A Hybrid Inverse Aerodynamic Design Using a Variable Metric Optimization Algorithm

A. H. Souza
PME–EPUSP
alexandre.henrique@poli.usp.br

G. L. Oliveira
EMBRAER – Propulsion Systems Engineering
guilherme.oliveira@embraer.com.br

L. C. C. Santos
EMBRAER – CFD Environmental Systems – IME– USP
luis.castro@embraer.com.br

E. V. Volpe
PME–EPUSP
ernani.volpe@poli.usp.br

Abstract. *Over the last two decades, CFD has played an increasingly important role in aerospace design. For it provides a very cost-effective means for one to analyze different configurations. More recently, the development of inverse design methods has opened up new possibilities. On combining those methods with CFD, one can specify elaborate design goals and search for configurations that meet them. Among these methods, the well-known Modified Garabedian Mcfadden (MGM) is an attractive alternative, for its simplicity and effectiveness. In essence, it compares the pressure distribution on control stations along the wing to target distributions. The location of those stations and the target distributions are both specified by the user. The method then changes those stations geometry so as to obtain the desired distributions. Wing lofting is gotten by interpolation between control stations. Geometric constraints are often a design requirement, owing to their role in the aerodynamic and structural specifications. The hybrid version of MGM enables one to impose such constraints. This paper proposes the use of a variable metric algorithm (VMA) as a cost-effective means to impose geometric constraints. Tests are run to compare the convergence performance of VMA with the preconditioned conjugate gradient algorithm (PCGA), which is currently in use.*

keywords: *MGM, Inverse Design, Variable Metrics*

1. Introduction

The enormous progress that has been achieved in computational fluid dynamics (cfd) over the last two decades has raised it to a crucial role in the aerospace industry (MacCormack, 1993; Santos, 1993). Flow simulation codes are used in virtually all phases of aircraft design, from conception to components design. As Jameson, 1997, notes, flow simulations now play a complementary role to that of wind tunnel tests in the aerospace industry.

In particular during the conceptual design phase, cfd makes for a very cost–effective means of analyzing different configurations, when compared to experimental testing. Even so, the cumulative costs of complex flow simulations can become prohibitive. That is the case when a trial–and–error approach is adopted to explore a large variety of possible designs, and the corresponding flow simulations build up in number. In addition, the odds of success of a such an approach rely heavily on the designer's experience, and they grow dimmer as the configurations variety increases, or as more elaborate conditions are imposed on the design.

Inverse design methods can have a very positive effect in this situation. On combining them with cfd codes, one makes for a more efficient way to explore the design space. In general, these methods compare actual surface pressure distributions to target distributions. Then they estimate geometry changes that should lead to the desired results. A number of inverse design methods have been proposed in the last two decades. Among them, the so–called Modified Garabedian–MacFadden method (MGM) remains an attractive alternative, for its simplicity and reliability.

More recently, a hybrid formulation of MGM has been developed by Santos, 1993, which enables one to impose geometric constraints on the inverse design. What sets the hybrid apart from the original formulation is the fact that it gives MGM the character of an aerodynamic optimization method. It does not simply estimate geometry changes that should lead to desired effects. Instead, it attempts to achieve those effects, while still satisfying a set of constraints. Upper and lower bounds for airfoil thickness are of special interest in inverse design applications. For they have direct bearings on the aerodynamic and structural design of an aircraft.

The purpose of this paper is to compare two implementations of the hybrid MGM, which make use of well–known optimization methods: the Preconditioned Conjugate Gradient (PCG) and the Variable Metric (VM) algorithms. Applications of both methods to airfoil inverse design are presented, with the aim of comparing their performance.

2. The Modified Garabedian–MacFadden Method

The conceptual foundations of MGM lie in the classic 2–D wavy–wall problem. In effect, it is based on the linearized small disturbance potential flow solution to that problem. It treats the airfoil upper and lower sides separately, and considers each one of them as a portion of a wavy–wall. For each portion, MGM compares the actual pressure coefficient distribution to a target distribution, which is previously specified by the user. On the basis of the difference between the two, it estimates geometry changes that should have the airfoil attain the desired distribution. Convergence is achieved when that difference falls below a prescribed level. The geometry changes are estimated by the model equation,

$$A(\delta z) + B\frac{\partial(\delta z)}{\partial x} - C\frac{\partial^2(\delta z)}{\partial x^2} = C_{pt} - C_p \tag{1}$$

where C_p represents the actual pressure coefficient distribution, C_{pt} stands for the corresponding target distribution. The x coordinate runs along the airfoil chord, from its leading edge to the trailing edge, and δz represents the airfoil contour changes. The coefficients A, B and C are arbitrary constants that are picked so as to make the inverse design cycles stable and, possibly, to accelerate convergence (Santos, 1998; Santos, 1999). Homogeneous Dirichlet boundary conditions are imposed at both ends of the domain: the leading and trailing edges.

$$\delta z|_{x=0} = \delta z|_{x=c} = 0 \tag{2}$$

The symbol c represents chord length. In the literature the RHS of eq. (1) is often replaced by the difference between squared velocity distributions $q_t^2 - q^2$, where q represents the dimensionless perturbation velocity in the flow direction $q \equiv u/U_\infty$ (Silva and Sankar, 1992). Both forms of the model equation are equivalent, since the relation between q^2 and C_p for steady potential flow is given by

$$q^2 = 1 - \frac{2}{(\gamma-1)M_\infty^2}\left[\left(1 + \frac{\gamma M_\infty^2}{2}C_p\right)^{\frac{(\gamma-1)}{\gamma}} - 1\right] \tag{3}$$

where M_∞ represents the far–field Mach number and γ is the specific–heat coefficients ratio $\gamma = c_p/c_v$. Owing to its conceptual foundations, MGM is suitable for the transonic flow regime.

The only piece of information from the flow solution the method actually needs is the C_p distribution. In that lies one of the most important features of MGM, which is its independence from the flow solver. It implies that an inverse design loop can be assembled on coupling the MGM routine with geometry and mesh generators, a flow solver and a post–processing unit to compute C_p distributions, as is shown in fig. 1

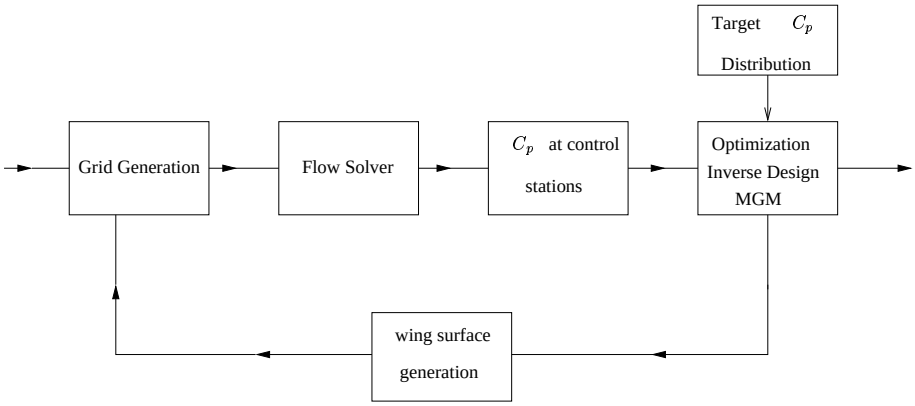

Figure 1: Inverse design loop. The MGM routine is nested in a loop along with geometry and mesh generators, a flow simulation code and a post–processing routine to evaluate C_p distributions

On accounting for its 2–D character, the use of MGM for wing design involves specifying a set of control stations along its span, as is depicted in fig. 2.a. The method is then applied to each one of them separately, thus changing their geometry— fig. 2.b. The wing lofting is gotten by surface interpolation between control stations.

Naturally, the fact that MGM is based on 2–D linearized potential flow raises important questions as to how effective it could be in 3–D applications, such as wing design. In addition to that, there is the question regarding the limitations of such a flow model, specially when the flow simulation may involve more complex models— such as the full Navier–Stokes equations, for example.

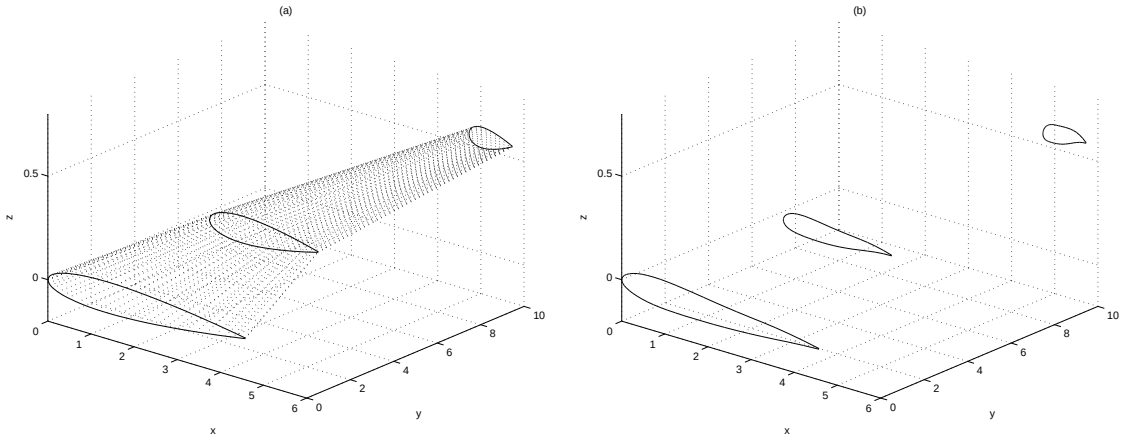

Figure 2: Schematic representation of wing inverse design: (a) original geometry showing three control stations. (b) geometry of control stations after 50 inverse design cycles. The wing tip region is not shown here.

As a matter of fact, both dimensionality and model limitations may indeed hinder or even prevent inverse design cycles from converging. That is likely to happen in regions where the flow field is strongly 3–D or where there is boundary layer separation.

On the other hand, for pressure driven flows without separation regions, the success of MGM is widely acknowledged in the literature (Bartelheimer, 1995; N. Hirose and Kawai, 1987; Santos, 1993; Silva and Sankar, 1992). Good results are reported even for flow fields that are moderately 3–D. That happens because the method responds primarily to pressure field variations. When those are dominant, then MGM can capture the most relevant part of the flow physics.

As a first account of our studies on the subject, this paper focuses on implementation issues, and the applications are limited to 2–D airfoil design tests. Wing design and other 3–D applications have been left for another paper on the subject.

2.1. Numerical Implementation

Owing to its linearity, eq. (1) could be solved analytically, on making use of Green's function method for instance. However, finite differences solutions are more appropriate for the hybrid formulation, as will be shown below.

Previous works have shown that one can get sufficiently accurate solutions on nonuniform grids (Santos, 1993). Under such conditions, the best results are gotten on approximating the first order derivative by a backward difference operator and the second derivative by a central one (Volpe, 2004). On making use of such differencing scheme, one can easily cast the MGM equation in the form of a tridiagonal linear system (Hirsch, 1994).

$$\mathbf{M}\delta z = \mathbf{R} \tag{4}$$

where $\mathbf{M}$ is the finite difference operator matrix, δz is a vector of point–wise geometry changes (δz_i) and $\mathbf{R}$ represents the RHS of the MGM equation, which can be either $(C_{pti} - C_{pi})$ or $(q_{ti}^2 - q_i^2)$. The above set (4) can certainly be tackled by Thomas algorithm (W. H. Press and Flannery, 1992), and that is precisely what is done in the conventional implementation of MGM.

On the other hand, the hybrid formulation does not seek to solve the set. Instead, it attempts to find δz that minimizes the error with respect to the actual solution, while satisfying a set of constraints. Hence it gives MGM the character of an optimization method. The finite differences form (4) makes it simple for one to approach the problem that way, and the overall framework allows for the imposition of geometric constraints (Santos, 1993). The error function is defined so as to ensure that only the actual solution results in the absence of constraints. On abiding by the usual notation and defining the variable $\mathbf{X} \equiv \delta z$ as an estimate of geometry corrections, the error function is defined as

$$\mathrm{F} = \frac{1}{2}\left|\mathbf{MX} - \mathbf{R}\right|^2 \tag{5}$$

It is a quadratic objective function, for which the gradient and Hessian are both given in closed form, by exact expressions with low computational cost

$$\nabla \mathrm{F}(\mathbf{X}) = \mathbf{M}^T(\mathbf{MX} - \mathbf{R}) \tag{6}$$

$$\mathbf{H}(\mathbf{X}) = \mathbf{M}^T\mathbf{M} \tag{7}$$

The above expressions for F and its gradient can be cast in an alternative form, which suits our purposes best

$$\mathrm{F} = \frac{1}{2}\mathbf{X}^T\mathbf{H}\mathbf{X} - \mathbf{R}^T\mathbf{M}\mathbf{X} - \frac{1}{2}\mathbf{R}^T\mathbf{R} \tag{8}$$

$$\nabla\mathrm{F} = \mathbf{H}\mathbf{X} - \mathbf{M}^T\mathbf{R} \tag{9}$$

The quadratic character of F, along with the fact that both its gradient and Hessian are readily available, seem to favor the use of a second–order minimization method. However, the imposition of constraints brings discontinuities in the Hessian, as will be shown below. Such discontinuities may represent a source of numerical ill–conditioning for second–order methods. Therefore, first–order methods are considered, instead.

In particular, two well–known optimization algorithms make for a very simple and robust implementation of the hybrid: The first–order Preconditioned Conjugate Gradient (PCG) and the Augmented Lagrange Multiplier (ALM)— Volpe, 2004. Hence the combination PCG–ALM is the method of choice to tackle the imposition of geometric constraints.

In essence, the Conjugate Gradient (CG) algorithm takes an initial estimate of the solution $\mathbf{X}$, choses a search direction $\mathbf{S}$ that depends on ∇F, and performs an 1–D search for the extremum in that direction. On finding the extremum along $\mathbf{S}$, it restarts the process from that point on. The iterations go on until a local extremum of F is gotten. In cases where ∇F and $\mathbf{H}$ are known in closed form, one can compute the 1–D step size α so as to reach an extremum along $\mathbf{S}$ in a single step— since it implies that $\nabla\mathrm{F}\perp\mathbf{S}$ (Vanderplaats, 1984).

The number of iterations it takes for the above algorithm to converge is of the order of $\sqrt{\kappa(H)}\ln(\sqrt{2/\varepsilon})$. Where $\kappa(H)$ represents the Hessian spectral condition number and ε is the ratio between initial and final errors. In this application, both the difference operator matrix and the Hessian ultimately depend on grid spacing Δx_i. It can be shown that a severely nonuniform grid can cause κ to reach very high values, thus hampering convergence (Volpe, 2004).

A pre–conditioning algorithm can be used to circumvent the problem. In essence, it performs a similarity transformation on the CG equations. The procedure leads to a space where the Hessian is transformed into a matrix of lower κ, and where the extremum is actually sought. The inverse transformation then takes the results back into the original system (Axelsson and Barker, 1984). The CG equations can be fully integrated with those of the pre–conditioning algorithm to form the PCG algorithm. On defining $\mathbf{g}^q \equiv \nabla\mathrm{F}(\mathbf{X}^q)$, to simplify the notation, one gets

$$\begin{cases} \alpha_q = \frac{-(\mathbf{g}^q)^T.\mathbf{h}^q}{(\mathbf{S}^q)^T\mathbf{H}\mathbf{S}^q} \\ \mathbf{X}^{q+1} = \mathbf{X}^q + \alpha_q\mathbf{S}^{q+1} \\ \mathbf{g}^{q+1} = \mathbf{H}\mathbf{X}^{q+1} - \mathbf{R}\mathbf{M} \end{cases} \qquad \begin{cases} \mathbf{h}^{q+1} = \mathbf{C}^{-1}\mathbf{g}^{q+1} \\ \beta_q = \frac{(\mathbf{g}^{q+1})^T.\mathbf{h}^{q+1}}{(\mathbf{g}^q)^T.\mathbf{h}^q} \\ \mathbf{S}^{q+1} = -\mathbf{h}^{q+1} + \beta_q\mathbf{S}^q \end{cases} \tag{10}$$

The main difference between this and the original CG algorithm lies in the definition of vector $\mathbf{h}$, which is used to compute the search direction $\mathbf{S}$. It is gotten as a product between the gradient $\mathbf{g}$ and the inverse of the preconditioning matrix $\mathbf{C}$. The matrix $\mathbf{C}$, in turn, is a function of the Hessian $\mathbf{H}$ (Axelsson and Barker, 1984). For both algorithms CG and PCG, the first iteration search direction is taken as $\mathbf{S}^1 = -\mathbf{g}^1$.

The geometric constraints of interest are upper and lower bounds for thickness, which are imposed on portions of an airfoil or wing. These are inequality constraints, which are enforced by the aforementioned ALM algorithm. They are cast in the form of functions G_k, which are positive in the range where the constraints are satisfied— for instance $x_k \geq x_{min_k} \Rightarrow G_k = x_k - x_{min_k}$. The constraints are introduced in the original objective function as an external penalty function, which is only nonzero when any one of the constraints is violated (Vanderplaats, 1984).

$$\Phi(\mathbf{X}, r_p) = \mathrm{F}(\mathbf{X}) + r_p \sum_{j=1}^{m} \left\{ \max\left[\frac{\lambda_j}{2r_p} - G_j\ ,\ 0 \right] \right\}^2 \tag{11}$$

The symbol Φ represents the augmented objective function. The summation on the RHS is the external penalty function, the magnitude of which is controlled by the parameter r_p. The λ_j represent Lagrange multipliers. Both r_p and λ_j change in the iterative process, so as to ensure convergence, while keeping the penalty function null within the feasible region, *i.e.* where all constraints are satisfied. Naturally, the PCG algorithm is applied to the augmented function, as opposed to the original one. The corresponding gradient and Hessian are given by

$$\frac{\partial\Phi}{\partial x_k} = \frac{\partial\mathrm{F}}{\partial x_k} - 2r_p \sum_{j=1}^{m} \max\left[\frac{\lambda_j}{2r_p} - G_j\ ,\ 0 \right] \frac{\partial G_j}{\partial x_k} \tag{12}$$

$$H_{jk} = \frac{\partial^2 F}{\partial x_j \partial x_k} + \begin{cases} 2r_p \Rightarrow\ j = k \text{ and } G_k\ <\ \lambda_k/2r_p \\ 0 \quad \text{for } j \neq k \end{cases} \tag{13}$$

Equation (13) shows that $\mathbf{H}(\Phi)$ is given by the sum of $\mathbf{H}(\mathrm{F})$ and a contribution to its main diagonal that depends on the constraints. The last term represents the aforesaid discontinuities.

Convergence is verified on the basis of two complementary criteria: the magnitude of the objective function gradient, and the magnitude of that function variation between consecutive iterations. The primary criterion is obviously the gradient magnitude, which should be the only one effective in the absence of constraints. However, the second criterion is essential in the presence of constraints. For the minimum of Φ can be at the boundary of the feasible region, in which case it may not be an actual extremum ($\nabla\Phi \neq 0$). Then the rationale is to set the criteria up in a hierarchy, so as to ensure the gradient prevails in all but those cases, where the constraints actually hamper convergence. That is accomplished on prescribing the objective function variation an accuracy level that is much more stringent than that of the gradient.

3. The Variable Metric Algorithm

The CG algorithm and its preconditioned counterpart PCG represent an improvement over the Steepest Descent algorithm. Because they carry on information on previous iterations, when computing a new search direction $\mathbf{S}^q$. Whereas the latter only uses the gradient to that end— $\mathbf{S}^q = -\mathbf{g}^q$. In both cases, CG and PCG, that is accomplished through the scalar parameter β, in eq. (10). Variable Metric algorithms (VM) have a similar feature, but instead of conveying that information through a single parameter, they store it in a n dimensional array. Since they keep more information on previous steps, one would expect these methods to be more efficient (Vanderplaats, 1984).

The basic idea is to create an array that approximates the inverse of the Hessian matrix as the optimization progresses. The search direction at iteration q is defined as

$$\mathbf{S}^q = -\tilde{\mathbf{H}}\mathbf{g}^q \tag{14}$$

where $\tilde{\mathbf{H}}$ is a matrix that approaches the inverse of the Hessian during the optimization process for quadratic functions. Owing to this feature, the VM algorithms should have convergence characteristics similar to second–order methods.

Given the search direction $\mathbf{S}^q$, the 1–D search is performed the same way as before, with α_q as given by eq.(10). Furthermore, at the first iteration $\tilde{\mathbf{H}}$ is taken as the identity matrix, so that the first search direction is also along the steepest descent. At subsequent iterations $\tilde{\mathbf{H}}$ is given by

$$\tilde{\mathbf{H}}^{q+1} = \tilde{\mathbf{H}}^q + \mathbf{D}^q \tag{15}$$

where $\mathbf{D}$ is a symmetric update matrix

$$\mathbf{D}^q = \frac{\sigma + \theta\tau}{\sigma^2}\mathbf{p}\mathbf{p}^T + \frac{\theta - 1}{\tau}\tilde{\mathbf{H}}^q\mathbf{y}(\tilde{\mathbf{H}}^q\mathbf{y})^T - \frac{\theta}{\tau}\left[\tilde{\mathbf{H}}^q\mathbf{y}\mathbf{p}^T + \mathbf{p}(\tilde{\mathbf{H}}^q\mathbf{y})^T\right] \tag{16}$$

The change vectors $\mathbf{p}$ and $\mathbf{y}$, in turn, are defined as

$$\begin{cases} \mathbf{p} = \mathbf{X}^q - \mathbf{X}^{q-1} \\ \mathbf{y} = \mathbf{g}^q - \mathbf{g}^{q-1} \end{cases} \tag{17}$$

and the scalar parameters σ, τ and θ are defined as follows

$$\begin{cases} \sigma = \mathbf{p}.\mathbf{y} \\ \tau = \mathbf{y}^T\tilde{\mathbf{H}}\mathbf{y} \\ \theta = 0 \;\Rightarrow\; \text{DFP method} \\ \theta = 1 \;\Rightarrow\; \text{BFGS method} \end{cases} \tag{18}$$

Where the acronyms DFP and BFGS stand for Davidson–Fletcher–Powell and Broydon–Fletcher–Goldfarb–Shanno, respectively. These are two of the most popular variable metric methods. In our tests we have made use of the BFGS method, for it has shown slightly better performance in our applications. Other than for the changes in the computation of the search direction $\mathbf{S}^q$, the algorithm follows the same steps that where outlined for the CG, in sec. 2.1.

An illustrative comparison of these algorithms can be drawn on applying them to the same test function that is often used in the literature: $F_t(x, y) = 10x^4 - 20x^2y + 10y^2 + x^2 - 2x + 5$. Its minimum is the point $(1, 1)$ and no constraints are imposed on the problem. Figure 3 depicts a comparison between the PCG and the VM algorithms. The steepest descent is also included, to provide a basis for comparison. The three algorithms take the origin as the starting point. Apart from the first step, which is in the same direction for all of them, the picture clearly shows that they take very distinct paths to approach the minimum. The steepest descent exhibits its distinctive "zig–zag" pattern, which implies the highest step count among them (64). As for the other two, the VM algorithm has converged with the lowest step count (8), which represents a very significant advantage over the PCG $(26$ steps).

These results seem to favor the VM algorithm as the most promising among them. It must be borne in mind, though, that F_t bears no relation to the actual objective function, not to mention the absence of constraints in the test. Moreover, the steepest descent algorithm preforms so poorly that it is out of the question. As for the other two, VM and PCG, it remains to be seen which one performs better in actual applications.

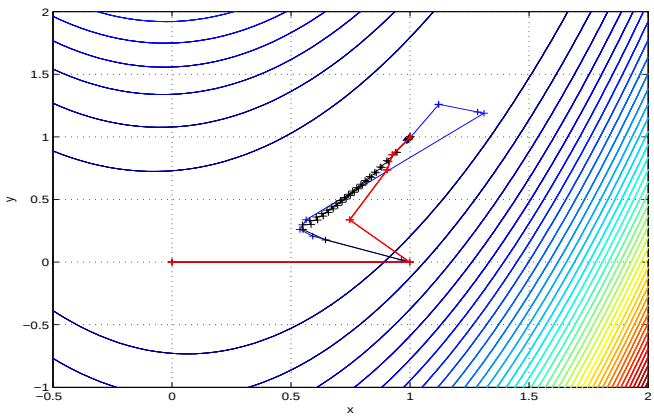

Figure 3: Performance test. Steepest descent, black, 64 it.; PCG, blue, 26 it.; VM, red, 8 it.

4. Airfoil Application Tests

The tests were performed on taking a known airfoil, under fixed flow conditions, as the starting point. The target C_{pt} distribution was prescribed as the C_p distribution of another known airfoil, under the same flow conditions. The procedure enables one to validate the algorithms implementation. Furthermore, the exact same tests were done on using the PCG and the VM algorithms, for the purposes of comparison. Some results are presented and discussed below.

Before proceeding to test results, it is worth making a distinction between the terms *cycle* and *iteration*, regarding the way they are used here. The former denotes a full cycle of the inverse design loop, as is shown in figure 1. Whereas the latter refers to the number of steps it takes for the minimization algorithm to reach the objective function minimum.

The first test was done with inactive constraints. That is, the maximum thickness bound was assigned a value much higher than that of the airfoils, and the minimum bound was assigned zero— so as to ensure the iterations would not cross constraint boundaries. The idea was to compare the PCG and VM performance in the absence of constraints. The results are presented in the figure 4. The picture on the left shows the how the C_p distribution changes in the process. Whereas

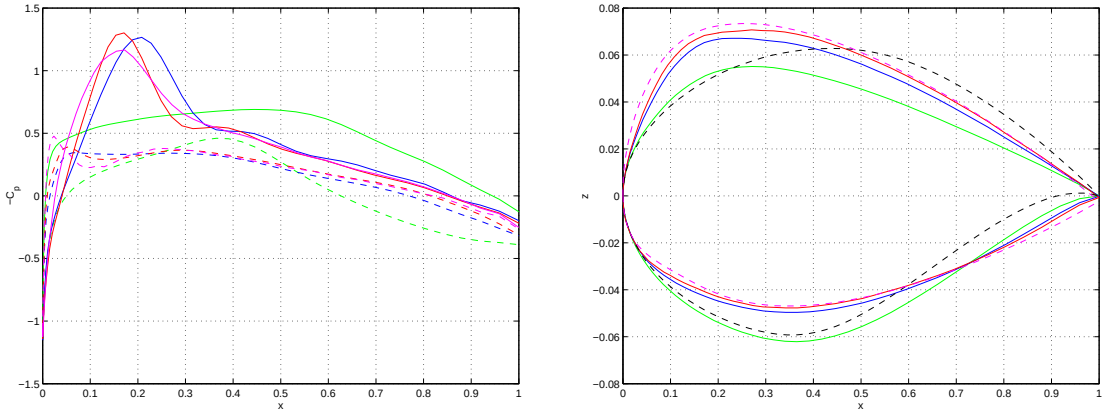

Figure 4: Validation test. From RAE 2822 to NACA 23010. Left: C_p distributions. Solid lines, upper side; dashed lines, lower side. Magenta, RAE 2822; black, NACA 23010; red, 1st. cycle; green 5th cycle; blue, 10th cycle. Right: Airfoil contours, same color code.

the picture on the right depicts the geometry evolution in the inverse design cycles. As expected, both PCG and VM algorithms have converged to the same results, within about 15 inverse design cycles. However the numbers of iterations required by PCG and VM at each cycle are quite different. The evolution of these numbers in the process is presented in fig. 7.a. Two aspects are rather noteworthy in the curves: 1. The upper side has required higher numbers of iterations than the lower side, which is apparently a result of grid differences between them. 2. The PCG clearly outperforms the VM on the upper side, while it shows no significant performance gains on the lower side.

Another test was done with active constraints. This time, the original airfoil did not meet minimum thickness constraints, but the one that corresponds to the target C_{pt} was picked so as to meet all constraints. Results are presented in fig.

5. As before, both PCG and VM have converged to the same result within 15 cycles. On the other hand, fig. 7.b. shows a

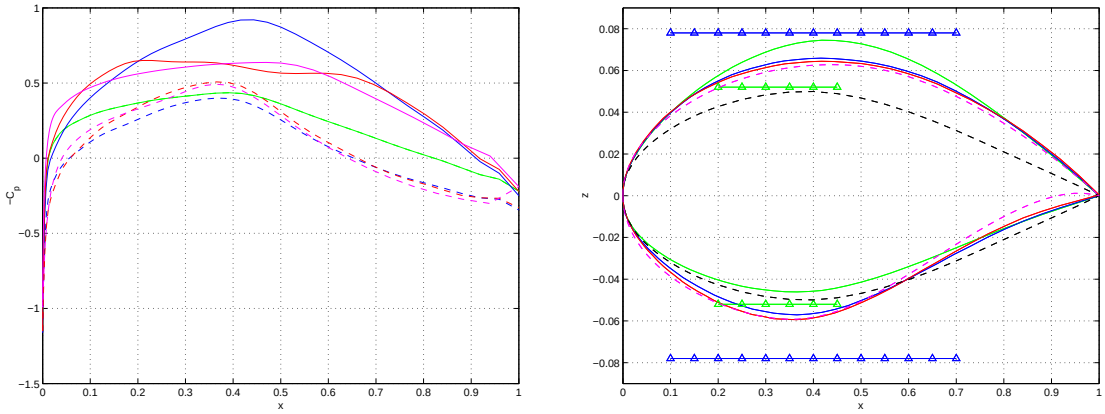

Figure 5: Validation test. From NACA 64012 to RAE 2822. Left: C_p distributions. Solid lines, upper side; dashed lines, lower side. Magenta, NACA 64012; black, RAE 2822; red, 1st. cycle; green 5th cycle; blue, 10th cycle. Right: Airfoil contours, same color code. Maximum thickness constraints blue △; minimum thickness constraints green △.

significant increase in the number of iterations— probably because of the active constraints. Furthermore, the step counts of PCG and VM, for both upper and lower sides, all seem to hover around the same average value.

Finally, a test was performed where both the original and the target geometries were picked so as not to meet minimum thickness constraints. The results that are shown in fig. 6 were gotten within 15 cycles, on using both methods PCG and VM. The algorithms performance is shown in fig. 7.c. Here again, one sees a large rise in the step counts for both of them,

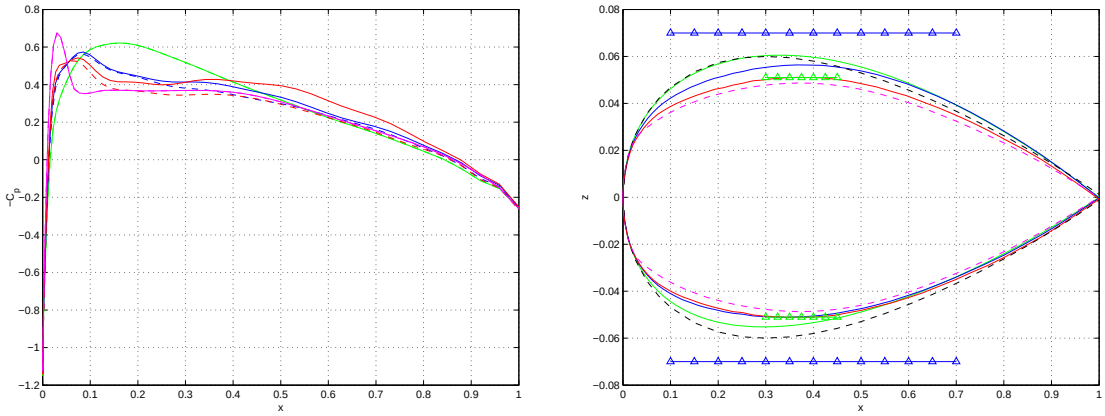

Figure 6: Validation test. From NACA 0012 to ONERA M6. Left: C_p distributions. Solid lines, upper side; dashed lines, lower side. Magenta, NACA 0012; black, ONERA M6; red, 1st. cycle; green 5th cycle; blue, 10th cycle. Right: Airfoil contours, same color code. Maximum thickness constraints blue △; minimum thickness constraints green △.

which is apparently caused by the active constraints. The differences between upper and lower sides seem to be relatively minor. Different from the others, though, in this case the VM outperforms PCG, since its step count is noticeably lower than the latter.

5. Conclusions

The above results are illustrative of our findings. Both PCG and VM have shown to converge to the same solution at each inverse design cycle, which is the bottom line for the applications. In regard to the number of iterations, while it is plain that active constraints raise them by a large amount, there seems to be no clear evidence of the superiority of any one of the methods over the other.

As for computational costs, the greatest difference between them lies in the preconditioning matrix inversion, which only PCG requires. The size of matrix $\mathbf{C}$ is proportional to the grid size. Thus, that operation may indeed represent a major liability for PCG, unless it could outperform VM by converging much faster than it.

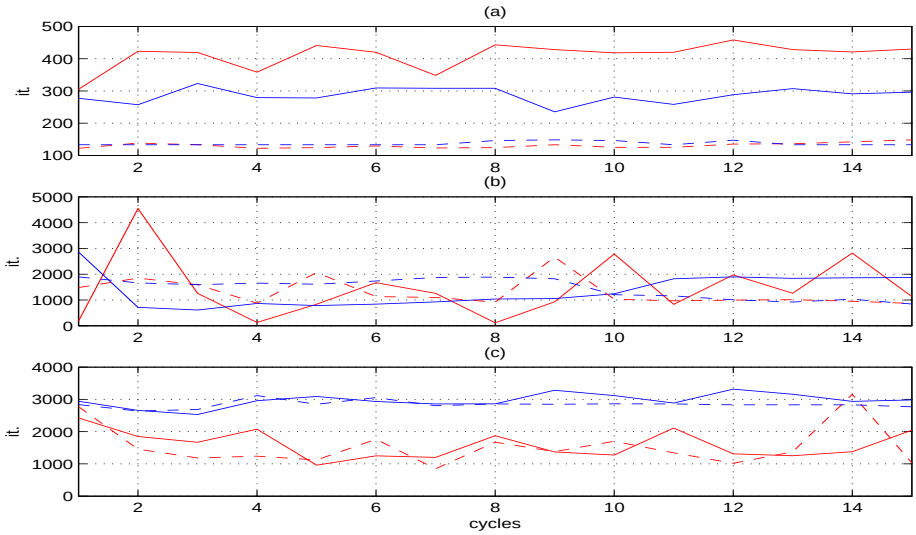

Figure 7: Algorithm iterations per inverse design cycle. Red, VM. Blue, PCG. Solid lines, upper side. Dashed lines, lower side. (a) test shown in fig. 4. (b) test shown in fig. 5. (c) test shown in fig. 6.

Lacking clear evidence of a superior performance in iteration counts, the VM algorithm may still be favored in the inverse design applications. That is because, in principle, it can deal with larger grids at lower computational costs.

6. Acknowledgments

The authors are grateful for the support provided by EMBRAER and FAPESP, under grant no. 2000/13768–4. Moreover, the first author (A. H.) gratefully acknowledges the support provided by FAPESP, under grant no. 04/02451–0.

7. References

Axelsson, O. and Barker, V. A., 1984, “Finite Element Solution of Boundary Problems: Theory and Computation”, Computer Sceince and Applied Mathematics, Academic Press, Inc., Cambridge, MA, 1st edition.

Bartelheimer, W., 1995, An Improved Integral Equation Method for the Design of Transonic Airfoils and Wings, Published by the AIAA, Inc. with permission.

Hirsch, C., 1994, “Numerical Computation of Internal and External Flows”, Vol. I of “Wiley Series in Numerical Methods in Engineering”, John Wiley & Sons, NY, 1st edition, Fundamentals of Numerical Discretization.

Jameson, A., 1997, Re–Engineering the Design Process through Computation, “35th Aerospace Sciences Meeting & Exhibit”, Reno, NV. American Institute of Aeronautics and Astronautics, AIAA, AIAA-97-0641.

MacCormack, R., 1993, A perspective on a Quarter Century of CFD Research, “11th AIAA Computational Fluid Dynamics Conference”, Orlando, FL. American Institute of Aeronautics and Astronautics, AIAA, AIAA-93-3291.

N. Hirose, S. T. and Kawai, N., 1987, Transonic Airfoil Design Procedure Utilizing a Navier-Stokes Analysis Code, “AIAA Journal”, Vol. 25, No. 3, pp. 353–359.

Santos, L. C. C., 1993, “A Hybrid Inverse Optimization Method for Aerodynamic Design of Lifting Surfaces”, PhD thesis, Georgia Institute of Technology.

Santos, L. C. C., 1998, Modelling Auxiliary Geometric Equations for Inverse Design Metods, Published by the AIAA, Inc. with permission, AIAA98-2405.

Santos, L. C. C., 1999, Convergence Acceleration Strategies for an Inverse Design Method, Published by the AIAA, Inc. with permission, AIAA99-0184.

Silva, D. H. and Sankar, L. N., 1992, An Inverse Method for the Design of Transonic Wings, “1992 Aerospace Design Conference”, number 92–1025 in proceedings, pp. 1–11, Irvine, CA. AIAA.

Vanderplaats, G. N., 1984, “Numerical Optimization Techniques for Engineering Design: With Applications”, Series in Mechanical Engineering, McGraw-Hill, N.Y., 1st edition.

Volpe, E. V., 2004, A11 – Inverse Aerodynamic Design Module, Technical Report, EMBRAER – FAPESP – EPUSP, SP, Research Project: Advanced Applications of Computational Fluid Dynamics to High Performance Aircraft.

W. H. Press, S. A. Teukolsky, W. T. V. and Flannery, B. P., 1992, “Numerical Recipes in C: The Art of Scientific Computing”, Cambridge University Press, New York, NY, 2nd edition.