INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids **30**: 127–136 (1999)

MESH ADAPTION AND AUTOMATIC DIFFERENTIATION IN A CAD-FREE FRAMEWORK FOR OPTIMAL SHAPE DESIGN

BIJAN MOHAMMADI[a,*] AND OLIVIER PIRONNEAU[b]

[a] *INRIA, BP 105, 78153 Le Chesnay Cedex, France*
[b] *University of Paris VI, Paris, France*

SUMMARY

A new approach for optimal shape design based on a CAD-free framework for shape and unstructured mesh deformations, automatic differentiation for the gradient computation and mesh adaption by metric control in 2D is presented. The CAD-free framework is shown to be particularly convenient for optimization when the mesh connectivities and control space size are variable during optimization. Constrained optimization for a transonic regime has been investigated in both 2D and 3D. Copyright © 1999 John Wiley & Sons, Ltd.

KEY WORDS: shape design; unstructured mesh adaption; automatic differentiation

1. INTRODUCTION

The aim of this paper is to show how to couple mesh adaption and shape optimization tools. Here, not only the mesh in the computational domain, but also the number and repartition of the control parameters over the shape will change during optimization. Therefore, general shape and unstructured mesh deformation tools are necessary. In particular, these tools have to be, as much as possible, CAD-free, in the sense that *the only geometrical entity available during optimization should be the mesh*. Another reason that justifies this requirement is that CAD-based shape deformation tools are quite complicated to use and also difficult to take into account when evaluating the Jacobian. In [1], the current authors showed that all the mesh points can be considered on the body (in both 2D and 3D) as control points as long as the initial local regularity of the shape is preserved during the design process. This makes it possible, and easy, to consider all kinds of shapes.

Local mesh adaption by metric control is a powerful tool for getting mesh-independent results at a reduced cost [2–4]. However, mesh adaption has never been used in optimization procedures because it implies a change of the design space and mesh connectivities after each adaption, and this is difficult to take into account. Moreover, these procedures are not differentiable at all. Therefore, optimization is usually performed on meshes with fixed connectivities and on a given control space (i.e. no control parameters are created during optimization). However, as for direct problems, the need for adapted meshes is clear in design processes.

* Correspondence to: INRIA, BP 105, 78153 Le Chesnay Cedex, France.

CCC 0271–2091/99/100127–10$17.50

As a gradient method is used in the current shape optimization, it is necessary to have a good evaluation of the Jacobian of the 'discrete' cost function with respect of control parameters. It is clear that if the number of control parameters is large, an adjoint method is necessary [5,6]. But classical adjoint methods are often built in continuous level, after a linearization of the governing equations, and do not account for the inconsistencies of the numerical schemes (like numerical dissipation). In this work, automatic differentiation (AD) is used in reverse mode to produce the Jacobian of the discrete cost function [7–11]. Hence, the non-consistencies of the numerical schemes are naturally taken into account in the Jacobian. Moreover, the cost of this evaluation is independent of the number of control parameters, as in a classical adjoint method. Using this approach, the authors have studied [1,12,13] the impact of using different gradients based on different numerical fluxes (Roe and Osher with and without a MUSCL-type second-order reconstruction [14–16]) used for the solution of the Euler equations, on the optimization procedure. They also showed that this approach does not suffer from a change in the nature of the equations (from parabolic to hyperbolic). Viscous turbulent configurations have also been considered [12] by taking account of a $k-\varepsilon$ two-equation turbulence model [17] and special wall-laws (including pressure and convection effects [18,19]) in the Jacobian. It was noticed that in an optimization procedure involving mesh deformation, wall-laws are more suitable than low Reynolds modeling (i.e. solving the flow up to the wall), as this former approach requires much finer meshes for which conformal mesh deformation is hard to achieve. In particular, the authors showed that once the difficulties coming from the non-linearities of the operators treated, it is easier to perform optimization for a turbulent flow configuration than for the corresponding inviscid flow (i.e. at the same Mach number).

As this work includes several tools, only a short description of each of them is given. The governing equations and the flow solver are described in Section 2. Section 3 is devoted to the shape and mesh deformation tools. A short description of the mesh adaption approach is given in Section 4. The adaptive optimization algorithm and related details, including the Jacobian computation by AD in reverse mode, are described in Section 5. Numerical results are presented in Section 6.

2. FLOW EQUATIONS AND SOLVER

The cost function to be minimized under geometric and aerodynamic constraints is denoted $J(x, U(x))$. Here x indicates the geometrical description of a configuration and the flow pattern ($U(x)$) around this shape is the solution of the steady Euler system of fluid dynamics in conservation form:

$$\nabla\cdot(F(U)) = 0, \tag{1}$$

where U is the vector of conservative variables (i.e. $U = (\rho, \rho\vec{u}, \rho(C_vT + \frac{1}{2}|\vec{u}|^2))^t$) and F represents the advective operator. This system has four equations in 2D and the system is closed using the equation of state $p = p(\rho, T)$. For the sake of simplicity, the Euler equations are only considered, but the extension is already available for viscous turbulent flows [1,12].

The flow solver is based on a finite volume–Galerkin approach on unstructured triangular meshes (tetrahedra in 3D). A Roe [14] approximate Riemann solver has been used together with MUSCL reconstruction and Van Albada limiters [15]. The time-dependent equation

$$\frac{\partial U}{\partial t} + \nabla \cdot F(U) = 0$$

is marched in time to a steady state. The time discretization is based on a four-stage Runge–Kutta scheme. Inflow and outflow boundary conditions are of characteristic type [20], and for outflow boundaries, care has been taken to correctly treat subsonic configurations. More technical details can be found in [21,22].

3. SHAPE AND MESH DEFORMATION TOOLS

The set of tools needed for shape and mesh deformation from a variation of control parameters ($x_c(n_c)$) is described as:

$$\delta x_c \to \delta x_w \to \delta x_m,$$

where $x_w(n_w)$ denotes the discretization points on the geometry and $x_m(N)$ the internal mesh nodes.

For 2D applications, control points are usually fitted by splines. Splines have two features:

1. They permit $n_c \ll n_w$.
2. They smooth the variations of control points when propagating to body discretization points.

But in this work, following the solution (through a local metric), the control space changes during optimization as points may be created or removed on the body. Therefore, the positions of the points have to be redefined after each adaption over the shape. Of course, this is possible in 2D but much more difficult in 3D. In all cases, this increases the difficulty by introducing another layer in the adaptive optimization. Moreover, general 3D surface splines are quite complicated to handle, especially on unstructured meshes. To avoid 3D difficulties and reduce the dependencies between two adaptions, the following general framework for shape deformation in 2D and 3D is introduced:

1. All the nodes on the shape are control points (i.e $n_c = n_w$).
2. To avoid oscillations, a smoothing operator is defined over the shape. This can be, for instance, a few 'local' Jacobi iterations to solve the following system:

$$(I - \varepsilon\Delta)\delta\tilde{x}_w = \delta x_w, \tag{2}$$

where $\delta\tilde{x}_w$ is the smoothed shape variation for the shape nodes and δx_w is the variation given by the optimization tool. By 'local' it is meant that if the predicted shape is locally smooth, it remains unchanged during this step.

In the past, this framework was used for optimal shape design in 2D and 3D [1,12,13,23] for cases with fixed connectivity and control space. It was shown that configurations involving several thousands of control points can be easily treated.

Once $(\delta\tilde{x}_w)$ is known, these variations have to be expanded over all the mesh. This is done by solving a volumic elasticity system that is also of the form of (2).

4. MESH ADAPTION BY METRIC CONTROL

A brief description of the metric evaluation is given. The mesh generation technique is not described here. It is based on a Delaunay algorithm [24,25].

The key idea is to modify the scalar product used in the mesh generator for distance, surface (or volume) evaluations. Hence, the Delaunay approach will generate equilateral elements in a new metric, and not in the Euclidean one. This scalar product is based on the evaluation of the Hessian of the variables of the problem. Indeed, for a P^1–Lagrange discretization of a variable u, the interpolation error is bounded by:

$$\mathscr{E} = |u - \Pi_h u|_0 \leq ch^2 |D^2 u|_0, \tag{3}$$

where h is the element size, $\Pi_h u$ the P^1 interpolation of u and D^2u its Hessian matrix. This matrix being symmetric, one has:

$$D^2u = \begin{pmatrix} \partial^2 u/\partial x^2 & \partial^2 u/\partial x\,\partial y \\ \partial^2 u/\partial x\,\partial y & \partial^2 u/\partial y^2 \end{pmatrix} = \mathscr{R}\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}\mathscr{R}^{-1},$$

where $\mathscr{R}$ is the eigenvectors matrix of D^2u and λ_i its eigenvalues (always real). Using this information, the following metric tensor $\mathscr{M}$, is introduced:

$$\mathscr{M} = \mathscr{R}\begin{pmatrix} \tilde{\lambda}_1 & 0 \\ 0 & \tilde{\lambda}_2 \end{pmatrix}\mathscr{R}^{-1}, \tag{4}$$

where

$$\tilde{\lambda}_i = \min\left(\max\left(|\lambda_i|, \frac{1}{h_{\max}^2}\right), \frac{1}{h_{\min}^2}\right),$$

with $h_{\min}$ and $h_{\max}$ being the minimal and maximal edge lengths allowed in the mesh.

Now, if an equilateral mesh with edges of length of 1 in the metric $\mathscr{M}/(c\mathscr{E})$ is generated by a Delaunay procedure, the interpolation error $\mathscr{E}$ is equi-distributed over the edges a_i of the mesh. More precisely, one has

$$\frac{1}{c\mathscr{E}}\, a_i^T\, \mathscr{M}\, a_i = 1. \tag{5}$$

More details on the ingredients used in the metric definition for inviscid and viscous laminar and turbulent flows involving shocks and boundary layers can be found in [2–4,26].

5. OPTIMIZATION PROBLEM

The following minimization problem is considered:

$$\min_{x_c} J(x_c, U(x_c)),$$

$$E(x_c, U(x_c)) = 0, \quad g_1(x_c) \leq 0, \quad g_2(U(x_c)) \leq 0,$$

where $x_c \in R^{n_c}$ describe the control parameters, $U \in R^N$ the flow, $E \in R^N$ the state equation and $g_{1,2}$ direct ('geometrical', on x_c) and indirect (on $U(x_c)$, such as a given lift) constraints.

Indirect constraints on U have been taken into account in the cost function by penalty. Geometrical constraints are of two types:

1. shape deformation is allowed between two limiting surfaces (curves in 2D),
2. the volume of the shape can only increase.

The first constraint has been taken into account by projection and the second constraint by penalty in the cost function.

5.1. The gradient

As said above, in this approach all the nodes on the wall belong to the design space. This makes an evaluation of the Jacobian out of reach, for instance, by finite difference. AD in reverse mode is used through the automatic differentiator Odyssée. This approach is similar to an adjoint method. More details about Odyssée and its use for similar applications can be found in [1,9–12,23,27,28].

5.2. Adaptive optimization algorithm

The gradient is used in a gradient method to solve the optimization problem. The minimization tool is quite simple. It is based on a gradient method with a fixed descent step. But more sophisticated methods can be used as long as they do not require the solution of large systems (proportional to n_c) as the number of nodes on the wall can be quite large, especially for 3D applications. Moreover, as in this approach, the number of control variables may vary during optimization due to mesh adaption, the optimization tool should not be sensitive to this.

The adaptive optimization algorithm is given with only one optimization after each adaption. But several iterations of optimization can be made on the same mesh.

At step i of the adaption, the mesh, the solution, the metric and the cost $J(x_{c_i}, U(x_{c_i}))$ are denoted by $\mathscr{H}_i$, $\mathscr{S}_i$, $\mathscr{M}_i$ and $J(x_i)$ respectively.

5.3. Algorithm

$\mathscr{H}_0$, $\mathscr{S}_0$, given,
do $i = 0, \ldots, iadapt$
define the control space (wall nodes): $\mathscr{H}_i \to x_i$,
compute the gradient: $(x_i, \mathscr{H}_i, \mathscr{S}_i) \to \mathrm{d}J(x_i)/\mathrm{d}x_i$
$\tilde{x}_i = P_i\left(x_i - \lambda \dfrac{\mathrm{d}J(x_i)}{\mathrm{d}x_i}\right)$,
deform the mesh: $(\tilde{x}_i, \mathscr{H}_i) \to \tilde{\mathscr{H}}_i$,
update the solution over the deformed mesh: $(\tilde{\mathscr{H}}_i, \mathscr{S}_i) \to \tilde{\mathscr{S}}_i$,
compute the metric: $(\tilde{\mathscr{H}}_i, \tilde{\mathscr{S}}_i) \to \mathscr{M}_i$,
generate the new mesh using this metric: $(\tilde{\mathscr{H}}_i, \mathscr{M}_i) \to \mathscr{H}_{i+1}$,
interpolate the previous solution over the new mesh: $(\tilde{\mathscr{H}}_i, \tilde{\mathscr{S}}_i, \mathscr{H}_{i+1}) \to \bar{\mathscr{S}}_{i+1}$,
compute the new solution over this mesh: $(\mathscr{H}_i, \bar{\mathscr{S}}_{i+1}) \to \mathscr{S}_{i+1}$,
end do.

In the previous algorithm, P_i is the projection operator that changes after each adaption as the control space changes.

6. RESULTS

In this section some results for constrained optimization problems at various Mach number are presented. All these computations have been performed on a workstation making about 10 Megaflop with less than 64 Megabytes of memory in simple precision. The 2D configuration, including mesh adaptions, gradient computations and flow solutions require less than 2 h, while the 3D case has been solved overnight. Concerning memory requirements, due to the optimization described in the previous section, the reverse mode requires approximately 10 times more memory than the direct solver. An estimation of the memory required by the direct solver is given by $50 \times N$ words (1 word = 32 or 64 bits depending on architectures) in 2D, where N is the number of nodes. This is more than what is necessary in a structured solver because that the data structures involved are much more complicated in an unstructured approach.

In the following examples, when global constraints are present, the different penalty coefficients in the cost function are initially chosen for the different quantities involved to have variations of the same order of magnitude. During optimization, they are reduced with the same ratio as the cost function.

The drag reduction problem with constraints on the lift and the volume has been considered. The cost function for all these cases is given by:

$$J(x) = C_d + \alpha|C_l - C_l^0| + \beta|\mathrm{Vol} - \mathrm{Vol}_0|,$$

where C_d is the drag coefficient, C_l and C_l^0 are the actual and initial lift coefficients, and Vol and Vol_0 the actual and initial volumes. To be sure that the solutions are mesh-independent, a final computation has been done on the final shape until convergence. The initial and final (after 20 adaptions and optimizations) meshes and the isoMach distributions are shown.

6.1. Drag reduction for a Naca 0012

The initial airfoil is the NACA 0012. The design takes place at Mach number 0.754 and an incidence of 2°. The drag coefficient has been reduced by more than 10% while the lift and volume slightly increased (Figure 1).

6.2. Drag reduction for a business jet

This is the same drag reduction problem but for a complex 3D configuration. Indeed, the solution is far from being mesh-independent. The aim is to show that this approach including the CAD-free framework and AD in reverse mode enables general 3D configurations to be treated. Here the engines are treated as being flow-through. The platform of the aircraft and its volume are constraints. In this case, more than 5000 control points were involved (Figure 2). The design takes place at Mach number of 0.85 and zero incidence. The aim is to reduce the depression over the cockpit and the shocks over the wings, engines and empenage. The drag has been reduced by about 5% while the lift and volume remained almost unchanged (Figures 3–5).

7. CONCLUDING REMARKS

A new approach for optimal shape design on variable control spaces and on meshes with variable connectivities has been presented. The reverse mode of AD has been used to produce

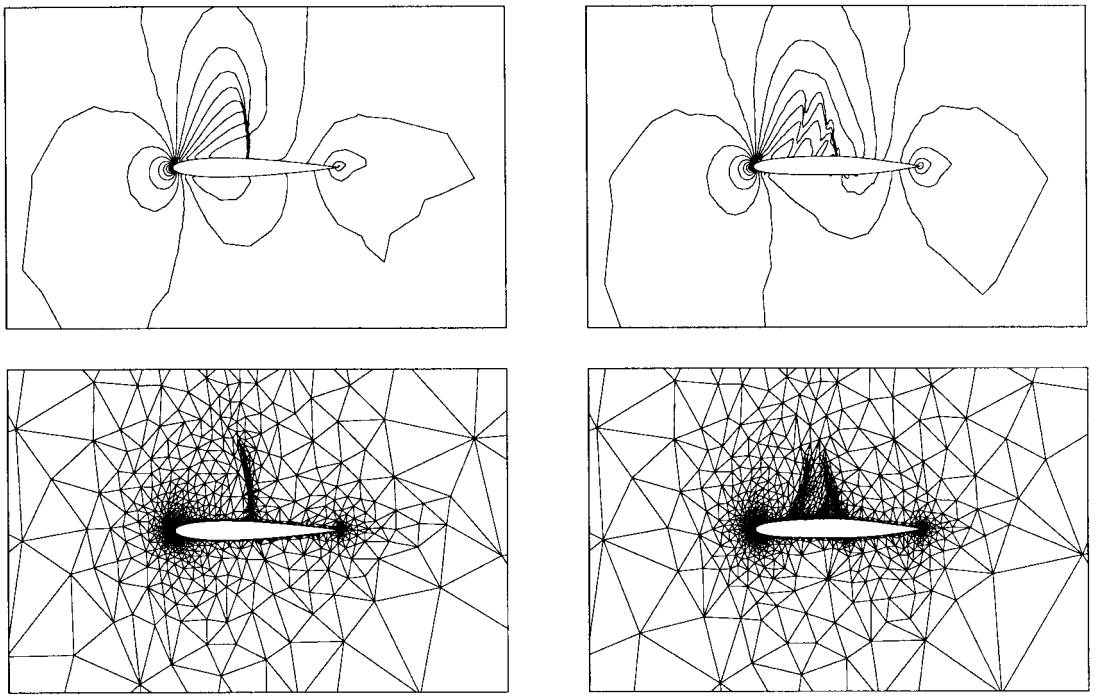

Figure 1. Transonic drag reduction: isoMach and mesh adaption evolution.

the gradients of the discrete operators. In this approach, the mesh is part of the optimization procedure and function of the solution. This is an easy way to avoid mesh dependencies in the optimization as well as in the direct problem.

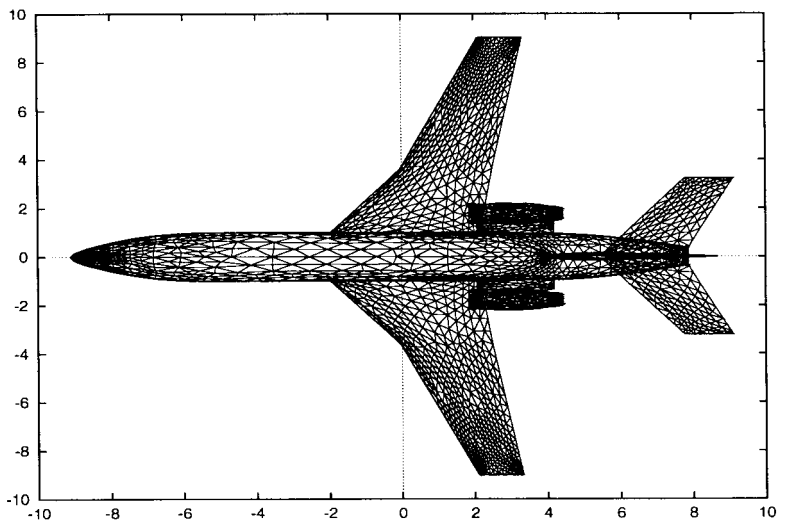

Figure 2. Optimization of a business jet: upper surface discretization, all these nodes are control points.

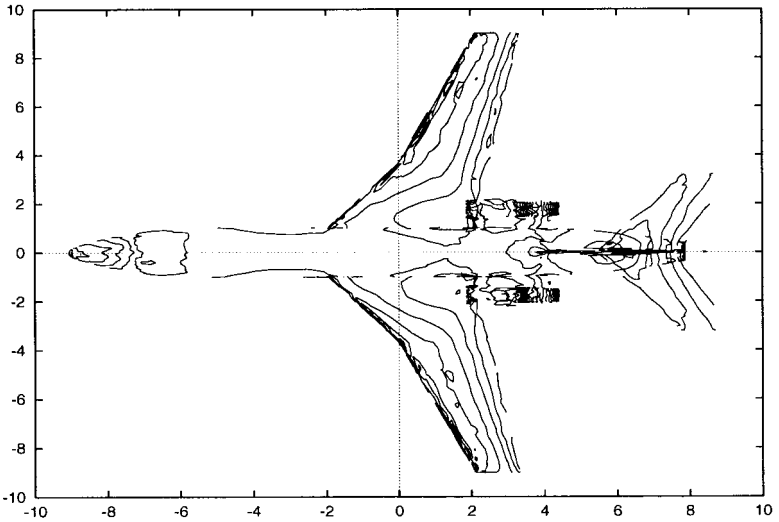

Figure 3. Optimization of a business jet: isoMach distribution, the target is to remove the depression over the cockpit and to smooth the shocks on the wings and engines.

Other automatic differentiator tools exist. The most popular are probably Adifor (informations at bischof@mcs.anl.gov) developed at Argonne National Laboratory and Adol-c (informations at griewank@math.tu-dresden.de). It seems that due to its wide distribution and a good user support provided, Adifor is becoming a reference. However, currently, only Odyssée produces the reverse mode for the whole solver. On the other hand, unlike Odyssée, higher-order derivatives are only available through Adifor and Adol-c. These tools are in free access. A state-of-the-art on AD can be found in the proceedings of the recent SIAM workshop help in Santa Fe. The 2D Navier–Stokes solver and the mesh generation and adaption tools are in free access under ftp://ftp.inria.fr/inria/projects/gamma/.

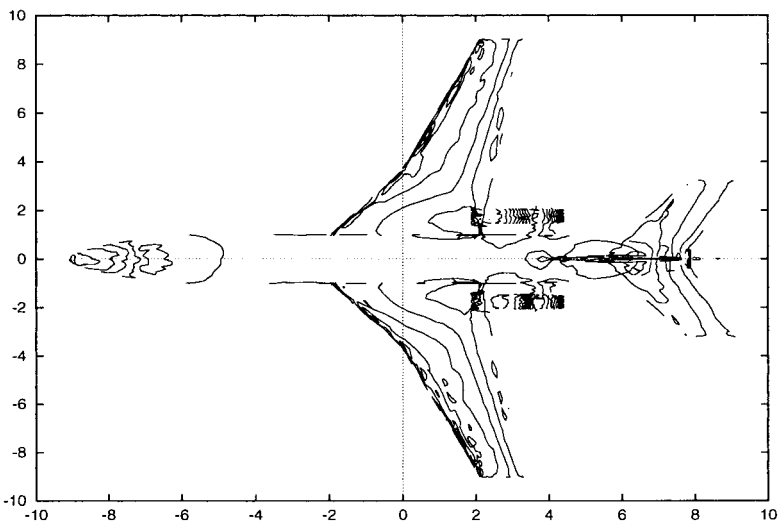

Figure 4. Same isoMach distribution, after five optimization steps. The depression over the cockpit has been reduced and the shocks on the wings and engines smoothed.

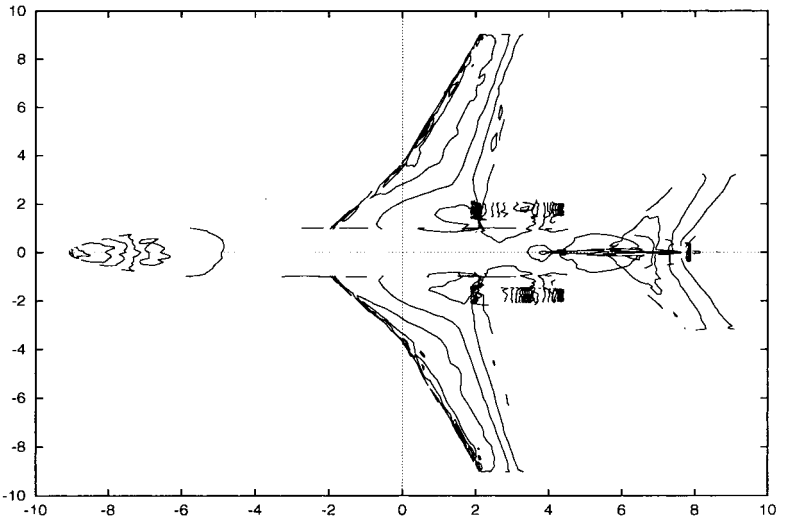

Figure 5. Same isoMach distribution, after ten optimization steps. The depression over the cockpit has almost been removed.

ACKNOWLEDGMENTS

The authors would like to thank Professors Griewank, Le Tallec and Hafez for the interest they brought to this work. Thanks also to N. Rostaing-Schmidt and C. Faure of INRIA Sophia Antipolis for their assistance in using Odyssée. Informations on Odyssée can be obtained from rostaing@sophia.inria.fr or faure@sophia.inria.fr.

REFERENCES

1. B. Mohammadi, 'A new optimal shape design procedure for inviscid and viscous turbulent flows', *Int. J. Numer. Methods Fluids*, **25**, 183–203 (1997).
2. H. Borouchaki, M.J. Castro-Diaz, P.L. George, F. Hecht and B. Mohammadi, 'Anisotropic adaptive mesh generation in two dimensions for CFD', *5th Int. Conf. on Numerical Grid Generation in Computational Field Simulations*, Mississippi State University, MI, 1996.
3. M. Castro-Diaz, F. Hecht and B. Mohammadi, 'Anisotropic grid adaption for inviscid and viscous flows simulations', *Int. J. Numer. Methods Fluids*, **25**, 175–191 (1997).
4. H. Borouchaki, P.L. George and B. Mohammadi, 'Delaunay mesh generation governed by metric specifications. Part II: applications', *Finite Elem. Anal. Des.*, **25**, 85–109 (1997).
5. O. Pironneau, *Optimal Shape Design for Elliptic Systems*, Springer, New York, 1984.
6. A. Jameson, 'Optimum aerodynamic design via boundary control', *AGARD Report 803*, Von Karman Institute Courses, 1994.
7. A. Griewank, D. Juedes and J. Srinivasan, 'ADOL-C, a package for the automatic differentiation of algorithms written in C/C++', *Tech. Rep. MCS-P180-1190*, ANL, Argonne, IL, 1991.
8. A. Griewank, 'Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation', *Optim. Methods Software*, **1**, 35–54 (1995).
9. J.C. Gilbert, G. Le Vey and J. Masse, 'La différentiation automatique de fonctions représentées par des programmes', *Rapport de Recherche 1557*, INRIA, 1991.
10. N. Rostaing-Schmidt, 'Différentiation automatique: Application à un probléme d'optimisation en météorologie', *Ph.D. Thesis*, University of Nice, 1993.
11. C. Faure, 'Splitting of algebraic expressions for automatic differentiation', *Proc. 2nd SIAM Int. Workshop on Computational Differentiation*, Santa Fe, NM, 1996.
12. B. Mohammadi, 'Optimal shape design, reverse mode of automatic differentiation and turbulence', *AIAA Paper 97-0099*, 1996.
13. M. Hafez, B. Mohammadi and O. Pironneau, 'Optimum shape design using automatic differentiation in reverse mode', *ICNMFD Conf.*, Monterey, CA, 1996.

14. P.L. Roe, 'Approximate Riemann solvers, parameters vectors and difference schemes', *J. Comput. Phys.*, **43** (1981).
15. G.D. Van Albada and B. Van Leer, 'Flux vector splitting and Runge–Kutta methods for the Euler equations', *ICASE 84-27*, 1984.
16. S. Osher and S. Chakravarthy, 'Upwind difference schemes for the hyperbolic systems of conservation laws', Math. Comput. (1982).
17. B.E. Launder and D.B. Spalding, *Mathematical Models of Turbulence*, Academic Press, New York, 1972.
18. B. Mohammadi and O. Pironneau, *Analysis of the K–Epsilon Turbulence Model*, Wiley, New York, 1994.
19. B. Mohammadi and O. Pironneau, 'Unsteady separated turbulent flows computations with wall-laws and $k-\varepsilon$ model', *Comput. Method Appl. Mech. Eng.*, **148**, 393–405 (1997).
20. J. Steger and R.F. Warming, 'Flux vector splitting for the inviscid gas dynamic with applications to finite difference methods', *J. Comp. Phys.*, **40**, 263–293 (1983).
21. A. Dervieux, 'Steady Euler simulations using unstructured meshes', *VKI lecture series 1884-04*, 1985.
22. B. Mohammadi, 'CFD with NSC2KE: a user guide', *Tech. Rep. No. 164*, INRIA, 1994.
23. B. Mohammadi, 'Différentiation automatique par programme et optimisation de formes aérodynamiques', *MATAPLI 07/96*, 1996.
24. P.L. George, F. Hecht and E. Saltel, 'Fully automatic mesh generator for 3D domains of any shape', *Impact Comp. Sci. Eng.*, **2**, 187–218 (1990).
25. P.L. George, *Automatic Mesh Generation. Applications to Finite Element Method*, Wiley, New York, 1991.
26. F. Hecht and B. Mohammadi, 'Mesh adaption by metric control for multiscale phenomena and turbulence', *AIAA Paper 97-0859*, 1997.
27. B. Mohammadi and O. Pironneau, 'New progress in optimum shapes design', in Hafez-Oshima (ed.), *CFD Review 95*, Wiley, New York, 1996.
28. B. Mohammadi, 'Automatic differentiation and non-linear PDE', *ECCOMAS*, Paris, 1996.

