A Hierarchical approach for shape optimisation

François Beux, Alain Dervieux

Rapports de Recherche

N°1868

UNITÉ DE RECHERCHE
INRIA-SOPHIA ANTIPOLIS

Programme 6

Calcul scientifique, Modélisation et Logiciel numérique

A HIERARCHICAL APPROACH FOR SHAPE OPTIMIZATION

François BEUX
Alain DERVIEUX

Février 1993

Institut National
de Recherche
en Informatique
et en Automatique

2004 route des Lucioles
B.P. 93
06902 Sophia-Antipolis
France

A HIERARCHICAL APPROACH FOR SHAPE OPTIMIZATION

UNE APPROCHE HIERARCHIQUE EN
OPTIMISATION DE FORME

François BEUX, Alain DERVIEUX

INRIA Sophia-Antipolis
2004 Route des Lucioles 06360 VALBONNE
FRANCE

UNE APPROCHE HIERARCHIQUE EN OPTIMISATION DE FORME

Résumé

On considère l'application d'une méthode de gradient au contrôle optimal d'un système dont la simulation est coûteuse. Une stratégie s'inspirant des méthodes multiniveau est appliquée pour travailler avec un nombre croissant de paramètres. Cette stratégie est appliquée au problème académique de l'optimisation de la forme d'une tuyère pour des écoulements subsoniques et transsoniques régis par les équations d'Euler.

A HIERARCHICAL APPROACH FOR SHAPE OPTIMIZATION

abstract

We consider the gradient method applied to the optimal control of a system for which each simulation is expensive. A method for increasing the number of unknowns, and relying on multilevel ideas is tested for the academic problem of shape optimization of a nozzle in a subsonic or transonic Euler flow.

Contents

Introduction

The ultimate purpose of simulation studies of industrial process/products is the automated computer–aided optimization of some **parameters**, that are often **shapes**.

From this point of view, Automated Computer–Aided Shape Optimization is thought as an **external software optimization loop** calling another software level for **simulation** of one (or several) physical process(es) (ex: flow simulation, ...).

In many simplified cases, the optimization can be reduced to an inverse problem, that can, ideally, be just as costly as a single simulation; however, in the most general cases, it seems necessary to organize the optimization as a **succession of simulations with increasingly better parameters** deduced by an optimization software kernel. In that general case, the simulation code is considered as a **black box**; the most reluctant consequence, is that **finding N optimal parameters will be paid at least by N simulations**.

In practical cases this figure is much larger, so that either only simulations with rather simplified models can be used, either/or only a very small number of parameters is considered for optimization.

For aerodynamics, 2D Full Potential flows begin to be used ([2], [10], [11], [13], [6]); often less than 10 parameters are used.

If we examine more accurately the CPU cost of this kind of optimization, we note that for a standard optimizer, a cost that is a linear function of the number N of unknown parameters is an **ideal** situation that generally is not attained. Instead, the **problem stiffness increases** with the number N, and is paid by a N^α complexity ($\alpha > 1$).

An additional question is, starting from a shape optimization problem, **how to derive a parametrization** with an adhoc number N of unknowns.

Many answers to this question have been already proposed : smooth continuous curves can be used (Beziers, splines, ...), see for example ([11], [13], [6]) for recent studies. Another point of view is to solve inverse problems in order to obtain a sample of physically relevant shapes (e.g. "aerodynamic shapes", deduced from prescribed pressure distributions [1]).

This chapter presents a study of a hierarchical strategy in which the number of parameters is progressively increased. This is applied to an academic problem of nozzle inverse shape optimization.

A family of embedded parametrizations are build for describing smoothly the nozzle shape.

Although the ideas are rather simple, we think they have no been yet much tested, and we present a first set of comparisons between the hierarchical strategy and standard one, using the steepest (conjugate or not) gradient method, in the case of unconstrained optimization. We emphasize that faster optimizer such as GMRES could be applied [12]; this might deserve further study, in particular since the hierarchical strategy can be **combined** with fast optimizers ; however, in this paper, we stay at a basic level.

A last characteristic of the present study is that Euler flows are considered; we have shown in [4] that for a precise approximation, an adjoint method could be applied, yielding, by an analytical differentiation, a gradient for the functional.

In the case where a gradient is not analytically computed, finite-differences, relying on perturbations of parameter will produce (costly !) an "approximate gradient".

Both approaches (analytical gradient, approximate gradient) are considered in the sequel and the advantages of the hierarchical strategy are evaluated in both cases.

The plan is therefore the following : in Section 1, we formalize the interaction between a descent method and a parametrization.

The proposed hierarchical parametrization is depicted in Section 2, and the multilevel optimization strategies in Section 3.

Applying the method to solve (by optimization) a two–point boundary value problem is already an interesting experiment, as demonstrated by Section 4. In Section 5 we present the studied shape optimization problem. Section 6 is devoted to shape optimization with an analytical gradient, Section 7 is devoted to the "approximate gradient" case.

1 Parametrization in optimization

Let j be a real-valued functional defined on an Hilbert space E.
We consider the optimization problem :

$$Find\ \bar{u} \quad such\ that \quad j(\bar{u}) = \min_{u \in E} j(u) \ . \tag{1}$$

Let us consider the problem of introducing a parametrization in an optimization process.
Let F be a second Hilbert space and P a linear continuous mapping from F to E. Then we construct the following iterative algorithm :

$$\begin{cases} u_0 \quad given \\ \\ u_{n+1} = u_n - \rho P P^* \Lambda_E j'(u_n) \end{cases} \tag{2}$$

where $P^* \in \mathcal{L}(E,F)$ is the adjoint of P, Λ_E is the canonic isomorphism between E' and E and $j'(u_n)$ is the Fréchet derivative of j at u_n. For all $u \in E$, we define the gradient of j at u in the following way: $grad_E\, j(u) = \Lambda_E\, j'(u)$.

Lemma 1 :

Algorithm (2) is the application of the gradient method to the minimization problem:

$$Find\ \bar{v} \quad such\ that \quad joP(\bar{v}) = \min_{v \in F} joP(v) \ ; \tag{3}$$

Proof :

Let v_0 , v_1 be two elements of F, we have:

$$< grad_F\, joP(v_0), v_1 >_F = ((joP)'\,(v_0)\ ,\ v_1)_F \ = (j'(P(v_0))oP'(v_0)\ ,\ v_1)_F \quad ,$$

where the notation (,) represents the duality between a Hilbert space and his dual and $<\ ,\ >$ represents the scalar product of a Hilbert space.
Actually, using the linearity of P, we obtain :

$$\begin{aligned} < grad_F\, joP(v_0), v_1 >_F = & \ (j'(P(v_0))oP\ ,\ v_1)_F \\ = & \ (j'(P(v_0))\ ,\ P(v_1))_E \\ = & \ < grad_E\, j(P(v_0))\ ,\ P(v_1) >_E \\ = & \ < P^*\, grad_E\, j(P(v_0))\ ,\ v_1 >_F \quad . \end{aligned}$$

Thus we have proved that $P^* grad_E\, j(u)$ corresponds to the gradient of joP in the space F. Then the following algorithm solves (3) with a gradient method :

$$\begin{cases} v_0 \quad given \\ \\ v_{n+1} = v_n - \rho P^* \Lambda_E\, j'(P(v_n)) \end{cases}$$

Finally, applying the operator P we obtain :

$$P(v_{n+1}) = P(v_n) - \rho P P^* \Lambda_E\, j'(P(v_n)) \ ,$$

which corresponds to Algorithm (2).

Lemma 2 :

Algorithm (2) is a weak descent method for the optimization problem (1), i.e.

$$j(u_{n+1}) \leq j(u_n) \qquad for\ \rho_n\ small\ enough. \tag{4}$$

Proof :

From Taylor formula, we have :

$$j(u + Pv) = j(u) + < grad_E\, j(u), Pv >_E \ + o(v) \ .$$

Let ρ be a real strictly positive, and $v = -\rho P^* grad_E\, j(u)$, then we obtain:

$$j(u + Pv) = j(u) + < grad_E\, j(u), -\rho P P^* grad_E\, j(u) >_E \ + o(\rho)$$

$$j(u + Pv) = j(u) - \rho < P^* grad_E\, j(u), P^* grad_E\, j(u) >_F \ + o(\rho) \ .$$

From previous expressions, we deduce that if ρ is small enough we have $j(u + Pv) \leq j(u)$; i.e. with a good choice of the step ρ the property (4) is verified.

Remark:

We can notice that we have not a descent direction on a strict sense (i.e. $j(u_{n+1}) < j(u_n)$). In fact, we can have simultaneously $P^* grad_E\, j(u) = 0$ and $grad_E\, j(u) \neq 0$; in this case we obtain the solution of the minimization problem (3) which is not equivalent to the problem (1).

2 Hierarchical parametrization and interpolation

We are interested in using a set of parametrizations involving different numbers of parameters; the number of parameters describes the **coarseness** of the parametrization or level. Coarser and finer levels will be applied alternatively like in a multigrid process. The purpose of this section is to show how these different parametrizations can be mathematically derived from finer ones by a mechanism of **interpolation**. The simplest interpolation we can have is the **linear interpolation**; let x_j a new node located inside $[x_i\,,\,x_{i+1}]$, we have :

$$(Pf)_j = \overline{\alpha}_j f_i + \alpha_j f_{i+1}\ , \quad \text{with } \alpha_j = \frac{x_j - x_i}{x_{i+1} - x_i} \text{ and } \overline{\alpha}_j = 1 - \alpha_j \ .$$

Let l be the number of parameters on the coarser level and m the total parameter number ($m \geq l > 0$), we have then $P \,\epsilon\, \mathcal{L}(I\!R^l, I\!R^m)$.
This is a particular case of the previous section with $F = I\!R^l$ and $E = I\!R^m$.

Thus, if we suppose that we have p new nodes between two old nodes, we obtain the following expression of the matrix associated to the operator P:

$$\begin{pmatrix}
1 & 0 & & & & \\
\overline{\alpha}_1 & \alpha_1 & 0 & & & \\
\vdots & \vdots & & & & \\
\overline{\alpha}_p & \alpha_p & 0 & & & \\
0 & 1 & 0 & & & \\
0 & \overline{\alpha}_1 & \alpha_1 & 0 & & \\
 & \vdots & \vdots & & & \\
0 & \overline{\alpha}_p & \alpha_p & 0 & & \\
0 & 0 & 1 & 0 & & \\
0 & 0 & \overline{\alpha}_1 & \alpha_1 & 0 & \cdots \\
 & & \vdots & \vdots & & \\
0 & 0 & \overline{\alpha}_p & \alpha_p & 0 & \cdots \\
 & & & & \vdots &
\end{pmatrix}$$

This operator produces an interpolated curve that presents discontinuous derivatives (at old nodes); let us define a smoother interpolation; the basic principle that we shall use here is to define new points such that high order finite differences are minimal; a first consequence is that P is sparse, with a rather small bandwidth.

We distinguish two families of such interpolations:

Explicit parametrization :
$(Pf)_j$ is directly defined from values $f_{i1}\,,\,\cdots\,,\,f_{il}$ of f on existing coarse nodes: $(Pf)_j$ = linear combination of $f_{i1}\,,\,\cdots\,,\,f_{il}$.

Implicit parametrization :

(Pf) is defined from an implicit equation :

$$\sum_{k \epsilon fine} \lambda_{ik} (Pf)_k \;=\; \sum_{q \epsilon coarse} \mu_{iq}\, f_q \;\;.$$

Only parametrizations of the first family are now presented.

A first explicit example:

For each interval $[x_i, x_{i+1}]$ we take the third degree polynomial $\mathcal{P}$ such that:
$\mathcal{P}(x_i) = f_i\;,\;\mathcal{P}(x_{i+1}) = f_{i+1}\;,\;\dfrac{d\mathcal{P}}{dx}(x_i) = f_i^{'}$ and $\dfrac{d\mathcal{P}}{dx}(x_{i+1}) = f_{i+1}^{'}$ with the following notation:

$$f_i^{'} = \frac{f_{i+1} - f_{i-1}}{2\,(x_{i+1} - x_i)} \;\;.$$

Then, the operator P is defined as follows: for each new node x_j, we have:
$(Pf)_j = \mathcal{P}(x_j)$.

Let $\varphi_0\;,\;\varphi_1\;,\;\varphi_2\;,\;\varphi_3$ four functions defined on $[x_i, x_{i+1}]$ by :

$$\varphi_0(x) = \frac{(x_i - x)^2(2x + (x_i - 3x_{i+1}))}{(x_i - x_{i+1})^3} \qquad , \qquad \varphi_2(x) = \frac{(x - x_i)^2(x - x_{i+1})}{(x_{i+1} - x_i)^2}\;,$$

$$\varphi_1(x) = \frac{(x_{i+1} - x)^2(2x + (x_{i+1} - 3x_i))}{(x_{i+1} - x_i)^3} \qquad , \qquad \varphi_3(x) = \frac{(x - x_i)(x - x_{i+1})^2}{(x_{i+1} - x_i)^2}\;;$$

then the polynomial $\mathcal{P}$ can be written in the following way:

$$\mathcal{P}(x_j) = \varphi_1(x_j)\; f_i \;+\; \varphi_0(x_j) f_{i+1} \;+\; \varphi_3(x_j) f_i^{'} \;+\; \varphi_2(x_j) f_{i+1}^{'} \;\;.$$

If we report now the expressions of $f_i^{'}$ and $f_{i+1}^{'}$, we obtain finally the interpoled value $(Pf)_j$:

$$(Pf)_j = G(\overline{\alpha}_j) f_{i-1} + F(\alpha_j) f_i + F(\overline{\alpha}_j) f_{i+1} + G(\alpha_j) f_{i+2} \ , \tag{5}$$

with $\quad F(\alpha) = \alpha(\frac{1}{2} + 2\alpha - \frac{3}{2}\alpha^2) \quad$ and $\quad G(\alpha) = -\frac{1}{2}\alpha(1-\alpha)^2 \ .$

Thus, in this case, P is given by:

$$\begin{pmatrix}
0 & 1 & 0 & & & & & \\
G(\overline{\alpha}_1) & F(\alpha_1) & F(\overline{\alpha}_1) & G(\alpha_1) & 0 & & & \\
\vdots & \vdots & \vdots & \vdots & & & & \\
G(\overline{\alpha}_p) & F(\alpha_p) & F(\overline{\alpha}_p) & G(\alpha_p) & 0 & & & \\
0 & 0 & 1 & 0 & 0 & & & \\
0 & G(\overline{\alpha}_1) & F(\alpha_1) & F(\overline{\alpha}_1) n & G(\alpha_1) & 0 & & \\
 & \vdots & \vdots & \vdots & \vdots & & & \\
0 & G(\overline{\alpha}_p) & F(\alpha_p) & F(\overline{\alpha}_p) & G(\alpha_p) & 0 & & \\
0 & 0 & 0 & 1 & 0 & 0 & & \\
 & 0 & G(\overline{\alpha}_1) & F(\alpha_1) & F(\overline{\alpha}_1) & G(\alpha_1) & 0 & \cdots \\
 & & \vdots & \vdots & \vdots & \vdots & & \\
 & 0 & G(\overline{\alpha}_p) & F(\alpha_p) & F(\overline{\alpha}_p) & G(\alpha_p) & 0 & \cdots \\
 & & & 0 & 1 & 0 & 0 & \cdots \\
 & & 0 & G(\overline{\alpha}_1) & F(\alpha_1) & F(\overline{\alpha}_1) & G(\alpha_1) & \cdots \\
 & & & \vdots & \vdots & \vdots & \vdots & \\
 & & 0 & G(\overline{\alpha}_p) & F(\alpha_p) & F(\overline{\alpha}_p) & G(\alpha_p) & \cdots \\
 & & & & & & \vdots &
\end{pmatrix}$$

A second explicit example : nested parametrization

We begin to interpolate at the middle of each $[x_i, x_{i+1}]$. If we use the previous interpolation (with $p = 1$ and $\alpha_j = \frac{1}{2}$) , we obtain :

$$(P_1 f)_j = \frac{9}{16}(f_i + f_{i+1}) - \frac{1}{16}(f_{i-1} + f_{i+2}) \ . \tag{6}$$

In Figure 1 the symbol " $\Box$ " represents the values of f_i of the first parametrization while the symbol " $\times$ " the interpolated values $(P_1 f)_j$; then, starting from the new set of points, we calculate by the same interpolation rule, an other set of points (symbolized in Figure 1 by " + "), and we repeat this procedure until all points are generated.

Let h be the number of employed levels, then we can summarize this interpolation by Equation (6) and the following system:

$$\begin{cases} P_k \in \mathcal{L}(I\!R^{m_k-1}, I\!R^{m_k}) \quad \text{with} \quad m_k = 2m_{k-1} - 1 \quad \text{and} \quad m_0 = l\,,\ m_{h-1} = m \\ \\ \forall\, k = 1, \cdots, h-2 \\ \quad (P_{k+1}f)_j = \dfrac{9}{16}((P_kf)_i + (P_kf)_{i+1}) - \dfrac{1}{16}((P_kf)_{i-1} + (P_kf)_{i+2}) \\ \\ P = P_{h-1}\, o\, P_{h-2}\, o\, \cdots\, o\, P_1 \end{cases} \tag{7}$$

$(Pf)_j$ is still expressed by linear combination of $f_{i1}\,, \cdots, f_{il}$ but the maximum number of terms (equal to four for the precedent case and to two for the linear interpolation) is here variable. Although we do not have yet any theorical argument to prove it, we observe that this process yields smooth discrete curves; this will be illustrated in the numerical experiments of Sections 4 - 7.

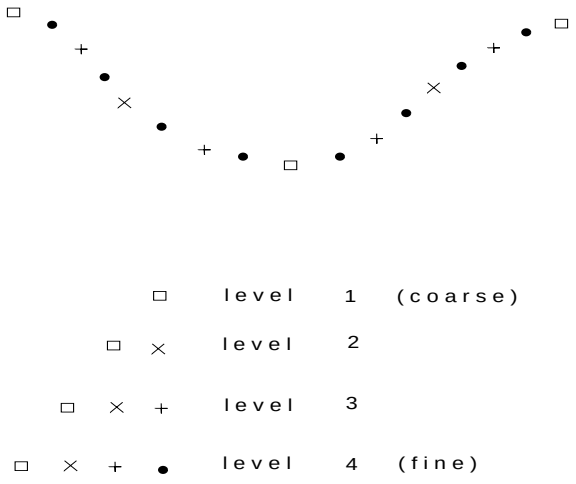

Figure 1: Sketch of the nested interpolation

3 Multilevel optimization strategy

The basic idea is to compute a descent direction with alternatively different levels of parametrization.
At first, we consider the **Full V-cycle** strategy (according to the multigrid theory [7],[8]), which consists of the following phases :
we start doing several iterations on the coarsest level, then, in a second phase and for several other iterations, the optimization is made alternately on coarsest level and on next level, etc.

In Figure 2 this strategy is plotted for three levels.

We can also use a "**nested iteration**"strategy : beginning on the coarsest level, we converge successively on each level until the finest one (see Figure 3).

We have to choose some criteria of changing the phase in our multilevel strategy. The main difficulty is to decide when we have sufficiently converged.

We summarize here the algorithm giving a descent direction at the n^{th} optimization iteration :

- first, we have computed a gradient $g_n = grad\, j(u_n)$.
- Assuming that we are on level i (with l_i parameters); we can associate to this level an interpolation $P_{i,n} : I\!R^{l_i} \rightarrow I\!R^m$ defined in the previous section. $g_n^F = P_{i,n}^* \, g_n$ is then a descent direction in the space $I\!R^{l_i}$ (see Section 1).
- Finally, we obtain a descent direction on the finest level given by the following expression:
$$d_n = P_{i,n} \, P_{i,n}^* \, g_n \, .$$

Remarks

- If the level i is the finest level we have obviously $d_n = g_n$.
- We have as many different PP^* as number of levels.
The relationship between a level and an optimization iteration depends of the choice of multilevel strategy and of the criterion. For example, in the case presented in Figure 2, the same interpolation P is used for the iterations 4, 6, 9 and 12.

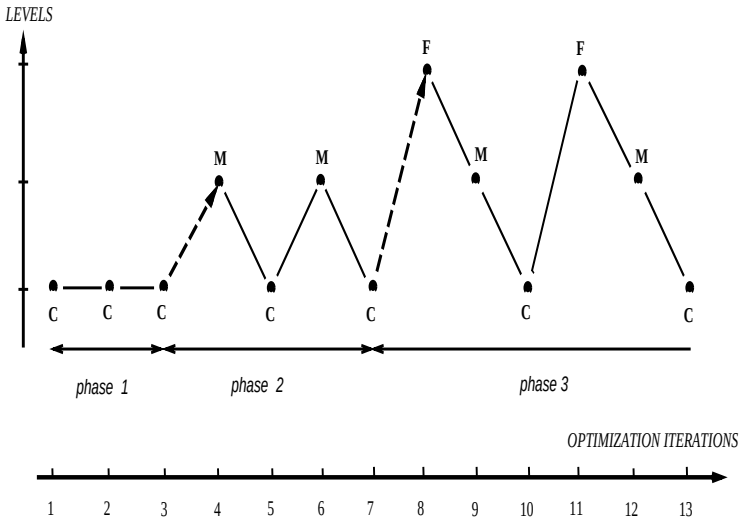

Figure 2: Sketch of full V-cycle strategy on 3 levels (C for coarse, M for medium, F for fine)

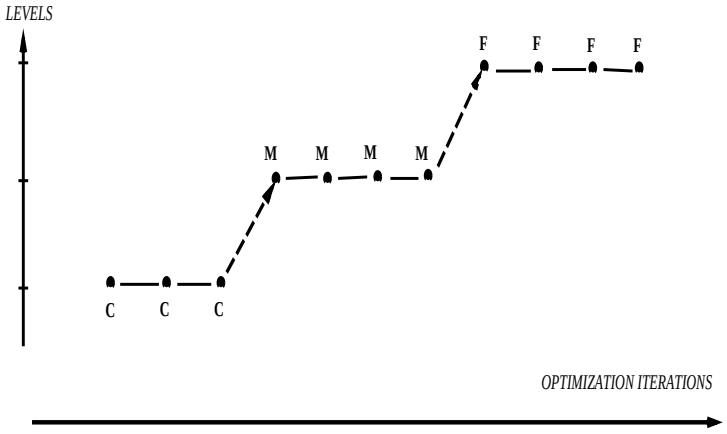

Figure 3: Sketch of nested iteration strategy on 3 levels (C for coarse, M for medium, F for fine)

4 A simplified linear example

4.1 Problem definition

We try to solve the following 1-D problem :

$$\begin{cases} -\Delta u + u = f \qquad \text{on } [0,1] \\ \\ u(0) = u(1) = 0 \end{cases} \tag{8}$$

Using the spatial discretization of $]0,1[$ below:

$$\Delta x = \frac{1}{N} \ , \quad x_j = j\Delta x \ \text{ for } \ j = 0, \cdots, N \ ,$$

and a finite volume approach, we obtain for each cell $C_i = [x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}]$:

$$\int_{C_i} u - \Delta u = \int_{C_i} u - \int_{C_i} \Delta u = \int_{C_i} f \ .$$

We choose as unknown the mean value of u on C_i and we suppose f constant on C_i, equal to f_i :

$$\Delta x \, u_i - (u'|_{i+\frac{1}{2}} - u'|_{i-\frac{1}{2}}) = \Delta x \, f_i \ .$$

Using a centered approximation of the derivative:

$$-\frac{1}{\Delta x^2}\,(u_{i+1} + u_{i-1}) + (1 + \frac{2}{\Delta x^2})\,u_i = f_i \ ,$$

thus, we have to solve the following linear system :

$$AW = b \ , \tag{9}$$

$$\text{where } \begin{cases} W = (u_i)_i \\ \\ b = (f_i)_i \end{cases} , \quad A = \begin{pmatrix} (1 + \frac{2}{\Delta x^2}) & -\frac{1}{\Delta x^2} & 0 & \cdots \\ \\ -\frac{1}{\Delta x^2} & (1 + \frac{2}{\Delta x^2}) & -\frac{1}{\Delta x^2} & 0 \\ \\ 0 & \ddots & \ddots & \ddots \\ \\ \cdots & 0 & -\frac{1}{\Delta x^2} & (1 + \frac{2}{\Delta x^2}) \end{pmatrix}$$

This matrix is symmetric positive definite .
Thus, in this case solving the linear system (9) is equivalent to minimize the following

quadratic function :

$$J(W) = \frac{1}{2} < AW, W > - < b, W >$$

and we apply a steepest descent method to minimize J on $\mathbb{R}^n$.
Let W_k be a solution at step k of the optimization, the derivative of J at W_k is defined here by :

$$\forall\, W \in \mathbb{R}^n \quad J'(W_k).W =< AW_k - b, W > \ ;$$

we know also that $g_k = AW_k - b$ is a descent direction.
If now, we use a hierarchical approach, we obtain the following expression for the descent direction (see Section 1) :

$$h_k = PP^* g_k \quad .$$

Finally, employing the Polak-Ribière conjugation, a new descent direction is found:

$$d_k = h_k + \frac{< h_k, h_k - h_{k-1} >}{< h_{k-1}, h_{k-1} >}\, d_{k-1} \quad .$$

We choose, for descent step, the real $\tilde{\rho}$ which minimizes:
$J(W_k + \tilde{\rho} d_k) = \frac{1}{2} < Ad_k, d_k > \tilde{\rho}^2 + < g_k, d_k > \tilde{\rho} \ + \ J(W_k)$,
and we obtain then the following algorithm that solve the linear system (9):

$$W^{k+1} = W^k - \frac{< g_k, d_k >}{< Ad_k, d_k >}\, d_k \quad .$$

4.2 Numerical experiments

We solve the linear system (9) for successives values of N :
$N = 2^{10} + 1 = 1025$, $N = 2^{11} + 1 = 2049$, $N = 2^{12} + 1 = 4097$, $N = 2^{13} + 1 = 8193$. For the right hand side b, we choose to take $f_i = 1$ for all i. Then we compare the convergence history (logarithm of the gradient as a function of the number of optimization iterations) for different strategies (see in Figures 4, 5, 6 ,7):
a conjugate gradient method without any hierarchical approach (with $PP^* = I_d$), a Full V-cycle (FVC) strategy with and without conjugation (i.e. we have respectively for descent direction d_k and h_k).
We see in the four Figures the same behaviour: if the conjugate gradient method convergence varies linearly with respect to the unknown's number N , conversely, for the hierarchical strategies, the convergence seems independent of N (it is, likely, proportional to the number of levels, $log_2 N$). The two hierarchical methods converge respectively with near 300 iterations and less than 500 iterations; here an iteration represents a good measure of CPU cost. We can notice that the addition of the conjugation deteriorates the Full V-cycle strategy.

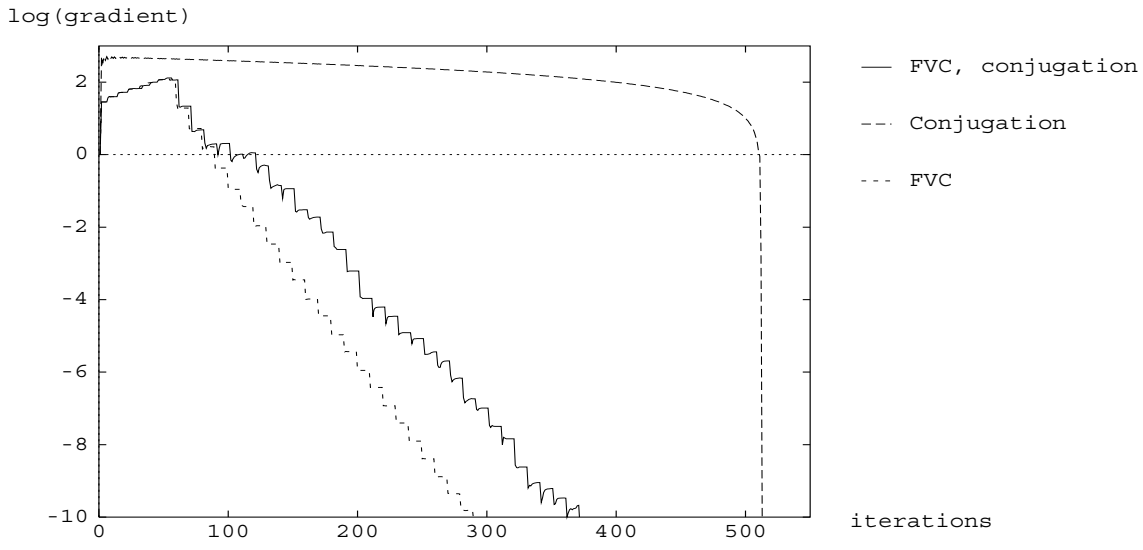

Figure 4: Convergence history of the gradient (N=1025)

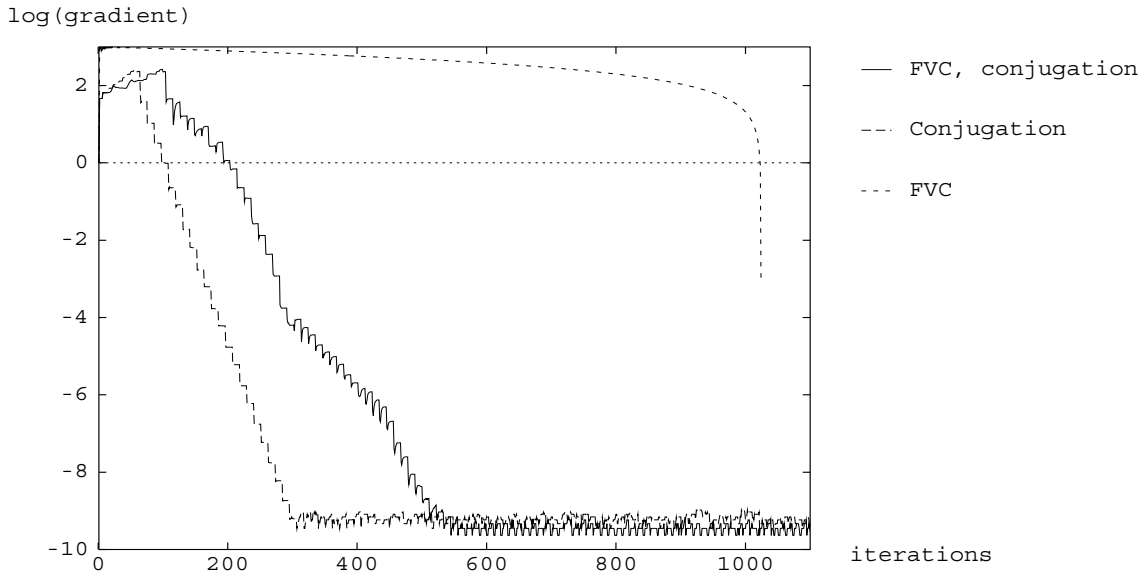

Figure 5: Convergence history of the gradient (N=2049)

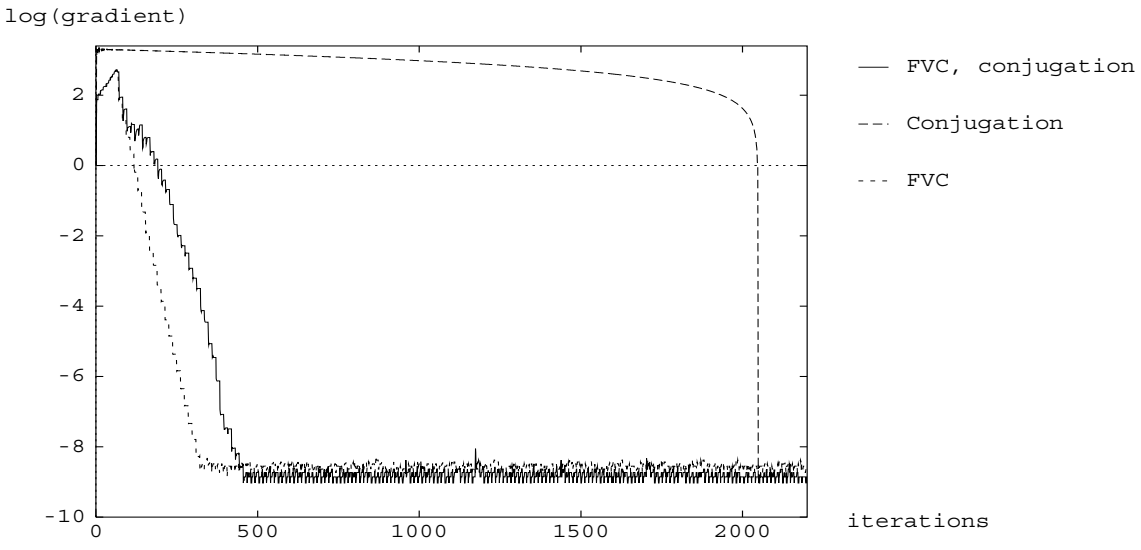

Figure 6: Convergence history of the gradient (N=4097)

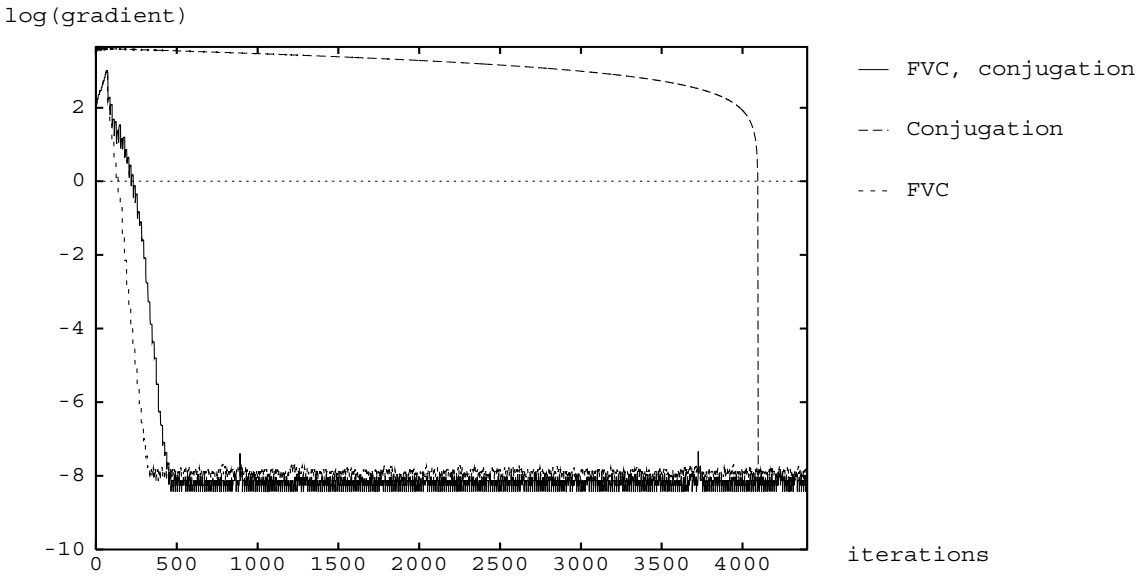

Figure 7: Convergence history of the gradient (N=8193)

5 Shape optimization problem

Since the background information in a flow simulation code can be huge, we do not describe it in details, but we give only the **basic informations** to understand what is going on; readers interested in performing this type of computation will find the complete information set in [4] and [9] for both flow simulation and gradient derivation by the adjoint method.

5.1 Continuous problem

We consider a family of 2D nozzles; since they are symmetric, we can solve the problem for an half of nozzle. The geometry is defined as follows :

- the axis is a straight segment (equation : $y = 0$) ,
- the wall is straight, of height 0.5 for x in $]-2, 0[$ and x in $]2, 4[$,
- the part of the wall to be optimized is the graph a function γ of x, for x belonging to $[0, 2]$,
- the wall shape is smooth enough.

The flow of an ideal gas is going through the nozzle entering at $x = -2$ $(0 < y < .5)$ and exiting at $x = 4$ $(0 < y < .5)$.

The global geometry is sketched in Figure 8.

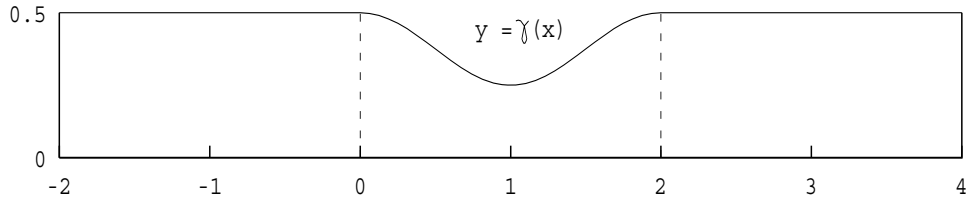

Figure 8: Nozzle geometry

To any shape γ smooth enough we associate the domain Ω_γ bounded by the nozzle and the solution W_γ of the steady Euler equations :

$$F(W_\gamma)_x + G(W_\gamma)_y = 0 \ \text{ in } \ \Omega_\gamma \ ,$$

with in/out boundary conditions for $x = -2$ and $x = 4$, and with slip boundary condition at nozzle axis and wall. From the flow variable W_γ, we can extract the pressure distribution

P_γ on the upper variable wall :

$$\left\{ \begin{array}{l} P_\gamma \ : \ [0,2] \longrightarrow I\!R \\ P_\gamma(x) = \ pressure \ of \ W_\gamma \ at \ y = \gamma(x) \end{array} \right.$$

A cost functional j is then defined as follows :

$$j(\gamma) = \int_0^2 (P_\gamma - P_{desired})^2 \ dx \ \ ,$$

in which $P_{desired}$ is a given target pressure distribution.
The **optimization problem** is :

$$Find \ \gamma \ = \ Arg \ \min_\gamma j(\gamma) \ .$$

Remark : a variant can be to prescribe a pressure distribution in other parts of the domain; the present study still holds (while usual inverse methods would not). We have described an example in Section 6.4.

5.2 Discrete problems

The family of domain parametrized by functions γ is discretised and covered by a family of "concertina-like" meshes with rows lying on fixed vertical lines as sketched in Figure 9.

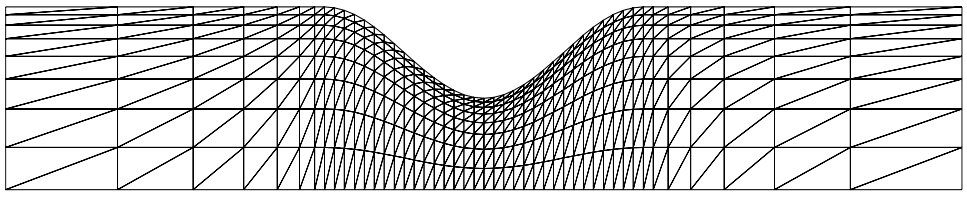

Figure 9: Domain discretization

The shape γ is defined by the node ordinates : $\gamma \in I\!R^N$. The Euler equations are discretized with an **first-order accurate wind olume heme** with nodes at vertices (dual cells built with medians) and the van Leer flux Vector Splitting ([16], [4]).

We can then obtain a **discrete cost function** :

$$j \ : \ I\!R^N \rightarrow I\!R, \ \gamma \rightarrow j(\gamma) = \sum_{i \, \epsilon \, \gamma \ curve} (P^i_\gamma - P^i_{desired})^2 \ .$$

In the sequel, for the sake of comparison with less general methods, we have chosen to construct the distribution $P_{desired}$ from a given shape and solving the corresponding discrete Euler system [3]:

$$P_{desired} = P_{\gamma_{given}} \ ;$$

As a consequence we expect to find $\gamma_{opt} = \gamma_{given}$, and $j(\gamma_{opt}) = 0$;
the purpose of the experiment is to see how fastly we find it.

This "optimization inverse problem" is considered with the following target :

$$\gamma_{given} \ : \qquad y(x) = \frac{3}{8} + \frac{1}{8} sin(\pi(x + \frac{1}{2}))$$

and the initial geometry before optimization is defined by

$$\gamma_{initial} \ : \qquad y(x) = \frac{1}{2} \quad .$$

The chosen mesh has 423 nodes with 31 shape-control nodes ($N = 31$) (see Figure 9). The Mach contours of the flow obtained with a 1900-nodes mesh in subsonic case (see Section 6.5), are zoomed in Figure 10; unaccuracy (in particular loss of symmetry) is explained mainly by the first-order accuracy of the scheme.

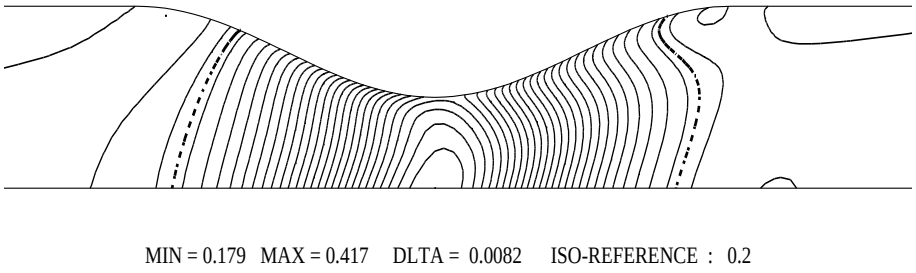

Figure 10: Mach contours

6 Optimization with gradient

6.1 Gradient method

It has been shown in [4] that the above discrete cost functional could be differentiated by introducing an adjoint state to the Euler system. In this section, we apply a gradient method or a conjugate-gradient (CG) method using the gradient of j; as conjugate-gradient, a Polak-Ribière orthonormalization is applied as in Section 4; for both algorithms, steepest versions are used; they involve a 1-D minimization consisting of firstly a dichotomic research, and finally a parabolic interpolation that ensures a quadratic final convergence; the total expense for one global gradient/conjugate-gradient iteration is one adjoint state solution and 4-5 cost evaluations (involving 4-5 state-equation solutions). As it is shown in the experiments of [4], an high precision (up to 100 Jacobi iterations) must be achieved in solving the linear systems for cost function and adjoint state evaluations. In the 1-D minimization, which requires 3-4 cost evaluations, an high accuracy is not necessary : in this procedure the precision in solving the different systems is 5 times less than in the previous case.

6.2 One level experiments

First we present results obtained using only one level (31 optimization parameters) for a subsonic flow. In Figure 11 the convergence of j to zero is plotted as a function of gradient iteration; the non-conjugate gradient is compared to the conjugate one. Without conjugation, the convergence is rather slow, while we obtain more than five order of magnitude in one hundred gradient iterations for the conjugate gradient even if some difficulties are encountered at the beginning (see Figure 12).

6.3 Full V-cycle, sensitivity to parametrization

We return now to usual (non conjugate) gradient and we apply a Full V-cycle strategy with four levels (respectively 3, 7, 15 and 31 unknowns). We implement differents parametrizations : a linear interpolation, a cubic one and a nested parametrization with cubic interpolation. With respect to the one-level algorithm, we observe a tremendous acceleration. In fact, as it is sketched on Figures 13 and 14, the solution at 20^{th} gradient iteration is already rather good. The convergence is faster with a smoother interpolation, and considerably more fast with a nested parametrization (see Figure 16). Unfortunately, we did not obtain yet an efficient combination of hierarchy with conjugate gradient.

6.4 Nested iteration with conjugate gradient

We apply the conjugate gradient method successively on five levels with respectively 1, 3, 7, 15, 31 unknows while the interpolation is nested cubic one; the conjugation vector is not reset to zero between two phases. This strategy appears to be already very efficient; in fact, we are rapidly very close of the desired shape (see Figure 15). A comparison with a one level

conjugate gradient can be seen in Figure 17 where is sketched the decimal logarithm of the cost functional with respect to the cost evaluations number (the gradient iterations number may be obtained dividing approximately the cost evaluations number by 4-5).

We have used this strategy for solving Test Case T1 of Brite-Euram 1082 Workshop (see [3]). The change with respect to the above test case is that the pressure is now prescribed on the fixed symmetry axis (at the bottom of the computational domain). We have plotted in Figure 18 respectively the decimal logarithm of the cost and of the gradient norm with respect to the cost evaluation number (the values are normalized by the initial value). We obtain a rather good initial convergence: a reduction of five orders of magnitude in 20 gradient iterations (i.e. 100 cost evaluations). Then, the convergence becomes more slow due to a lower numerical sensitivity of pressure on axis with respect to nozzle shape.

6.5 Sensitivity to mesh size

We have seen in [4] that in the one-level case, a refinement of the mesh slows down significantly the convergence. We take here a finer mesh with $1900 = 20 \times 95$ nodes and 63 unknowns (we have previously $423 = 9 \times 47$ nodes and 31 unknowns) and we use the strategy of the Section 6.3, i.e. a Full V-cycle beginning on the three-parameter level. Finally, we obtain, with less than one hundred iterations, respectively six order of magnitude for the finer mesh and seven order of magnitude for the coarser one (see Figure 19 where $log(j)$ is normalised by the initial value).
Thus the hierarchical approach seems to be a good answer for the mesh refinement problem. This confirms the study carried out on a 1-D linear example in Section 4.

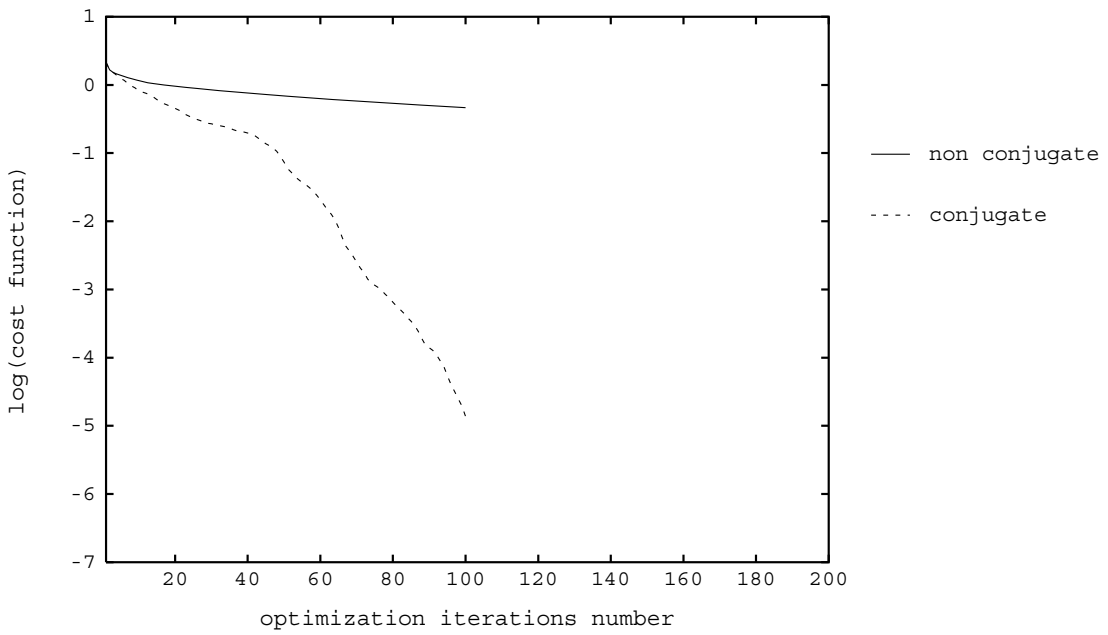

Figure 11: Convergence history for simple and conjugate gradient

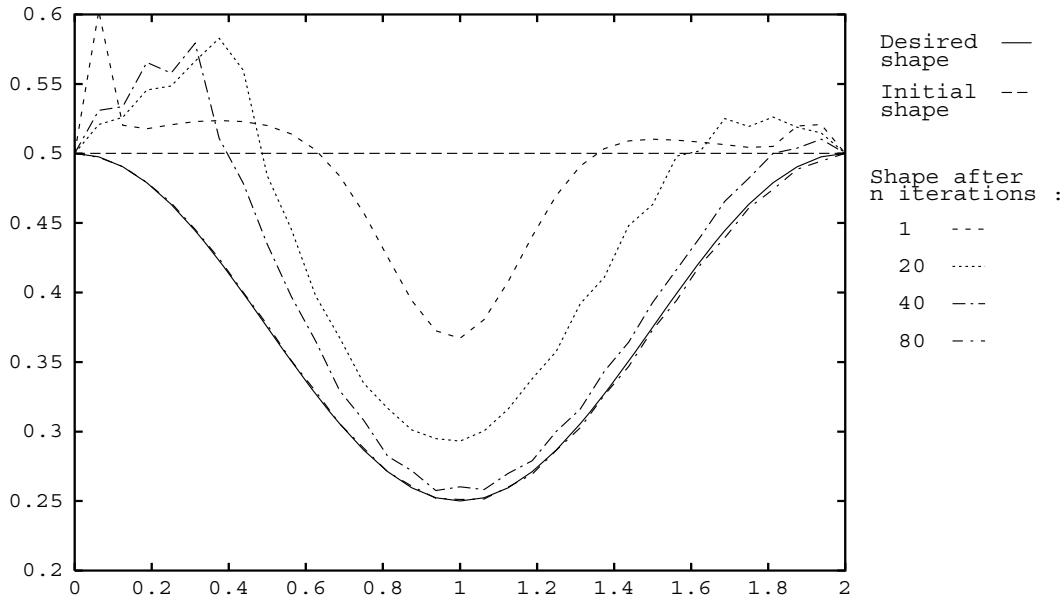

Figure 12: Successives shapes : Conjugate gradient

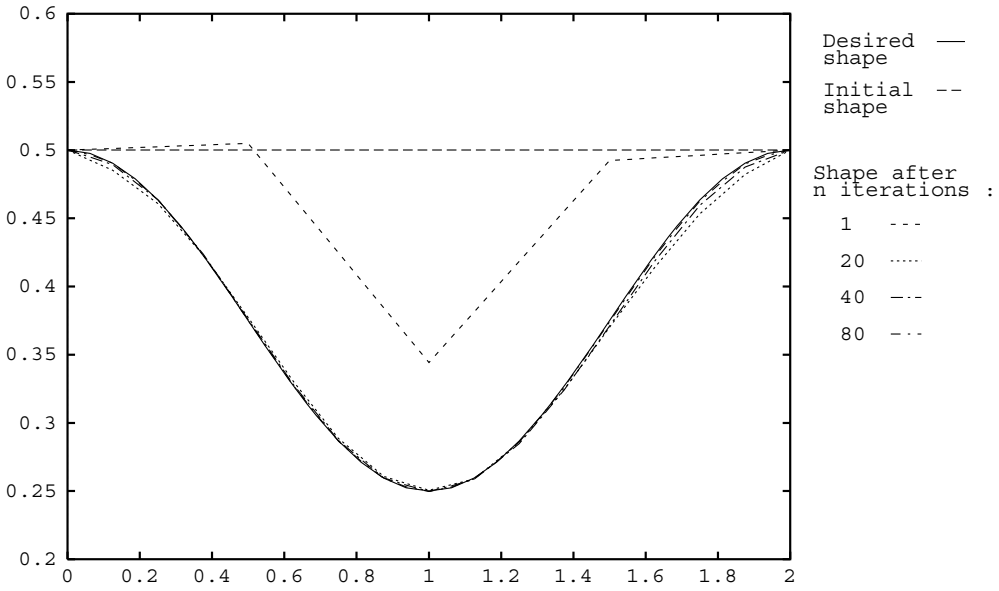

Figure 13: Successives shapes : Full V-cycle, linear interpolation

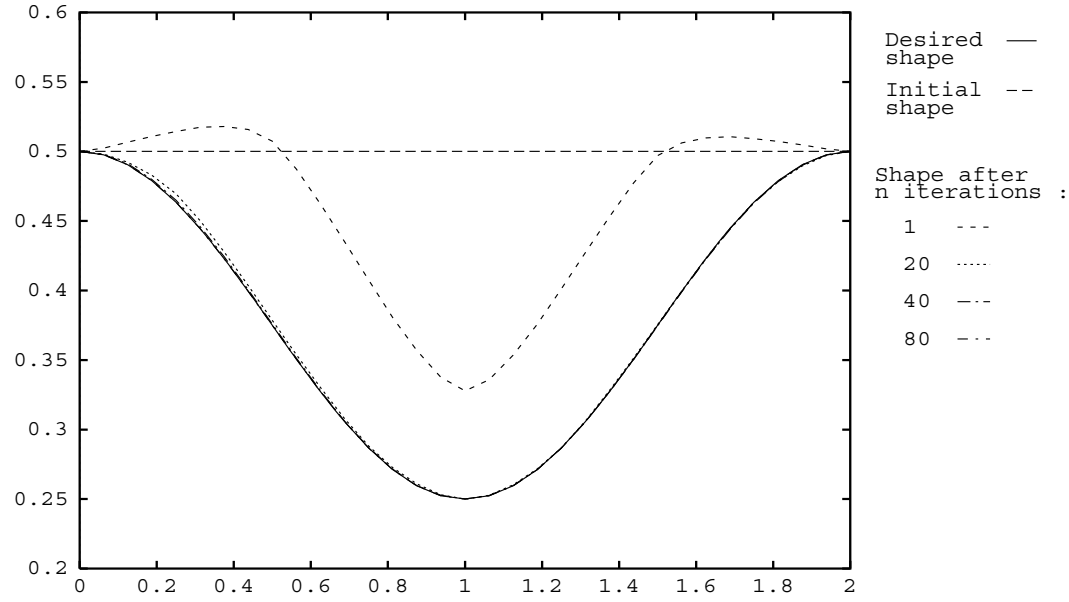

Figure 14: Successives shapes : Full V-cycle, nested cubic interpolation

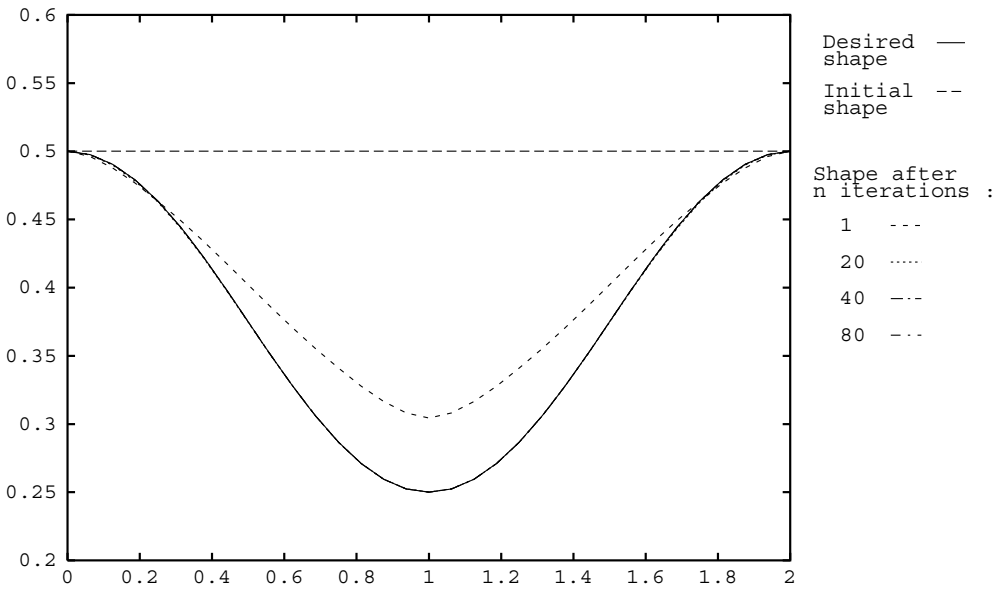

Figure 15: Successives shapes : Nested iteration + conjugation

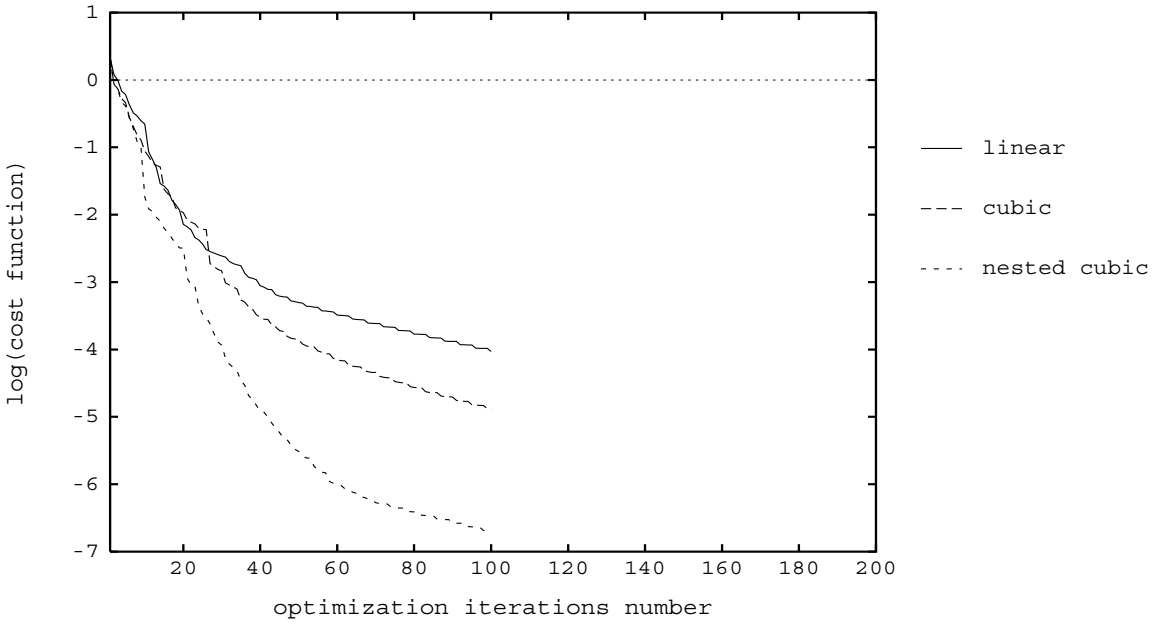

Figure 16: Convergence history: Full V-cycle

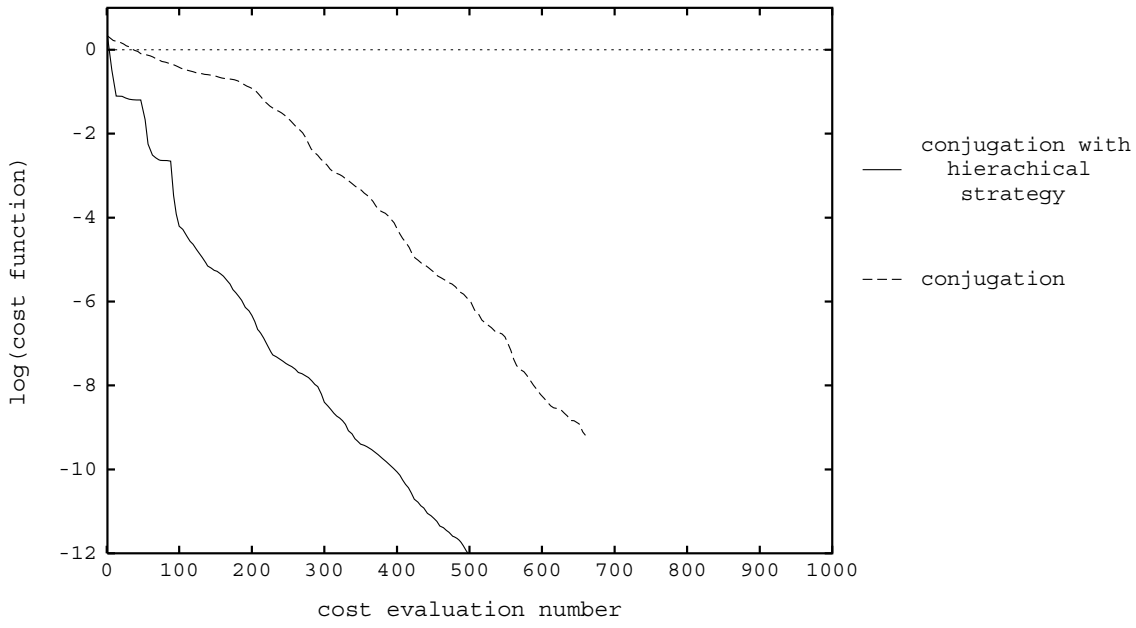

Figure 17: Convergence history

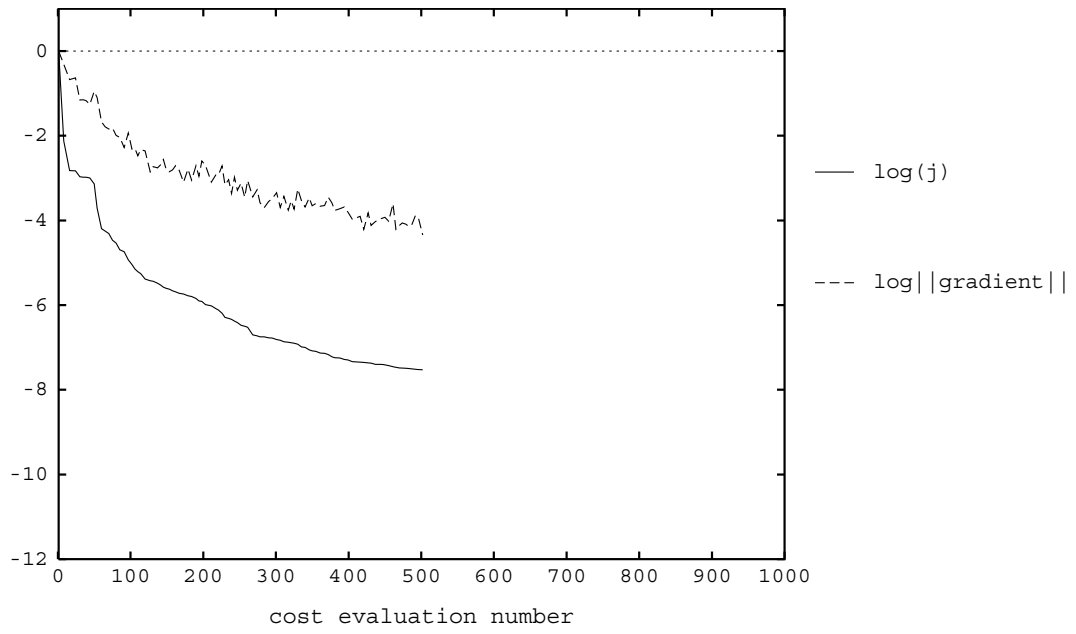

Figure 18: Cost function and gradient convergence history

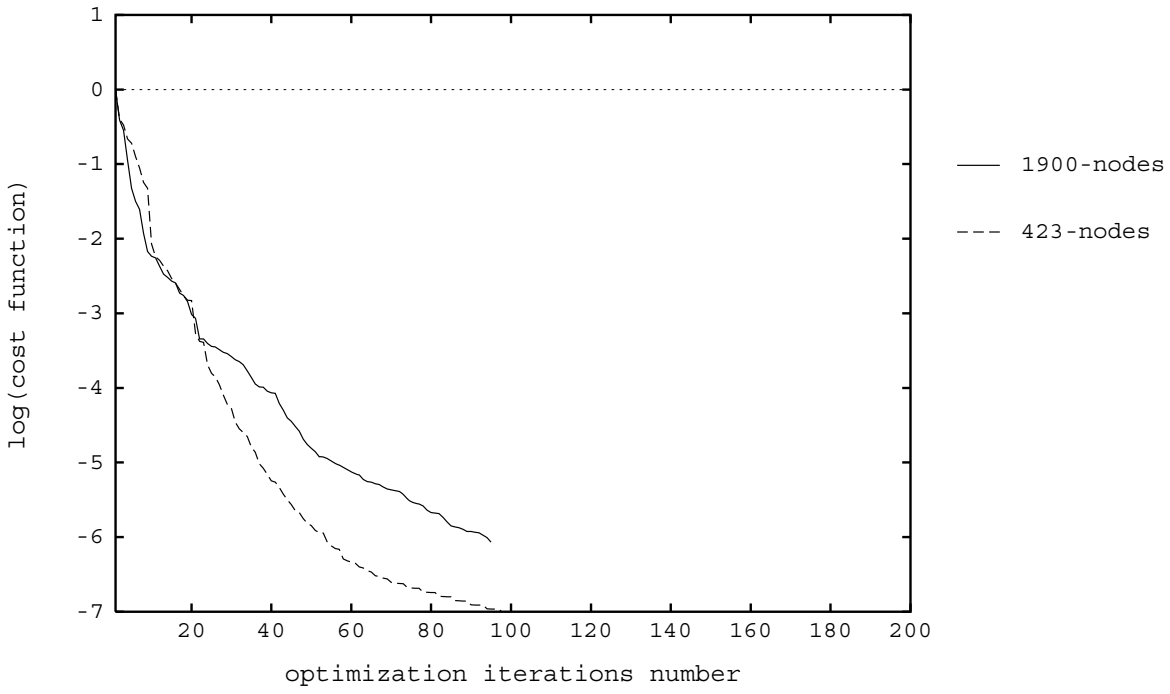

Figure 19: Comparaison of the convergence history of the two meshes

6.6 A transonic case

We consider now a transonic flow in the same nozzle geometry as in Section 6.4; the farfield Mach number is equal to 0.5 instead of 0.2 . The general features of the obtained flow (isomach contours) are shown in Figure 20.
The hierarchical method is applied: a nested iteration on five levels with conjugation and a nested cubic interpolation. We use the 423-nodes mesh as in Section 6.4.
In Figure 21, we plot the following quantities: (1) the logarithm of the cost, (2) the logarithm of the gradient and (3) the logarithm of the deviation from the optimal shape ($||\gamma - \gamma_{desired}||$) as functions of iteration optimization number.
These values are normalized by the initial value.

The successive nozzle shapes are shown on Figure 22.
The results appear even better than for the subsonic case (compare with Figures 15, 17 and 18).

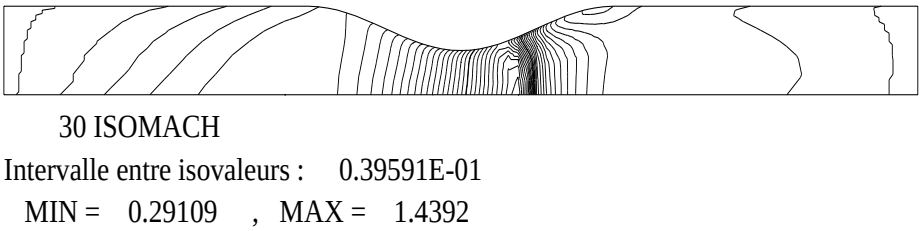

Figure 20: Distribution of the final nozzle shape

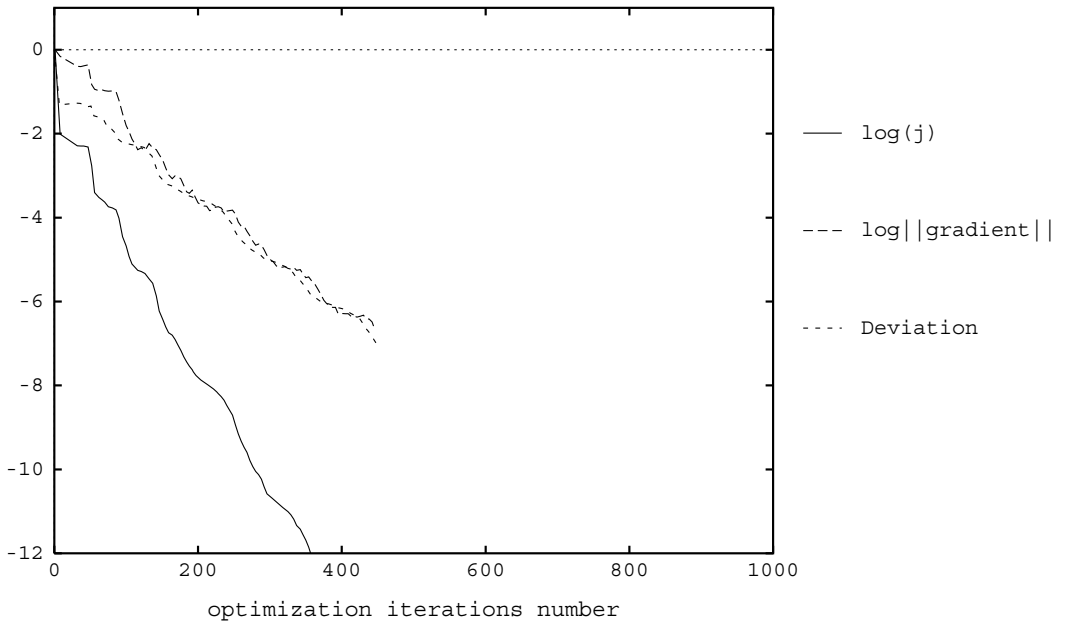

Figure 21: Convergence history

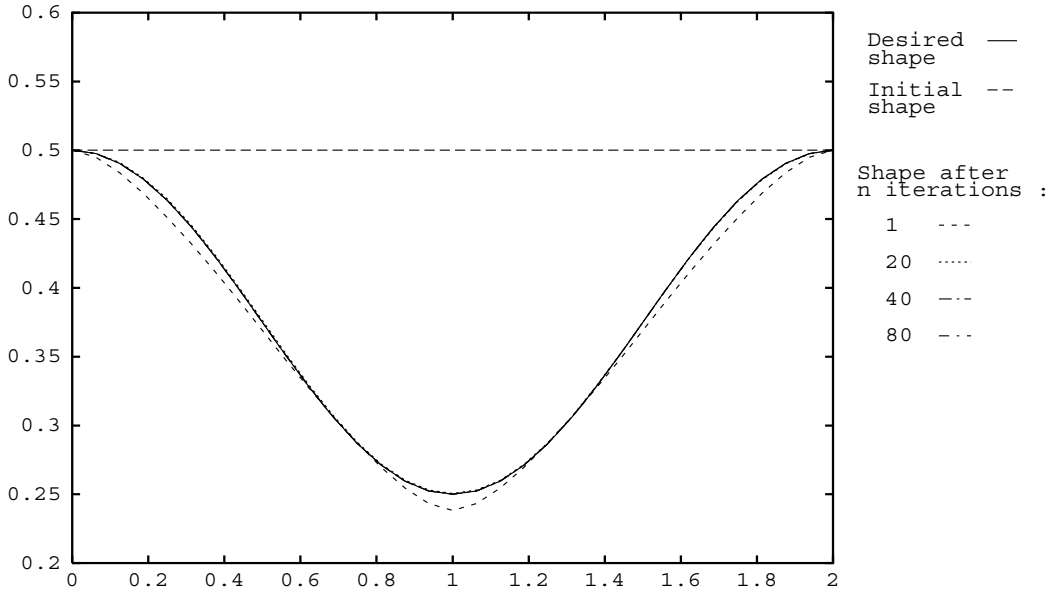

Figure 22: Successives shapes

7 Optimization with an approximate gradient

7.1 Algorithm

We use now an approximate gradient obtained by divided finite differences instead of an exact gradient; thus we have to compute only the cost function (but N times if we have N gradient components to obtain).
In fact we apply a divided finite differences gradient approach :

$$\begin{cases} u_0 \quad given \\ \\ u_{n+1} = u_n - \rho\, \tilde{g} \end{cases} \tag{10}$$

where the approximate gradient $\tilde{g}$ is defined by :

$$\forall\, i \,\epsilon\, \{1, \cdots, m\} \qquad \tilde{g}_i \;= \frac{j(u_n + \varepsilon\, e_i) - j(u_n)}{\varepsilon} \tag{11}$$

and e_i is the i^{th} canonical base vector of E and ε is small enough.

7.2 Implementation

We use here the same notation as in Section 1 where the Hilbert space E and F are respectively $\mathbb{R}^m$ and $\mathbb{R}^l$ $(m \geq l > 0)$. According to Lemma 2, we know that $d = PP^* grad_E\, j(u_n)$ is a descent direction in E at the n^{th} optimization iteration of the algorithm (2).
Here the gradient of j is not computed exactly, thus, instead of d, we have an approximate direction $\tilde{d} = P\tilde{g}_F$ where $\tilde{g}_F$ is an approximate gradient in the space F.
Let $v_0 \,\epsilon\, F$, the following equalities may be written also for all v elements of F:

$$< P^*\, grad_E\, j(P(v_0)), v >_F = < grad_F\, joP(v_0), v >_F \;.$$

Then using the Fréchet derivate definition and the linearity of P we obtain the following expression :

$$\forall\, v \,\epsilon\, F \qquad < grad_F\, joP(v_0), v >_F = \lim_{\theta \to 0} \frac{j(P(v_0) + \theta P(v)) - j(P(v_0))}{\theta} \;\;.$$

We can also define $\tilde{g}_F$ as the divided finite differences approximation of $P^* grad_E\, j(u_n)$ by

$$\forall\, i \,\epsilon\, \{1, \cdots, l\} \qquad (\tilde{g}_F)_i \;= \frac{j(u_n + \varepsilon P(e_i)) - j(u_n)}{\varepsilon} \;\;, \tag{12}$$

where e_i is the i^{th} canonical base vector of F and ε a small fixed real, chosen in practice equal to 10^{-8} (in Section 4.2 of [4] the sensitivity to the choice of ε value).

Finally our iterative algorithm is the following one :

$$\begin{cases} u_0 \quad given \\ \\ u_{n+1} = u_n - \rho P \, \tilde{g}_F \end{cases} \tag{13}$$

where $\rho = arg \min_{r > 0} (u_n - r \, P \, \tilde{g}_F)$.

7.3 Numerical experiments

We consider the subsonic flow in a nozzle as defined in Section 5 and we compare the convergence speed for an exact gradient computation and for an approximate gradient one. In both cases, we use a Full V-cycle on 5 levels (1, 3, 7, 15 and 31 optimization parameters) for hierarchical strategy. We compare first the hierarchical approach gradient with the simple gradient and the conjugate one.
We see in Figure 25 that, if we consider the convergence with respect to the iterations number, we obtain rather same results with the approximation gradient and the exact one (it just demonstrates that both gradients are computed accurately enough).
While, if we evaluate the CPU cost (see Figure 23 for the exact gradient and Figure 24 for the gradient obtained by divided finite differences), it appears that the convergence is rather costly for the standard approximate gradient approach but not so bad for the combination with the hierarchical approach. We try to explain it in Table 1 which summarizes the CPU times (in second on CONVEX 210) and the number of cost computations the 47 first iteration. Note that at 47^{th} iteration, only the two hierarchical computations have converged of nearly 4 orders of magnitude, as asserted by the values of log(j) given in the last column. We have splitted in three principal steps our optimization iteration. For the approximate approach, the gradient computation step is the most expensive one in the entire computation. In fact, in the case without hierarchy, we have to do $N + 1$ cost evaluations for the gradient computation and only 1 cost evaluation for obtaining the cost function and few others for the 1-D minimization; finally in our case 90% of the CPU times is used to compute the gradient. With a hierarchical approach, we use much less cost evaluations for the gradient computation. As a conclusion, for the exact gradient approach, using or not a hierarchical strategy does not affect the CPU times per gradient step while for the approximate approach, the hierarchical strategy is three times less expensive.

	computation of		1-D minimization	total	value of log(j)
	cost	gradient			
exact gradient	420.6 s 47	65.4 s 0	1000.1 s 149	1486.1 s 196	-0.12
exact gradient with hierarchy	630.2 s 47	68.5 s 0	2338.1 s 222	3036.8 s 269	-3.55
approximate gradient	388.5 s 47	12050.1 s 1457	966.1 s 147	13404.7 s 1651	-0.14
approximate gradient with hierarchy	627.4 s 47	3964.3 s 269	2071.6 s 203	6663.3 s 519	-3.64

Table 1: Comparison of CPU times (seconds CONVEX 210) and number of cost evaluations for different steps and for different strategies

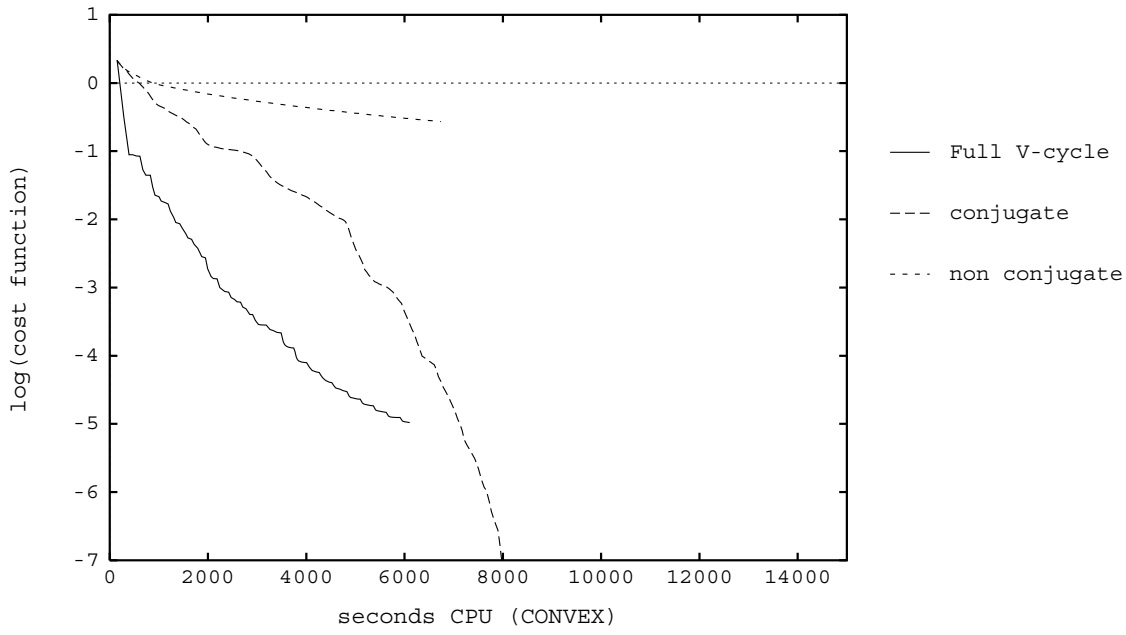

Figure 23: Exact Gradient: convergence history

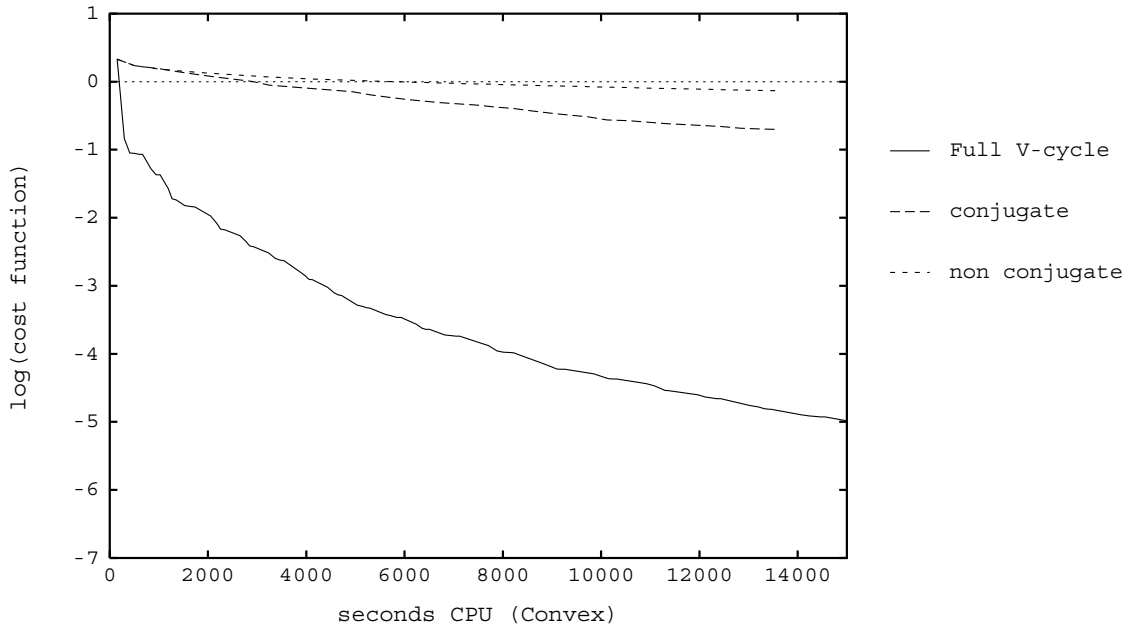

Figure 24: Approximate gradient: convergence history

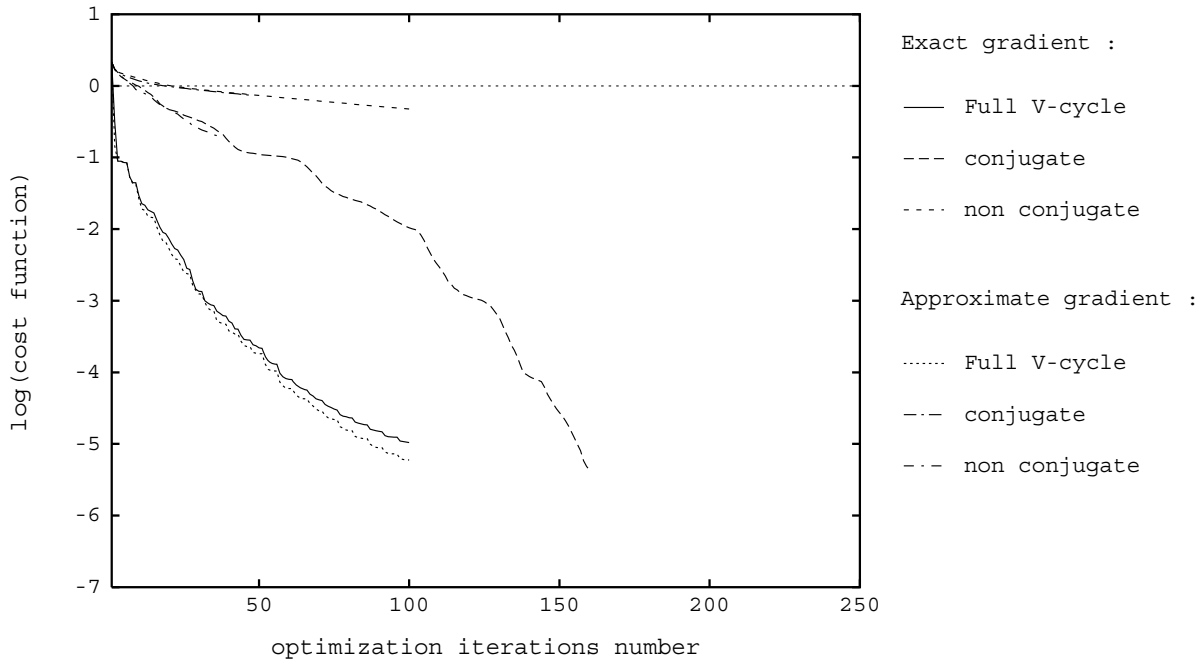

Figure 25: Comparison with respect to iteration number

7.4 Application to second-order spatial accuracy

It may be rather tedious to implement an exact gradient for a second-order accurate upwind formulation because the approximation is much more complex to differentiate. In a previous work, we have tried to mix a gradient computed with the first-order approximation with a second-order state equation, but it seems to be not really interesting (see Appendix 3 of [4]). Now the hierarchical approach allows us to consider a direct alternative since we can use a divided finite differences approximation for the gradient with a second-order accuracy.
To obtain second-order accuracy, we use a standard MUSCL approach ([16]) We have employed here, as in the first-order case, a Full V-cycle strategy on 5 levels. In Figure 26 the decimal logarithm is plotted with respect to the number of optimization iterations. This convergence is only slightly slower, after a better starting.
The obtained flow (Mach contours) is zoomed in Figure 27; if we compare with Figure 10 (obtained with a quite finer mesh), it appears that the second-order accuracy solution captures better some features of the flow. In particular, in the region of the nozzle center the isolines are more perpendicular to the axis and the maximum values are located on the wall and not on the axis.

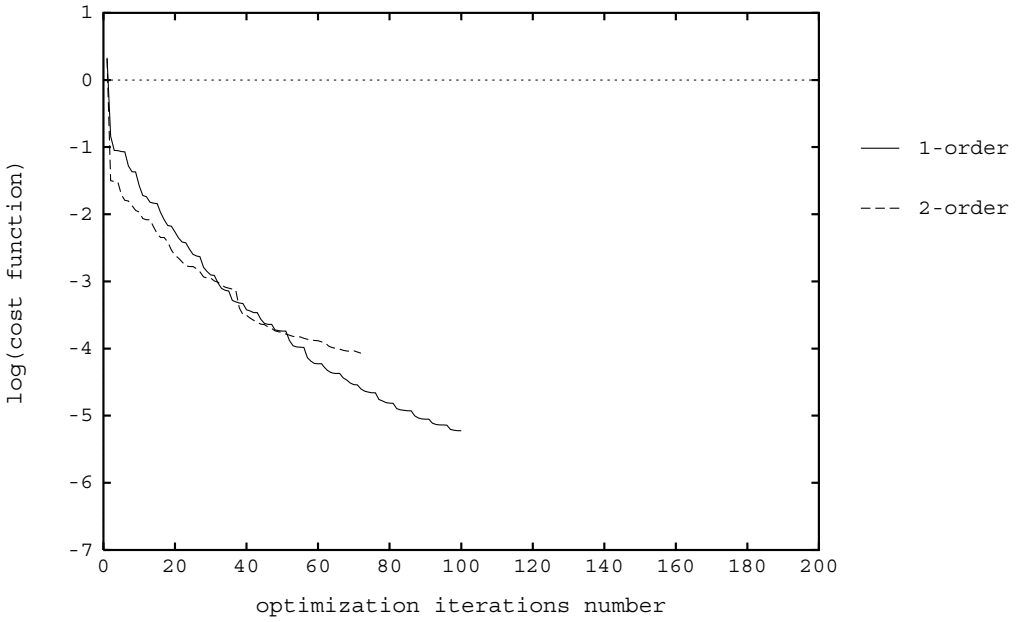

Figure 26: Comparison of first- and second-order accuracy in the Euler solver

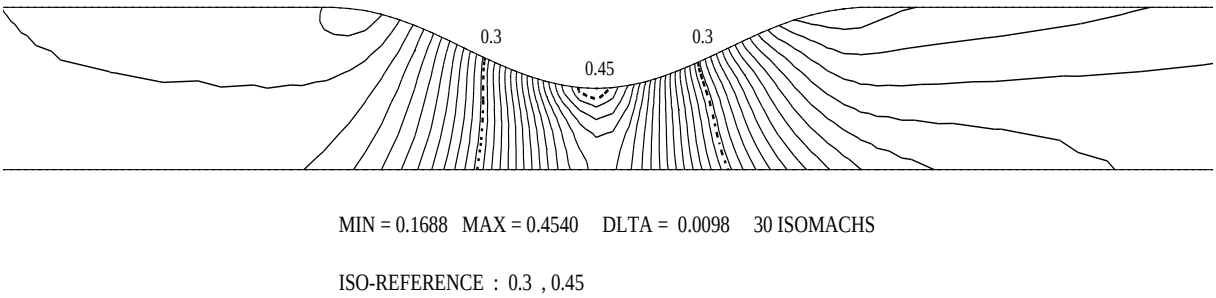

Figure 27: Approximate gradient with a second-order spatial accuracy: Mach contours

8 Conclusion

We have explored a rather natural way to introduce a hierarchy of levels in a gradient-based optimization process, and we have applied it to a shape optimization problem in which smooth parametrization is a severe prerequisite, as showed in [4]. With the options presented, a rather good answer is obtained in ten gradient iterations on coarser levels; the convergence of the global set of parameters needs at least 50 gradient iterations. A rather large number of parameters (up to 63) could then be handled progressively and the convergence is not deteriorated by increasing the nodes number, contrary to the one level case. Optimization can be stopped when a too small sensitivity of the flow with respect to the shape is detected.

In the case of approximate gradients computed by divided differences, the hierarchical approach is still more favourable.

One important feature of the presented approach is the possibility of further extensions; in particular, the application to minimization with constraints is under study.

Many efforts are still necessary to reach an algorithm that would be efficient enough for industrial use: finding a way to combine hierarchical parametrization with faster basic optimization (conjugate gradient, GMRES, $\cdots$) should hopefully produce some progress.

Instead of using a descent method another way consists in solving simultaneously mesh variables, flow equation and shape optimality (see [14, 15]). It could be interesting to combine this type of method with a multilevel approach : we have made in [5] a first investigation in this direction.

A last remark is that the presented hierarchical approach can be used as a system solver, that would be more easily applicable than multigrid (but a priori algebraically more complex than $O(N)$).

Acknowledgements

The authors wish to dedicate this paper to Jean Céa at the occasion of his 60^{th} birthday in acknowledgements of his contribution to Optimum Design Theory.
This work was supported by Brite Euram 1082 Contract (Optimum Design in Aerodynamics) .

References

[1] P.V. AIDALA, W.H. DAVIS, Jr., and W.H. MASON, *"Smart Aerodynamic Optimization"*, AIAA Applied Aerodynamics Conference, Danvers Massachusetts (1983).

[2] F. ANGRAND, R. GLOWINSKI, J. PERIAUX, P. PERRIER, O. PIRONNEAU, et G. POIRIER, *"Optimum design for potential flows"*, Proc. Finite Elements in Flows Problems Calgary (1980).

[3] F. BEUX, *" Shape optimization of an Euler flow in a nozzle "*, Actes du Workshop BRITE EURAM AREA 5 "Optimum Design in Aerodynamics", Volume I, à paraitre dans "Notes on numerical Fluid Mechanics" aux éditions Vieweg Verlag.

[4] F. BEUX, A. DERVIEUX, *" Exact-gradient shape optimization of a 2D Euler flow "*, Rapport de Recherche INRIA N^o 1540 (1991) et dans Finite Elements in Analysis and Design (Eds Elsevier), **12**, p. 281-302 (1992).

[5] F. BEUX, N. MARCO, A. DERVIEUX, *"Shape optimization for complex flows: toward coupled approaches"*, INRIA Contribution to BE-1082 24th-month synthesis, Barcelona (1992).

[6] M. BORSI, *" Aerodynamic design and optimization at Alenia D.V.D "*, Computational Methods in Applied Sciences, ECCOMAS, Brussels (1992).

[7] A. BRANDT, *" Multigrid techniques "*, Guide with Applications to Fluid Dynamics, G.M.D-Studien, N^o 85 (1984).

[8] W.L. BRIGGS, *A multigrid tutorial*, SIAM, Philadelphia (1987).

[9] L. FEZOUI, B. STOUFFLET, *"A class of implicit upwind schemes for Euler simulations with unstructured meshes"*, Rapport de Recherche INRIA N^o 517 (1986).

[10] A. JAMESON, *" Aerodynamic design via control theory. "*, Report 1824 MAE, Princeton University, New Jersey (1988).

[11] D. JOANNAS, B. MANDEL, J. PERIAUX, B. STOUFFLET, *"Optimal shape design for airfoils"*, Brite Euram "Optimum Design In Aerodynamics" contract AERO-0026C proposal 1082, Mid-Term Report (1992).

[12] O. PIRONNEAU, A. VOSSINIS *"Comparison of some optimization algorithms for optimum shape design in aerodynamics"*, Rapport de Recherche INRIA N^o 1392 (1991).

[13] V. SELMIN, M. BORSI, *"Direct optimization of transonic airfoils"*, Brite Euram "Optimum Design In Aerodynamics" contract AERO-0026C proposal 1082, Mid-Term Report (1991).

[14] S. TA'ASAN, *"One Shot Methods for Optimal Control of Distributed Parameter Systems I: Finite Dimensional Control"*, ICASE Report N^o 91-2, NASA Contractor Report 187497 (1991).

[15] S. TAA'SAN, G. KURUVILA, *"Aerodynamic design and optimization in One Shot"*, AIAA paper 92-0025 (1992).

[16] B. VAN LEER, *"Flux Vector Splitting for the Euler equations"*, Lecture notes in Physics, **170**, p. 405-512 (1982).