JOURNAL OF COMPUTATIONAL PHYSICS **98**, 74–89 (1992)

A Comparison of Optimization-Based Approaches for a Model Computational Aerodynamics Design Problem

PAUL D. FRANK AND GREGORY R. SHUBIN

Boeing Computer Services, P.O. Box 24346, Seattle, Washington 98124-0346

Received April 26, 1990; revised October 17, 1990

The objective of this paper is to compare three optimization-based methods for solving aerodynamic design problems. We use the Euler equations for one-dimensional duct flow as a model problem. The optimization methods are (i) the black-box method with finite difference gradients, (ii) a modification where gradients are found by an algorithm based on the implicit function theorem, and (iii) an all-at-once method where the flow and design variables are simultaneously altered. The three methods are applied to the model problem and compared for efficiency, robustness, and implementation difficulty. We also show that the black-box (implicit gradient) method is equivalent to applying the "variational" or "optimal control" approach to design optimization directly to the discretized analysis problem, rather than to the continuous problem as is usually done. The black-box method with implicit gradients seems to provide a good compromise of features, and can be retrofitted to most existing analysis codes to turn them into design codes. Although the all-at-once method was found to be less robust than the black-box methods, when it succeeded it was considerably more efficient. © 1992 Academic Press, Inc.

1. INTRODUCTION

Most of the effort in devising schemes for solving computational aerodynamics problems has focused on the forward, or *analysis* problem: given the shape of the airfoil (or aircraft), what will be the flow of air over it? Of more direct use in designing an aircraft is the solution of the more difficult inverse, or *design* problem: given the flow, what shape will produce it? Recently, due to improved methods for solving the analysis problem, and also due to increases in available computational power, there has been renewed interest in attacking the design problem.

Many different approaches to solving the design problem have been developed; these are nicely summarized in [1]. For our purposes, these approaches can be separated into two fundamental classes. In the first class, one attempts to solve the inverse problem by (essentially) manipulating the equations governing the geometry and the flow so that the geometry can be solved for, once the flow is specified. In the second class, a method for solving the forward problem is used iteratively, employing an optimization strategy to vary the airfoil shape in some systematic way until (close to) the desired flow is obtained. The second class of methods, while generally much more computationally intensive than the first, offers more promise for handling difficult geometries and complex flow phenomena and takes advantage of existing methods for solving the associated analysis problems.

While our ultimate goal is to solve design problems for realistic, three-dimensional transonic aircraft, our short-term objective is to handle the simpler case of a two-dimensional transonic airfoil in inviscid flow. Even in such a simple case, little is apparently known about whether the analysis (much less the design) problem is well posed. Indeed, computational evidence [2] suggests that the analysis problem for transonic airfoil flows governed by the full-potential equation may possess more than one solution for given airfoil shape and free-stream conditions. In general, we might expect the design problem, being an inverse problem, to potentially suffer from ill-posedness. Whether this is the case is difficult to discern from the literature.

The objective of this paper is to compare several optimization-based approaches for solving the design problem. To do so, we introduce a very simple model problem. The analysis problem for this model is well known and has been widely used for testing numerical methods for flows with shocks; it is the problem of determining the steady, one-dimensional flow of an inviscid fluid in a duct with a specified spatially-variable cross-sectional area. The design problem for the model is to determine the duct shape from the flow solution. One not-so-well-known fact about the duct flow model is that, under certain circumstances, the governing Euler equations can be reduced to a single ordinary differential equation. This fact and the way the duct geometry enters the problem, make it relatively easy to study and understand both the analysis and design problems for this simple model. Additionally, except for one-dimensionality, the flow phenomena exhibited by solutions of the model are quite similar to those in two-dimensional inviscid flow over an airfoil; this point is illustrated in Fig. 1. Thus, we may hope to gain some insight into the

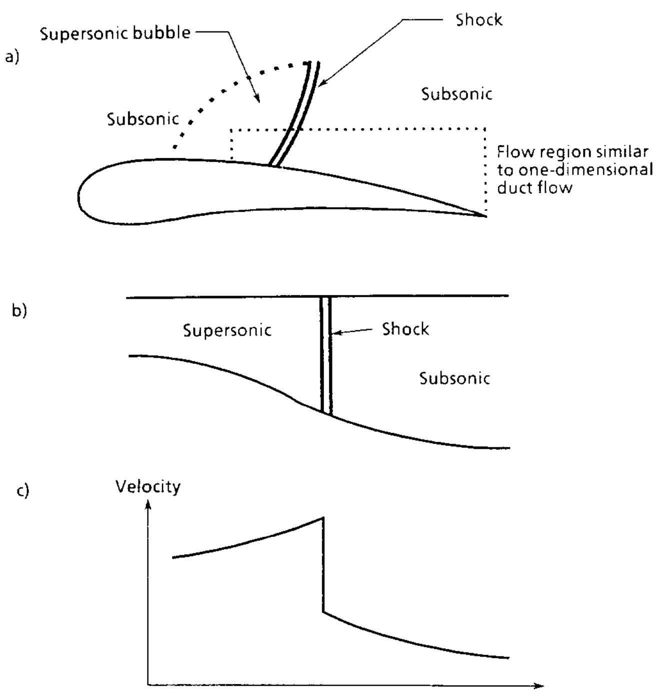

FIG. 1. Analogy between (a) flow over a transonic airfoil and (b) flow in a duct. Of course, the flow in (a) is two-dimensional, but is assumed one-dimensional in (b). In either case, the velocity along the airfoil surface or along the duct length is qualitatively given in (c)

nature of the airfoil design problem by studying the vastly simpler duct flow model.

In Section 2 below, we show the reduction of the Euler equations to a single equation, describe how to solve it, and discuss the posedness of the analysis problem. Additionally, we introduce three discretizations of the analysis problem (possessing different degrees of continuity) that will be used later. We also introduce the design problem for this model and discuss its posedness. In Section 3 we present three distinct optimization methods for solving the design problem, and discuss the relationships between them. In Section 4 we display some computational results using the three methods, and discuss the trade-offs between them. In Section 5 we present our conclusions.

2. MODEL PROBLEM

2.1. *Continuous Analysis Problem*

The steady flow of an inviscid fluid in a duct of variable cross-sectional area $A(\xi)$ is governed by the Euler equations

$$\mathscr{F}_\xi + \mathscr{G} = 0, \qquad 0 \leqslant \xi \leqslant 1, \tag{1}$$

where

$$\mathscr{F} = \begin{pmatrix} \rho u A \\ (\rho u^2 + p) A \\ (\rho E + p) u A \end{pmatrix}, \qquad \mathscr{G} = \begin{pmatrix} 0 \\ -pA_\xi \\ 0 \end{pmatrix},$$

ξ is distance along the duct, ρ is density, u is velocity, $E = e + u^2/2$, where e is specific internal energy, and p is pressure. Here, the subscript ξ means differentiation with respect to ξ, and it is assumed that $A(\xi)$ is a given, differentiable function. The pressure p is given by the equation of state for a perfect gas, $p = (\gamma - 1)\rho e$, where $\gamma > 1$ is the gas constant. (For air, $\gamma = 1.4$.) We assume supersonic inflow at $\xi = 0$ and subsonic outflow at $\xi = 1$. Under these circumstances, it is proper to specify three boundary conditions at $\xi = 0$ and one boundary condition at $\xi = 1$ [3].

We now show how, under these conditions, (1) can be reduced to a single ordinary differential equation in u. (See, e.g., [4].) The first and third components of (1) can be integrated to give

$$\rho u A = C, \qquad \gamma e + u^2/2 = H,$$

where the constants C (the mass flow rate) and H (the total enthalpy) are evaluated at the inflow boundary $\xi = 0$. Using these relations to eliminate ρ and e from the second component of (1) gives, after some algebra,

$$f_\xi + g = 0, \tag{2}$$

where

$$f(u) \equiv u + \bar{H}/u,$$

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

and $\bar{\gamma} = (\gamma - 1)/(\gamma + 1)$ and $\bar{H} = 2H\bar{\gamma}$ are given constants. Equation (2) is fully equivalent to (1); no approximations have been made in the derivation.

As an aside, we note that a useful model for testing schemes for nonlinear conservation laws can be obtaining by introducing an unphysical time dependence as

$$u_t + f_\xi + g = 0 \tag{3}$$

which, though not equivalent to the time-dependent Euler equations, gives the same solution when a steady state is reached. (It also rejects falsely stabilized shocks in regions of decreasing duct area ($A_\xi < 0$) that can be obtained as a solution of the time-independent equation (1) or (2) [5], and therefore it does capture some of the essence of the time-dependent Euler equations.)

By applying standard results for hyperbolic conservation laws like (3), appropriately specialized for steady solutions, we can immediately state some facts about the behavior of the solutions of the steady flow equation (2). The flow is sonic (where f' vanishes) at $u = u_* \equiv \sqrt{\bar{H}}$, supersonic for $u > u_*$, and subsonic for $u < u_*$. Shock jumps from u_{L} (left of the shock) to u_R (right of the shock) satisfy the Rankine–Hugoniot jump relation $f_L = f_R$, or,

$$u_L \cdot u_R = \bar{H}. \tag{4}$$

The entropy condition states that such jumps are always from supersonic to subsonic, or

$$u_L > u_* > u_R. \tag{5}$$

Since the flow is supersonic at $\xi = 0, f'$ is positive and it is proper to specify one boundary condition, say $u = u_{\text{in}}$ there. Similarly, the flow is subsonic at the outflow boundary $\xi = 1, f'$ is negative, and it is proper to specify $u = u_{\text{out}}$ there. For smooth solutions, (2) can be integrated in closed form to give

$$Au(2H - u^2)^r = K, \tag{6}$$

where $r = 1/(\gamma - 1)$ and K is a constant. It can be shown that K, which is a function of entropy, increases across a shock.

Now we pose our analysis problem, specified so that the solution contains a single shock at ξ_s, is supersonic for $0 < \xi < \xi_s$, and subsonic for $\xi_s < \xi < 1$.

Analysis Problem. Given:

$$A(\xi), \qquad A_\xi > 0 \tag{7a}$$

Find:

$$u(\xi) \text{ satisfying } \begin{cases} f_\xi + g = 0, \\ \quad \text{away from shocks;} \\ u_L \cdot u_R = \bar{H} \text{ and } u_L > u_* > u_R, \\ \quad \text{at shocks;} \\ u(\xi = 0) = u_{\text{in}} > u_* \\ u(\xi = 1) = u_{\text{out}} < u_* \\ \text{and other technical conditions.} \end{cases} \tag{7b}$$

The technical conditions amount to certain relationships between u_{in} and u_{out} that must hold in order for a solution to exist. Basically, given u_{in} (say), u_{out} must be larger than the value of u that would be obtained by assuming the shock to be at $\xi = 0$ (with the flow being entirely subsonic for $\xi > 0$), and less than or equal to the value of u that would be obtained by having the shock at $\xi = 1$ (with the flow being entirely supersonic for $\xi < 1$).

An algorithm for solving (7) is as follows: From $A(0)$ and u_{in}, determine K_1, that value of K satisfying (6). Similarly, from $A(1)$ and u_{out}, determine K_2. The solution curves of u as a function of A as determined by (6) for the two values of K are shown in Fig. 2. Since $A(\xi)$ is increasing for increasing ξ, we know that the solution must be as shown in

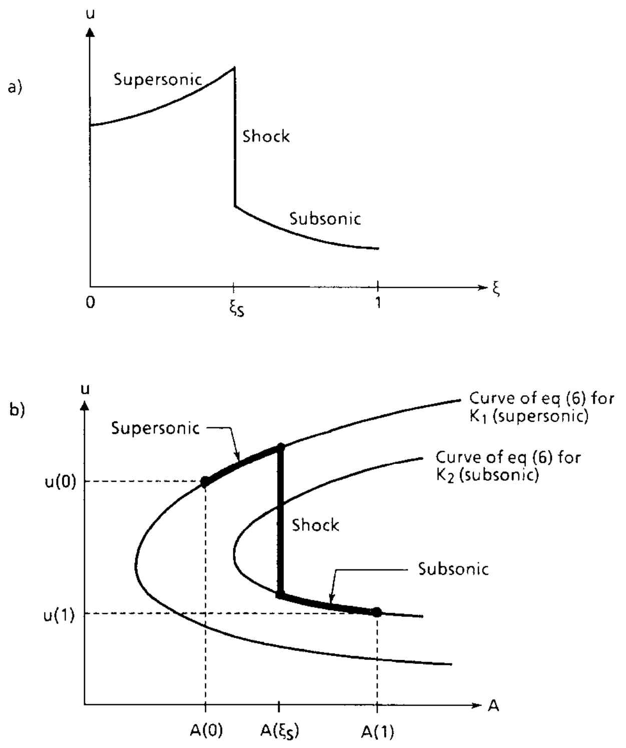

FIG. 2. Solution algorithm for continuous analysis problem.

Fig. 2. It remains to determine the shock location ξ_s. At the shock, A is the same from left to right, and (4) and (5) are satisfied. Defining $z(u) = u(2H - u^2)^r$, the shock conditions are satisfied if

$$\frac{1}{K_1} z(u_L) - \frac{1}{K_2} z\left(\frac{\bar{H}}{u_L}\right) = 0.$$

Recalling that $K_1 < K_2$, it can be shown that this equation has a unique solution under the stated conditions.

The analysis problem (7) given above is a well-posed problem. Existence, uniqueness, and continuous dependence of the solution on the data follow from a detailed analysis of the algorithm given above.

2.2. *Discrete Analysis Problem*

Of course, in practice analysis problems cannot generally be solved analytically, but are instead approximated numerically. We therefore introduce three discretization methods for (2) to be used in solving (7); these methods differ in their degrees of continuity, which has an effect on the results obtained with the design optimization methods presented later.

Let the ξ-coordinate be discretized by a uniform, cell-centered grid with centers at $\xi_j = (j - 1/2)h$, $h = 1/J$, 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 (2) is given by

$$w_j \equiv \frac{f_{j+1/2} - f_{j-1/2}}{h} + g_j = 0. \tag{8}$$

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 [6] with coefficients b_m for $m = 1, 2, ..., M$ 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 $u_{J+1} = u(\xi = 1)$.

It remains to prescribe the fluxes $f_{j+1/2}$ as functions of u_j and u_{j+1}. Three such prescriptions, f^{G}, f^{EO}, and f^{AV}, corresponding to the Godunov, Engquist–Osher, and artificial viscosity methods for numerically approximating hyperbolic conservation laws, are given in the Appendix. The Godunov flux f^{G} corresponds roughly to the first-order upwind scheme frequently used in computational aerodynamics and is a C^0 function of its arguments. The Engquist–Osher flux f^{EO} is a slight perturbation of f^{G} that makes it C^1. The artificial viscosity flux f^{AV} is entirely different, and is C^∞. The abilities of these schemes for sharply representing computed shock waves vary somewhat inversely to the degree of continuity, with the Godunov scheme having about one grid cell interior to a shock, the Engquist–Osher scheme two cells, and the artificial viscosity scheme many cells. Because continuity is an issue later, we will refer to these schemes as the C^0, C^1, and C^∞ difference schemes, respectively.

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

Discrete Analysis Problem.
Given: $b_m, m = 1, ..., M$ (spline coefficients describing $A(\xi)$)
Find: u_j satisfying

$$W(u) = 0. \tag{9}$$

Here W is the vector of discretized equations (8) for $j = 1, 2, ..., J$ and the boundary conditions on u. We will refer to the method for solving the analysis problem as the *analysis code*. The actual method employed in the analysis code may be Newton's method (or a variant), some other iterative method (e.g., multigrid), or a time-marching scheme that approximates a time-dependent differential equation like (3). In any case, we note (as an aside) that there is generally no guarantee that the nonlinear system of equations approximating the well-posed problem (7) will have a unique solution; this point is illustrated in [4, 7].

2.3. *Continuous Design Problem*

We next turn our attention to posing the design (or inverse) problem: given the flow solution $u(\xi)$, what is the duct geometry $A(\xi)$? In other words, we want to find that duct geometry $A(\xi)$ such that the solution of (7) is some specified function $\hat{u}(\xi)$. It is not clear what one wants to specify about $\hat{u}$. Obviously, it is possible to specify $\hat{u}$ in such a way that no choice of A will yield it; for example, a specification of $\hat{u}$ that does not satisfy (4) or (5) cannot be the solution of (7). This is, of course, precisely the situation faced in practice: it is generally impossible to specify a desired flow field that is certain to be realizable as the solution of the forward problem for some geometry (not to mention some reasonable geometry). In this sense, any realistic design problem for our model is going to be improperly posed. If we were able to know that $\hat{u}$ had to have a single shock satisfying the jump (4) and entropy (5) conditions and restricted our specifications of $\hat{u}$ to functions that are realizable as solutions of (7), it would then be possible to show continuous dependence of A on $\hat{u}$.

Another way to look at the design problem is to consider viewing (2) as an equation for A, given u. (This is the "non-optimization" view of the design problem referred to in the Introduction.) This equation is

$$(\bar{\gamma}u - \bar{H}/u)\, A_\xi + (u + \bar{H}/u)_\xi\, A = 0. \tag{10}$$

One difficulty that is immediately apparent is that the coefficients are discontinuous across a shock; this problem could be circumvented by including an appropriate viscous term in (2) that would slightly smear out the shock, and then, by taking the appropriate limit of viscosity, to zero. A more serious concern is that fact that (10) is a first-order *linear* equation, so that only one boundary condition on A can be specified. Assuming that $A(0)$ is specified and (10) integrated in the positive x direction, there is no guarantee that the correct value $A(1)$ would be achieved. In the context of the analogy suggested in Fig. 1, this would correspond to an airfoil that overlaps or does not close at the trailing edge.

Given these difficulties, we are led naturally to consider quasisolutions [8]; that is, we want to solve

Design Problem. Given: $\hat{u}(\xi)$
Find: $A(\xi)$, $A_\xi > 0$, such that $u(\xi)$ satisfies (7b) and $\|u(\xi) - \hat{u}(\xi)\|_2$ is minimized.

Obviously there exists at least one solution to this problem; if the minimizer is unique, the problem is apparently well-posed. We have not established whether there is a unique solution to this design problem as posed above. We do note that this particular objective function puts a large premium on getting the shock located correctly and that precise location of shocks may not be as important in practical design problems for airfoils or aircraft.

2.4. *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

Discrete Design Problem. Given: $\hat{u}_j, j = 1, ..., J$
Find: b_m, $m = 1, 2, ..., M$ (spline coefficients describing $A(\xi)$) such that (9) is satisfied and $\frac{1}{2}\sum_{j=1}^{J}(u_j - \hat{u}_j)^2$ is minimized.

Later we will consider three variations on this problem that amount to leaving $A(\xi)$ unconstrained, requiring $A_\xi > 0$, and requiring $A_{\xi\xi}$ have the "correct" sign. The latter two translate into simple linear constraints on the B-spline coefficients b_m.

3. APPROACHES TO FORMULATING DESIGN PROBLEMS USING OPTIMIZATION

Most of the recent literature on the aerodynamic design problem features specific optimization approaches or specific design problems. In this section we present a general view of the problem of optimal design. In particular, we consider three different methods for formulating the design problem as an optimization problem. General concepts for these approaches are illustrated by their application to the duct design problem discussed in Section 2. Except for these illustrations, this section is independent of the previous material.

Two fundamental design issues are the choice of *design variables* and the selection of an *objective function* for the optimization. Often the design variables specify the shape of an object, such as an airfoil. For example, the duct design variables are the coefficients of the piecewise cubic splines specifying the duct area function $A(\xi)$. In general, design variables can be any quantities that affect the analysis result. In addition, there are many choices for the optimization objective function. For example, the objective function for the duct problem is to come as close as possible to some specified velocity distribution. For airfoil design the objective might be to minimize the drag of an airfoil over several operating conditions.

In addition to the objective function there are usually design constraints. Some examples of design constraints are that an airfoil be sufficiently thick to ensure structural integrity, or that the pressure in a given flow region not exceed a specified level. In the duct design problem we will, for example, sometimes require that the duct area be monotonically increasing.

Though selection of appropriate design objective and constraint functions are important, the correct choices are problem-dependent and little general guidance can be given. There are, however, some issues regarding the methods presented in Sections 3.1–3.4 that are more generic. These include efficiency, optimization trial designs that cannot be solved by the analysis code, and potential discontinuities in optimization problem derivatives due to analysis techniques such as shock-handling schemes and grid refinement. Indeed, one optimization difficulty that affects all of the methods discussed here is the "mismatch" in the continuity requirements of analysis schemes based on Newton's method and those for efficient optimization algorithms. That is, convergence results for Newton-based analysis schemes require C^1 continuity of the discretized differential equations (though this is infrequently achieved in practice), whereas efficient optimization codes require C^2 continuity of the objective and constraint functions with respect to the design variables. (See, e.g., [9] for a discussion of these continuity requirements.) This mismatch can adversely affect the optimization process because the design problem cannot have a higher degree of continuity than the associated analysis problem.

3.1. *The Black-Box Method*

The *black-box* method is the most direct approach to optimal design. In the black-box method the analysis code is repeatedly invoked as the design variables are altered by the optimization code. Since the analysis code is independent of the optimization code, it may be treated as a black box.

If the design is characterized by a vector x_D of n_D *design variables* then the optimal design problem is given by

$$\begin{aligned} &\text{minimize } \mathtt{f}(x_D), \\ &\quad x_D \in \mathbf{R}^{n_D} \qquad\qquad (11) \\ &\text{subject to } C(x_D) \geq 0, \end{aligned}$$

where $\mathtt{f}(x_D)$ is the objective function[1] and $C(x_D)$ is a vector of m_D constraint functions. In the black-box method, each evaluation of $\mathtt{f}(x_D)$ requires a solution by the analysis code.

Often, the function $\mathtt{f}$ will be formulated in terms of *flow variables* x_F. The flow variables are the physical variables on the discretization grid, such as velocities or pressures. For example, the objective for the duct design problem is a function of velocities on the grid cells. In this situation, $\mathtt{f}$ is dependent on the design variables x_D in an indirect manner. That is, the variables x_D are linked to the flow variables x_F via the differential equations or the discretization of these equations, since the flow variables will change when (for example) the geometry is altered. In general, $\mathtt{f}$ will have both a direct dependence on x_D and an indirect dependence on x_D, due to the dependence of x_F on x_D. Thus, one could consider the objective function to be $\mathtt{f}(x_F(x_D), x_D)$. The

[1] Note that the typewriter font $\mathtt{f}$ is used for the objective function to distinguish it from the flux function f introduced in Section 1.

term $x_F(x_D)$ indicates that, given x_D, the value of x_F is obtained by solving an analysis problem.

The constraints C may also have an indirect dependence on the design variables. However, they will often be shape constraints formulated directly in terms of x_D. For example, one version of the duct design problem requires that the duct area increase monotonically. This constraint can be formulated in terms of the coefficients of the piecewise polynomials that define the area function $A(\xi)$.

One of the drawbacks of the black-box approach is high computational cost. Typically, efficient optimization codes (see, e.g., [10]) for solving (11) require computation of $\nabla_D \mathrm{f}$ and $\nabla_D C$, the gradients of the objective funcion and constraints with respect to the design variables. Computing these derivatives by one-sided finite differences requires solving n_D analysis problems, where each problem corresponds to a perturbation of a different component of x_D. One mitigating factor is that solving these perturbed analysis problems should be considerably cheaper than solving arbitrary problems, at least when the analysis code employs an iterative solver. This is due to the availability of the solution of the "nearby" problem at the nominal value of x_D as a starting guess for the iteration at the perturbed value of x_D. In the next section we show that the first derivatives for the design problem can often be computed by another method for considerably less cost than solving n_D analysis problems.

Another difficulty with the black-box approach is that the analysis process may fail for some values of x_D generated by the optimizer. The failure may be due to lack of robustness of the analysis code or to the design having no solution in the domain of the analysis model (e.g., a physically unrealistic geometry, or one that has no steady-state flow solution). One approach to handling this problem is to impose constraints that prohibit designs that have no analysis solution. For example, an airfoil could be constrained to exclude shapes with sharp indentations. Another approach is to assign very large objective function values to unanalyzable designs so that they are not selected as improved designs. A minimum requirement for the black-box approach is the existence of an analysis solution for the initial design input to the optimizer.

An advantage of the black-box approach is that the analysis code can be used essentially without modification. Thus, there is no need to tamper with complicated discretization schemes such as those used in most advanced computational aerodynamics codes.

3.2. *Implicit Gradient for the Black-Box Scheme*

In this section we describe a method, based on the implicit function theorem, for "cheaply" computing derivatives required in the optimization. Similar methods are mentioned in Ref. [11] and the citations therein. For simplicity, the unconstrained version of (11) is considered. However, the results apply to the constrained problem as well.

Assume that the analysis problem has been discretized 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 x_D, the analysis code solves $W(x_F)=0$, where x_F is the vector of n_F flow variables and W is a vector of n_F nonlinear equations. Since the analysis problem is an implicit function of x_D it can be viewed as solving

$$W(x_F, x_D)=0 \tag{12}$$

for x_F, given a design specified by x_D.

Suppose that x_F and x_D are considered as subsets of the n_F+n_D vector x given by

$$x \equiv (\quad x_F \quad | \; x_D); \tag{13}$$

the Jacobian (first-derivative) matrix of (12) is then

$$J=\left[\begin{array}{c|c} \quad J_F \quad & J_D \end{array}\right], \tag{14}$$

where J is $n_F \times (n_F+n_D)$, J_F is the $n_F \times n_F$ Jacobian with respect to the flow variables, and J_D is the $n_F \times n_D$ Jacobian with respect to the design variables. (The partitioned view of the Jacobian implies $n_F \gg n_D$; this will usually be the case.) Note that J_F is often available in analysis codes, especially those based on Newton's method and variants.

Consider the function $\tilde{\mathrm{f}}(x_F, x_D)$, where $\tilde{\mathrm{f}}$ is the same as the black-box method objective function f, except that x_F and x_D are considered to be independent of each other. The function $\tilde{\mathrm{f}}(x_F, x_D)$ is then equivalent to the black-box method objective function $\mathrm{f}(x_F(x_D), x_D)$ only when (12) is satisfied. Computing gradients of $\tilde{\mathrm{f}}$ is considerably simpler than computing gradients of f. This is due to the fact that the partial derivatives of $\tilde{\mathrm{f}}$ with respect to variables in x_D can be computed with the assumption that x_F is fixed. In contrast, the partial derivatives of f with respect to variables in x_D must account for the fact that x_F is a function of x_D.

Usually $\nabla_F \tilde{\mathrm{f}}$ and $\nabla_D \tilde{\mathrm{f}}$, the gradients of $\tilde{\mathrm{f}}$ with respect to the flow variables and the design variables, respectively, are available as analytic expressions or can easily be computed by finite differences. For example, the discrete design problem for duct flow has $(\nabla_F \tilde{\mathrm{f}}(x))_j = u_j - \hat{u}_j$ and $\nabla_D \tilde{\mathrm{f}}(x)=0$. However, the black-box method requires $\nabla_D \mathrm{f}$, the gradient of f with respect to the design variables x_D. Theorem 3.1 provides an efficient way to compute $\nabla_D \mathrm{f}$, given $\nabla_F \tilde{\mathrm{f}}$ and $\nabla_D \tilde{\mathrm{f}}$.

THEOREM (3.1). *If $W(\bar{x}_F, \bar{x}_D) = 0$ and $W(x_F, x_D)$ is C^1 in a neighborhood of $\bar{x} = (\bar{x}_F, \bar{x}_D)$, with J_F nonsingular at $\bar{x}$ then*

$$\nabla_D \mathrm{f}(\bar{x}_D) = \nabla_D \tilde{\mathrm{f}}(\bar{x}) - J_D^{\mathrm{T}} J_F^{-\mathrm{T}} \nabla_F \tilde{\mathrm{f}}(\bar{x}). \tag{15}$$

(*Here, superscript* T *indicates transpose.*)

Proof. The implicit function theorem (see, e.g., [12]) guarantees the existence of a C^1 (unique, locally one-to-one) function mapping x_D to x_F in a neighborhood of $\bar{x}_D$, such that $W(x_F, x_D) = 0$. Suppose $x_D = \bar{x}_D + \Delta x_D$ and, correspondingly, $x_F = \bar{x}_F + \Delta x_F$. The Taylor series for $W(x_F, x_D)$ expanded about $\bar{x}$ is

$$W(x_F, x_D) = 0 = W(\bar{x}_F, \bar{x}_D) + J_F \Delta x_F + J_D \Delta x_D + O(\|\Delta x\|^2). \tag{16}$$

Since $W(\bar{x}_F, \bar{x}_D) = 0$, Eq. (16) yields

$$\Delta x_F = -J_F^{-1} J_D \Delta x_D + O(\|\Delta x\|^2). \tag{17}$$

Since the mapping from x_D to x_F is locally one-to-one, $\Delta x \to 0$ as $\Delta x_D \to 0$. Thus, Eq. (17) indicates that at $\bar{x}$ the partial derivative of component i of x_F with respect to component j of x_D is the (i, j) component of $M \equiv -J_F^{-1} J_D$, where M is an $n_F \times n_D$ matrix. Accounting for changes in f due directly to changes in x_D and, via the chain rule, to the corresponding changes in x_F gives

$$\nabla_D \mathrm{f}(\bar{x}_D) = \nabla_D \tilde{\mathrm{f}} + M^{\mathrm{T}} \nabla_F \tilde{\mathrm{f}}(\bar{x}),$$

which is equivalent to (15). ▌

The following algorithm could be used for computing $\nabla_D \mathrm{f}$ using Eq. (15). First compute $\nabla_F \tilde{\mathrm{f}}$ and $\nabla_D \tilde{\mathrm{f}}$, solve $J_F^{\mathrm{T}} y = \nabla_F \tilde{\mathrm{f}}$ for y and then compute $\nabla_D \mathrm{f} = \nabla_D \tilde{\mathrm{f}} - J_D^{\mathrm{T}} y$. Note that, if it is difficult to solve linear systems with the matrix J_F^{T}, the linear algebra in (15) can be rearranged as $(J_F^{-1} J_D)^{\mathrm{T}} \nabla_F \tilde{\mathrm{f}}(\bar{x})$, requiring n_D solves with J_F.

Thus, computation of $\nabla_D \mathrm{f}$ by the implicit method requires computing J_D and solving the linear system $J_F^{\mathrm{T}} y = \nabla_F \tilde{\mathrm{f}}$. Computation of J_D by forward finite differences requires n_D evaluations of $W(x_F, x_D)$. Note that *evaluation* of $W(x_F, x_D)$ (sometimes referred to as "computing the residuals") is usually significantly cheaper than *solving* $W(x_F, x_D) = 0$, i.e., solving the analysis problem. Solving $J_F^{\mathrm{T}} y = \nabla_F \tilde{\mathrm{f}}$ is trivial if the analysis code computes a factorization of J_F. However, if an iterative method such as pre-conditioned conjugate gradient is used in the analysis code, then the iterative solver must be adapted to solve the transposed system.

Some analysis codes do not provide J_F or an iterative solver for systems involving J_F; an example is a time-dependent code where the steady-state solution is found by stepping through time. The implicit gradient scheme can still be used in this case, provided that J_F can be computed efficiently using sparse finite differences (see, e.g., [13]). The sparse difference approach only requires that the analysis code provide the values of $W(x_F, x_D)$ when values of x_F and x_D are input; most codes, if not already in this form, can be easily modified to produce W.

In general, computing implicit gradients is much cheaper than computing gradients by finite differences. This is because the finite difference gradient computation requires the solution of n_D analysis problems. In contrast, computing the gradient implicitly requires n_D evaluations of the flow equations $W(x_F, x_D)$ and one solve of a linear system with the matrix J_F^{T} (or n_D solves with J_F). A disadvantage of the implicit scheme is that some (perhaps substantial) modification of the analysis code is required.

3.3. *The All-at-Once Method*

In deriving the implicit gradient method the objective function $\tilde{\mathrm{f}}$ and the discretized differential equations W were considered to be functions of the independent sets of variables x_D and x_F. Thus, one could consider a design method where both x_D and x_F are treated as optimization variables and the flow equations $W(x_F, x_D) = 0$ are treated as equality constraints. This *all-at-once* method can be described formally as

$$\begin{aligned} &\underset{x \in \mathbf{R}^{(n_F + n_D)}}{\text{minimize}}\ \tilde{\mathrm{f}}(x_F, x_D), \\ &\text{subject to } C(x_F, x_D) \geq 0, \\ &\qquad\qquad W(x_F, x_D) = 0, \end{aligned} \tag{18}$$

where $x = (x_F, x_D)$ and the vector C consists of the design constraints as in (11). An iteration of the optimization now involves simultaneous modification of both x_F and x_D. A similar approach to the design problem is described in [14].

An advantage of the all-at-once method over the black-box method is the probability of requiring considerably fewer equivalent solutions of the large discretized system $W(x) = 0$. This is because the black-box method requires the solution of $W(x_F) = 0$ for each change in x_D. However, in the all-at-once method, each change in x_D requires the computational equivalent of only one step of a Newton solver for $W(x_F) = 0$.

Another advantage of the all-at-once approach is that it does not require the existence of solutions to the analysis problem for all values of the design variables generated in the course of the optimization. All that is required is that the residual of the system $W(x_F, x_D)$ be computable for the values of x_F and x_D generated by the optimizer. However, by definition, the analysis problem must be analyzable at the optimal value for x_D.

A big disadvantage of the all-at-once method is that the optimization code is not isolated from the analysis code. That is, since the optimization code must simultaneously change the analysis and design variables, it must contain all the specialized software required for an analysis. In particular, even if the number of design variables is small, the optimizer must include code for handling large analysis problems; for example, sparse matrix factorization codes or codes that compute preconditioners and conjugate gradient iterations. Consequently, the all-at-once optimization code may have to be modified significantly for application to each new analysis problem.

Another disadvantage of the all-at-once method compared to the black-box method was discovered in tests on the duct design problem: the all-at-once method is much more susceptible to derivative discontinuities arising from finite difference schemes designed to sharply approximate shocks. These discontinuities are caused by certain "switches" in the difference schemes that trigger changes in the difference formulas based on the computed flow solution. The last of the "if" tests associated with the Godunov scheme, shown in Appendix A, typifies such low-continuity switches. Often, the activation of these switches is associated with the movement of shocks.

The black-box optimization method "sees" only converged analysis solutions, and thus does not experience the difference formula switches that occur in the course of converging an analysis solution. It does, however, indirectly experience the discontinuities caused by these switches when shocks move from one grid cell to the next between successive converged analysis solutions. In contrast, the all-at-once optimization method works directly with the difference formulas, since they define equality constraints for the optimization problem. Thus, this method frequently encounters the discontinuities due to the switches as the optimizer traverses (x_F, x_D)-space.

3.4. *The Variational Approach*

To facilitate the description of the variational approach to optimal design, optimality conditions for the finite-dimensional, discretized problem are discussed first. Consider the all-at-once problem (18) with no design constraints, i.e., no $C(x_F, x_D) \geqslant 0$ constraints. The *Lagrangian*, $L(x_F, x_D, \lambda)$, for (18) is then given by

$$L(x_F, x_D, \lambda) \equiv \tilde{f}(x_F, x_D) - W^{\mathrm{T}}(x_F, x_D)\lambda, \qquad (19)$$

where λ is a vector of n_F *Lagrange multipliers*. The first-order optimality condition for (18) is

$$\nabla L(x_F, x_D, \lambda) = 0. \qquad (20)$$

In formulating variational or optimal control (see, e.g., [15]) methods for the design problem one derives relationships corresponding to (19) and (20) in an infinite-dimensional space (i.e., for the continuous problem). In this approach, the vectors x_F and x_D are replaced by continuous variables, the discrete equations $W(x_F, x_D) = 0$ are replaced by the original differential equation, sums in the objective function are replaced by integrals over the domain and the Lagrange multiplier vector is replaced by an *adjoint function*. The adjoint function is specified by an operator equation, the *adjoint equation*, which is derived by imposing optimality conditions or descent conditions on the gradient of the infinite-dimensional "Lagrangian." Once the adjoint has been derived, both the original differential equation and the adjoint are discretized and used in an optimization procedure.

The variational approach to design has been described by several authors, and has been applied in many disciplines for a long time. (See, e.g., [16, 17] for applications to petroleum reservoir problems.) In aerodynamics, Miele [18] describes variational methods based on satisfying optimality conditions and Jameson [19] describes a variational method for finding an infinite-dimensional gradient of the design problem. Using Jameson's approach, the gradient can be computed for the cost of solving the original analysis problem and the adjoint problem. This can be significantly cheaper than the n_D analyses required for the black-box method using finite difference gradient computations.

Detailed derivations of variational methods for certain problem classes are given in the references. Here, a result is presented that shows that a particular variational method (one representative of those used in practice) applied to the discretized problem is equivalent to the black-box method with implicit gradients.

Note that in the "variational" method described below the discretization of the problem occurs prior to derivation of the optimality formulas. In contrast, in the control theoretic methods (e.g., [19]) optimality conditions are derived prior to discretizing the problem.

Given the assumptions of Theorem 3.1, consider the first-order optimality conditions for the discretized design problem, given by (20). Let $\nabla_\lambda L$, $\nabla_F L$, and $\nabla_D L$, respectively denote the gradient of the Lagrangian with respect to the Lagrange multipliers, the flow variables, and the design variables. The optimality conditions (20) are then given by

$$\nabla_\lambda L(x_F, x_D, \lambda) = 0 \Rightarrow W(x_F, x_D) = 0 \qquad (21)$$

$$\nabla_F L(x_F, x_D, \lambda) = 0 \Rightarrow \nabla_F \tilde{f}(x) - J_F^{\mathrm{T}}\lambda = 0 \qquad (22)$$

$$\nabla_D L(x_F, x_D, \lambda) = 0 \Rightarrow \nabla_D \tilde{f}(x) - J_D^{\mathrm{T}}\lambda = 0, \qquad (23)$$

Equation (21) requires that the discretized system of equations be satisfied.

Conditions (21)–(23) constitute a nonlinear system of $(2 * n_F) + n_D$ equations in as many unknowns. Algorithm A is an iterative method for solving this system; it is the same algorithm that would be obtained by applying optimal control or variational techniques to the "discrete Lagrangian" (19) rather than to its continuous analogue (see, e.g., [20]).

Algorithm A. Variational Method for the Discretized Design Problem.

Step 1. Given a value for the design variables x_D, solve (21) for x_F.

Step 2. Compute $\nabla_F \tilde{\mathrm{f}}(x)$ and J_F. Then solve (22) for λ.

Step 3. Compute $\nabla_D \tilde{\mathrm{f}}$ and J_D. Compute $\nabla_D L(x_F, x_D, \lambda)$ using (23). If $\nabla_D L(x_F, x_D, \lambda) = 0$ then STOP, the first-order optimality conditions are satisfied. Otherwise, use $\nabla_D L(x_F, x_D, \lambda)$ to obtain a new value for x_D which reduces the optimization objective function. GO TO Step 1.

We note that in Step 3, the new value for x_D does not necessarily reduce the residual in (23), but instead seeks to directly reduce the objective function f. However, it will be shown below that at a minimizer for f, (23) will be satisfied.

The relation between Algorithm A and the black-box method using implicit gradients is indicated in Proposition 3.2.

Proposition (3.2). *The variational method given by Algorithm A is equivalent to the black-box method using implicit gradients.*

Proof. In the black-box method each new value of x_D requires solving an analysis problem for x_F, just as in Step 1 of Algorithm A. In Step 2 of Algorithm A, solving (22) for λ yields $\lambda = J_F^{-\mathrm{T}} \nabla_F \tilde{\mathrm{f}}(x)$. Using (15) and this value for λ in (23) gives

$$\nabla_D L(x_F, x_D, \lambda) = \nabla_D \tilde{\mathrm{f}}(x) - J_D^{\mathrm{T}} J_F^{-\mathrm{T}} \nabla_F \tilde{\mathrm{f}}(x) \equiv \nabla_D \mathrm{f}(x_D). \tag{24}$$

At each iteration of an optimization code implementing the black-box method, a new value for x_D is computed which reduces the objective function. These values for x_D are obtained by computing *descent directions* for the objective function, where descent directions are vectors that have negative inner products with $\nabla_D \mathrm{f}(x_D)$. Equation (24) indicates that this technique is consistent with step 3 of Algorithm A. The black-box optimization method terminates at an optimal solution when $\nabla_D \mathrm{f}(x_D) = 0$, which is equivalent to the termination condition in step 3 of Algorithm A. ▌

Proposition 3.2 provides insight into comparing the black-box method with implicit gradients to the variational scheme of Jameson. To compute a gradient for the design problem, both methods require solution of the analysis problem. Additionally, the implicit gradient scheme requires solving the linear system $J_F^{\mathrm{T}} y = \nabla_F \tilde{\mathrm{f}}$, whereas Jameson's method requires solving (a discretization of) the adjoint equations.

A potential disadvantage of Jameson's approach is that the adjoint equations may be difficult to derive for complex design problems. Another potential drawback arises from deriving the gradient formula prior to discretizing the problem. The discretized adjoint problem may not be discretely adjoint to the discretized analysis problem. This mismatch could cause the computed gradient of the discretized design problem to be inaccurate.

3.5. *A Solution Method for Nonlinear Optimization Problems*

In Sections 3.1–3.4 we discussed posing the design problem as a nonlinear optimization problem. To provide insight into the optimization method used in our testing, we now discuss the sequential quadratic programming (SQP) method. A detailed description of the SQP method appears in [10].

The SQP method is an iterative method where at each iterate x, a step is taken based on solving a local model of the problem. The objective function for the model problem is a quadratic approximation to the Lagrangian at x and the model constraints are a linearization of the constraints at x. That is, at each iteration the SQP method computes the step p from x that solves the quadratic program (QP) given by

$$\begin{aligned} &\underset{p \in \mathbf{R}^n}{\text{minimize}}\ g^{\mathrm{T}} p + \tfrac{1}{2} p^{\mathrm{T}} Q p, \\ &\text{subject to } Jp \geqslant 0, \end{aligned} \tag{25}$$

where the n vector g is the gradient of the Lagrangian, the $n \times n$ matrix Q is an approximation of the Hessian (second-derivative) matrix of the Lagrangian and the $m \times n$ matrix J represents the linearized constraints.

Solving the QP is also an iterative process, where at each QP iteration there is a different *active set* of constraints. The active set consists of equality constraints and exactly satisfied inequalities. Each QP iteration requires solving the equality constrained problem (EP) given by

$$\begin{aligned} &\underset{p \in \mathbf{R}^n}{\text{minimize}}\ g^{\mathrm{T}} p + \tfrac{1}{2} p^{\mathrm{T}} Q p, \\ &\text{subject to } \hat{J} p = 0, \end{aligned} \tag{26}$$

where the $t \times n$, $t \leqslant n$, matrix $\hat{J}$ represents the active set of linearized constraints.

Solving EP usually requires solving linear systems involving $\hat{J}$ and the projection of Q into the null space of $\hat{J}$. These linear systems are modified each time an inequality is added to or deleted from the active set.

The direction to the solution of the QP subproblem is used to find an improved iterate for the original problem. If the new iterate is not optimal, a new QP subproblem is formed and solved.

One advantage of the SQP method is that QP solver iterations only involve evaluations of the QP model. No extra evaluations of the original objective function and constraints are required. This is important in the black-box approach because each evaluation of the objective function requires solving an analysis problem. A disadvantage of the SQP method is that only the linearized constraints are satisfied at each major iteration. The nonlinear constraints are not guaranteed to be satisfied until an optimal point is found. For example, when using an SQP code in conjunction with the all-at-once method, the discretized flow equations may not be satisfied until an optimal point is found.

4. NUMERICAL RESULTS

In this section we present numerical results obtained by applying the design methods discussed in Section 3 to the discrete problem for duct flow described in Section 2. The testing was done on a Sun SPARC workstation.

The optimization code used was NPSOL version 2.0, a product of the Systems Optimization Laboratory, Stanford University. NPSOL is an implementation of the SQP method described in Section 3.5. NPSOL 2.0 computes a secant approximation to the Hessian (second derivative) matrix and the user supplies first derivatives. Forward finite differences were used to compute first derivatives for the black-box (finite difference gradient) method. NPSOL is intended for small to moderate size problems in that it treats all matrices as dense. This is acceptable for the duct design problems since the total of design and flow variables never

The design variables (called x_D in Section 3) were the B-spline coefficients describing the duct geometry $A(\xi)$. The two end values of A were fixed at $A(0)=1.05$ and $A(1)=1.745$. The tests were run for the case of two design variables ($n_D=2$) and 10 design variables ($n_D=10$). The linear duct shown in Fig. 3a was the initial design for each run.

Velocities along the duct were the flow variables (called x_F in Section 3) for the duct design problem. We took $J=40$ grid cells, so there were $n_F=40$ flow variables. The boundary conditions were $u_0=1.299$ and $u_{41}=0.506$. The flow variables resulting from an analysis of the linear duct, using the C^0 difference scheme, appear as crosses in Fig. 3b.

In the black-box method optimization runs Newton's method was used for the analyses and the runs were "warm started." That is, the initial values for the flow velocities were taken from the preceding analysis. The initial velocity profile for the first analysis in an optimization run was a linear profile connecting the boundary conditions.

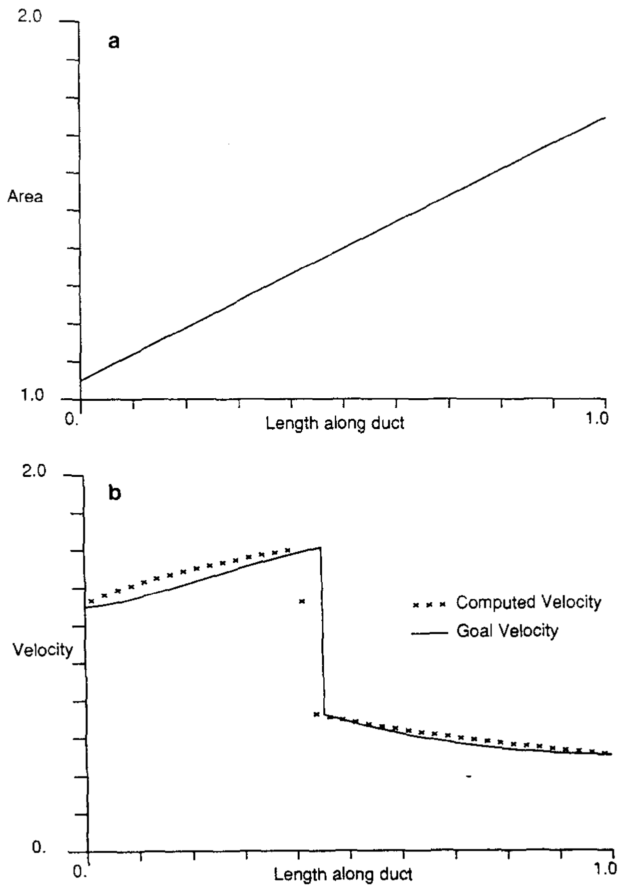

FIG. 3. (a) Area profile, linear duct using C^0 scheme. (b) Velocity profile, linear duct, using C^0 scheme.

The velocities $\hat{u}_j$ used as the design goal were the evaluations on the computational grid of the analytic solution for a duct with a cross-sectional area given by a sinusoidal

continuous curve in Fig. 3b.

Figure 4 shows the optimal solutions for the $n_D=2$ duct design problem using the C^0, C^1, and C^∞ difference

TABLE I

Test Results for $n_D=2$, No Constraints

Problem formulation	Difference scheme	Opt. found?	No. itrns	No. fevals	No. equiv. Newton steps	Time (s)
Bbox (fd grad)	C^0	yes	10	16	181	11.7
Bbox (impl grad)	C^0	yes	11	20	166	10.4
All-at-once	C^0	*lm*	—	—	—	—
Bbox (fd grad)	C^1	yes	7	13	156	9.4
Bbox (impl grad)	C^1	yes	7	13	108	6.9
All-at-once	C^1	no	—	—	—	—
Bbox (fd grad)	C^∞	yes	9	16	187	12.6
Bbox (impl grad)	C^∞	yes	9	16	131	9.2
All-at-once	C^∞	yes	6	7	6	6.7

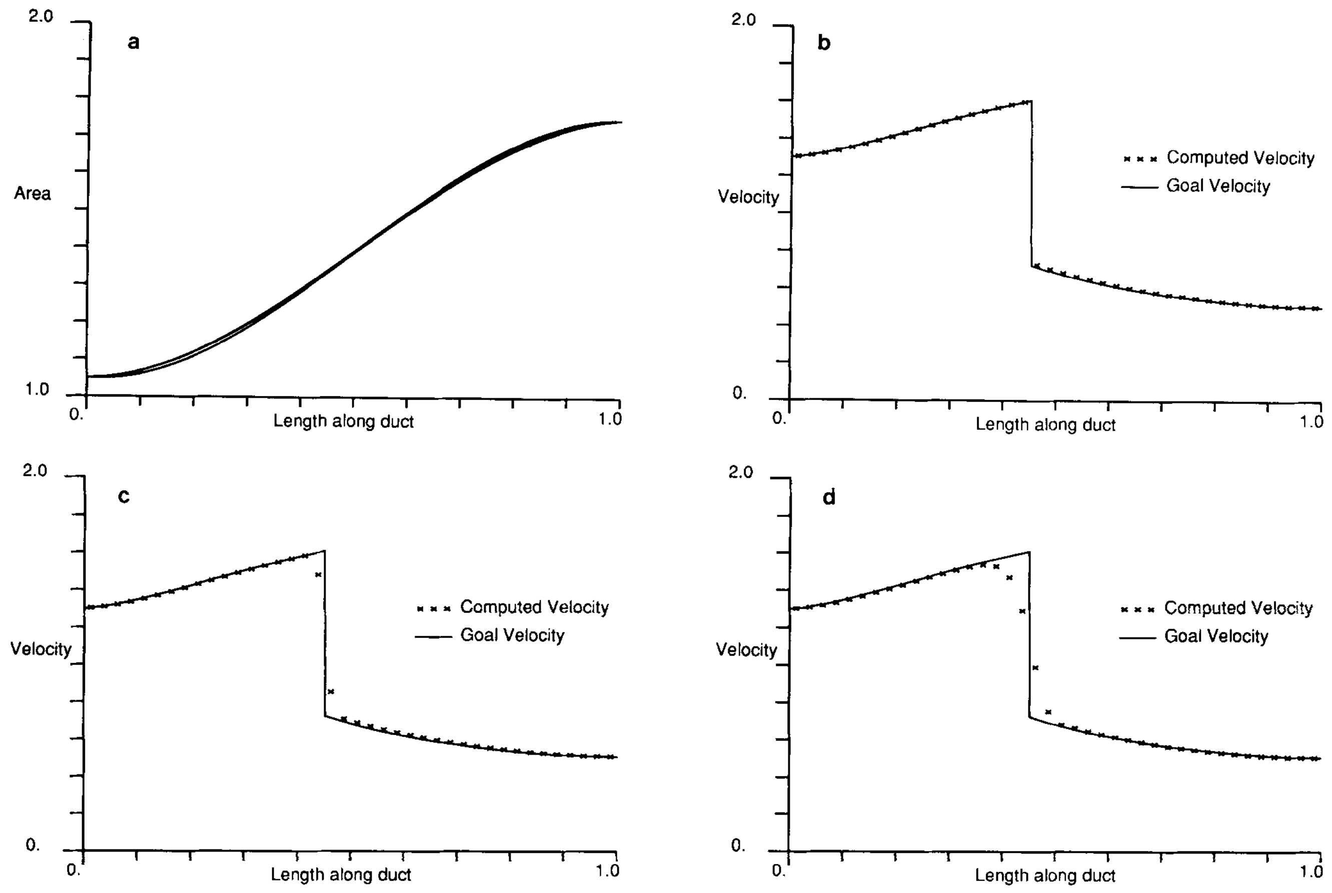

FIG. 4. (a) Area profiles, optimal ducts with two design variables. (b) Velocity profile, optimal duct with two design variables, using C^0 scheme. (c) Velocity profile, optimal duct with two design variables, using C^1 scheme. (d) Velocity profile, optimal duct with two design variables, using C^∞ scheme.

schemes. The optimal ducts, shown in Fig. 4a, are only slightly different. However, the optimal fit to the velocity design goal (Figs. 4b–d) clearly improves as the continuity of the difference scheme is decreased. This is not surprising since the design goal has a maximally sharp shock.

Table I gives the numerical results for tests with $n_D = 2$. The following notation is used in Tables I–IV. "Bbox (fd grad)" and "Bbox (impl grad)," respectively, denote the black-box scheme with finite difference gradients and gradients computed using the implicit method. "Opt. Found?" indicates whether or not the optimization code converged to the optimal solution. If the optimization code

TABLE II

Test Results for $n_D = 10$, No Constraints

Problem formulation	Difference scheme	Opt. found?	No. itrns	No. fevals	No. equiv. Newton steps	Time (s)
Bbox (fd grad)	C^0	yes	31	48	1216	105.4
Bbox (impl grad)	C^0	yes	31	48	317	35.9
All-at-once	C^0	yes	19	30	19	32.3
Bbox (fd grad)	C^1	yes	30	46	1143	97.1
Bbox (impl grad)	C^1	yes	28	45	315	27.2
All-at-once	C^1	yes	19	31	19	33.8
Bbox (fd grad)	C^∞	yes	28	40	998	68.9
Bbox (impl grad)	C^∞	yes	29	42	257	23.3
All-at-once	C^∞	yes	11	12	11	18.2

TABLE III

Test Results for $n_D = 10$, First Derivative Constraints

Problem formulation	Difference scheme	Opt. found?	No. itrns	No. fevals	No. equiv. Newton steps	Time (s)
Bbox (fd grad)	C^0	yes	55	85	2046	175.7
Bbox (impl grad)	C^0	yes	58	89	418	55.8
All-at-once	C^0	*lm*	—	—	—	—
Bbox (fd grad)	C^1	yes	18	28	675	43.6
Bbox (impl grad)	C^1	yes	19	28	133	13.7
All-at-once	C^1	yes	93	146	246	150.1
Bbox (fd grad)	C^∞	yes	16	23	544	36.7
Bbox (impl grad)	C^∞	yes	17	24	108	12.2
All-at-once	C^∞	yes	7	10	32	13.5

TABLE IV

Test Results for $n_D = 10$, First and Second Derivative Constraints

Problem formulation	Difference scheme	Opt. found?	No. itrns	No. fevals	No. equiv. Newton steps	Time (s)
Bbox (fd grad)	C^0	yes	41	64	1557	134.2
Bbox (impl grad)	C^0	yes	41	64	315	41.4
All-at-once	C^0	*lm*	—	—	—	—
Bbox (fd grad)	C^1	yes	18	30	713	46.3
Bbox (impl grad)	C^1	yes	16	26	122	13.4
All-at-once	C^1	yes	121	202	463	214.2
Bbox (fd grad)	C^∞	yes	15	23	580	39.3
Bbox (impl grad)	C^∞	yes	15	23	141	13.9
All-at-once	C^∞	yes	14	22	55	24.7

converged to a local minimizer that is not optimal then "*lm*" appears in the "Opt. Found?" column. All problem formulations with "yes" in this column, for a given difference scheme, converged to the same solution. The number of optimization iterations, number of objective function evaluations reported by NPSOL and CPU time are indicated in the "No. itrns," "No. fevals," and "Time" columns, respectively. The number of gradient evaluations is approximately the same as the number of function evaluations.

The "No. equiv. Newton steps" column indicates the number of times the optimization method requires a computation that is equivalent to the work of a Newton step on the discretized analysis problem, solve $W(x_F) = 0$. This data is included in Tables I–IV because this work will dominate the computation cost for large problems. Inclusion of equivalent Newton step results is intended to provide a more meaningful basis for performance evaluation than would be obtained by solely considering CPU times on a small problem.

TABLE V

Efficiency Comparison of Versions of the Black-box Method (in Terms of Ratios of Equivalent Newton Steps)

Comparison	$n_D = 2$	$n_D = 10$
Avg. advantage of Bbox (impl grad) over Bbox (fd grad); both using warm-start analyses	1.3	3.9
Avg. advantage of using warm-start analyses in Bbox (fd grad)	2.8	4.7
Avg. advantage of using warm-start analyses in Bbox (impl grad)	1.6	2.1

TABLE VI

Equivalent Newton Steps for Bbox (impl grad) vs All-at-once Method

n_D	Constraints	Avg. ratio of equivalent Newton steps for Bbox (impl grad) over all-at-once method
2	none	21.8
10	none	18.9
10	1st Deriv.	2.2
10	1st and 2nd Deriv.	1.4

The test results in Table I indicate that the black-box scheme with implicit gradients is always more efficient than the black-box scheme with finite difference gradients. In addition, Table I indicates that the all-at-once scheme is less robust than the black-box schemes. As indicated in Section 3.3, the all-at-once scheme is much more susceptible to difficulties due to low-continuity finite difference schemes. However, when the all-at-once method does work, as in the C^∞ case, it is much more efficient than the black-box schemes. This is particularly true in terms of equivalent Newton steps.

Figure 5 shows the optimal solutions for the $n_D = 10$ duct design problem using the C^0, C^1, and C^∞ difference schemes. The $n_D = 10$ case allows enough degrees of freedom for "wavy" ducts to be generated in the optimization process. It is clear that a strangely shaped optimal duct results from the C^∞ scheme that allows a "smearing" of the shock.

To make the optimal duct design more physically reasonable, design constraints were imposed. Figure 6 shows the $n_D = 10$ optimal solutions when first-derivative positivity (monotonicity) constraints are imposed for three discretization schemes. The optimal duct for the C^0 scheme is unaffected by the monotonicity constraint. The monotonicity constraint makes the optimal duct for the C^1 scheme indistinguishable from that for the C^0 scheme. The

TABLE VII

Summary Comparison of Design Problem Formulations

Problem formulation	Robustness	Computational cost	Independence of optimization and analysis codes
Bbox (fd grad)	High	High	High
Bbox (impl grad)	High	Medium	Medium
All-at-once	Low	Low	Low

optimal duct for the C^∞ scheme with monotonicity constraints is considerably smoother than the optimal unconstrained duct, but it is still somewhat ugly.

In addition to the monotonicity constraints, second-derivative constraints were imposed that required the duct curvature to have the same sign as the optimal $n_D = 2$ ducts shown in Fig. 4. Figure 7 shows the $n_D = 10$ optimal solutions when both types of constraints are imposed for the three difference schemes. The second-derivative constraint has little effect on the optimal C^0 and C^1 scheme ducts. However, the new constraint results in an acceptable optimal design for the C^∞ scheme.

Tables II–IV, respectively, give the numerical results for the $n_D = 10$ design problem with no constraints, monotonicity constraints, and both monotonicity and curvature constraints. As in the $n_D = 2$ case, the black-box scheme using implicit gradients is always more efficient than the black-box scheme with finite-difference gradients. The results are summarized in the first row of Table V. The advantage of using the implicit gradient scheme increases as the number of design variables increases. This is to be expected since n_D is the number of analyses required to compute the gradient by one-sided finite differences, whereas n_D is only a secondary factor in the computation cost for the implicit gradient method.

Table V also summarizes the efficiency gained in both black-box methods by using warm-start analyses. Not surprisingly, since finite difference gradients are computed by analyzing slightly modified problems, the black-box method with finite difference gradients obtains more benefit from warm starts than the black-box method with implicit gradients.

As in the $n_D = 2$ case, Tables II–IV indicate that the all-at-once method is less robust than the black-box methods but, when it works, it is usually more efficient. Table VI compares the number of equivalent Newton steps required for the all-at-once method with those required for the most efficient black-box method, on problems where they both computed the optimal design. The all-at-once method has a significant advantage in the unconstrained case. However, adding design constraints reduces this advantage. A partial explanation of this trend is revealed by the discussion of

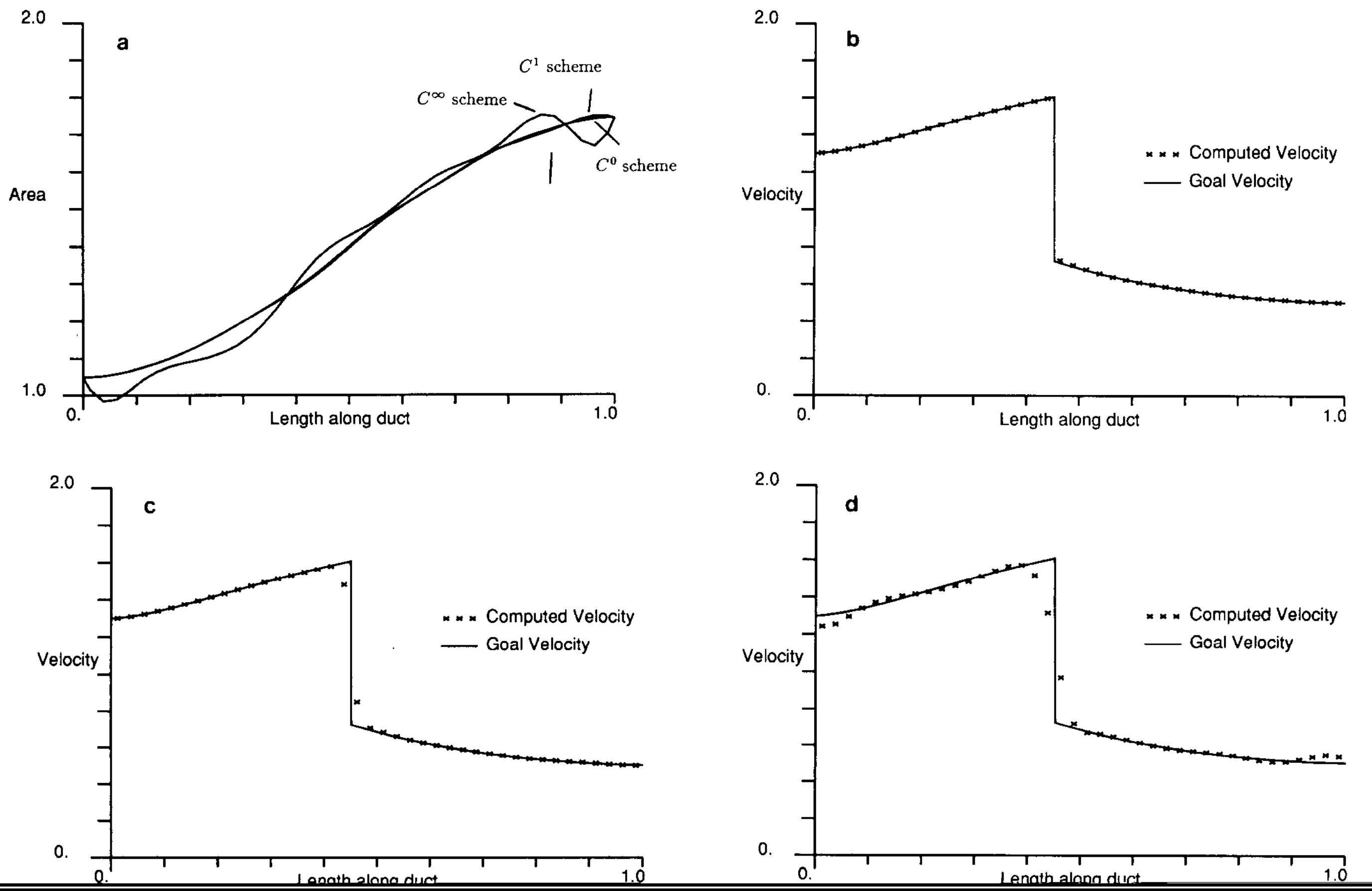

FIG. 5. (a) Area profiles, optimal ducts with 10 design variables, no design constraints. (b) Velocity profile, optimal duct with 10 design variables, no design constraints, using C^0 scheme. (c) Velocity profile, optimal duct with 10 design variables, no design constraints, using C^1 scheme. (d) Velocity profile, optimal duct with 10 design variables, no design constraints, using C^∞ scheme.

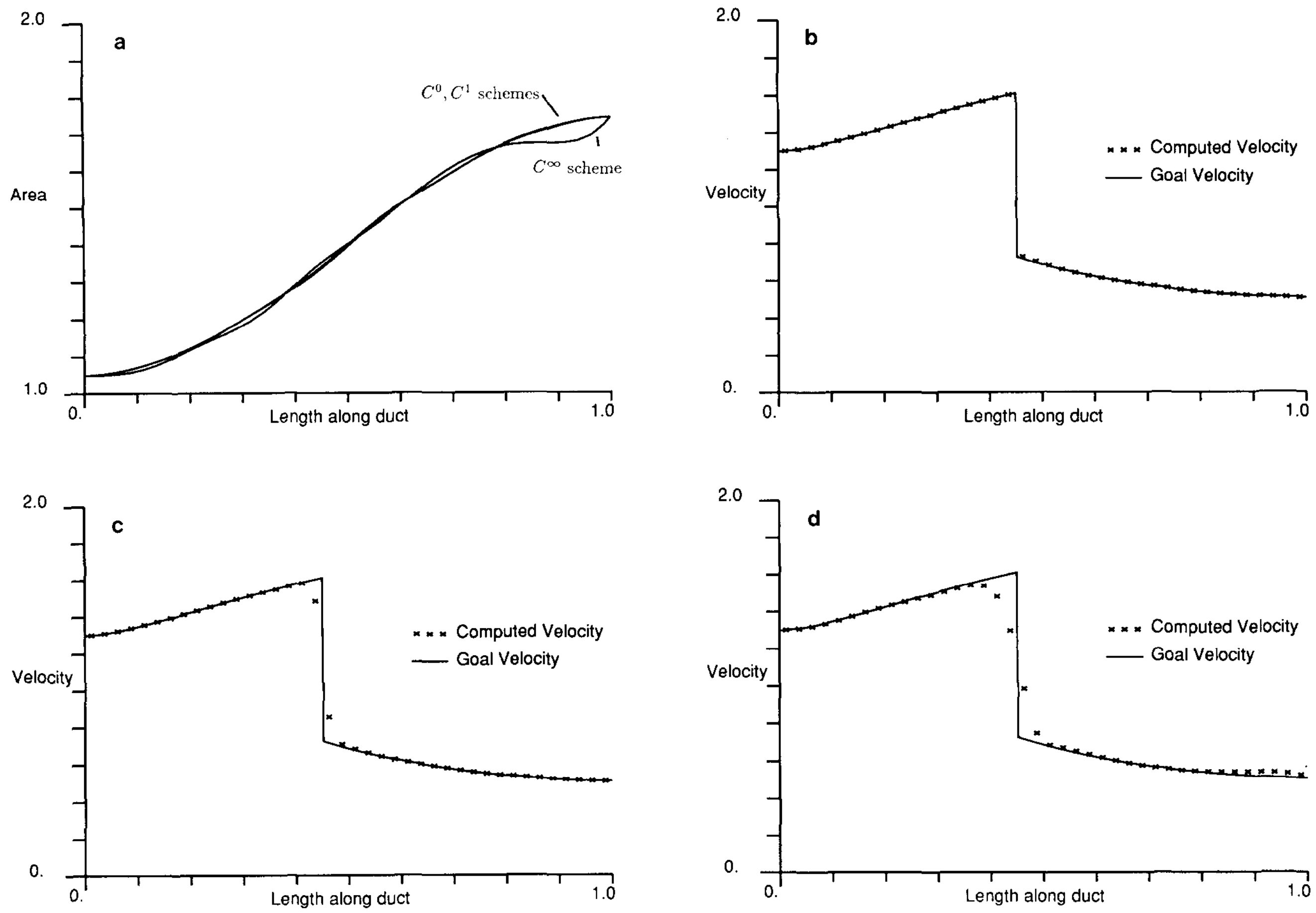

FIG. 6. (a) Area profiles, optimal ducts with 10 design variables, first derivative constraints. (b) Velocity profile, optimal duct with 10 design variables, first derivative constraints, using C^0 scheme. (c) Velocity profile, optimal duct with 10 design variables, first derivative constraints, using C^1 scheme. (d) Velocity profile, optimal duct with 10 design variables, first derivative constraints, using C^∞ scheme.

SQP methods in Section 3.5. That is, the all-at-once method works with a much larger QP than the black-box methods. Thus, when constraint inequalities enter and leave the active set, the all-at-once method must perform linear computations on much larger problems than the black-box methods. Despite this disadvantage, the all-at-once method always required many fewer equivalent Newton steps on C^∞ scheme problems than the black-box methods.

Based on the duct design tests, several summary statements can be made. A general trend is that increasing the continuity of the shock scheme reduces the difficulty in the optimization runs, but increases the degree to which the design must be constrained. Summary observations comparing the three problem formulations are given in Table VII. They are compared based on robustness, computational cost, and the extent to which they allow independence of the optimization and analysis codes. (The two black-box methods tied for first in the robustness category.)

The test results indicate the desirability of improving the robustness of the all-at-once method so that its efficiency advantage can be exploited. One way to do this is to give the all-at-once method a very good initial estimate of the solution for both the flow and design variables. This idea was tested on the $n_D = 10$ design problem, using the C^0 difference scheme, with both monotonicity and curvature design constraints. The initial flow and design variables were taken from the optimal solution computed by the all-at-once method on the C^∞ version of this problem. The all-at-once method then converged to the optimal solution of the C^0 problem at the cost of 143 equivalent Newton steps. The total cost for both the C^∞ initial solution and the final run on the C^0 problem was 198 equivalent Newton steps. This is an improvement over the 315 equivalent Newton steps required by the best black-box method.

Finally, we note that tests (not reported here) indicate that, as suggested by examination of the continuous design problem in Section 2, the spline coefficients obtained in solving the discrete design problem depend continuously on the data $\hat{u}_j$.

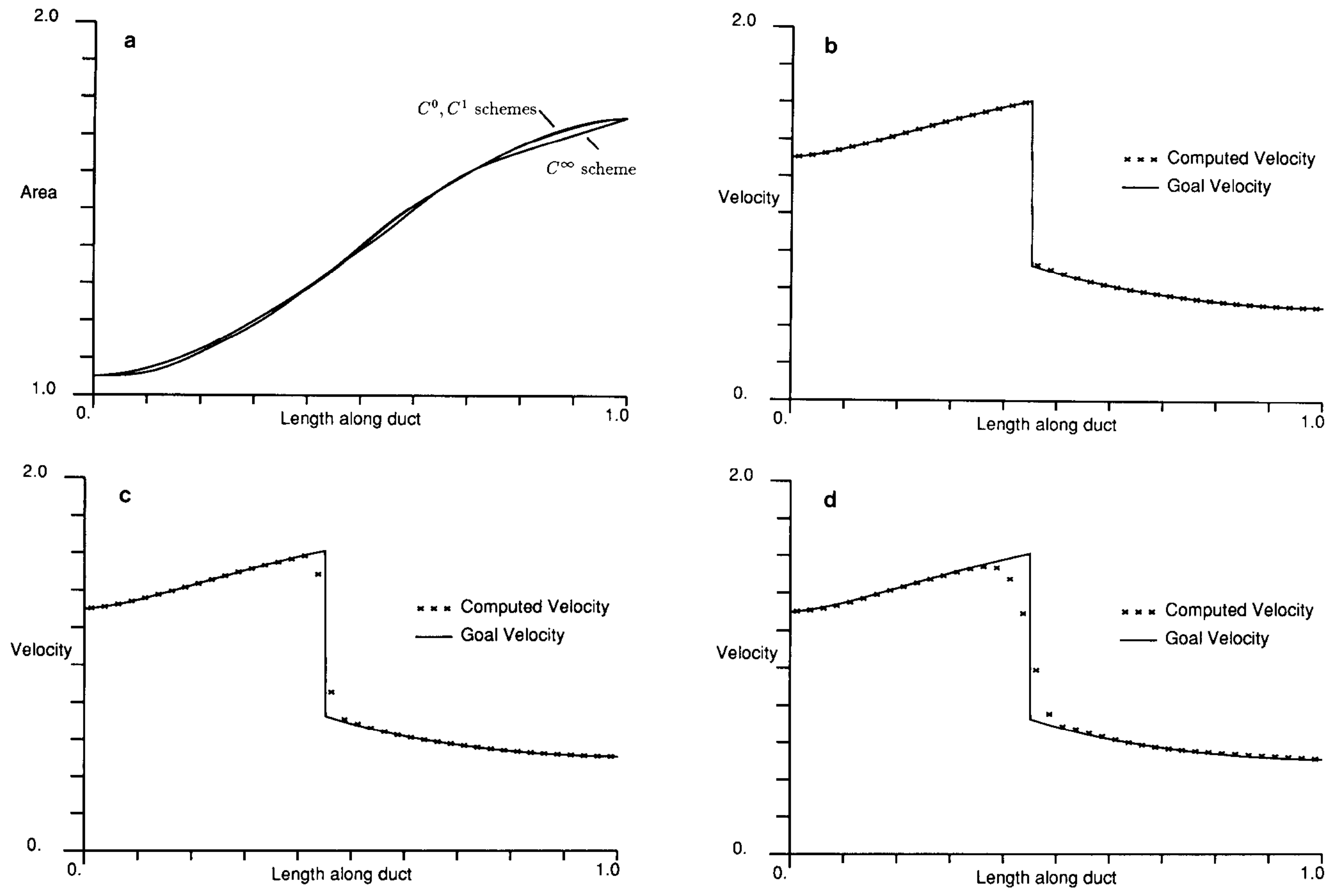

FIG. 7. (a) Area profiles, optimal ducts with 10 design variables, first and second derivative constraints. (b) Velocity profile, optimal duct with 10 design variables, first and second derivative constraints, using C^0 scheme. (c) Velocity profile, optimal duct with 10 design variables, first and second derivative constraints, using C^1 scheme. (d) Velocity profile, optimal duct with 10 design variables, first and second derivative constraints, using C^∞ scheme.

5. CONCLUSIONS

We have presented three methods for formulating design problems as optimization problems. The first is the black-box method where the optimization code is completely separated from the analysis code, and the optimization code repeatedly invokes the analysis code to provide values of the flow variables that are used to evaluate the objective function of the optimization. Most of these invocations of the analysis code are made by the optimization code in order to evaluate finite difference approximations to the gradients of the objective function (and constraints) with respect to the design variables. In general, this is very costly. We therefore presented a modification of the black-box method where these gradients are found by an algorithm based on the implicit function theorem. This black-box method with implicit gradients inherits most of the good properties of the black-box finite-difference gradient method (good robustness, considerable independence of the optimization and analysis codes), while substantially reducing the computational cost.

We also showed that this black-box (implicit gradient) method was equivalent to applying the "variational" or "optimal control" approach to design optimization directly to the discretized analysis problem, rather than to the un-discretized (continuous) problem as is usually done. A further analysis of this relationship will be presented in a future paper.

The black-box method with implicit gradients can be retrofitted to most existing analysis codes to turn them into design codes. The amount of work required depends on the solution methodology employed in the analysis code. The largest task is to solve linear systems with a coefficient matrix that is the transpose of the Jacobian of the discretized flow equations with respect to the flow variables. In many cases (primarily in schemes based on Newton's method), this Jacobian is already computed by the analysis code. In other cases, it can readily be obtained by sparse differencing. In all cases, a solution method for the transpose of the Jacobian needs to be provided. While this is trivial if a direct factorization of the Jacobian is employed in the analysis code, such will rarely be the case for large scale (three-dimensional) problems. It remains to be determined how iterative methods can best be adapted to solve transposed systems.

The other method we introduced was the all-at-once method where the optimization simultaneously varies the flow and design variables, and the discretized flow equations are viewed as equality constraints on the optimization. The primary difference between the all-at-once approach and the black-box approach is that the discrete flow equations are not required to be satisfied in the optimization iteration until the optimal solution is reached. Obviously, the optimization methodology and the flow equation solution methodology need to be tightly integrated in this approach, so that code independence is sacrificed. We found in our tests on the model duct flow problem that the all-at-once approach was less robust than the black-box approach, often failing to converge or converging to an undesirable local minimum. This was especially true when difference schemes of low continuity (those giving the sharpest shocks) were employed. However, when the all-at-once approach succeeded, it was dramatically less expensive than the other approaches. Since expense is a key issue for large problems, further investigation of how to increase the robustness of the all-at-once method seems justified.

APPENDIX A: FLUXES FOR THE DIFFERENCE SCHEMES

For completeness, we give here the formulas for the fluxes appearing in the Godunov, Engquist–Osher, and artificial viscosity schemes:

Godunov.

$$f^G_{j+1/2} = \begin{cases} f_{j+1}, & \text{if } u_j, u_{j+1} < u_*; \\ f_j, & \text{if } u_j, u_{j+1} > u_*; \\ f_*, & \text{if } u_j < u_* < u_{j+1}; \\ \max(f_j, f_{j+1}), & \text{if } u_{j+1} < u_* < u_j. \end{cases}$$

Here, f_{j+1} means $f(u_{j+1})$, etc., and f_* means $f(u_*)$.

Engquist–Osher. $f^{\mathrm{EO}} = f^{\mathrm{G}}$, with the last line replaced by $f_j + f_{j+1} - f_*$.

Artificial viscosity. $f^{\mathrm{AV}}_{j+1/2} = 1/2(f_{j+1} + f_j - \alpha(u_{j+1} - u_j))$. We took $\alpha = 1$ in the tests reported here.

ACKNOWLEDGMENT

The authors gratefully acknowledge John T. Betts for several helpful discussions and for suggesting that they pursue the all-at-once approach.

REFERENCES

1. J. W. Slooff, in *Proceedings of International Conference on Inverse Design Concepts in Engineering Sciences, University of Texas at Austin, 1984*, edited by G. S. Dulikravich, p. 1.
2. M. D. Salas, A. Jameson, and R. E. Melnik, in *AIAA Computational Fluid Dynamics Conference, Danvers, MA, 1983*, p. 48.
3. G. R. Shubin, A. B. Stephens, and H. M. Glaz, *J. Comput. Phys.* **39**, 364 (1981).
4. A. B. Stephens and G. R. Shubin, *J. Comput. Phys.* **55**, 175 (1984).
5. P. Embid, J. Goodman, and A. Majda, *SIAM J. Sci. Stat. Comput.* **5**, 21 (1984).
6. C. de Boor, *A Practical Guide to Splines* (Springer-Verlag, New York, 1985).
7. J. Lorenz, *Math. Comput.* **46**, 45 (1986).
8. A. N. Tikhonov and V. Y. Arsenin, *Solutions of Ill-Posed Problems* (Winston, Washington, DC, 1977).
9. J. E. Dennis and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations* (Prentice-Hall, Englewood Cliffs, NJ, 1983).
10. P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization* (Academic Press, New York, 1982).
11. R. G. Voigt, "Requirements for Multidisciplinary Design of Aerospace Vehicles on High Performance Computers," ICASE Report No. 89-70, 1989 (unpublished).
12. T. M. Apostol, *Mathematical Analysis* (Addison–Wesley, Reading, MA, 1960).
13. A. R. Curtis, M. J. D. Powell, and J. K. Reid, *J. Inst. Math. Appl.* **13**, 117 (1974).
14. M. H. Rizk, "Aerodynamic Optimization by Simultaneously Updating Flow Variables and Design Parameters," AGARD Paper No. 15, May 1989 (unpublished).
15. A. E. Bryson, Jr. and Y.-C. Ho, *Applied Optimal Control* (Hemisphere, Washington, DC, 1975).
16. W. H. Chen, G. R. Gavalas, J. H. Seinfeld, and W. L. Wasserman, *Soc. Pet. Eng. J.* **14** 593 (1974).
17. C. Chavent, M. Dupuy, and P. Lemonnier, *Soc. Pet. Eng. J.* **15**, 74 (1975).
18. A. Miele, *Theory of Optimum Aerodynamic Shapes* (Academic Press, New York, 1965).
19. A. Jameson, "Aerodynamic Design via Control Theory," ICASE Report No. 88-64, 1988 (unpublished).
20. W. L. Wasserman, A. S. Emanuel, and J. H. Seinfeld, *Soc. Petr. Eng. J.* **15**, 347 (1975).