + U(T).P(X).(p(0) - p(T)) = 0, \quad \forall U.$$

In strong form, this reduces to

$$\frac{\partial J}{\partial W} + P(X).\frac{\partial p}{\partial t} + \left(\frac{\partial E}{\partial W}\right)^* p(t) = 0 \tag{14}$$

together with the periodicity condition

$$p(0) = p(T). \tag{15}$$

As the operator with nice convergence properties is $-P(X)\frac{\partial}{\partial t} + \left(\frac{\partial E}{\partial W}\right)^*$ (observe the minus sign before the first operator), the time variable is replaced by a reverse time variable $\bar{t} = T - t$, and we finally solve

$$P(X).\frac{\partial \bar{p}}{\partial \bar{t}} = \left(\frac{\partial E}{\partial W}\big(T - \bar{t}, X, W(T - \bar{t})\big)\right)^* .\bar{p}(\bar{t}) + \frac{\partial J}{\partial W}\big(X, W(T - \bar{t})\big) \tag{16}$$

and

$$\bar{p}(0) = \bar{p}(T),$$

where $\bar{p}(\bar{t}) = p(T - \bar{t})$. We can see in (16) that we now need to know $W(t)$ on the whole period, and as the adjoint equation needs to be solved backwards in time, we cannot solve both equations simultaneously. Up to now, the only known technique to fix this problem is an appropriate incomplete storage of the state variables.

Remark 1 *In the above procedure, all gradients $\frac{\partial J}{\partial W}$ or $\frac{\partial E}{\partial W}$ are computed using automatic differentiation.*

The aeroelastic case

Let us study the discrete version of the problem described in 1.2. We suppose now that we have a mesh X_{int} inside Ω_{int}. The smallest eigenvalue and normalised eigenmode of the structural problem is first obtained by solving

$$\begin{cases} \frac{\alpha^2}{2} - \frac{1}{2}\left\langle M(X_{int})\vec{V}_1, \vec{V}_1 \right\rangle = 0, \\ \alpha\left[-\lambda_1 M(X_{int}) + K(X_{int})\right]\vec{V}_1 = \vec{0}. \end{cases}$$

Here $\vec{V}_1$ is the eigenmode corresponding to λ_1, α is a normalisation constant for the mode $\vec{V}_1$, $M(X_{int})$ is the mass matrix and $K(X_{int})$ the stiffness matrix associated to problem (5),

when discretized say by a P_1-Lagrange finite elements method. We add to this problem the boudary conditions $\vec{V}_1 = \binom{V_1^1}{V_1^2} = \binom{0}{0}$ in A and $V_1^2 = 0$ in B, so that the problem has a unique solution.

Then we set

$$\omega = \omega_0 \sqrt{\frac{\lambda_1}{\lambda_0}}, \quad T = \frac{2\pi}{\omega},$$

$$\vec{d}(t) = \vec{V}_1 \sin(\omega t), \quad \vec{V}(t) = \frac{d}{dt}\vec{d}(t) = \omega \vec{V}_1 \cos(\omega t), \tag{17}$$

and we add these new variables in the input variables of the residual term E in equation (1) in order to take into account the periodic deformation of the airfoil given by $\vec{d}(t)$. The new state equation characterizing the flow is now:

$$\begin{cases} W^\infty(0) = W^\infty(T), \\ P(X_{ext}) \frac{\partial W^\infty}{\partial t}(t) = E\left(X_{ext}, \vec{d}(t), \vec{V}(t), W^\infty(t)\right) \text{ on }]0,T[. \end{cases} \tag{18}$$

With this full set of state variables, we can define a generic cost function as

$$j(X_{int}, X_{ext}) = \frac{1}{T}\int_0^T J\left(X_{ext}, \vec{d}(t), \vec{V}(t), W^\infty(t)\right) dt.$$

We can see that we have now two meshes: X_{int} for the structure and X_{ext} for the flow. We will consider that they came from a global (possibly non conforming) mesh X and that we have two linear operators $\Pi_{ext} : \mathbb{R}^{2\times ns} \to \mathbb{R}^{2\times ns_{ext}}$ and $\Pi_{int} : \mathbb{R}^{2\times ns} \to \mathbb{R}^{2\times ns_{int}}$ such that X_{ext} and X_{int} are given restrictions of X:

$$X_{ext} = \Pi_{ext}(X) \quad \text{and} \quad X_{int} = \Pi_{int}(X).$$

By writing

$$j_0(X) = j(\Pi_{int}(X), \Pi_{ext}(X)),$$

we have a global cost function whose gradient is given by

$$\frac{dj_0}{dX} = \frac{\partial j}{\partial X_{int}} \cdot \frac{d\Pi_{int}}{dX} + \frac{\partial j}{\partial X_{ext}} \cdot \frac{d\Pi_{ext}}{dX}.$$

We have now to compute $\frac{\partial j}{\partial X_{int}}$ and $\frac{\partial j}{\partial X_{ext}}$, and we can consider that X_{int} and X_{ext} are independant variables. The Lagrangian writes now

$$\begin{aligned}
&L\left(X_{int}, X_{ext}, \lambda_1, \vec{V}_1, \omega, T, \vec{d}, \vec{V}, W, \lambda_1^*, \vec{V}_1^*, \omega^*, T^*, \vec{d}^*, \vec{V}^*, p\right) \\
&\quad = \lambda_1^* \left[\frac{\alpha^2}{2} - \frac{1}{2}\left\langle M\left(X_{int}\right)\vec{V}_1, \vec{V}_1\right\rangle\right] + \alpha \vec{V}_1^* . \left[-\lambda_1 M\left(X_{int}\right) + K\left(X_{int}\right)\right] . \vec{V}_1 \\
&\qquad + \omega^*\left(\lambda_0 \omega^2 - \omega_0^2 \lambda_1\right) + T^*\left(\omega T - 2\pi\right) \\
&\quad + \frac{1}{T}\int_0^T \vec{d}^*\left(t\right) . \left(\vec{d}\left(t\right) - \vec{V}_1 \sin\left(\omega t\right)\right) dt + \frac{1}{T}\int_0^T \vec{V}^*\left(t\right) . \left(\vec{V}\left(t\right) - \omega \vec{V}_1 \cos\left(\omega t\right)\right) dt \\
&\qquad + \frac{1}{T}\int_0^T p\left(t\right) . \left(-P\left(X_{ext}\right) . \frac{\partial W}{\partial t} + E\left(X_{ext}, \vec{d}\left(t\right), \vec{V}\left(t\right), W\left(t\right)\right)\right) dt \\
&\qquad\quad + \frac{1}{T} p\left(0\right) . P\left(X_{ext}\right) . \left(W\left(T\right) - W\left(0\right)\right) \\
&\qquad\quad + \frac{1}{T}\int_0^T J\left(X_{ext}, \vec{d}\left(t\right), \vec{V}\left(t\right), W\left(t\right)\right) dt.
\end{aligned}$$

As seen previously, the flow equation can be integrated by parts which leads to

$$\begin{aligned}
&L\left(X_{int}, X_{ext}, \lambda_1, \vec{V}_1, \omega, T, \vec{d}, \vec{V}, W, \lambda_1^*, \vec{V}_1^*, \omega^*, T^*, \vec{d}^*, \vec{V}^*, p\right) = ... \\
&\quad + \frac{1}{T}\int_0^T \frac{\partial p}{\partial t} . \left(-P\left(X_{ext}\right) . W\left(t\right) + E\left(X_{ext}, \vec{d}\left(t\right), \vec{V}\left(t\right), W\left(t\right)\right)\right) dt \\
&\qquad + \frac{1}{T}\left(p\left(0\right) - p\left(T\right)\right) . P\left(X_{ext}\right) . W\left(T\right) + ...
\end{aligned}$$

Let us differentiate with respect to the state variables to get the adjoint equations, and let us start with the flow variables W. We find a problem analogous to (14)-(15):

$$\begin{cases}
p\left(0\right) = p\left(T\right), \\
-P\left(X_{ext}\right)\dfrac{\partial p}{\partial t}\left(t\right) = \left(\dfrac{\partial E}{\partial W}\left(X_{ext}, \vec{d}\left(t\right), \vec{V}\left(t\right), W^{\infty}\left(t, X_{ext}\right)\right)\right)^* p\left(t\right) \\
\qquad\qquad + \dfrac{\partial J}{\partial W}\left(X_{ext}, \vec{d}\left(t\right), \vec{V}\left(t\right), W^{\infty}\left(t\right)\right) \qquad \text{on }]0, T[,
\end{cases} \tag{19}$$

where W^{∞} is the periodic solution of (18). By differentiating with respect to $\vec{d}$ and $\vec{V}$, we have then the adjoint interface conditions

$$\vec{d}^*\left(t\right) + \left(\frac{\partial E}{\partial \vec{d}}\right)^* . p\left(t\right) + \frac{\partial J}{\partial \vec{d}} = 0 \quad \text{and} \quad \vec{V}^*\left(t\right) + \left(\frac{\partial E}{\partial \vec{V}}\right)^* . p\left(t\right) + \frac{\partial J}{\partial \vec{V}} = 0.$$

The differentiation with respect to the period T gives, by using the fact that some factors are equal to zero when the state equations are satisfied,

$$T^*\omega + \frac{1}{T}p\left(0\right).P\left(X_{ext}\right).\frac{\partial W^\infty}{\partial t}\left(T\right) + \frac{1}{T}J\left(X_{ext},\vec{d}\left(T\right),\vec{V}\left(T\right),W^\infty\left(T\right)\right) - \frac{1}{T^2}\int_0^T J\left(X_{ext},\vec{d}\left(t\right),\vec{V}\left(t\right),W^\infty\left(t\right)\right)\,dt = 0.$$

We then differentiate with respect to the pulsation ω, yielding

$$2\omega^*\lambda_0\omega + T^*T - \frac{1}{T}\int_0^T \vec{d}^*\left(t\right).\vec{V}_1 t\cos\left(\omega t\right)\,dt - \frac{1}{T}\int_0^T \vec{V}^*\left(t\right).\vec{V}_1\left(\cos\left(\omega t\right) - \omega\times t\sin\left(\omega t\right)\right)\,dt = 0.$$

Finally, we differentiate with respect to $\vec{V}_1$ then to λ_1 to get the adjoint structural equations:

$$-\lambda_1^* M\left(X_{int}\right)\vec{V}_1 + \alpha\left[-\lambda_1 M\left(X_{int}\right) + K\left(X_{int}\right)\right].\vec{V}_1^* - \frac{1}{T}\int_0^T \vec{d}^*\left(t\right)\sin\left(\omega t\right)\,dt - \frac{1}{T}\int_0^T \vec{V}^*\left(t\right)\omega\cos\left(\omega t\right)\,dt = 0$$

and

$$-\alpha\vec{V}_1^*.M\left(X_{int}\right).\vec{V}_1 - \omega^*\omega_0^2 = 0.$$

We can see that (19) can be solved by the same way as (14)-(15). Then, using $\omega T = 2\pi$, we have the explicit formulae:

$$T^* = -\frac{1}{2\pi}p\left(0\right).P\left(X_{ext}\right).\frac{\partial W^\infty}{\partial t}\left(T\right) - \frac{1}{2\pi}J\left(X_{ext},\vec{d}\left(T\right),\vec{V}\left(T\right),W^\infty\left(T\right)\right) + \frac{1}{2\pi}j\left(X_{int},X_{ext}\right),$$

and

$$\omega^* = \frac{1}{2\lambda_0\omega}\left\{-T^*T + \frac{1}{T}\int_0^T \vec{d}^*\left(t\right).\vec{V}_1 t\cos\left(\omega t\right)\,dt + \frac{1}{T}\int_0^T \vec{V}^*\left(t\right).\vec{V}_1\left(\cos\left(\omega t\right) - \omega\times t\sin\left(\omega t\right)\right)\,dt\right\}.$$

Finally, λ_1^* and $\alpha\vec{V}_1^*$ are solutions of the following system:

$$\begin{bmatrix} 0 & -{}^t\vec{V}_1.M\left(X_{int}\right) \\ -M\left(X_{int}\right).\vec{V}_1 & -\lambda_1 M\left(X_{int}\right) + K\left(X_{int}\right)\end{bmatrix}\begin{pmatrix}\lambda_1^* \\ \alpha\vec{V}_1^*\end{pmatrix} = \begin{pmatrix}\omega^*\omega_0^2 \\ \frac{1}{T}\int_0^T \vec{d}^*\left(t\right)\sin\left(\omega t\right)\,dt + \frac{1}{T}\int_0^T \vec{V}^*\left(t\right)\omega\cos\left(\omega t\right)\,dt\end{pmatrix}$$

When we have computed all the state variables and all their adjoint variables, it remains only to differentiate L with respect to X_{int} and X_{ext}. The case of X_{ext} is the same as the periodic case with fixed period, and $\frac{\partial j}{\partial X_{ext}}$ is thus given by (13). Concerning X_{int}, we have

$$\frac{\partial j}{\partial X_{int}} = -\frac{\lambda_1^*}{2}\left\langle \frac{dM}{dX_{int}}(X_{int})\,\vec{V}_1, \vec{V}_1 \right\rangle + \alpha \vec{V}_1^*.\left[-\lambda_1 \frac{dM}{dX_{int}}(X_{int}) + \frac{dK}{dX_{int}}(X_{int})\right].\vec{V}_1.$$

5 Numerical results

5.1 Steady computations

In the experiment described here, we start from the mesh of figure 11, the mesh of each new shape is adapted based on the mesh used for the previous shape, and the lifting is given by the iterative algorithm of § 2.3.

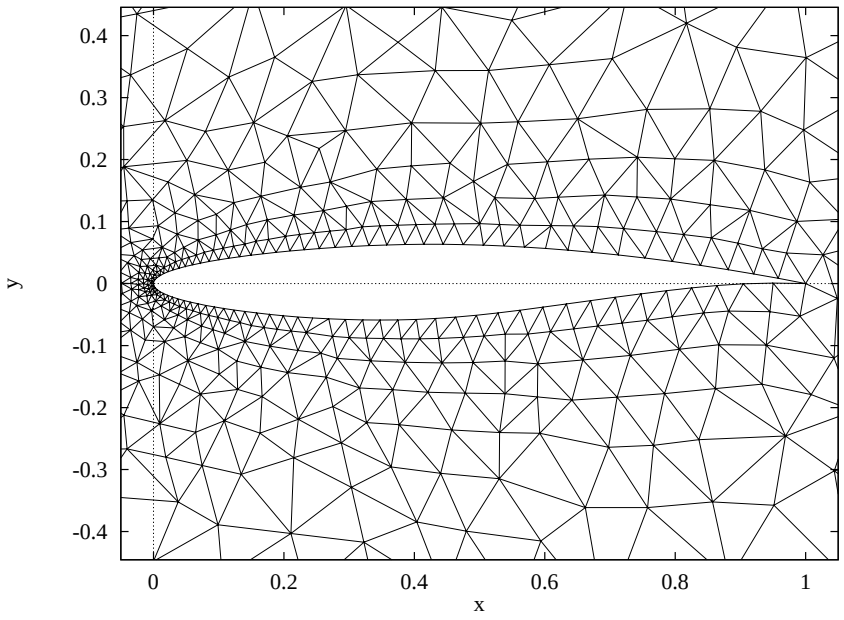


Figure 11: Initial mesh

The mesh adaptor is `BAMG` [4]. The metric used for the adaption is built according to the criterium of interpolation error: on the adapted mesh, the interpolation error of the choosen field should be lower than on the old mesh. The question is: what field shall we use to monitor this interpolation error? The answer given here is to compute the metric corresponding to the flow variables (density, x-velocity, y-velocity, energy) and to their adjoint variables, and

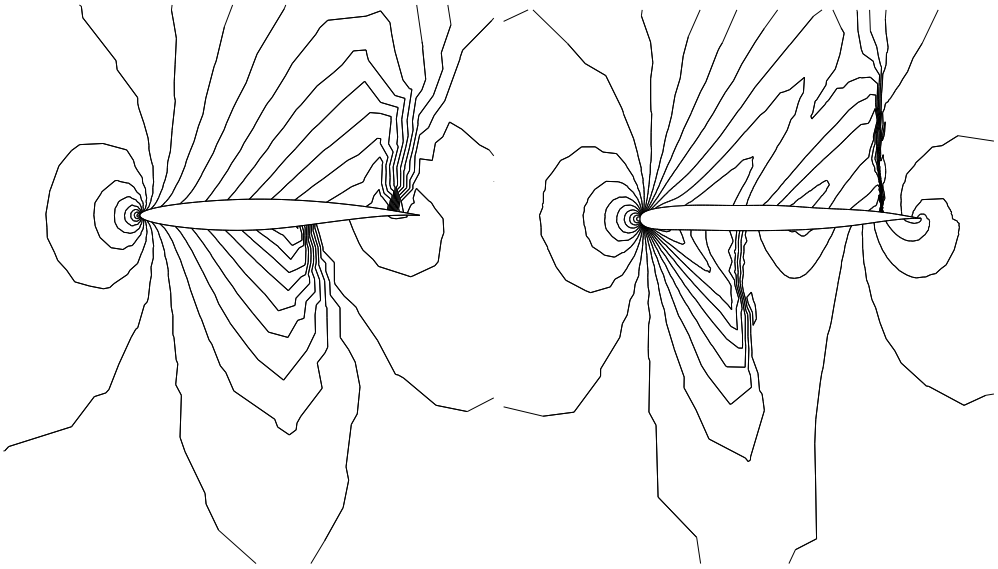
then to intersect all these metrics by taking the lowest length among those required for the different variables [13]. Indeed, if we look at the interpolation error of the gradient in the case of a Dirichlet problem, we can see that it depends both on the interpolation error on the state variable and also on the adjoint variable.

The flow equations are the steady Euler equation, the Mach number is 0.85. Parametrisation is the same as in 3.3. We have then 16 parameters. The cost function is the drag. We have 3 constraints:

- 2 inequality constraints:
  - the lift must remain greater than 95% of the initial lift;
  - the airfoil area must remain greater than 95% of the initial area;
- 1 equality constraint, which is a fixed point in the geometry. Here, the leading edge is fixed.

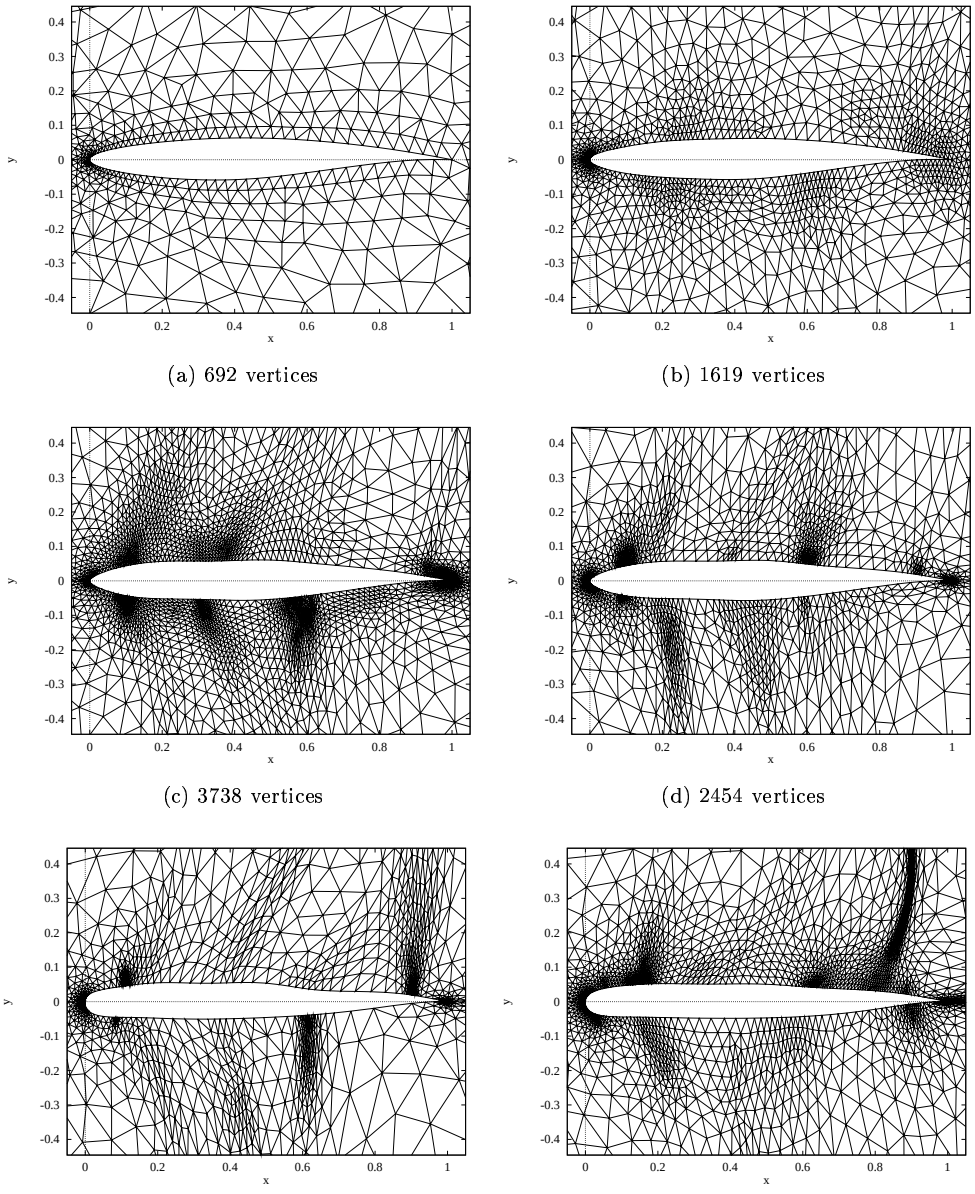
With this first mesh, we obtain a drag reduction greater than 50% in 5 optimisation iterations, and the constraint on the lift is satisfied. The figure 12 shows the inital and the final isolines of the density. We can see that the accuracy of the computation increased during the optimisation.

Figure 13 shows the various meshes obtained during the optimisation. The mesh (a) is the initial guess. To avoid the failure of the linear search, the number of vertices was limited to 6000.



(a) initial density (b) final density

Figure 12: Initial and final iso-density



(a) 692 vertices

(b) 1619 vertices

(c) 3738 vertices

(d) 2454 vertices

(e) 1930 vertices

(f) 3122 vertices

Figure 13: Meshes of each optimisation iteration

5.2 Unsteady computations

We give an exemple of result for the problem described page 4. The cost function is the function (6) with J given by (7). The pulsation ω and the deformation $\vec{d}(t,.)$ are obtained by solving the problem (5), with the following characteristics of the structure of the airfoil:

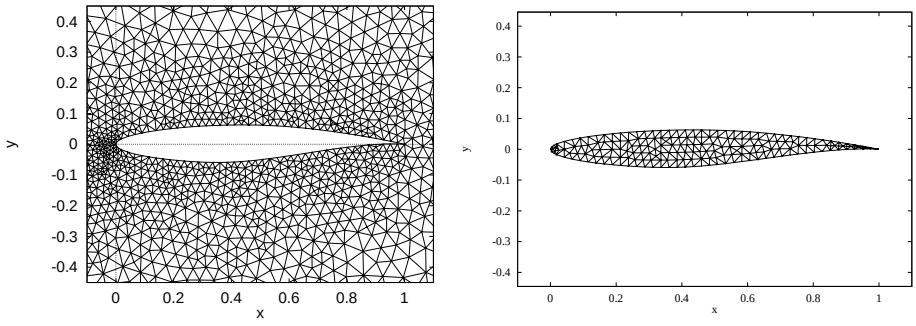
- a Young modulus of 217 MPa;
- a Poisson coefficient of 0.25;
- a density of 7800 kg/m^3.

These data are tipical of an IN 100 turbine blade (ONERA data). The corresponding Lamé coefficients are $\lambda_L = \mu_L = 86.8$ MPa.

The geometrical constraints are the same as in the steady case. We add in one of the two examples presented below a constraint on the mean lift $\frac{1}{T}\int_0^T C_L(\gamma,t)\,dt$, which must remain above 95% of its initial value, where $C_L(\gamma,t)$ is the instantaneous lift of the airfoil γ.

The flow is inviscid and the Mach number is 0.85.

The initial meshes are given by figure 14. Note that the boundary points of the two meshes are identical. During the optimisation, all the meshes are obtained by the explicit lifting described in § 2.3, and the lifting $\mathcal{R}$ used to compute the gradient according to the algorithm of figure 5 is the iterative lifting of § 2.3. The number of shape parameters is $p = 10$.



(a) initial mesh of the flow (2001 nodes)

(b) initial mesh of the structure (197 nodes)

Figure 14: Initial meshes

We present the results only with few comments because we did not find any comparison in the litterature.

Figure 15 shows the convergence of the cost for the problem with or without the constraint on the mean lift. We can see the small gain (11% without the constraint and 7% with it). Adding the constraint on the mean lift greatly decreased the speed of convergence of the minimisation algorithm. Moreover, figure 16 shows that the result of the optimisation without the constraint on the mean lift satisfies nevertheless this constraint. This shows once more the great dependency of the result with respect to the problem and the existence of several local minima.

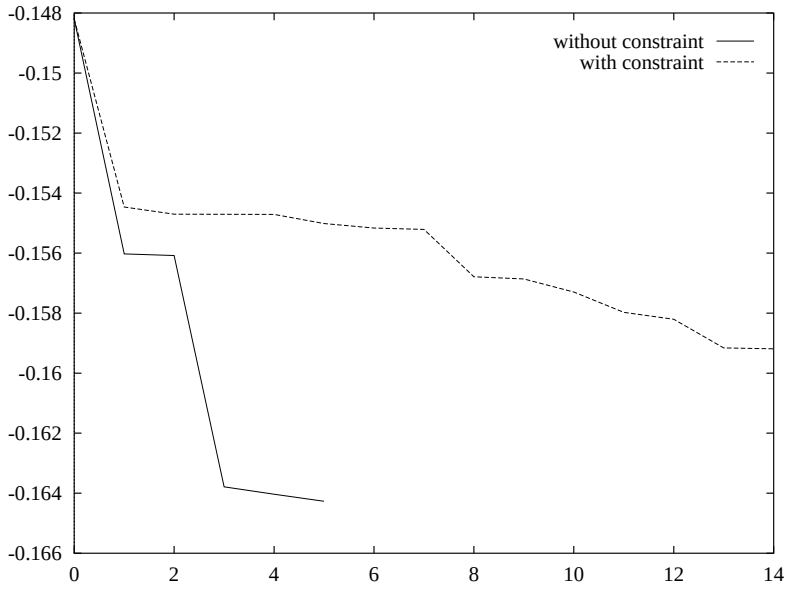


Figure 15: Cost convergence for problems with and witout the constraint on the mean lift

During the optimisation, we see a global decrease of the mean drag. Nevertheless, figure 17 shows that energy and drag are not correlated together. This shows the interest in minimizing not only the drag. We also see on figures 15, 17 and 19 that there are several main steps in the optimisation process, separated by large jumps. We can think that this is due to variations in the pulsation, which would mean that the function which gives the pulsation from the shape has discontinuities, but remains roughly piecewise differentiable.

We can also say that both optimisations were made with only 10 shape parameters. A greater number of shape parameter would have maybe given better performances, but on the other hand we might have oscillating shapes.

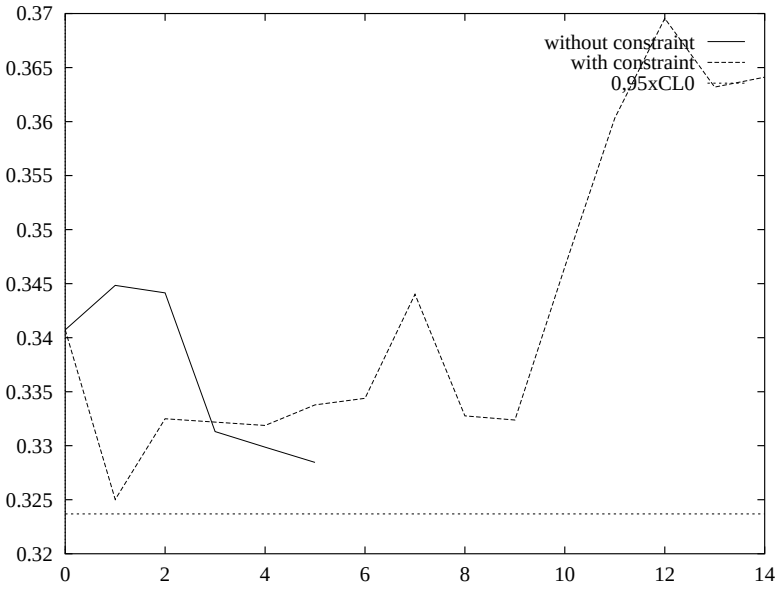


Figure 16: Mean lift during optimisations

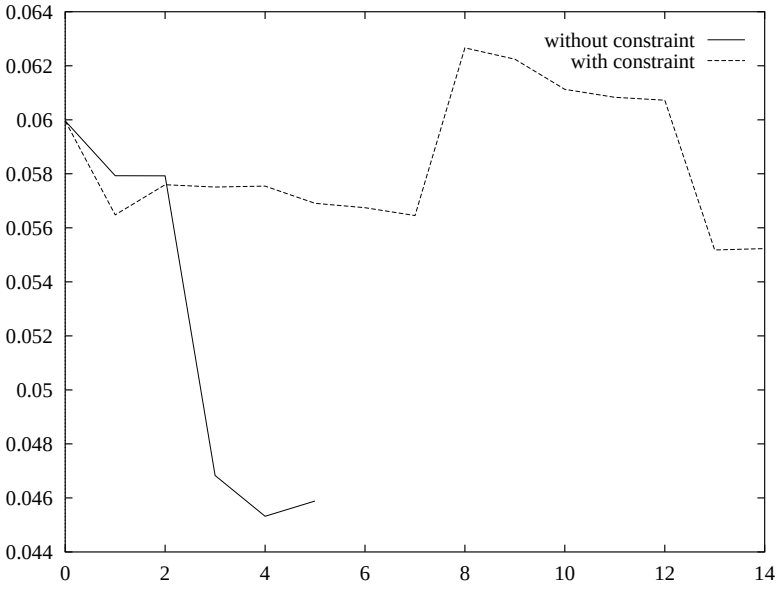


Figure 17: Mean drag during optimisations

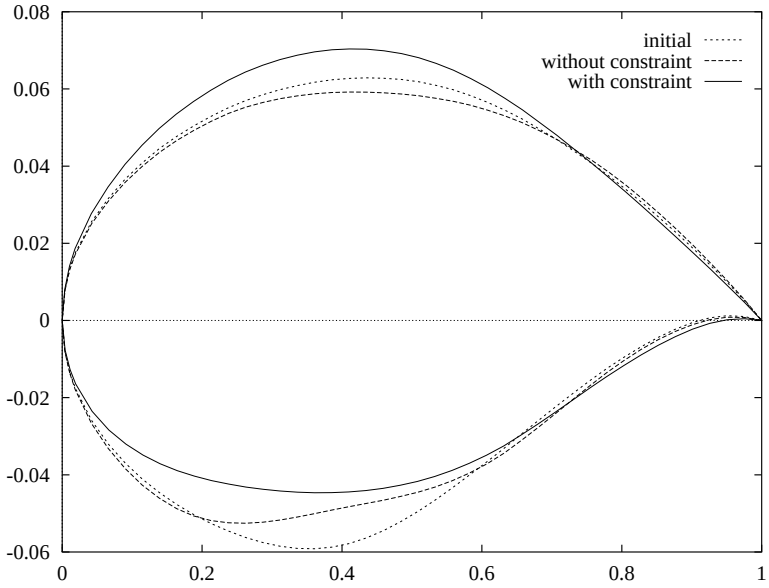
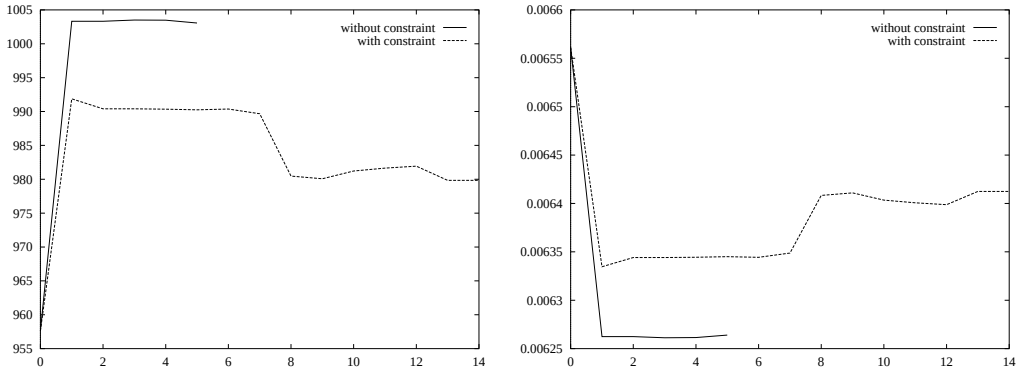


Figure 18: Intial and finals airfoils for the two problems



(a) pulsations (b) periods

Figure 19: Pulsations and periods during optimisations (physical values)

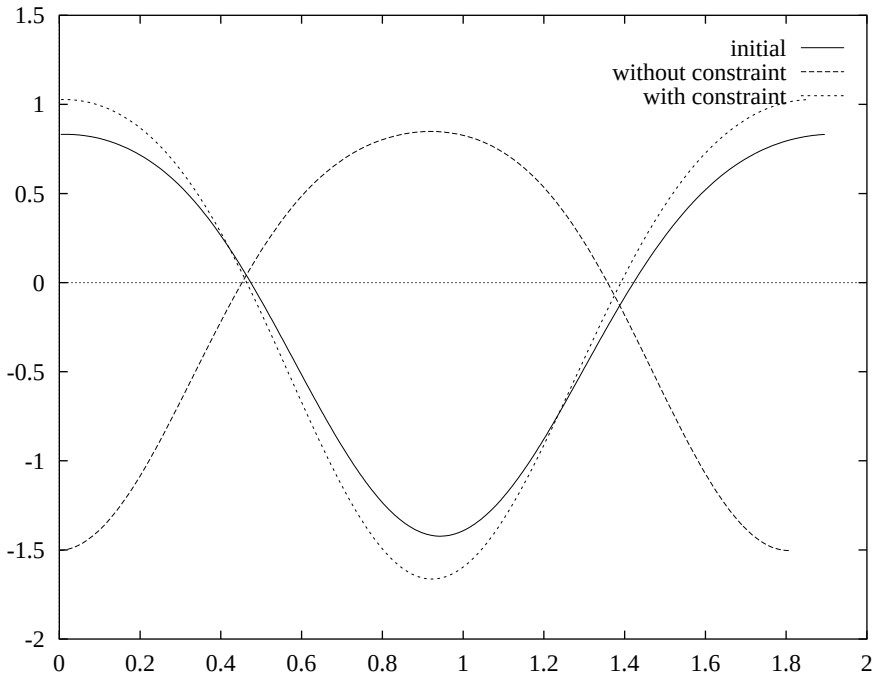


Figure 20: Instantaneous energy for intial and finals airfoils (with respect to the adimensionnal time)

6 Conclusion

We have seen in this report that a mesh independent strategy can be used in optimum design, without affecting too much the performance of the minimisation algorithm. Nevertheless, a lot remains to do in order to know what is the good way to control the number of vertices, the linear search and so on, to avoid the tuning of parameters and to have a good self-control of the minimisation algorithm.

For the aeroelastic problem, because of the numerical errors (coarse meshes of 2001 vertices), we do not pretend to have discovered physically meaning solutions. The most important is to have shown that the gradient of a complicated cost function can be computed with sufficient accuracy so that a minimisation algorithm can work properly, even when using an interior point algorithm which requires precision. Let us keep in mind that the cost function includes:

- an elasticity problem;
- an unsteady flow computation;
- an aeroelastic coupling.

Thus, the strategy seen in 4.3 allows to compute all the adjoint variables of the problem and the final gradient.

References

[1] Mohammed Barkatou and Antoine Henrot. Un résultat d'existence en optimisation de forme en utilisant une propriété géométrique de la normale. *ESAIM: Control, Optimisation and Calculus of Variations*, Vol. 2:pp. 105–123, 1997.

[2] Christèle Faure and Yves Papegay. O∂yssée version 1.6 - the user's reference manual. Technical Report RT-0211, INRIA Rocquencourt, november 1997.

[3] J.W. He and O. Pironneau. Optimisation du profil d'aile dans un fluide visqueux. Rapport de Recherche R94015, Laboratoire d'Analyse Numérique, Université Pierre et Marie Curie, 1994.

[4] Frédéric Hecht. `BAMG`: Bidimensional anisotropic mesh generator, mars 1998.

[5] J. Herskovits, E. Laporte, P. Le Tallec, and G. Santos. A quasi-newton interior point algorithm applied to constrained optimum design in computational fluid dynamics. *European Journal of Finite Elements*, Vol. 5(n. 5–6):pp. 595–517, 1996.

[6] E. Laporte. *Optimisation de formes pour écoulements instationnaires.* PhD thesis, Ecole Polytechnique, 1998.

[7] Mohamed Masmoudi. *Outils pour la Conception Optimale de Formes.* Thèse d'état, Université de Nice, 1987.

[8] Bijan Mohammadi. Fluid dynamics computation with `nsc2ke`: an user-guide: release 1.0. Technical Report RT-0164, INRIA Rocquencourt, 1994.

[9] Jérome Monnier. *Optimisation de Forme pour un Système Couplé Fluide-Thermique.* Thèse de 3e cycle, Université de Nice, 1995.

[10] François Murat and Jacques Simon. Sur le contôle par un domaine géométrique. Rapport de Recherche LAN76015, Laboratoire d'Analyse Numérique, Université Pierre et Marie Curie, 1976.

[11] Arian Novruzi. *Contribution en optimisation de formes et applications.* Thèse de 3e cycle, Université Henri Poincaré Nancy 1, september 1997.

[12] Elijah Polak. *Computational Methods in Optimization*, volume 77 of *Mathematics in Science and Engineering.* Academic Press, 1971.

[13] Ramm. Personnal communication, 1998.

[14] Bernard Rousselet. *Quelques Résultats en Optimisation de Domaines.* Thèse d'état, Université de Nice, décembre 1982.

Unit´e de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unit´e de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unit´e de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unit´e de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unit´e de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399