Incomplete Sensitivities and BFGS Methods for 3D Aerodynamic Shape Design

Andrea Dadone, Bijan Mohammadi, Nicola Petruzzelli

.

INRIA
INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Incomplete Sensitivities and BFGS Methods for 3D Aerodynamic Shape Design

Andrea Dadone, Bijan Mohammadi and Nicola Petruzzelli

N ° 3633

March 1999

THÈME 4

ISSN 0249-6399

INRIA
ROCQUENCOURT

Incomplete Sensitivities and BFGS Methods for 3D Aerodynamic Shape Design

Andrea Dadone, *Bijan Mohammadi †and Nicola Petruzzelli ‡

Thème 4 — Simulation et optimisation
de systèmes complexes

Projet M3N

Rapport de recherche n ° 3633 — March 1999 — 20 pages

Abstract: We investigate the practicability of an optimization algorithm based on the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method for 3D shape design problems using approximate sensitivities of objective functions, where the contribution of the partial derivatives of the flow state with respect to the control variables is neglected. Therefore, it is worthwhile to investigate how an optimization method based on the Hessian behaves in this context. Indeed, the Hessian should be far from its real value if the gradient approximation is wrong. The optimization methodology is characterized by an unstructured CAD-free framework for shape and mesh deformations, an automatic differentiation of programs for the computation of the gradient of the cost function, and an unstructured flow solver. The redesign of transonic and supersonic wings has been considered and the performance of the BFGS method has been analyzed in comparison with a steepest descent method. Taking into account that a line search is too expensive to be carried out in such problems, a step size proportional to the gradient modulus has been employed for updating the control variables. Numerical results show that the BFGS method does not suffer from the approximation used in the evaluation of sensitivities, and leads to an effective improvement of the efficiency of the optimization methodology. These results can be then considered an *a posteriori* justification for incomplete sensitivities

Key-words: Aerodynamic Shape Design, BFGS method, Automatic Differentiation, Incomplete sensitivities, Unstructured Flow solver, CAD-free Framework.

* e-mail: Dadone@imedado.poliba.it
† e-mail: Bijan.Mohammadi@inria.fr
‡ e-mail: Nicola.Petruzzelli@inria.fr

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

(Résumé : tsvp)

Gradient incomplets et une méthod BFGS pour l'optimisation de formes aérodynamiques 3D.

Résumé : On présente notre algorithme d'optimisation de formes aérodynamiques 3D basé sur la méthode Broyden-Fletcher-Goldfarb-Shanno (BFGS). Une particularité de ce travail est l'utilisation d'un calcul approché pour les gradients de la fonctionnelle coût ainsi que les contraintes par rapport aux points de contrôle. Cette approximation implique que le Hessien approché utilisé dans BFGS devrait être inexact. Ceci est donc un test a *posteriori* pour l'approximation des sensibilités que nous utilisons. La méthodologie d'optimisation est basée sur un solveur non structuré, la différentiation automatique pour le calcul des gradients discrets et une paramétrisation de forme 'CAD-Free' à partir du maillage surfacique. On présente l'application de la méthode à l'optimisation d'ailes en régime transonique et supersonique. La performance de la méthode BFGS a été comparé avec celle d'un algorithme de gradient en utilisant un pas proportional au module du gradient. Les résultats numériques montrent que la méthode BFGS converge malgré l'approximation utilisée dans le calcul du gradient. Cette approximation implique une réduction appréciable du coût de l'optimisation.

Mots-clé : Optimisation de Formes Aérodynamique, Méthode BFGS, Différentiation Automatique, Gradient Incomplet, Solveur non Structuré, Paramétrisation baseé sur le Maillage.

Contents

1 Introduction

Many existing methods for solving a shape design problem are based on algorithms in which the minimization of the objective function is achieved using its gradient or sensitivity derivatives with respect to the design or control parameters. Usually, these optimization methodologies consist of a sequence of global iterations, each of them requiring a computation of the objective function, a computation of the sensitivity derivatives, and an appropriate change of the design variables according to a given gradient-based optimization algorithm. The procedure ends when the minimum of the objective function is reached. The design process may require a substantial amount of computational resources. The high computational cost comes mainly from the repeated calculation of the objective function, each evaluation requiring the solution of the flow governing equations. An additional significant cost is due to the calculation of the gradient of the objective function. Hence, the development of efficient optimization algorithms is needed, in order to reduce the number of gradient and functional evaluations necessary to reach the minimum.

There is a large number of iterative methods aimed at minimizing a function given its gradient [1, 2, 3, 4]. Standard techniques, such as line search methods, determine, at each iteration, a search direction and then find the minimum of the function using a simple one-dimensional optimization method. The simplest line search method is the steepest descent (SD) method, in which the search direction coincides with the direction of the gradient. More efficient methods define the search direction by using an approximation to the Hessian matrix of the objective function computed by using the gradient variations evaluated during the optimization process. Such methods are based on a quadratic approximation of the objective function, more effective than a linear approximation (used in the SD method) in predicting the directions along which substantial progress can be made. In this work we investigate the practicability of one of these methods, the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method [1], for optimal shape design problems, using approximate objective function sensitivities. The approximation consists in an incomplete evaluation of the sensitivities, being the partial derivatives of the flow state with respect to the control variables neglected. Such an approach comes from the observation that, when the cost function can be expressed through information over the shape (for instance through boundary integrals), the gradient is dominated by the sensitivities with respect to the shape [5, 6]. In fact, this is often the case in industrial applications, where a typical optimization problem is the drag reduction problem or the aerodynamic moment reduction problem.

The essential ingredients of our approach for optimal shape design are an unstructured flow solver, an automatic differentiation methodology for the computation of sensitivities, and a geometric model based on a CAD-free framework [7, 8, 9, 10]. The methodology has already been validated for various constrained optimization problems for compressible and incompressible flow configurations. [7, 8, 9, 10, 11, 12]. In all

the analyzed cases an optimization algorithm based on a SD method has been used. This work represents a further development of the methodology and aims at devicing an efficient and effective method for 3D fluid dynamic shape optimization problems.

As this work includes several tools, only a short description of each of them is given. In Section 2 we will formulate the optimization problem with a definition of the objective function. Section 3 is devoted to a short description of the framework of the methodology and of the optimization strategy. Subsequently, in Section 4 we review the classical algorithms which are used in the numerical solution of unconstrained optimization problems. Finally, in Section 5, an optimization algorithm based on the BFGS method will be tested by means of two wing design problems and its performances will be compared to the ones of an algorithm based on the SD method.

2 Problem Statement

Consider the following fluid flow optimization problem:

$$\begin{cases} \text{minimize} & J(x, q(x), U(q)), \\ \text{subject to} & E(x, q(x), U(q)) = 0, \\ \text{and} & g_1(x) \leq 0, \quad g_2(U(q)) \leq 0, \end{cases} \tag{1}$$

where J denotes the objective function, $x \in R^n$ describes the design parameters, $q(x)$ the geometrical entities (normals, surfaces, volumes), U the flow, E denotes the state equation and $g_{1,2}$ direct ('geometrical' on x) and indirect or state ('physical' on $U(q)$) constraints.

One of the geometrical constraints usually limits the control space. For a 3D shape optimization problem, this can be done by defining two limiting surfaces between which shape variations δx are allowed and then by imposing that:

$$f_1(x) \leq \delta x \leq f_2(x). \tag{2}$$

Other geometrical constraints enter the problem under consideration. In this work the redesign of planar wings is considered. Then, the second constraint requires that the original wing planform should remain unchanged. The last geometrical constraint considered is that the volume V of the wing has to be conserved or increased during the optimization process. The first and second constraints can be taken into account by using a projection operator, which filters the variations of the design parameters. The last constraint can be introduced in the cost function as a penalty term.

Finally, physical constraints concern the aerodynamic coefficients and characteristics of the flow. For instance, it may be required to reduce the drag conserving or increasing the lift or to minimize the aerodynamic moment coefficient for given lift and drag coefficients. Physical constraints are introduced in the cost as penalty terms.

By introducing penalty terms in the cost function in order to take into account the constraints, the constrained minimization problem, Eqs. (1), is transformed into a unconstrained minimization problem with respect to the shape geometry and the flow state. Such an approach defines a new functional, the penalty function, such that its minimum represents an optimal solution.

3 Methodology Framework

This section briefly outlines all the components of our optimization methodology, with emphasis on aspects related to 3D applications.

3.1 Flow Equations and Solver

This section is devoted to the description of the state equation $E(x, q(x), U(q)) = 0$, which represents the steady Euler equations apt to describe flows of compressible inviscid fluid. The numerical scheme used for its solution will be also outlined.

The three-dimensional Euler equations for unsteady flows in conservative form can be written as:

$$\frac{\delta U}{\delta t} + \nabla \cdot (F(U)) = 0, \tag{3}$$

where $U = (\rho, \rho\vec{u}, \rho E)^T$ is the vector of conserved variables and F represents the advective operator. This system is closed using the equation of state $p = p(\rho, T)$.

To discretize the equations of motion a finite-volume-Galerkin approach has been applied. A polygonal bounded domain of computation has been defined and divided into tetrahedrical cells. The equations have been solved using the flux-difference splitting method of Roe [13]. A MUSCL-type extrapolation, with a formal second-order accuracy in space, has been applied to extrapolate the physical variables onto the left and right sides of each cell edge. The Van Albada limiter has been used to compute oscillation-free solutions. The steady-state solution of Eq. (3) has been obtained with an explicit time integration using a four-stage Runge-Kutta scheme. To accelerate the convergence of the scheme, a local time stepping technique has been employed. The initial solution and boundary conditions are classical. In particular, inflow and outflow boundary conditions have been prescribed according to the theory of characteristics. On solid walls, the only contribution to the convective fluxes is the pressure. Additional details for finite-volume-Galerkin technique and for boundary conditions treatment can be found in references [13, 14, 15, 16, 17].

3.2 CAD-Free Geometric Model

Shape deformation and grid generation are performed using a CAD-free geometric model. This is based on the following points:

1. All the nodes on the body are control points.
2. A grid movement technique is used to generate a new mesh.

The assumption in point 1 allows to make shape deformation without interfacing the design process to the CAD-system and avoids problems related to the propagations of control point variations to the other body discretization points. To preserve the initial local regularity of the shape during the design process and to avoid oscillations, a 'local' smoothing operator has then been defined over the shape, which consists in performing a few Jacobi iterations to solve the following system:

$$\begin{cases} (I - \varepsilon\Delta)\delta\tilde{x}_w = \delta x_w, \\ \delta\tilde{x}_w = \delta x_w = 0 \quad \text{on wedges}, \end{cases} \tag{4}$$

where $\delta\tilde{x}_w$ is the smoothed shape variation for the body points, δx_w is the variation given by the optimization tool, and ε is a positive parameter. In order to perform a local smoothing, the parameter ε has a value proportional to the norm of the second derivatives of the deformation and is set to zero (this is done during the Jacobi loops) if

$$\frac{\delta_{ij}(\delta x_w)}{(\delta x_w)_T} < 10^{-3},$$

where $\delta_{ij}(\delta x_w)$ is the difference between the variations of the two nodes of each of the segments of a surface triangle and $(\delta x_w)_T$ the mean variation on this triangle.

The grid movement technique mentioned at point 2 consists in creating a new grid by perturbing the coordinates of the original grid points in a way which is consistent with the perturbation of the body points. This is done by solving a volumic elasticity system, similar to Eq. (4):

$$(I - \eta\Delta)\delta x_m = \overline{\delta x_m}, \tag{5}$$

under the conditions:

$$\begin{cases} \delta x_m = 0 & \text{at inflow and outflow boundaries}, \\ \frac{\partial \delta x_m}{\partial n} = 0 & \text{at slip boundaries}, \\ \delta x_m = \overline{\delta x_m} & \text{at wall nodes}. \end{cases} \tag{6}$$

In Eq. (5), $\overline{\delta x_m}$ is the extension of δx_w over the mesh defined by

$$\begin{cases} \overline{\delta x_m} = \delta x_w & \text{for mesh nodes on the wall}, \\ \overline{\delta x_m} = 0 & \text{for internal nodes}, \end{cases} \tag{7}$$

and η is a positive parameter. To avoid mesh degeneration, the value of the parameter η is proportional to the inverse of the local element size. Such an approach forces the new grid to remain topologically identical to the initial one.

The shape and the mesh deformation tools have been already used in optimization problems in 2D and 3D configurations for compressible flows [7]. They have been proved to be apt to treat extremely fine meshes with a computational cost equivalent to the one needed to perform few flow iterations.

3.3 Incomplete Gradient Evaluation

All ingredients presented, i.e., the flow solver and the shape and mesh deformation tools, have to be differentiated in order to compute the gradient of the objective function. More exactly, it is given by:

$$\frac{dJ}{dx} = \frac{\partial J}{\partial x} + \frac{\partial J}{\partial q}\frac{\partial q}{\partial x} + \frac{\partial J}{\partial U}\frac{\partial U}{\partial q}\frac{\partial q}{\partial x}. \tag{8}$$

In our approach for optimal shape design, we use an automatic differentiation (AD) methodology [18, 19, 20, 21] to produce the sensitivity derivatives of the discrete objective function. Usually, AD tools can be operate with two basic procedures which are referred to as forward and reverse modes. We use the reverse mode procedure, which is closely related to adjoint methods [3, 22, 23], and then suitable to compute gradients when the number of the design variables is very large.

The most difficult part in the computation of the gradient of the objective function (8) is the differentiation of the flow state U, especially when the solution of the Navier-Stokes equations and turbulence models are involved. Indeed, the physical solver part is often nondifferentiable and contains several nested loops (for instance, the time integration loop and loops over nodes and elements in a finite element solver) [10]. This is a real difficulty for the reverse mode of AD, especially for time dependent problems, as we need to store all the intermediate states. In the past, it has been shown how to use automatic differentiation in reverse mode to perform this task [3, 10]. More precisely, we computed the following sensitivities:

$$\delta x_w \rightarrow \delta x_m \rightarrow \delta q \rightarrow \delta U \rightarrow \delta J(x, \delta q, U),$$

Through different numerical experiences, in References [5, 6] it has been noticed that, when the cost function is based on local information around the shape (through boundary integrals for instance), the dominant part of the gradient comes from the partial derivatives with respect to the shape. Hence, for this class of problem, References [5, 6] suggest to neglect the derivative with respect of the state U in Eq.(8). This also means that, for small variations of the shape, the state remains almost unchanged. In other words, only the shape and the mesh deformation tools together with the geometrical part of the flow solver have to be differentiated.

3.4 Optimization Procedure

Having defined all the components of the design methodology, it is now possible to outline the complete optimization procedure. We denote the cost function $J(x, q(x), U(q))$ by $J(x)$. The strategy used to solve the problem (1) is the following:

1. Start with an initial set of design variables x.
2. Solve the flow equations.
3. Compute the gradient: $\frac{dJ(x)}{dx}$.
4. Compute the design variables variations $\overline{\delta x}$ according to a given gradient-based optimization algorithm.
5. Define the new shape deformation: $\delta x = \Pi(\overline{\delta x})$, where Π is the projection operator over the admissible space.
6. Smooth the deformations.
7. Deform the mesh.
8. Solve the flow equations for the new geometry starting from the last flow solution.
9. Repeat steps 3 to 8 until the objective function reaches its minimum, or, equivalently, until the gradient becomes sufficiently small.

4 Optimization Algorithms

There is a huge number of methods designed to optimize a smooth function $f(x)$ when its vector gradient $\nabla f(x)$ can be evaluated [1, 2, 3]. In unconstrained optimization, the most traditional approach are the *line search algorithms*. In this section some general features of such methods are described.

A typical line search algorithm proceeds as follows:

$$\left\{\begin{array}{l} 1. \quad \text{Start with an initial guess} \quad x^0 \quad \text{for the design parameters.} \\ 2. \quad \text{For} \quad k = 0, 1, 2, ..., \text{until satisfactory convergence is achieved :} \\ 2.1 \quad \text{determine a direction of search} \quad \mathbf{s}^k; \\ 2.2 \quad \text{find} \quad \alpha^k \quad \text{to minimize} \quad f(x^k + \alpha \mathbf{s}^k) \quad \text{with respect to} \quad \alpha; \\ 2.3 \quad \text{set} \quad x^{k+1} = x^k + \alpha^k \mathbf{s}^k. \end{array}\right. \tag{9}$$

Different methods correspond to different choices for $\mathbf{s}^k$ in step 2.1, based on information available in the model. In particular, our attention is focused on methods in which the direction of search $\mathbf{s}^k$ satisfies the *descent property*, i.e. ${\mathbf{s}^k}^{\mathrm{T}} \Delta x^k < 0$. This ensures that the slope $\mathrm{d}f/\mathrm{d}\alpha = \mathbf{s}^{\mathrm{T}} \Delta x$ is always negative at $\alpha = 0$ (unless x^k is a stationary point) and then the function $f(x^k + \alpha \mathbf{s}^k)$ can be reduced for some $\alpha^k > 0$.

In the *steepest descent* (SD) *method* $\mathbf{s}^k = -\nabla x^k$ for all k, i.e. the method searches in the steepest descent direction along which the objective function decreases most rapidly at x^k.

In order to obtain a more suitable search direction, some methods operate a deflection of the gradient vector. These methods are formulated by using a *quadratic model*, i.e. by constructing a local quadratic approximation of the objective function. A quadratic model takes into account the curvature of the level lines of the objective function. Accordingly, then, it is more effective than the linear approximation (used for the SD method) in predicting directions along which substantial progress can be made, so that methods with a superlinear rate of convergence can be derived.

In the *quasi-Newton's methods* the search direction is given by $\mathbf{s}^k = -\mathbf{H}^k \nabla f(x^k)$, where $\mathbf{H}$ is a symmetric definite matrix, constructed at each iteration, which approximates the inverse of the Hessian matrix of the objective function. In the literature there are several formulas for the calculation of $\mathbf{H}^{k+1}$ from $\mathbf{H}^k$. The attempt is to construct $\mathbf{H}^{k+1}$ by using second derivative information obtained by using the variation of the gradient with respect to its value at the previous kth iteration. The Broyden-Fletcher-Goldfarb-Shanno (BFGS) update formula is one of the most successful scheme [1]. It is given by:

$$\mathbf{H}^{k+1} = \mathbf{H}^k + \left(1 + \frac{\gamma^{k^{\mathrm{T}}} \mathbf{H}^k \gamma^k}{\delta^{k^{\mathrm{T}}} \gamma^k}\right) \frac{\delta^k \delta^{k^{\mathrm{T}}}}{\delta^{k^{\mathrm{T}}} \gamma^k} - \left(\frac{\delta^k \gamma^{k^{\mathrm{T}}} \mathbf{H}^k + \mathbf{H}^k \gamma^k \delta^{k^{\mathrm{T}}}}{\delta^k \gamma^k}\right), \tag{10}$$

where the vectors δ^k and γ^k are defined as:

$$\delta^k = \alpha^k \mathbf{s}^k = x^{k+1} - x^k, \tag{11}$$

$$\gamma^k = \nabla f\left(x^{k+1}\right) - \nabla f\left(x^k\right). \tag{12}$$

Even if $\mathbf{H}^k$ is definite positive and thus ensures the descent property, some restrictions must be placed on the choice of the search direction $\mathbf{s}^k$, when $\mathbf{s}^k$ is close to be orthogonal to the steepest descent vector, in order to avoid the possibility to make negligible reductions in $f(x)$. This possibility can be avoided if $\mathbf{s}^k$ satisfies an *angle criterion*, i.e. if the angle θ^k between $\mathbf{s}^k$ and $-\nabla f\left(x^k\right)$ is uniformly bounded away from orthogonality:

$$\theta^k \leq \frac{\pi}{2} - \mu \quad \forall k, \tag{13}$$

where $\mu > 0$ is independent of k and $\theta^k \in [0, \pi/2[$. In the SD method this criterion is satisfied with $\theta^k = 0$. In the case of Newton-like methods, it can be proved that a sufficient condition for the angle criterion is that the spectral condition number κ^k of $\mathbf{H}^k$ is bounded above, independently of k, i.e.:

$$\theta^k \leq \frac{\pi}{2} - \kappa^{k^{-1}} \quad \forall k. \tag{14}$$

In this case, there are two possibilities to satisfy the above condition. One can modify $\mathbf{H}^k$ such that k^k is bounded and a well conditioned system is obtained. Alternatively and more simply, one can add some multiple of $-\nabla f\left(x^k\right)$ to $\mathbf{s}^k$, in order to satisfy the condition (13). Obviously, the superlinear convergence property can degrade to a linear one, when these modifications operate.

In general, it is not possible to perform an *exact line search*, i.e. to compute exactly the solution α^k in step 2.2 of the algorithm (9). As a consequence, step 2.2 must be approximated, i.e., an *inexact line search* has to be employed. In the literature there are several iterative methods to compute the minimum of a one-dimensional function $f(\alpha)$. When the angle criterion is satisfied, it is possible to state a global convergence theorem for descent methods with an inexact line search performed under appropriate conditions on α^k. This implies that, in general, no convergence is guaranteed when line search is not performed. Quasi-Newton methods usually operate resetting $\mathbf{H}^k$ every $n+1$ iterations (n being the number of the control parameters) to the initial matrix $\mathbf{H}^1$. Obviously, this is not a practicable strategy when treating very large control space problems. An alternative strategy is to operate a continuing update, without any resets. In general, in order to assure robustness, $\mathbf{H}^k$ is reset every $m << n$ and, at the limit, every $m = 2$ iterations. Although the global convergence of such schemes requires rather stringent conditions, the local convergence rate behavior is often asymptotically superlinear, even without performing a line search.

In the present work, a continuing update of the Hessian will be operated. Moreover, a step size α^k proportional to the gradient modulus will be used to update the control variables. This implies that in general total convergence theorem does not hold and then the decreasing of the objective function is not *a priori* guaranteed.

5 Numerical Results

In order to analyze the performance of the BFGS algorithm, two optimization problems have been considered, involving a wing in transonic and supersonic flow conditions. In both cases the goal of the optimization was to minimize the drag coefficient, the lift coefficient and the volume of the wing being constrained to be greater than or equal to their initial values. The penalty objective function has been defined as:

$$J(x, q(x), U(q)) = \frac{C_d}{C_d^\circ} + \alpha\ max\left(0, \frac{(C_l^\circ - C_l)}{C_l^\circ}\right) + \beta\ max\left(0, \frac{(V^\circ - V)}{V^\circ}\right) \qquad (15)$$

where α and β are penalty parameters, C_d and C_d° the actual and the initial drag coefficients, C_l and C_l° the actual and the initial lift coefficients, V and V° the actual and the initial volumes. In both test cases, the minimization problem has been solved using the BFGS method as well as the SD method. The same penalty parameters values

have been obviously utilized. When using the BFGS method, at each optimization step, the variation of the design parameters δ^k required by the Hessian updating formula (10) is the one obtained by applying the projection operator. It has been numerically observed that a small correction of line direction is required in order to satisfy, at each iteration, the angle condition $\theta \leq \frac{\pi}{2} - \mu$, the value of μ having been assumed equal to 30^o. As a consequence, the angle condition has been ensured by adding a fraction of the gradient vector to the search vector. This ensures to take advantage of the quadratic information of the objective function, namely to avoid search directions too close to the gradient direction.

5.1 Transonic Flow

The initial geometry corresponds to the ONERA-M6 wing. This wing is characterized by a swept planar planform with a value of the aspect ratio equal to 7.55 and a sweep angle equal to 26.6 degrees at the quarter-chord line. The present computation has been performed using an unstructured grid with nearly 48000 tetrahedra. Nearly 2500 grid points have been placed on the surface wing. The upper surface grid for the baseline configuration is shown in Fig. 1. The flow is characterized by a free-stream Mach number equal to 0.85 and an angle of attack equal to 3 degrees. The solution of the flow equations has been considered converged to a steady state solution when the residual dropped to 10^{-6}. Convergence has been obtained approximately in about 900 iterations with a value of the CFL number equal to 0.9. The initial values of the lift coefficient, C_l°, of the drag coefficient, C_d°, and of the Volume, V°, are 0.1385, 0.0237 and 0.05890 respectively. Fig. 2 shows the convergence history of the objective function *J(x)*, obtained using the BFGS method and the SD method. The abscissa in Fig. 2 is the required work, with a unit of work considered as the computational time required to run a single analysis solution to convergence [24]. The evaluation of the required work has been made by taking into account, at each optimization step, the computational time required by the flow solution, by the gradient evaluation, and by the shape and mesh deformation. The cost of the last three terms is negligible, because it corresponds to the computational cost for one flow iteration. The additional work needed to update the Hessian in the BFGS method is equal to the cost required to run seven explicit flow iterations, namely 0.0078 work units. As refers to the required work, this means that one global step of the BFGS optimization procedure is approximately equivalent to one global step of the SD optimization procedure. Fig. 2 shows that both methods reduce the objective function by approximately 22%, the BFGS method being nearly two times faster than the SD method. Obviously, the wing shapes obtained by means of the two methods are practically coincident. BFGS method reaches the minimum in 98 iterations, while the SD method requires 105 iterations.

The initial and optimized wing shapes at different spanwise locations along the wing together with the corresponding pressure coefficient distributions are shown in Fig.

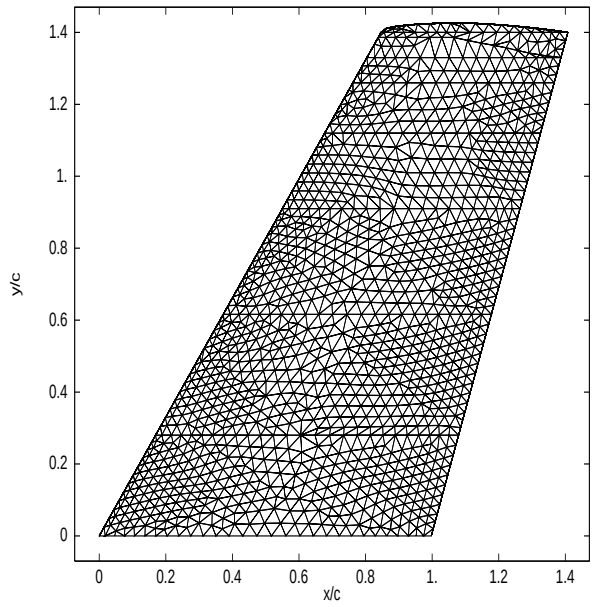


Figure 1: Transonic Flow. Upper surface grid baseline configuration.

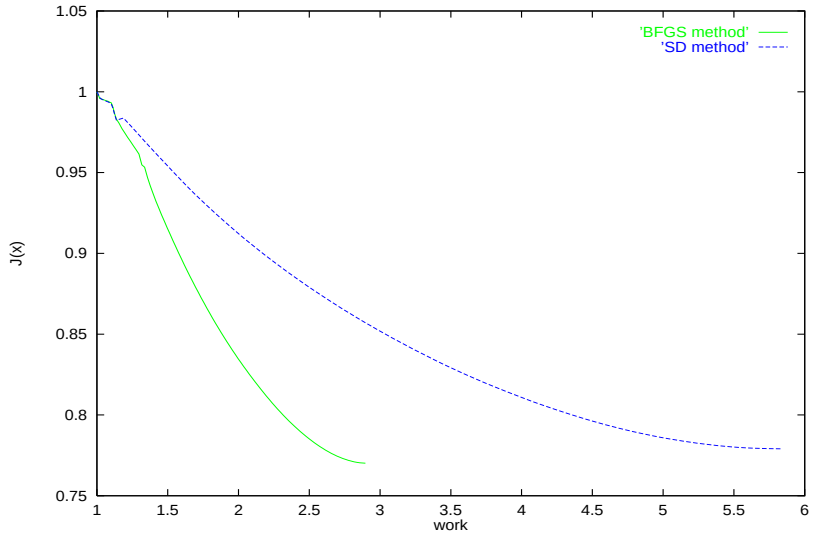
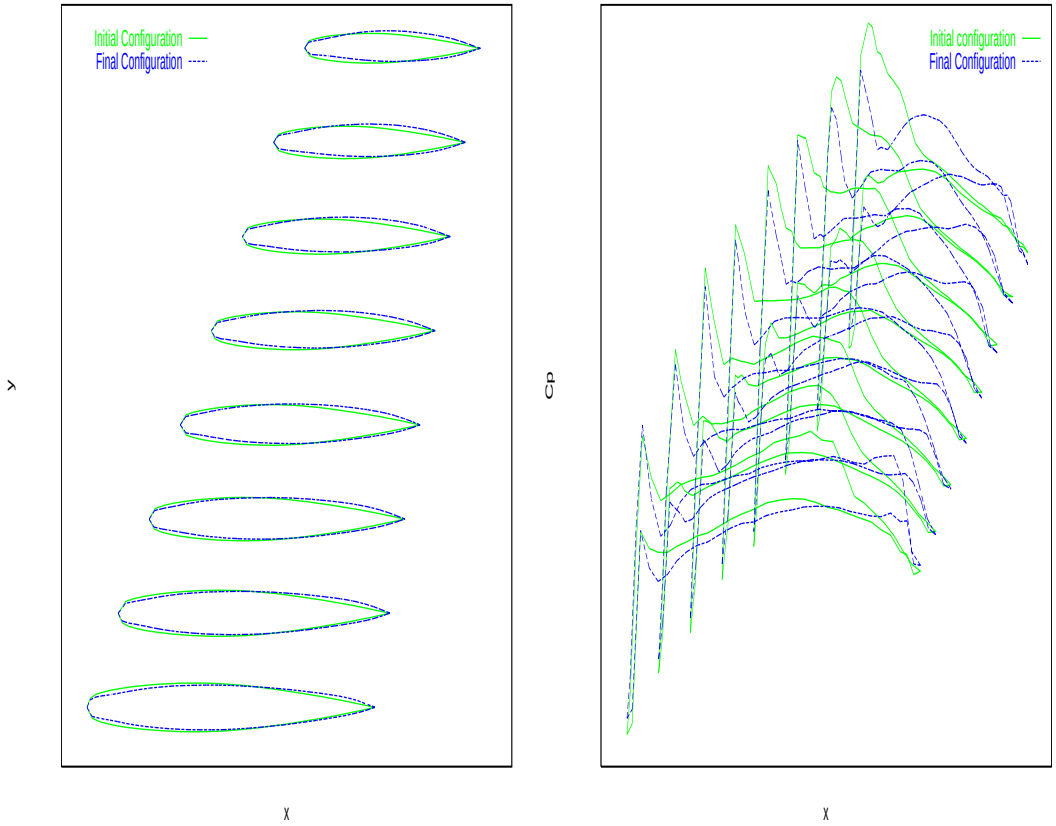


Figure 2: Transonic Flow. Convergence for the objective function. One unit of work corresponds to the time to converge a single flow analysis.



(a) Initial and optimized wing shapes (deformed, for a better view).

(b) Initial and optimized pressure coefficient distributions.

Figure 3: Transonic Flow. BFGS method. Optimal design results at 0%, 12.5%, 25%, 37.5%, 50%, 62.5%, 75%, 87.5%, 100% semispan.

3. The by-section definition of the wing is not available so that the plotted cross-section results have been obtained by interpolation. The three dimensional nature of the optimization process is proven by the spanwise variations of the shape modifications, as shown in Fig 3a. Moreover, Fig 3b outlines that the shock located on the upper surface of the initial wing has been slightly smoothed. The drag coefficient has been reduced by 19.3%, the lift coefficient has been increased by 19.8%, while the volume remained almost unchanged (variation lower than 0.5%).

5.2 Supersonic Flow

The initial geometry is an Agard wing with an aspect ratio of 3.3 and a sweep angle of 45 degrees at the leading edge line. The computational mesh has 120000 tetrahedra, 2800 grid points being located on the surface wing. The upper surface grid for the

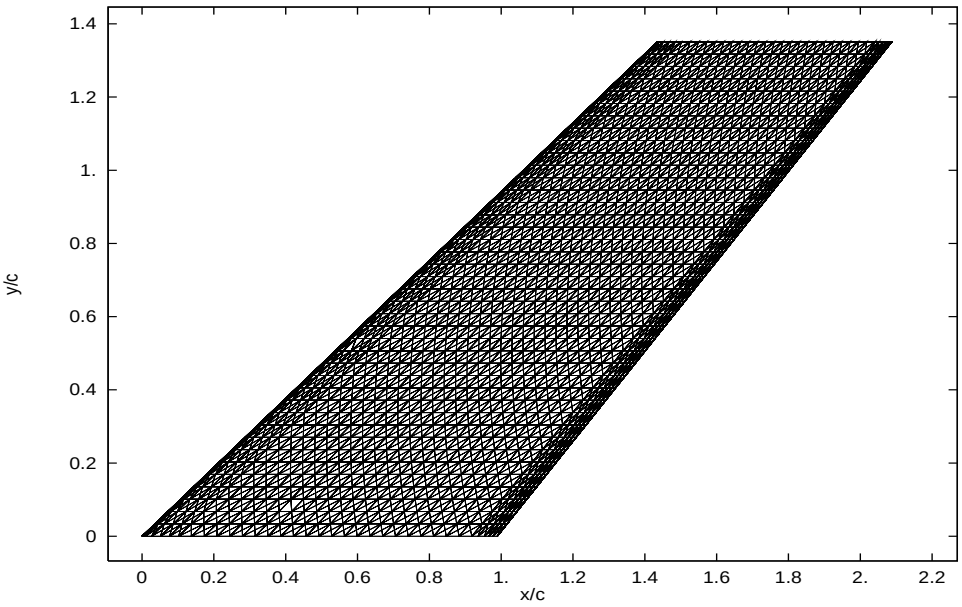


Figure 4: Supersonic Flow. Upper surface grid baseline configuration.

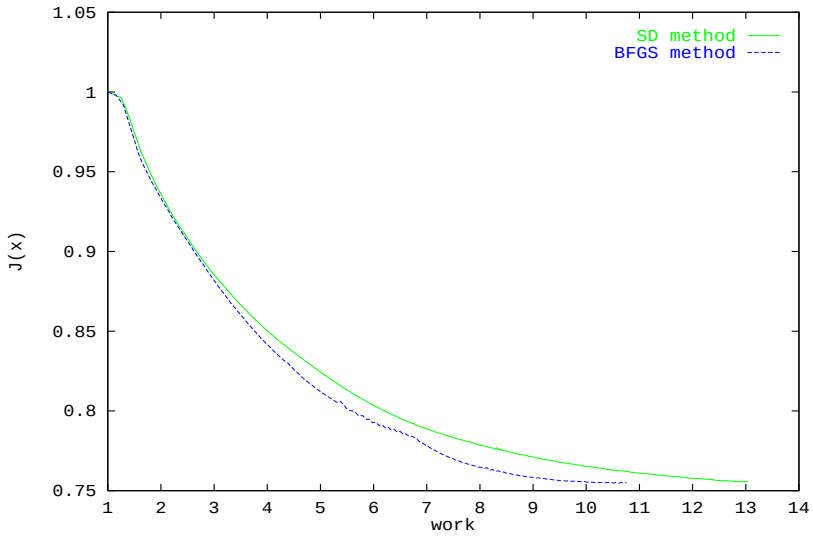
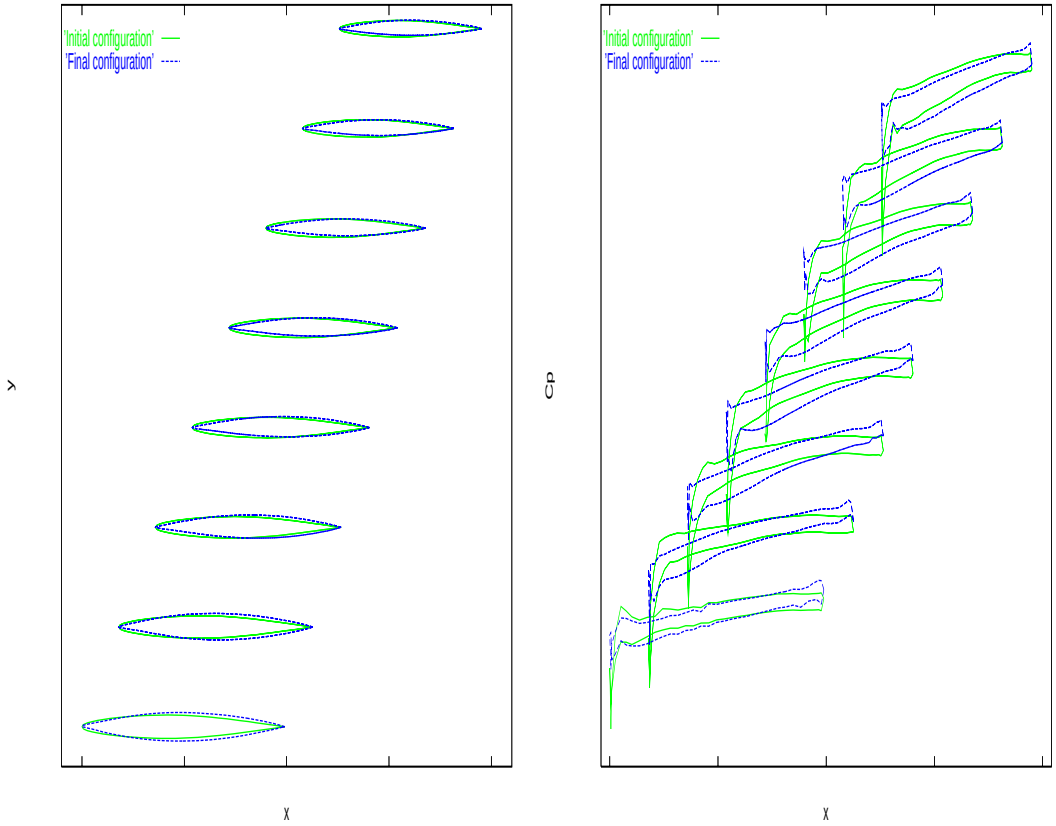


Figure 5: Supersonic Flow. Convergence for the objective function. One unit of work corresponds to the time to converge a single flow analysis.

baseline configuration is shown in Fig. 4. The flow is characterized by a free-stream Mach number of 2 and an angle of attack of 2 degrees. The flow solver requires nearly 300 iterations to drop the residual by three orders of magnitude with a value of the CFL number equal to 1.3. The initial values of the lift coefficient, C_l°, of the drag coefficient, C_d°, and of the Volume, V°, are 0.04850, 0.00697 and 0.030545 respectively. Fig. 5 shows the convergence history of the objective function *J(x)* versus the required work, for the BFGS method and the SD method. The work required by the Hessian computation is equal to 0.027 and is equivalent to the work required to perform 8 flow iterations. The convergence plots contain several slope discontinuities, caused by the constraints activation. In the present test case, the volume and lift constraints represent more stringent design conditions than in the previous test case. The BFGS method exhibits a more rapid reduction of the objective function after the initial iterations. The objective function has been reduced by 23.5%. The BFGS method reaches the minimum in 150 iterations, while the SD method requires 450 iterations. In this design problem the efficiency improvement is less attractive in comparison to the previous problem. This is partly explained by the fact that, at each step of the optimization procedure, the search direction given by the BFGS method is very close to the steepest descent direction. An other reason lies in the more significant cost of the Hessian computation. Indeed, less iterations are required to converge the flow equations in the present supersonic test case. As a consequence, the cost of the Hessian computation represents a greater percentage of the cost of the flow solution with respect to the previous transonic design problem. Fig. 6 shows the initial and the optimized wing shapes at various spanwise locations along the wing together with the corresponding pressure coefficient distributions. The drag coefficient has been reduced by 28%, the lift coefficient has been increased by 4%, while the volume remained almost unchanged.

6 Concluding Remarks

In this work we have presented the application of a line search optimization algorithm, based the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method, to 3D aerodynamic shape optimization problems. The optimal shape design methodology is characterized by an incomplete evaluation of the objective function, being the partial derivatives of the flow state with respect to the control variables neglected. The optimization algorithm has been tested by means of two drag minimization problems for a transonic and a supersonic wing. Its performances have been compared with the ones of an algorithm based on the steepest descent (SD) method. A step size proportional to the gradient modulus has been employed to update the control variables, because a line search is too expensive for such problems. The results obtained indicate that the BFGS method is more efficient than the SD method and does not suffer from the approximation in the evaluation of the gradient. Thus, these results can be considered an *a posteriori* validation of the approximation used in the gradient evaluation.



(a) Initial and optimized wing shapes (deformed, for a better view).

(b) Initial and optimized pressure coefficient distributions.

Figure 6: Supersonic Flow. BFGS method. Optimal design results at 0%, 12.5%, 25%, 37.5%, 50%, 62.5%, 75%, 87.5%, 100% semispan.

7 Acknowledgment

The research of the first author has been supported by the Italian Agency MURST (Ministero dell'Universita' e della Ricerca Scientifica e Tecnologica), while the thirth author has been supported by ESF (European Science Foundation). The authors acknowledge the helpful discussion of G. Medić and M. Stanciu of INRIA Rocquencourt and of G. Pascazio of Politecnico di Bari on this subject.

References

[1] R. Fletcher (1987), *Practical Methods of Optimization*, Wiley, New York.

[2] M. S. Bazaraa, H. D. Sherali, C. M. Shetty (1993), *Nonlinear Programming - Theory and Algorithm*, Wiley, New York.

[3] O. Pironneau (1984), *Optimal Shape Design for Elliptic Systems,* Springer-Verlag, New York.

[4] G. N. Vanderplaats (1980), *Numerical Optimization Techniques for Engineering Design*, McGraw-Hill Book Company, New York.

[5] B. Mohammadi (1997), *A New Optimal Shape Design Procedure for Inviscid and Viscous Turbulent Flows*, International Journal for Numerical Methods in Fluids, vol. 25,183-203.

[6] B. Mohammadi (1998), *An Unified Formulation for Shape Optimization and Flow Control*, to appear in International Journal for Numerical Methods in Fluids.

[7] B. Mohammadi (1996),*Optimal Shape Design, Reverse Mode of Automatic Differentiation and Turbulence*, AIAA paper 97-0099.

[8] B. Mohammadi, J.M.Malé, N. Rostaing Schmidt (1995), *Automatic Differentiation in Direct and Reverse Modes: Application to Optimum Shapes Design in Fluid Mechanics*, proc. SIAM workshop on AD, Santa Fe, SIAM.

[9] B. Mohammadi (1996), *Différentiation Automatique par Programme et Optimisation de Formes Aérodynamiques*, MATAPLI 07/96.

[10] B. Mohammadi (1997), *Practical Application of Automatic differentiation for optimization for flow problems*, VKI courses, 1997-05.

[11] G. Medić, B. Mohammadi, N. Petruzzelli, M. Stanciu (1999), *3D Optimal Shape Design for Complex Flows: Application to Turbomachinery*, AIAA paper 99-0130, To appear in Proceedings of the 37th AIAA Aerospace Sciences Meeting, Reno, Nevada, USA.

[12] G. Medić, B. Mohammadi, N. Petruzzelli, M. Stanciu (1998), *A new approach in optimal blade design*, GDR Optimization Meeting, Rennes.

[13] P.L.Roe (1981), *Approximate Riemann Solvers, Parameters Vectors and Difference Schemes,* J.C.P. Vol.43.

[14] P. Rostand, B. Stoufflet (1988), *Finite Volume Galerkin Methods for Viscous Gas Dynamics*, Rapport de Recherche INRIA 863.

[15] A. Dervieux (1985), *Steady Euler Simulations using Unstructured Meshes*, VKI lecture series, 1884-04.

[16] G.D.Van Albada, B. Van Leer (1984), *Flux Vector Splitting and Runge-Kutta Methods for the Euler Equations*, ICASE 84-27.

[17] J. Steger, R.F. Warming (1983), *Flux Vector Splitting for the Inviscid gas dynamic with Applications to Finite-Difference Methods*, J. Comp. Phys. 40, pp: 263-293.

[18] J.C. Gilbert, G. Le Vey, J. Masse (1991), *La différentiation automatique de fonctions représentées par des programmes*, Rapport de Recherche INRIA 1557.

[19] N. Rostaing-Schmidt (1993), *Différentiation automatique: Application à un problème d'optimisation en météorologie*, Ph.D. Thesis, University of Nice.

[20] C. Faure (1996), *Splitting of Algebraic Expressions for Automatic Differentiation,* In proc. of the Second SIAM Inter. Workshop on Computational Differentiation, Santa Fe.

[21] C. Faure, Y. Papegay (1997), *Odyssée Version 1.6. The User's Reference Manual*, Rapport Technique INRIA 0211.

[22] A. Jameson (1994), *Optimum Aerodynamic Design via Boundary Control*, AGARD Report 803, Von Karman Institute Courses.

[23] P. D. Frank, G. R. Shubin (1992), *A Comparison of Optimization-Based Approaches for a Model Computational Aerodynamics Design Problem*, Journal of Computational Physics, pp. 74-89.

[24] A. Dadone, B. Grossman (1998), *Progressive Optimization of Inverse Fluid Dynamic Design Problems*, MAD Center Report 98-05-01, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, to be published by Computers and Fluids.

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)
ISSN 0249-6399