Third International Conference on Inverse Design Concepts and Optimization in Engineering Sciences (ICIDES-III), Editor: G.S. Dulikravich, Washington D.C., October 23-25, 1991.

A COMPARISON OF TWO CLOSELY-RELATED APPROACHES TO AERODYNAMIC DESIGN OPTIMIZATION

G. R. Shubin and P. D. Frank

Boeing Computer Services

P.O. Box 24346, Mail Stop 7L-21, Seattle, WA. 98124-0346

1. ABSTRACT

The objective of this paper is to compare two closely-related methods for aerodynamic design optimization. The methods, called the "implicit gradient" approach and the "variational" (or "optimal control") approach, both attempt to obtain gradients necessary for numerical optimization at a cost significantly less than that of the usual black-box approach that employs finite difference gradients. While the two methods are seemingly quite different, they are shown to differ (essentially) in that the order of discretizing the continuous problem, and of applying calculus, is interchanged. Under certain circumstances, the two methods turn out to be identical. We explore the relationship between these methods by applying them to a model problem for duct flow that has many features in common with transonic flow over an airfoil. We find that the gradients computed by the variational method can sometimes be sufficiently inaccurate to cause the optimization to fail.

2. INTRODUCTION

We first define what we mean by "analysis" and "design" in the context of computational aerodynamics. In the "analysis problem" we seek to determine the aerodynamic flow, given a description of the geometry of an airfoil or aircraft. In the "design problem" we seek to do the inverse; given the flow, find the geometry that will produce it. Here, we are concerned with methods for solving the design problem that are based on coupling solutions of the discretized analysis problem with numerical optimization procedures.

In a previous paper [4] we compared three optimization-based approaches for solving computational aerodynamics design problems. (Actually, the methods apply to many computational physics design optimization problems.) The optimization methods are (i) the common "black-box" method with finite difference gradients, (ii) a modification where the gradients are found by an algorithm based on the implicit function theorem (hereafter called the implicit gradient approach), and (iii) an "all-at-once" method where the flow and design variables are simultaneously altered. We also showed that the implicit gradient approach was very closely related to a particular "variational" or "optimal control" approach to design optimization that has recently attracted interest (e.g., [5]). The purpose of the present paper is to further explore this relationship. (We note that the close relationship between nonlinear optimization and optimal control has apparently been known for some time[2][6]. However, this relationship appears to be little-known among practitioners in applications disciplines utilizing these mathematical techniques.)

The finite difference approach to obtaining gradients is conceptually the simplest, but it is ordinarily prohibitively expensive for practical problems, since it requires at least one solution of an analysis problem for each design parameter. Both the implicit gradient approach and the variational approach have the objective of determining gradients needed in an optimization

Third International Conference on Inverse Design Concepts and Optimization in Engineering Sciences (ICIDES-III), Editor: G.S. Dulikravich, Washington D.C., October 23-25, 1991.

procedure at a significantly reduced cost. Both approaches involve using "calculus-like" operations to derive the formulas employed in finding the gradients. As explained later, the procedures differ in that the order of applying calculus, and of discretizing the continuous problem, are interchanged. Because the implicit gradient approach applies to an already discretized analysis problem, it can be used to "retrofit" many analysis codes to produce inexpensive gradients for design optimization; see [7] for details.

3. MODEL PROBLEM

3.1 Continuous Analysis Problem

In [4] we showed how the steady flow of an inviscid fluid in a duct of variable cross-sectional area $A(\xi)$, governed by the Euler equations, can (under certain circumstances) be reduced to the single nonlinear ordinary differential equation

$$f_\xi + g = 0 \tag{1}$$

where

$$f(u) \equiv u + \bar{H}/u, \qquad g(u,\xi) \equiv \frac{A_\xi}{A}(\bar{\gamma}u - \bar{H}/u),$$

$u(\xi)$ is the fluid velocity, ξ is distance along the duct, and $\bar{\gamma}$ and $\bar{H}$ are given constants. Here, the subscript ξ means differentiation with respect to ξ. While a much more careful specification was given in [4], roughly speaking the continuous analysis problem is to find u, given a differentiable area function $A(\xi)$ and the specified boundary values $u(\xi = 0)$ and $u(\xi = 1)$. These boundary values are chosen so that the (weak) solution of (1) contains a shock.

3.2 Discrete Analysis Problem

Let the ξ-coordinate be discretized by a uniform, cell-centered grid with centers at $\xi_j = (j - 1/2)h$, $\Delta\xi = 1/J$, where J is the number of unknown grid values. Let U_j represent a piecewise constant approximation to u on each grid cell. Then, a conservative difference scheme for (1) is given by

$$W_j \equiv \frac{f_{j+1/2} - f_{j-1/2}}{\Delta\xi} + g_j = 0. \tag{2}$$

Here the source term $g_j = g(U_j, (A_\xi/A)_j)$ and we assume that the duct shape $A(\xi)$ is given by a piecewise cubic spline described in the B-spline basis with coefficients D_m for $m = 1, 2, \ldots, M$ and that $A(0)$ and $A(1)$ are fixed. $(A_\xi/A)_j$ is obtained by evaluating the spline and its derivative at ξ_j. The boundary conditions on U are $U_0 = u(\xi = 0)$ and $U_{J+1} = u(\xi = 1)$. The fluxes $f_{j+1/2}$, as functions of U_j and U_{j+1}, are chosen to correspond to the Godunov, Engquist-Osher, or Artificial Viscosity methods for numerically approximating hyperbolic conservation laws [4].

Once the discretization has been made, we are faced with solving a system of nonlinear algebraic equations. The system is

Given: D_m, $m = 1, \ldots, M$ (spline coefficients describing $A(\xi)$).
Find: U_j satisfying

$$W(U) = 0. \tag{3}$$

Third International Conference on Inverse Design Concepts and Optimization in Engineering Sciences (ICIDES-III), Editor: G.S. Dulikravich, Washington D.C., October 23-25, 1991.

Here W is the vector of discretized equations (2) for $j = 1, 2, \ldots, J$ and the boundary conditions on U.

3.3 Continuous Design Problem

We want to formulate the design problem as a minimization problem. It is:

Given: a desired (or goal) velocity $\hat{u}(\xi)$.
Let: $h(u) = \frac{1}{2}(u(\xi) - \hat{u}(\xi))^2, \mathtt{f}(u) = \int_0^1 h(u)d\xi$.
Find: $A(\xi)$ such that $u(\xi)$ satisfies (1) and $\mathtt{f}(u)$ is minimized.

3.4 Discrete Design Problem

We assume that a desired (or goal) velocity distribution $\hat{U}_j$ is given for each computational cell in the analysis problem. Then we have

Given: $\hat{U}_j, \quad j = 1, \ldots, J$.
Let: $H_j = \frac{1}{2}(U_j - \hat{U}_j)^2, F(U) = \sum_{j=1}^{J} H_j$.
Find: $D_m, \quad m = 1, 2, \ldots, M$ (spline coefficients describing $A(\xi)$) such that (3) is satisfied and $F(U)$ is minimized.

4. COMPARISON OF THE IMPLICIT GRADIENT APPROACH AND THE VARIATIONAL APPROACH

In this section, we compare two closely-related, optimization-based approaches to finding an approximate solution to the "Continuous Design Problem" posed above. In each case, function values needed in the optimization are obtained by solving a discrete analysis problem and evaluating a discrete form of the objective function (and constraints). The key question is how gradients needed in the optimization are computed:

1. **Implicit gradient approach.** Discretize the problem first to obtain the "Discrete Design Problem," then find a formula for the gradients by using the implicit function theorem.
2. **Variational (or control theory) approach.** Find a formula for the "gradients" for the continuous problem (i.e., in infinite dimensional space). This formula involves the solution of the analysis problem, and the solution of another differential equation called the adjoint problem. Discretize both the forward and adjoint problems, then evaluate the formula to get the gradient.

After the gradients are obtained, the function values and gradients are used in an optimization procedure to improve the current estimate of the design variables. As can be seen, these approaches differ, essentially, in that the order of discretizing, and of doing calculus-like operations, is interchanged.

4.1 Implicit Gradient Approach

The implicit gradient approach is a natural extension of the usual black-box method wherein gradients needed in the optimization are obtained by finite differences. We thus first introduce the black box method. We do so in a somewhat general setting, then specialize to the model problem.

We assume that the design problem has already been discretized. Let n_U and n_D be the number of flow variables U and design variables D, respectively. (In the duct flow model

Third International Conference on Inverse Design Concepts and Optimization in Engineering Sciences (ICIDES-III), Editor: G.S. Dulikravich, Washington D.C., October 23-25, 1991.

problem, the flow variables are the velocities, and the design variables are the spline coefficients describing the geometry.) Then we seek to solve

$$\begin{aligned} &\text{minimize } F(D) \ , \\ &D \in \mathbf{R}^{n_D} \\ &\text{subject to } C(D) \geq 0 \ , \end{aligned} \tag{4}$$

where $F(D)$ is the objective function and $C(D)$ is a vector of m_D constraint functions. In the black-box method, each evaluation of $F(D)$ requires a solution by the analysis code.

For simplicity, the unconstrained version of (4) is considered below. However, the results apply to the constrained problem as well.

As in our model problem, the function F will often be formulated in terms of the flow variables U. In this situation, F is dependent on the design variables D in an indirect manner. That is, the flow variables U are linked to the design variables D via the discretization of the differential equations, since the flow variables will change when the geometry is altered. In the general case, F will have both a direct dependence on D and an indirect dependence on D, due to the dependence of U on D. Thus, one could consider the objective function to be $F(U(D), D)$. The term $U(D)$ indicates that, given D, the value of U is obtained by solving an analysis problem.

Assume that the analysis problem has been discretized (as in Section 3.2) so that an analysis consists of solving a system of nonlinear equations. In this case function evaluations for the black-box method are computed as follows. Given a design specified by D, the analysis code solves $W(U) = 0$, where U is the vector of n_U flow variables and W is a vector of n_U nonlinear equations. Since the analysis problem is an implicit function of D it can be viewed as solving

$$W(U, D) = 0 \tag{5}$$

for U, given a design specified by D. When gradients are obtained by finite differences, each component of D is successively perturbed, and (5) is re-solved to get a perturbed value of U.

We now review how gradients can be obtained without recourse to finite differences. Suppose that U and D are considered as subsets of the $n_U + n_D$ vector X given by

$$X \equiv (\quad U \quad | \ D) ; \tag{6}$$

the Jacobian (first-derivative) matrix of (5) is then

$$J = \left[\begin{array}{c|c} & \\ \quad J_U \quad & J_D \\ & \end{array} \right] , \tag{7}$$

where J is $n_U \times (n_U + n_D)$, J_U is the $n_U \times n_U$ Jacobian with respect to the flow variables and J_D is the $n_U \times n_D$ Jacobian with respect to the design variables. (The partitioned view of the Jacobian implies $n_U \gg n_D$; this will usually be the case.) Note that J_U is sometimes available in analysis codes, especially those based on Newton's method and variants. J_D may, or may not, be easily obtainable. (The availability of J_U and J_D in computational aerodynamics codes is discussed in [7].)

Consider the function $\tilde{F}(U, D)$, where $\tilde{F}$ is the same as the black-box method objective function F, except that U and D are considered to be *independent* of each other. The function $\tilde{F}(U, D)$ is then equivalent to the black-box method objective function $F(U(D), D)$ only when

Third International Conference on Inverse Design Concepts and Optimization in Engineering Sciences (ICIDES-III), Editor: G.S. Dulikravich, Washington D.C., October 23-25, 1991.

(5) is satisfied. The gradients $\nabla_D \tilde{F}(X)$ and $\nabla_U \tilde{F}(X)$ are ordinarily "easy" to obtain because of the assumed independence.

However, the optimization code requires $\nabla_D F$, the gradient of the black-box objective function F with respect to the design variables D. As shown in [4], this gradient is given by

$$\nabla_D F(X) = \nabla_D \tilde{F}(X) - J_D^T J_U^{-T} \nabla_U \tilde{F}(X) . \tag{8}$$

Here, superscript T indicates transpose. The derivation of (8) assumes that we are at a solution of (5).

The following algorithm could be used for computing $\nabla_D F$ using (8):

i. Compute $\nabla_U \tilde{F}$ and $\nabla_D \tilde{F}$
ii. Solve $J_U^T \Lambda = -\nabla_U \tilde{F}$ for Λ
iii. Compute $\nabla_D F = \nabla_D \tilde{F} + J_D^T \Lambda$.

Note that the minus sign is associated with the second step of the algorithm to facilitate comparison with the variational approach later. Note also that, if it is difficult to solve linear systems with the matrix J_U^T, the linear algebra in (8) can be rearranged as $(J_U^{-1} J_D)^T \nabla_U \tilde{F}(X)$, requiring n_D solves with J_U. Observe that $J_U^{-1} J_D$ is the matrix of "sensitivities" of the solution U with respect to the design variables D.

We now apply this algorithm to the model problem and give a complete specification of one evaluation of a gradient during the optimization.

Implicit gradient algorithm for model problem:

1. Given the current estimate of the design variables D_m, solve the discrete analysis problem (3).
2. Compute $\nabla_U \tilde{F} = U - \hat{U}$ and $\nabla_D \tilde{F} = 0$.
3. Given the Jacobian J_U of the discretized flow equations with respect to the flow variables U, evaluated at the solution, solve $J_U^T \Lambda = -(U - \hat{U})$ for Λ
4. Given the Jacobian J_D of the discretized flow equations with respect to the design variables D, evaluated at the solution, compute the gradient $\nabla_D F = J_D^T \Lambda$

4.2 Variational Approach

In the variational approach, we deal first with the "Continuous Design Problem," and use calculus to derive an infinite dimensional "gradient." We then discretize the problem. Since it is somewhat cumbersome to present the methodology for a general case, we specialize to the model problem immediately.

For technical reasons that will become apparent later, it is desirable to augment the governing differential equation (1) with an artificial viscosity term $\epsilon u_{\xi\xi}$, giving

$$w(u, d) = -\epsilon u_{\xi\xi} + f_\xi + g(u, d) = 0. \tag{9}$$

Here, $d(\xi)$ is a function that controls A_ξ / A.

Recalling that $h(u) = \frac{1}{2}(u(\xi) - \hat{u}(\xi))^2$, the Lagrangian is

$$L = \int_0^1 h(u) d\xi + \int_0^1 \lambda(\xi) w(u, d) d\xi,$$

Third International Conference on Inverse Design Concepts and Optimization in Engineering Sciences (ICIDES-III), Editor: G.S. Dulikravich, Washington D.C., October 23-25, 1991.

and $\lambda(\xi)$ is an adjoint function that is the continuous analogue of Lagrange multipliers. Applying the calculus of variations, and doing the usual integration-by-parts, we find that the variation of the Lagrangian is

$$\delta L = [-\epsilon(\lambda \delta u_\xi - \lambda_\xi \delta u) + \delta(\lambda f)]_0^1$$

$$+ \int_0^1 (-\epsilon \lambda_{\xi\xi} - f_u \lambda_\xi + g_u \lambda + h_u)\delta u d\xi + \int_0^1 \lambda w_d (\delta d) d\xi.$$

(Note $w_d = g_d$.) The second term can be made to vanish by requiring that the adjoint equation

$$-\epsilon \lambda_{\xi\xi} - f_u \lambda_\xi + g_u \lambda = -h_u \tag{10}$$

be satisfied. In (10), f_u, g_u, and h_u are given functions of ξ, since they are evaluated at $u(\xi)$, the solution of (9). The integrated term $[\ ]_0^1$ vanishes since $\delta u(0) = \delta u(1) = 0$ and we choose $\lambda(0) = \lambda(1) = 0$ as the boundary conditions on the adjoint λ. Then, the "gradient" of the continuous design problem with respect to changes in the controlling function d is expressed by the variational formula

$$\delta \mathtt{f} = \int_0^1 \lambda w_d (\delta d) d\xi. \tag{11}$$

We now need to discretize (9), (10), and (11). We assume that (9) is discretized by one of the methods described in Section 3.2. Thus, the discretization of the analysis problem is assumed here to be the same as for the implicit gradient approach. (In general, of course, this need not be so.) While those discretizations (the G-, EO, and AV-schemes) are designed to solve the inviscid ($\epsilon = 0$) equation, they in fact all incorporate some kind of artificial viscous effects, either by upwinding (G and EO) or by explicit artificial viscosity (AV). That is why we added the viscous term in (9): so it would appear in equation (10), and thus guide us to reasonable discretizations of the adjoint equation.

Let the computational grid be as described in Section 3.2, and Λ_j be the approximation to λ on the grid. Noting that $h_u = u - \hat{u}$, let us take the discretization of the (10) to be given by

$$B\Lambda = -(U - \hat{U}),$$

where the difference operator B remains to be specified. Note that this equation is linear in Λ since (10) is linear in λ.

Finally, to discretize (11) we could use any reasonable quadrature formula. However, we choose to use the rectangle rule, which gives for the k-th component of the gradient

$$(\nabla_D F)_k = \sum_{j=1}^{J} (W_j)_{D_k} \Lambda_j.$$

Here, $(W_j)_{D_k}$ is the derivative of the $j-$th discrete flow equation with respect to the $k-$th design variable. In matrix notation, this is none other than

$$\nabla_D F = J_D^T \Lambda,$$

so we have again deliberately chosen the discretization to agree with Step 4 of the implicit gradient algorithm.

Third International Conference on Inverse Design Concepts and Optimization in Engineering Sciences (ICIDES-III), Editor: G.S. Dulikravich, Washington D.C., October 23-25, 1991.

Gathering these pieces together, a complete specification of one evaluation of a gradient in an optimization procedure is given below.

Variational algorithm for model problem:

1. Given the current estimate of the design variables D_m, solve the discrete analysis problem (3)
2. Compute $\nabla_U \tilde{F} = U - \hat{U}$ and $\nabla_D \tilde{F} = 0$.
3. Solve the discrete adjoint equation $B\Lambda = -(U - \hat{U})$ for Λ
4. Given the Jacobian J_D of the discretized flow equations with respect to the design variables D, evaluated at the solution, compute the gradient $\nabla_D F = J_D^T \Lambda$

As we have constructed this algorithm, it differs from the implicit gradient algorithm only in step 3. The two procedures are *identical* if we choose $B = J_U^T$, the transpose of the Jacobian of the analysis problem, evaluated at a solution of the analysis problem. Looked at another way, a particular choice of a discretization of the analysis problem, and the associated Jacobian J_U, suggests a specific choice of the discretization B of the adjoint problem, namely $B = J_U^T$. Pursuing this idea, let $(J_U)_G$,$(J_U)_{EO}$, and $(J_U)_{AV}$ denote the Jacobians associated with the G, EO, and AV schemes for the analysis problem, respectively. Then three possible discretizations of the adjoint are given by $B = (J_U)_G^T$,$B = (J_U)_{EO}^T$, and $B = (J_U)_{AV}^T$. We note that two of these, $(J_U)_G^T$ and $(J_U)_{EO}^T$, do not correspond to obvious discretizations of the adjoint equation (10). This is largely due to the careful treatment of "sonic points" (points where $f_u = 0$) and shocks in the G and EO schemes.

Let us call the discretizations of the forward and adjoint problems *incompatible* if $B \neq J_U^T$. This means that the discrete analysis problem and the discrete adjoint problem are *not discretely adjoint*. It is precisely the effect of such incompatibility that we want to test. Thus, to carry out such tests we may solve the forward problem with (say) the G-scheme, but choose the adjoint discretization to be $B = (J_U)_{EO}^T$. Such comparisons will be pursued in the Numerical Results section, below. There, we will use the notation $[G, (J_U)_{EO}^T]$ to refer to such a combination.

We may also look at (10) directly and ask "what is a good way to discretize this differential equation?" It turns out that, for our model and test cases, f_u changes sign once, and $g_u > 0$. For small ϵ, (10) is thus a singular perturbation, two-point boundary value problem with a turning-point. A good numerical method for such problems is the El-Mistakawy–Werle scheme; a complete specification of this scheme, and an analysis which applies directly to the cases tested below, is given in [1]. That analysis shows that, for our test cases, the adjoint function λ is "smooth" in the interior of the domain and has boundary layers at both ends. We will refer to this scheme for solving (10) as the EMW scheme. (In the results presented later, we took $\epsilon = 10^{-5}$ and used linear interpolation to move between the "point-centered" grid natural to the EMW scheme and the "cell-centered" grid used in the analysis solvers.)

4.3 What is the "correct" gradient?

When we use the variational formulation described above, and we choose B to be anything other than J_U^T, we will obtain a gradient different from the one obtained by the implicit gradient approach. This raises the issue of which gradient is "correct." There are two different philosophical points of view. The first holds that, since we are really computing an approximation to the continuous design problem, both gradients represent different approximations to the "continuous gradient," and hence neither is correct. The second holds that, irrespective of the continuous problem, our goal in computation is to solve the discrete design problem. We are more inclined to adopt the second point of view. Thus, we feel that (modulo finite precision arithmetic) the

implicit gradient is the correct one, and that the variational formulation only yields the correct gradient when the particular discretization of the adjoint represented by $B = J_U^T$ is chosen.

5. NUMERICAL RESULTS

In this section we present numerical results obtained by solving the discrete design problem for duct flow described in Section 3, utilizing gradients computed by the implicit gradient and variational methods of Section 4. As constructed in Section 4, these methods differ only in step three of the algorithms, and they are identical if in step three of the variational algorithm we choose $B = J_U^T$, the transpose of the Jacobian of the discrete analysis problem with respect to the flow variables. The specific algorithm used below is thus specified by the choice of B. We will first outline the optimization methods and test cases used. Then we will report on some tests using controlled amounts of gradient error, and compare the implicit gradient and variational methods.

5.1 Optimization Methods

The basic optimization code used was NPSOL version 2.0, a product of the Systems Optimization Laboratory, Stanford University. NPSOL is an implementation of the Sequential Quadratic Programming (SQP) method. NPSOL 2.0 computes a secant approximation to the Hessian (2nd derivative) matrix and the user supplies first derivatives. Results obtained with an optimization method similar to steepest descent are not reported here, but may be found in [8].

5.2 Test Cases

For our tests, the design variables (called D in Section 4) were the B-spline coefficients describing the duct geometry $A(\xi)$. The two end values of A were fixed at $A(0) = 1.050$ and $A(1) = 1.745$. Velocities along the duct were the flow variables (called U in Section 4) for the duct design problem. We took $J = 40$ grid cells, so there were $n_U = 40$ flow variables; this gives resolution about equal to what might be expected in practical computations. The boundary conditions were $U_0 = 1.299$ and $U_{41} = 0.506$. In the optimization runs Newton's method was used to solve the analysis problem (3), and the analyses were "warm started." That is, the initial guesses for the flow velocities were taken to be the solutions from the preceding analysis. The initial velocity profile for the first analysis in an optimization run was a linear profile connecting the boundary conditions. The goal velocities $\hat{U}_j$ were the evaluations on the computational grid of the analytic solution for a goal duct shape with a cross-sectional area given by a sinusoidal perturbation of the linear duct. These area and velocity profiles are the curves marked (X) in Figure 1. No constraints were imposed in these tests. Without constraining the geometry, it is possible for the optimizer to generate designs that cannot be analyzed (the analysis problem has no solution). In this case, we assign a large function value and return to the optimizer. The optimizations were allowed a maximum of 70 major iterations, which is considerably more than would be tolerable in practical use. (This corresponds, *very* roughly, to a maximum amount of work equivalent to 1000 linear system solutions with the Jacobian of the analysis problem.)

The majority of the tests were conducted with $n_D = 2$ design variables. For these tests, three initial guesses for the design variables were selected. These three guesses yield solutions of the analysis problem shown in Figure 1. A contour plot showing the dependence of the objective function on the design variables is displayed in Figure 2. (This plot is for the AV-scheme; the plots for the other schemes are similar.) Also shown are the locations of the optimum and of the three initial guesses of D. The contour plot shows a narrow valley with steep sides and a

relatively flat bottom. Descending the steep sides corresponds (roughly) to getting the shock in the correct location; this has the largest impact on reducing the objective function and is relatively easy for the optimizer. Moving along the valley bottom corresponds to getting the other details of the velocity profile right. This is much harder to do. Thus guess 1 corresponds to a relatively difficult problem, while guesses 2 and 3 correspond to problems that are somewhat easier.

5.3 Controlled gradient error tests.

Since we cannot directly control the gradient errors that are obtained when using incompatible discretizations of the forward and adjoint problems, we first conducted some controlled gradient error tests. In these tests we first obtained the correct gradient using the implicit gradient method, and then added controlled amounts of random error to the gradient. The quantitative results are given in [8]. We were surprised to find that the optimizations began to fail at fairly small amounts (a few percent) of gradient error.

The trust region methods for step size determination in optimization used in [3] apparently worked with a much higher level of relative error in the gradients. However, we found that trust region methods were not much better then line search methods (like in NPSOL) when applied to our model, which is apparently a "harder" problem than many standard optimization test cases.

5.4 Tests comparing the implicit gradient and variational approaches

We now proceed to compare results obtained with the implicit gradient and variational approaches.

The optimizations were run with the twelve combinations of analysis and adjoint solvers shown in Table 1. The discretizations of the analysis problem indicated by G, EO, and AV correspond to the Godunov, Engquist-Osher, and Artificial Viscosity schemes (described in Section 3.2), respectively. The discretizations of the adjoint problem are as described at the end of Section 4.2. Here, the notation $B = (J_U)_G^T$ means, for example, that the discretization of the adjoint differential equation in step 3 of the Variational Algorithm is given by the transpose of the Jacobian of the analysis problem when the Godunov scheme is used. The particular combinations $[G, (J_U)_G^T]$, $[EO, (J_U)_{EO}^T]$, and $[AV, (J_U)_{AV}^T]$ mean that the forward and adjoint solvers are discretely adjoint, and thus that the implicit gradient method is being used. In all other cases, the analysis and adjoint solvers are incompatible (not discretely adjoint).

The qualitative results of Table 1 show that the only reliable combinations of forward and adjoint solvers are those corresponding to the implicit gradient method. There does not seem to be any other discernible pattern in the results. An examination of more quantitative data, like final value of the objective function and specific amounts of work used, also yield little additional useful information. An examination of the gradients obtained by the variational method (not discretely adjoint) shows that the relative error compared to the correct (implicit) gradient is often more than a few percent, and that the gradients are in error both in direction and magnitude [8].

We carried out many of the same tests with an optimizer more like steepest descent, and also with the objective function "smoothed" by a method suggested by Jameson [5]. Such smoothing should reduce the impact of getting the shock location correct on the objective function. (It broadens the valley of Figure 2.) The necessary modifications to the variational approach are described in [8]. Again, we were unable to discern any pattern in the results: sometimes the modifications helped, sometimes they hurt.

Third International Conference on Inverse Design Concepts and Optimization in Engineering Sciences (ICIDES-III), Editor: G.S. Dulikravich, Washington D.C., October 23-25, 1991.

Additional test were carried out with $n_D = 10$ design variables, and the same conclusion was reached: the only reliable combinations of forward and adjoint solvers correspond to the implicit gradient method. That is, the forward and adjoint solvers should be discretely adjoint.

6. CONCLUSIONS

We have shown that two seemingly quite dissimilar approaches to design optimization can, under certain circumstances, be very closely related or even identical. The two approaches, the implicit gradient method and the variational method, both result in gradient calculations that are significantly cheaper than generating gradients by finite differences. The methods differ from each other (essentially) in that the order of discretizing the continuous problem, and of applying calculus, is interchanged. In the implicit gradient approach, the continuous problem is discretized first, and a formula for gradients needed in the optimization is derived by applying the implicit function theorem. In the variational method, calculus is applied first, and one then needs to solve two differential equation problems: the analysis (or forward) problem, and the adjoint problem. If the analysis problem is discretized the same way as for the implicit gradient approach, and if the adjoint is discretized by a method that corresponds to the transpose of the Jacobian of the forward discretization, then the methods are (modulo some details) the same. If the adjoint discretization is taken to be anything else, then the two methods generate different gradients and the variational method gradients are "in error." In our tests using a model for transonic duct flow, the gradient errors were generally small, but were nevertheless sufficient to cause the optimizations to slow down significantly or to fail altogether. For our model problem and optimization method, the only reliable combination of forward and adjoint discretizations is the one corresponding to the implicit gradient method.

7. REFERENCES

[1] Berger, A.E., Han, H., and Kellogg, R.B. A priori estimates and analysis of a numerical method for a turning point problem. *Mathematics of Computation*, 42:465–492, 1984.

[2] Canon, M.D., Cullum Jr., C.D., and Polak, E. *Theory of Optimal Control and Mathematical Programming*. McGraw Hill, 1970.

[3] Carter, R.G. Numerical experience with a class of algorithms for nonlinear optimization using inexact function and gradient information. Technical Report 89-46, ICASE, June, 1989.

[4] Frank, P.D. and Shubin, G.R. A comparison of optimization-based approaches for a model computational aerodynamics design problem. *Journal of Computational Physics*, to appear.

[5] Jameson, A. Aerodynamic design via control theory. Technical Report 88-64, ICASE, November, 1988.

[6] E. Polak. *Computational Methods in Optimization.* Academic Press, 1971.

[7] Shubin, G.R. Obtaining "cheap" optimization gradients from computational aerodynamics codes. Technical Report AMS-TR-164, Boeing Computer Services, June, 1991.

[8] Shubin, G.R. and Frank, P.D. A comparison of the implicit gradient approach and the variational approach to aerodynamic design optimization. Technical Report AMS-TR-163, Boeing Computer Services, April, 1991.

Third International Conference on Inverse Design Concepts and Optimization in Engineering Sciences
(ICIDES-III), Editor: G.S. Dulikravich, Washington D.C., October 23-25, 1991.

8. FIGURES AND TABLES

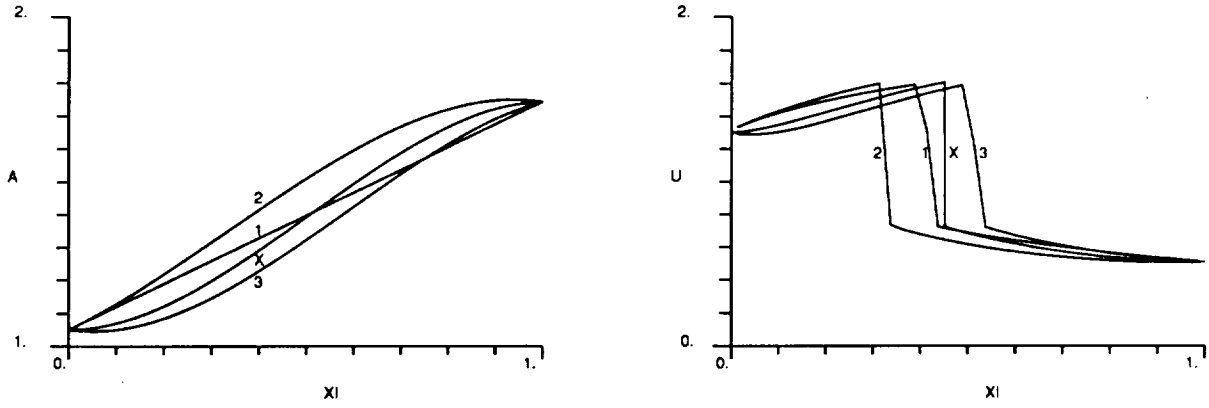

Figure 1: Area function A and corresponding velocity function U for Guesses 1, 2, 3, and optimal solution (X).

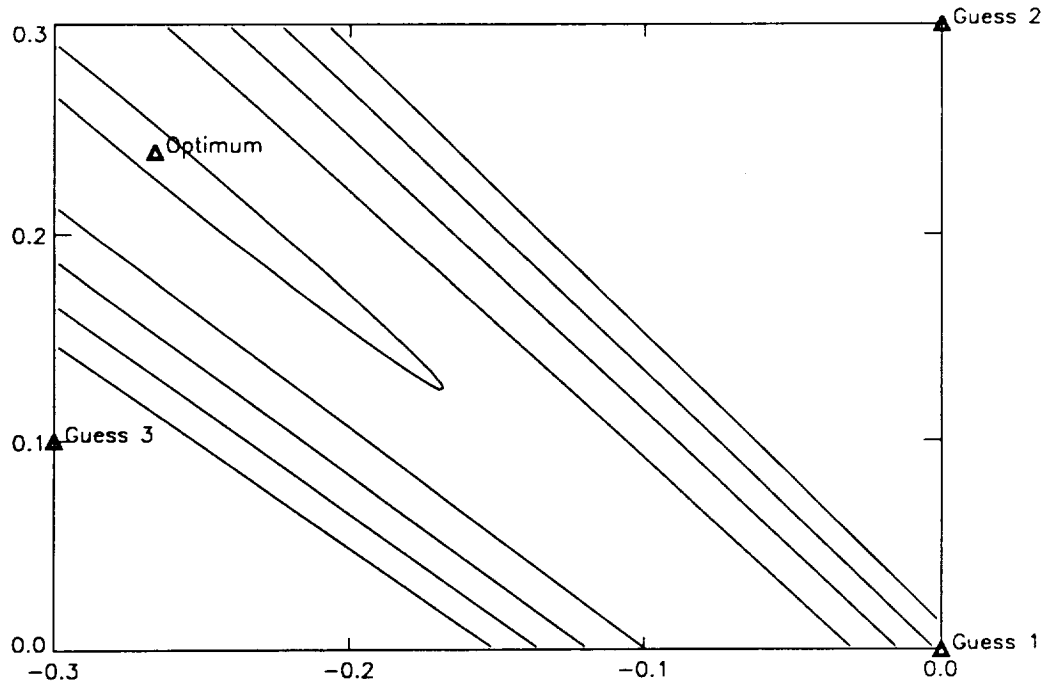

Figure 2: Contour plot of the objective function (for the AV-scheme) showing locations of Guesses 1, 2, 3, and the optimum. The two axes represent the two design variables (B-spline coefficients) describing the area function A

Third International Conference on Inverse Design Concepts and Optimization in Engineering Sciences (ICIDES-III), Editor: G.S. Dulikravich, Washington D.C., October 23-25, 1991.

Optimizer= NPSOL		Adjoint Discretization			
		$B = (J_U)^T_G$	$B = (J_U)^T_{EO}$	$B = (J_U)^T_{AV}$	$B = EMW$
Analysis Discret-ization	G	+	-	-	0
		+	-	-	-
		+	+	+	0
	EO	-	+	-	0
		-	+	0	0
		0	+	0	0
	AV	0	+	+	-
		-	+	+	0
		-	0	+	0

Table 1: Results obtained using NPSOL as the optimizer, for various combinations of forward and adjoint solvers. In each cell, the three entries correspond to initial guesses 1, 2, and 3 for the design variables. The designation (+) means that the optimization converged to the correct solution. The designation (o) means that the optimization got "close," but did not converge. The designation (-) means that the optimization did not succeed in getting close to the solution.