JOURNAL OF AIRCRAFT
Vol. 36, No. 1, January–February 1999

Constrained Multipoint Aerodynamic Shape Optimization Using an Adjoint Formulation and Parallel Computers, Part 1

James J. Reuther*
NASA Ames Research Center, Moffett Field, California 94035
Antony Jameson† and Juan J. Alonso‡
Stanford University, Stanford, California 94305
and
Mark J. Rimlinger§ and David Saunders§
NASA Ames Research Center, Moffett Field, California 94035

This paper is the first of a two-paper series (Reuther, J. J., Jameson, A., Alonso, J. J., Rimlinger, M. J., and Saunders, D., "Constrained Multipoint Aerodynamic Shape Optimization Using an Adjoint Formulation and Parallel Computers, Part 2," *Journal of Aircraft*, Vol. 36, No. 1, 1999, pp. 61–74), that describes a shape optimization method for aerodynamic design problems involving complex aircraft configurations and multiple design points that are subject to geometric constraints. The aerodynamic performance is evaluated using a set of high-fidelity governing equations discretized on body-conforming multiblock meshes. The design process is greatly accelerated through the use of an adjoint method for the calculation of sensitivity information and by means of a parallel implementation for distributed memory computers. This paper focuses on the details of the development and implementation of the complete design algorithm. A general adjoint formulation of the design optimization problem is presented along with a detailed derivation of the adjoint system for the Euler equations. The multiblock approach for the flow and adjoint solution algorithm and the mesh perturbation procedure are described. The extension of this approach to treat multipoint and constrained shape optimization of complete aircraft configurations is also discussed. Finally, the details of the parallel implementation are examined.

Nomenclature

A_i = Cartesian flow Jacobian matrix in the ith direction
C_A, C_N = coefficients of axial and normal force
C_D, C_L = coefficients of drag and lift
C_i = contravariant flow Jacobian matrix in the ith direction
C_p = pressure coefficient
c = chord length
E = total energy
F = boundary shape
F_i = contravariant inviscid fluxes
f_i = Cartesian inviscid fluxes
$\boldsymbol{G}$ = gradient vector
H = total enthalpy
I = cost function
J = det(K)
K_{ij} = mesh transformation Jacobian matrix components
M_∞ = freestream Mach number
p = static pressure
p_∞ = freestream static pressure
$Q_{i,j}$ = velocity transformation matrix
R = governing equations, residual
S = surface area
S_x, S_y, S_z = projected surface areas in Cartesian coordinate directions
t = time
U_i = contravariant velocity components
u_i = Cartesian velocity components
W = conservative contravariant flow variables
w = conservative Cartesian flow variables
x_i = Cartesian coordinates
α = angle of attack
γ = ratio of specific heats, C_p/C_v
δ = first variation
δ_{ij} = Kronecker delta function
λ_i = cost function weights for the ith flow condition
ξ_i = computational coordinates
ρ = fluid density
ψ = Lagrange multiplier, costate, or adjoint variable

Introduction

COMPUTATIONAL fluid dynamics (CFD) methods have traditionally been used to calculate the aerodynamic performance of existing aircraft configurations. However, much larger payoffs can be derived from the use of CFD methods as aerodynamic design tools. Yet, despite the fact that flow analysis has matured to the extent that Navier–Stokes solutions are routinely computed for very complex three-dimensional configurations, automated design methods that incorporate high-fidelity CFD modeling have only recently seen limited use for practical three-dimensional design problems.[1–7] The main obstacle preventing the use of CFD tools as a basis for aerodynamic design is the fact that their computational cost can be prohibitive for three-dimensional problems. With this issue in mind, our group has proposed a general strategy that uses the theory of control of partial differential equations[8] to devise computationally efficient adjoint-based design methods.

Received April 8, 1998; revision received June 15, 1998; accepted for publication June 22, 1998.

*Research Scientist, Research Institute for Advanced Computer Science, M/S 227-6. Member AIAA.
†T. V. Jones Professor of Engineering, Department of Aeronautics and Astronautics. Fellow AIAA.
‡Assistant Professor, Department of Aeronautics and Astronautics. Member AIAA.
§Research Scientist, Sterling Software, M/S 227-6.

Before examining our approach to CFD-based aerodynamic shape optimization, a brief background of previous methods is warranted. The simplest way of extending existing CFD analysis methods for the treatment of design problems is by direct coupling with a numerical optimization algorithm.[2,9–11] The essence of these design methods, which incur a heavy computational expense, is quite simple: a numerical optimization procedure is used to extremize a chosen aerodynamic cost function that is evaluated by the CFD code under consideration. The configuration is modified by the optimization algorithm through a set of user-specified design variables in an iterative fashion until a minimum is found.

In aerodynamic design applications, where the design space is often smooth, numerical optimization algorithms that use both the value of the cost function and its gradient usually have performance advantages over methods that rely on function evaluations alone. The simplest approach to obtain the necessary gradient information is the finite difference method. In this method, the components of the gradient are estimated by taking the differences in the values of the objective function resulting from independent perturbations to each of the design variables. This gradient information is then used by the numerical optimization algorithm to determine a direction along which an improvement should exist. After finding the minimum of the objective function along this search direction, the process is repeated until no further improvements to the design can be obtained. While the finite difference approach is relatively straightforward, it has one substantial drawback: for the design of complex configurations requiring hundreds or even thousands of design variables and multiple design points, the approach incurs an unaffordably high computational cost. Nevertheless, the finite difference approach can, in some cases, be considered an attractive option for aerodynamic design problems because alternative optimization strategies that do not rely on the use of gradient information, e.g., grid searches, genetic algorithms,[12] and simulated annealing, often require an even greater number of function evaluations.

An alternative strategy for aerodynamic design problems is the inverse design approach. Inverse methods attempt to achieve a prescribed target distribution using iterative modifications to the shape of the boundary. Their computational costs are proportional to that of a single flow analysis and, hence, they are far less expensive than finite difference-based optimization methods. Unfortunately, they require the specification of a target distribution which, in general, cannot be guaranteed to produce improvements in the design and may, in fact, prove to be unrealizable. In contrast, design methods that use a numerical optimization procedure are considerably more versatile since they permit a variety of aerodynamic figures of merit to serve as the objective function.

The use of numerical optimization for transonic aerodynamic shape design was pioneered by Hicks et al.,[9] who applied their method to the design of two-dimensional profiles using the potential flow equation. Their method was quickly extended to wing design by Hicks and Henne,[10] and later, through the work of Reuther et al.,[2] the approach was successfully extended to the design of supersonic wing–body transport configurations.[2] However, because all of these methods used the finite difference approach in conjunction with a serial implementation of the algorithm, they were limited to the very simple configurations and modest numbers of design variables owing to their lengthy computational run times. Despite the well-known importance of achieving a highly tuned nacelle/airframe integration for optimum aerodynamic performance, the designs included only the fuselage and wing.[2] The direct presence of the nacelles was omitted from the design process because the number of mesh points required for such a configuration is roughly double.

Recently, through the work of our group and that of other researchers, alternative, less-expensive methods for obtaining the desired gradients have been developed. Because a large part of the computational work in the finite difference optimization approach was spent in the calculation of the sensitivity information, these newly developed methods can greatly reduce the computational cost of design algorithms. In the most promising of these new methods, the adjoint approach, the gradient information at a single design point with respect to an arbitrary number of design variables can be obtained with the equivalent of two flow calculations (one flow or state solution and one adjoint or costate solution), instead of the $n + 1$ flow calculations (n being the number of design variables) that are required when using finite differences. Another advantage of the adjoint approach is that a high convergence level for the flow and adjoint systems is not required to obtain useful gradient information. This is in significant contrast to the highly converged flow solutions that are crucial to the accuracy of finite difference gradients.[13]

For the case of multiple design points at different flight conditions, the adjoint approach requires separate flow and adjoint solutions for each of the design points, yielding $2m$ equivalent flow calculations (where m is the number of design points). In contrast, the finite difference method requires $m(n + 1)$ flow solutions. Thus, for large numbers of design variables, and particularly for multiple design points, the finite difference method can quickly saturate the available computing resources, thereby making the use of the adjoint approach more appealing.

Despite large reductions in computational cost provided by an adjoint formulation, the aerodynamic optimization of complete configurations remains a formidable challenge. This statement is especially true with regard to the eventual inclusion of viscous effects or when viewed within the context of larger multidisciplinary design optimization (MDO) problems. The advent of reliable and efficient multiprocessor computers with various forms of distributed memory has enabling technology to permit the types of calculations presented in this work. This new class of computers has provided acceptable and cost-efficient turnaround for design calculations of full configurations subject to a variety of linear and nonlinear constraints at multiple flight conditions.

Our work has demonstrated that the combination of accurate and efficient flow analysis procedures and adjoint-based sensitivity analysis can result in practical design methods for the airfoils, isolated wings, wing–bodies, and complex configurations.[4,6,14,15] The intention of this work is to provide a comprehensive description of a design algorithm that takes into consideration the treatment of both geometric constraints and multiple design points. The method satisfies the fundamental requirements of reliable solution accuracy, acceptable computational cost, complex geometry treatment, and rapid design turnaround. This effort represents a substantial step toward the development of MDO methods that incorporate high-fidelity modeling of the physics within each discipline.

Formulation of the Adjoint Equations

The aerodynamic cost function at a given design point is a function of the flowfield variables and the physical location of the boundary of the geometry in question. Then

$$I = I(w, F)$$

and a change in F results in a change

$$\delta I = \frac{\partial I^T}{\partial w}\,\delta w + \frac{\partial I^T}{\partial F}\,\delta F \tag{1}$$

in the cost function. The governing equation R and its first variation express the dependence of w and F within the flowfield domain D

$$R(w, F) = 0, \qquad \delta R = \left(\frac{\partial R}{\partial w}\right)\delta w + \left(\frac{\partial R}{\partial F}\right)\delta F = 0 \tag{2}$$

Next, introducing a Lagrange multiplier ψ, we have

$$\delta I = \frac{\partial I^T}{\partial w}\delta w + \frac{\partial I^T}{\partial F}\delta F - \psi^T\left[\left(\frac{\partial R}{\partial w}\right)\delta w + \left(\frac{\partial R}{\partial F}\right)\delta F\right]$$
$$= \left[\frac{\partial I^T}{\partial w} - \psi^T\left(\frac{\partial R}{\partial w}\right)\right]\delta w + \left[\frac{\partial I^T}{\partial F} - \psi^T\left(\frac{\partial R}{\partial F}\right)\right]\delta F \quad (3)$$

Choosing ψ to satisfy the adjoint or costate equation

$$\left(\frac{\partial R}{\partial w}\right)^T \psi = \frac{\partial I}{\partial w} \quad (4)$$

the first term in Eq. (3) is eliminated, and we find that the desired gradient is given by

$$\boldsymbol{G}^T = \frac{\partial I^T}{\partial F} - \psi^T\left(\frac{\partial R}{\partial F}\right) \quad (5)$$

Because Eq. (5) is independent of δw, the gradient of I with respect to an arbitrary number of design variables can be determined without the need for additional flowfield evaluations. The main cost incurred is in solving the adjoint Eq. (4). In general, the complexity of the adjoint problem is similar to that of the flow solution. If the number of design variables is large, it becomes compelling to take advantage of the cost differential between one adjoint solution and the large number of flowfield evaluations required to determine the gradient using finite differences. To treat the multipoint design problem, a new composite cost function is developed as a weighted sum of the cost functions at each independent design point

$$I = \sum_{i=1}^{N} \lambda_i I_i$$

where the λ_i coefficients are the weights on the cost functions I_i at each of the design points. The composite gradient is obtained using a linear combination of the gradients calculated at each of the design points appropriately weighted by λ_i.

Issues of Importance for Design Problems

The development of aerodynamic design procedures that employ an adjoint formulation is currently being investigated by several groups of researchers. References 1, 7, 8, and 16–31 represent a partial list of recent work in this developing field by authors outside of our group, whereas Refs. 4, 6, 14, 15, and 32–43 provide a list of our own work in this area. However, as is the case in any new research field, many questions remain unanswered. The most salient issues of concern are the following: 1) use of discrete vs continuous adjoint formulations, 2) parameterization of the design space, 3) choice of optimization procedures, 4) level of coupling between design and analysis, and 5) treatment of geometric and aerodynamic constraints.

The first set of adjoint equations for transonic flow problems was derived and implemented by Jameson[33] and the approach was later followed by Lewis and Agarwal.[28] This adjoint system was obtained directly from the analytic governing equations and is thus termed a continuous adjoint. The adjoint differential equations with the appropriate boundary conditions may then be discretized and solved in a manner similar to that used in the flow solution algorithm. One may alternatively derive a set of discrete adjoint equations directly from the discrete approximation to the flow equations by following the procedure outlined in Eqs. (1–5). The resulting discrete adjoint equations can be viewed as one of the possible discretizations of the continuous adjoint system. This alternative is mentioned by Jameson,[34] but was not adopted in that work because of the complexity of the resulting discrete adjoint system. The approach, however, has been favored by Refs. 16–19, 20–22, and more recently by Ref. 7.

Both alternatives have advantages. The continuous approach provides the researcher with the ability to understand the meaning and behavior of the adjoint system and its related boundary conditions, whereas the discrete adjoint is simply a large system of coupled equations. On the other hand, the discrete approach maintains consistency between the flow solution and the gradient information and, if properly implemented, will give gradients that exactly match those obtained through finite differences. The continuous formulation produces gradients that are slightly inconsistent with those obtained by finite differences because of differences in the discretization between the state and costate systems. However, these slight inconsistencies are of the order of the truncation error in the solution and, therefore, must vanish in the limit of zero mesh *width*.

The discrete adjoint equations form a linear system of equations, whether derived directly by the discrete approach or by discretization of the continuous adjoint equation. The size and complexity of the resulting linear system is such that the use of a direct solution method is unrealistic except for two-dimensional problems and the smallest three-dimensional cases. Application of the continuous sensitivity analysis fosters a straightforward recycling of the flow solution algorithm for the solution of the adjoint equations because the methods applied to the original governing differential equations can be duplicated for the adjoint differential equations. When the discrete approach is used, the adjoint equations attain a complexity that makes them cumbersome to solve in terms of both storage and computational cost.[7,44] Moreover, the discrete method is subject to the difficulty that the discrete flow equations often contain nonlinear flux limiting functions that are not differentiable.

Resolution of items 3 and 4 in the previous list strongly hinges on the choice for item 2. In his original scheme, Jameson used the location of each surface mesh point as a distinct design variable.[33,34,45] In three-dimensional wing design cases this led to as many as 4224 design variables.[45] If this approach were followed for the design of a complete aircraft configuration and all of the points on the surface were allowed to move, the number of design variables would be in the tens of thousands. Such large numbers of design variables preclude the use of descent algorithms such as Newton or quasi-Newton approaches because of the high cost of matrix inversion/factorization operations. Therefore, Jameson[32] used a steepest descent method based on a suitably smoothed gradient. This method has the advantage that significant errors can be tolerated in the gradient evaluation during the early steps of the design. Thus, the approach favors tighter coupling of the flow solver, adjoint solver, and optimization procedure to accelerate the convergence of the overall design process. Ta'asan et al.[23] and Kuruvila et al.[24] have taken advantage of this situation by formulating the design problem as a *one shot* procedure, where all three systems are advanced simultaneously. The use of every surface grid point as an independent design variable also has the disadvantage of introducing high-frequency noise into the gradient, can cause simple descent methods to fail. In his original work, Jameson[34,46] addressed this poor conditioning by smoothing the corrections to the surface shape.

References 2 and 9–11 have parameterized the design space using sets of smooth functions that perturb the initial geometry. Using such a parameterization it is possible to work with considerably fewer design variables and eliminate the need for smoothing of the shape change corrections to filter out the high-frequency components. Note that because these pioneering efforts relied on finite difference gradients that required highly converged flow solutions to obtain even reasonable gradient accuracy, it was not possible to construct a one shot type design method. Nevertheless, the combination of a modest number of design variables and the reliance on highly con-

verged flow solutions allowed the use of quasi-Newton search strategies that improved the efficiency of the design process.

One simple choice of design variables for airfoils suggested by Hicks and Henne[10] has the following *sine bump* form:

$$b(x) = \{\sin[\pi x^{\log(1/2)/\log(t_1)}]\}^{t_2}, \qquad 0 \le x \le 1 \tag{6}$$

In this function, t_1 locates the maximum of the bump in the range $0 < x < 1$ at $x = t_1$, because the maximum occurs when $x^{\alpha} = \frac{1}{2}$, where $\alpha = \log \frac{1}{2}/\log t_1$, or $\alpha \log t_1 = \log \frac{1}{2}$. The parameter t_2 controls the width of the bump. These functions do not form a complete basis of the space of possible geometric variations; however, they have provided a versatile way of obtaining a reasonable parameterization of the design space. When skillfully distributed over the entire configuration, such analytic perturbation functions span a substantial design space. Furthermore, particular choices of these variables will concentrate the design effort in regions where refinement is needed while leaving the rest of the geometry virtually undisturbed. The disadvantage of these functions is that they are not orthogonal, and there is no simple way to form a basis for the design space with continuous functions that vanish at $x = 0$ and 1. For example, they do not guarantee that a solution of the inverse problem for a realizable target pressure distribution will be attained. Nevertheless, these functions have proven to be quite effective in achieving design improvements using limited numbers of design variables. Furthermore, their use in the design process may be accelerated either by tightly coupling the individual elements of the design process, as was the case when using the mesh points themselves, or through the use of higher-order optimization methods. Finally, they have the advantage that there is no need to smooth the resulting geometry during the design process because, by construction, higher frequencies are not admitted.

Design variables may also be chosen as the locations of the control points of B-spline curves that define the geometry.[1,18,27,47] Like the Hicks–Henne functions, B-splines allow for a greatly reduced number of design variables, thus permitting the use of higher-order optimization methods such as quasi-Newton procedures. Furthermore, local control is also possible by choosing only a limited number of active control points as design variables. This method in practice seems to have an advantage over the Hicks–Henne functions in that a more complete space of designs is spanned for a given number of design variables. Moreover, because B-spline curves and surfaces are now the almost universal choice in CAD environments, they provide a straightforward way of interfacing CAD and aerodynamic design. Despite these advantages, preliminary work[35] has shown that, in contrast to Hicks–Henne functions, the B-spline functions admit high-frequency noise in both the representation of the geometry and the calculated gradient. However, as explored by Lorentzen,[47] this difficulty may be overcome by a smoothing procedure similar to the one used in Jameson's original work.

The last item (5) in the previous list, the treatment of geometric and aerodynamic constraints, will also depend upon the choice of item 2. In the particular formulation presented here, a detailed description of constraint enforcement is discussed later in this paper as well as in Ref. 56.

Multiblock Flow Solution

In our most recent papers,[6,39,42,43] the Euler adjoint-based design formulation was extended to treat complete aircraft configurations using a new parallel multiblock implementation. This extension of the method required the replacement of our single-block flow and adjoint solvers[4,36,45] with their multiblock counterparts. Thus, during the development of the new multiblock flow solver, considerable care was taken to achieve the characteristics needed, not just for its use as an analysis tool, but also for its integration into design applications.

To use CFD in an automated design environment, the flow solver must meet fundamental requirements of accuracy, efficiency, robustness, and fast convergence. High accuracy is required because the improvements predicted by the method for the design in question can only be as good as the accuracy of the flow analysis. Efficiency of the flow solver is also critical because the optimization of the design will generally require the computation of many flow and adjoint solutions. The robustness of the method, i.e., its ability to obtain a flow solution for a variety of configuration shapes and flow conditions, is of great importance to guarantee the continuity of the design process. The last aspect, rapid convergence, is also of significant importance; in highly refined aerodynamic design applications, the benefit of aerodynamic optimization lies in obtaining the last few percentage points in aerodynamic efficiency. In such cases, the solutions must be highly converged such that the noise in the figure of merit is well below the level of realizable improvement.

In our three-dimensional single-block applications, Jameson's FLO87 code easily met all of the previously mentioned criteria. FLO87 achieves fast convergence with the aid of multigridding and implicit residual smoothing, typically obtaining solutions that converge to machine accuracy. The challenge addressed in Ref. 6 was to meet these strict convergence requirements within the framework of a cell-centered multiblock flow solver. Thus, the general strategy employed in developing the multiblock scheme consisted of using the same numerical algorithms used in the single-block code. This goal can be achieved by constructing and maintaining a halo of cells surrounding each block in the mesh. Then, updates to the internal cells within each block can be performed independently using single-block techniques. This strategy requires the identification of the halo cells and their corresponding *donor* cells (from which they inherit their values) in the interior of the neighboring blocks. The values of these halo cells need to be loaded with appropriate flowfield data at regular intervals in the main solution algorithm. A double halo is required so that the flowfield values of the complete stencil necessary to calculate the fluxes of all the internal cells in a block are available, even at the block boundaries.

The values of the flow variables stored correspond to the *centers* of the cells in the mesh. For conservation reasons, the fluxes (both convective and dissipative) are computed at the cell *faces* using the average of the values stored at the centers of the two adjoining cells. Therefore, the presence of a single-level halo for each of the blocks in the multiblock mesh is required for the calculation of the convective fluxes. The dissipative fluxes are composed of a blend of first- and third-order differences corresponding to terms that mimic second and fourth derivatives of the flow quantities.[48] To calculate the third-order differences, a second layer of halo cells is required at each block interface. Halo cells on the external boundaries of the entire computational domain are constructed and updated by suitable extrapolation or reflection, depending on whether the boundary condition applied corresponds to a solid wall or a far field. Coarse grids in the multigrid sequence are computed in the usual fashion, by coalescing groups of eight cells and repeating the preceding halo cell process. Once the halo configuration is set up for each block, standard methods for the spatial discretization and time integration (including artificial dissipation, implicit residual averaging, and multigridding) are employed to compute the flow solution within each individual block.

The presentation of the governing system of equations and the solution strategy follows that of many earlier works.[48–50] The three-dimensional Euler equations may be written as

$$\frac{\partial w}{\partial t} + \frac{\partial f_i}{\partial x_i} = 0 \quad \text{in} \quad D \tag{7}$$

where it is convenient to denote the Cartesian coordinates and velocity components by x_1, x_2, x_3 and u_1, u_2, u_3, and w and f_i are defined as

$$w = \begin{bmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ \rho u_3 \\ \rho E \end{bmatrix}, \qquad f_i = \begin{bmatrix} \rho u_i \\ \rho u_i u_1 + p\delta_{i1} \\ \rho u_i u_2 + p\delta_{i2} \\ \rho u_i u_3 + p\delta_{i3} \\ \rho u_i H \end{bmatrix} \tag{8}$$

Also

$$p = (\gamma - 1)\rho[E - \tfrac{1}{2}(u_i^2)] \tag{9}$$

$$\rho H = \rho E + p \tag{10}$$

Consider a transformation to computational coordinates ξ_1, ξ_2, ξ_3, where

$$K_{ij} = \left(\frac{\partial x_i}{\partial \xi_j}\right), \qquad J = \det(K), \qquad K_{ij}^{-1} = \left(\frac{\partial \xi_i}{\partial x_j}\right)$$

Introducing scaled contravariant velocity components

$$U_i = Q_{ij} u_j$$

where

$$Q = JK^{-1}$$

the Euler equations can now be written as

$$\frac{\partial W}{\partial t} + \frac{\partial F_i}{\partial \xi_i} = 0 \quad \text{in} \quad D \tag{11}$$

with

$$W = J \begin{bmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ \rho u_3 \\ \rho E \end{bmatrix}, \qquad F_i = Q_{ij} f_j = \begin{bmatrix} \rho U_i \\ \rho U_i u_1 + Q_{i1} p \\ \rho U_i u_2 + Q_{i2} p \\ \rho U_i u_3 + Q_{i3} p \\ \rho U_i H \end{bmatrix} \tag{12}$$

For multiblock meshes, the preceding notation applies to each block in turn. The flow is thus determined as the steady-state solution to Eq. (11) in all blocks subject to the flow tangency condition on all solid boundary faces

$$U_\eta = 0 \quad \text{on all} \quad B_S \tag{13}$$

where η takes the values 1, 2, or 3, depending on the direction that is normal to the face B_S, where a solid surface exists. At the far-field boundary faces, B_F, freestream conditions are specified for incoming waves, whereas outgoing waves are determined by the solution in the interior of the block.

The time-integration scheme follows that used in the single-block code.[48] The solution proceeds by calculating the cell fluxes, smoothing the residuals, and updating the flow variables at each stage of the time-stepping scheme and on each level of the multigrid cycle. The main difference in the integration strategy lies in the need to loop over all blocks during each stage of the Runge–Kutta scheme. The use of the double halo around each block permits standard single-block subroutines to be used, without modification, for the computation of the flowfield within each individual block.

The implementation of the implicit residual smoothing procedure is also different from that in the single-block code. In the single-block flow solver, tridiagonal systems of equations are solved using flow information from the entire grid. Thus, each residual is replaced by a weighted average of itself and the residuals of the entire grid. In the multiblock strategy, the stencil for the residual smoothing is restricted by the boundaries of each of the blocks in the mesh. This approach is taken to eliminate the need to solve scalar tridiagonal systems spanning the blocks, which would incur a penalty in communication cost. This change has no effect on the final solution, and in the present implementation it has not led to any deterioration in the rate of convergence.

Adjoint Formulation for the Euler Equations

The application of control theory to aerodynamic design problems is illustrated by the case of three-dimensional design using the Euler equations as the mathematical model for transonic flow. In our previous work,[14,37,45] the illustrative problem most often described used a cost function that reflected the difference between the existing pressure distribution and some specified target. In the case of transonic flow over conventional commercial transport wings, this aerodynamic figure of merit proves to be very effective because the tailoring of pressure distributions to achieve close to optimum performance is well understood by most experienced aerodynamicists. However, in the case of either supersonic design of three dimensional configurations or designs that involve complicated geometries, the specification of pressure distributions that will determine near-optimum performance is considerably more challenging. For this reason, from the outset of our development of the adjoint formulation, numerous cost functions and their combinations have been provided. In this paper, for illustrative purposes, we will use the coefficient of drag at a fixed lift as the cost function

$$\begin{aligned} I &= C_D \\ &= C_A \cos\alpha + C_N \sin\alpha \\ &= \frac{1}{S_{\text{ref}}} \int \int_{B_S} C_p (S_x \cos\alpha + S_y \sin\alpha)\, d\xi_1\, d\xi_2 \end{aligned}$$

where S_x and S_y define projected surface areas, S_{ref} is the reference area, and $d\xi_1$ and $d\xi_2$ are the two coordinate indices that lie in the plane of the face in question. Note that the integral in the preceding expression is carried out over all solid boundary faces in the multiblock mesh. The design problem is now treated as a control problem where the control function is the geometry shape, which is chosen to minimize I subject to the constraints defined by flow Eqs. (7–12). A variation in the shape will cause a variation δp in the pressure and, consequently, a variation in the cost function

$$\delta I = \tilde{\delta} C_D + \frac{\partial C_D}{\partial \alpha}\, \delta\alpha$$

where $\tilde{\delta} C_D$ is the variation in the cost function (coefficient of drag) as a result of changes in the design parameters with α fixed, whereas the second term describes the variation of the cost function caused by changes in α. To treat the more interesting problem of practical design, drag must be minimized at a fixed lift coefficient. Thus, an additional constraint is given by

$$\delta C_L = 0$$

or

$$\tilde{\delta} C_L + \frac{\partial C_L}{\partial \alpha}\, \delta\alpha = 0$$

Combining these two expressions to eliminate $\delta\alpha$ gives

$$\delta I = \bar{\delta} C_D - \frac{\left(\frac{\partial C_D}{\partial \alpha}\right)}{\left(\frac{\partial C_L}{\partial \alpha}\right)} \bar{\delta} C_L \tag{14}$$

which requires the estimation of slopes in the lift and drag curves. Because p depends on w through the equation of state [Eqs. (9) and (10)], the variation δp can be determined from the variation δw.

It is important to point out that if a fixed computational domain is used, the variations in the shape result in variations in the mapping derivatives Q_{ij}. Define the Jacobian matrices

$$A_i = \frac{\partial f_i}{\partial w}, \qquad C_i = Q_{ij} A_j \tag{15}$$

Then the equation for δw in the steady state becomes

$$\frac{\partial}{\partial \xi_i} (\delta F_i) = 0 \tag{16}$$

where in the domain

$$\delta F_i = C_i \delta w + \delta(Q_{ij}) f_j$$

and on solid surfaces

$$\delta F_\eta = \begin{bmatrix} 0 \\ Q_{\eta 1} \delta p \\ Q_{\eta 2} \delta p \\ Q_{\eta 3} \delta p \\ 0 \end{bmatrix} + p \begin{bmatrix} 0 \\ \delta(Q_{\eta 1}) \\ \delta(Q_{\eta 2}) \\ \delta(Q_{\eta 3}) \\ 0 \end{bmatrix} \tag{17}$$

on any B_S.

Now, multiplying Eq. (16) by a vector costate variable ψ, assuming the result is differentiable, and integrating by parts over the entire domain

$$\int_D \left(\frac{\partial \psi^T}{\partial \xi_i} \delta F_i \right) \mathrm{d}\xi_j - \int_B (\bar{n}_i \psi^T \delta F_i) \, \mathrm{d}\xi_j = 0 \tag{18}$$

where $\bar{n}_i$ are components of a unit vector normal to the boundary. Equation (18) can now be subtracted from Eq. (14) without changing the value of δI. Then, ψ may be chosen to cancel the explicit terms in δw and δp. For this purpose, ψ is set to the steady-state solution of the adjoint equation

$$\frac{\partial \psi}{\partial t} - C_i^T \frac{\partial \psi}{\partial \xi_i} = 0 \quad \text{in} \quad D \tag{19}$$

with the surface boundary condition

$$(\psi_2 Q_{\eta 1} + \psi_3 Q_{\eta 2} + \psi_4 Q_{\eta 3}) = \mathfrak{Q} \tag{20}$$

on all surfaces comprising B_S, where

$$\mathfrak{Q} = (1/\tfrac{1}{2} p_\infty \gamma M_\infty^2 S_{\text{ref}})[(S_x \cos\alpha + S_y \sin\alpha) + \Omega(S_y \cos\alpha - S_x \sin\alpha)]$$

$$\Omega = \frac{\left(\frac{\partial C_D}{\partial \alpha}\right)}{\left(\frac{\partial C_L}{\partial \alpha}\right)}$$

At internal block boundaries, the face integrals cancel from the contributions of adjacent blocks. At the far field, the choice of the adjoint boundary condition depends on whether the flow is subsonic or supersonic. For subsonic flow, as long as the outer domain is far from the configuration of interest, we may set

$$\psi_{1-5} = 0 \tag{21}$$

on all surfaces comprising B_F.

If, however, the far-field boundary is close to the body, then boundary conditions may be determined by setting $\psi_{1-5} = 0$ for incoming waves, with outgoing waves being extrapolated from the interior solution. It is noted that the waves in the adjoint problem propagate in the opposite direction to those in the flow problem as a result of the sign reversal in Eq. (19). For supersonic flows, the choice of boundary conditions at the outer domain can be developed from physical intuition as well as mathematical analysis. For a given geometry, a change in the surface at any particular point P will affect the flow in a conical region (referred to as the Mach cone) originating at P. Similarly, it is possible to determine the region of space where changes in either geometry or flowfield quantities will influence the solution and the aerodynamic properties at P. This region would also form a cone that points in the direction opposite to the Mach cone. It is the solution of this reverse problem that the adjoint represents. Changes in the flow conditions at a given point P are determined by contributions that arise from changes to the surfaces that exist within the reverse cone; changes to the geometry that occur outside of the reverse Mach cone will have no influence at P. The solution of the adjoint therefore determines the actual influence that any change in the geometry may have on the solution at P. The correct supersonic far-field boundary conditions for the adjoint equation that are consistent with this reversed character are

$$\psi_{1-5} = 0$$

at the downstream exit

$$\psi_{1-5} \tag{22}$$

extrapolated from the interior at the inflow boundary. Then, if the coordinate transformation is such that $\delta(JK^{-1})$ is negligible in the far field, we obtain the expression

$$\delta I = \frac{1}{S_{\text{ref}}} \int \int_{B_S} C_p[(\delta S_x \cos\alpha + \delta S_y \sin\alpha) + \Omega(\delta S_y \cos\alpha - \delta S_x \sin\alpha)] \, \mathrm{d}\xi_1 \, \mathrm{d}\xi_2 + \int_D \psi^T \frac{\partial}{\partial \xi_i} (\delta Q_{ij} f_j) \, \mathrm{d}\xi_k \tag{23}$$

from which the gradient components can easily be obtained by independent variations in the design variables, which, in turn, modify the values of the mesh metrics. Note that Eq. (23) does not require the recalculation of the flowfield because all explicit dependence on δw has been eliminated with the use of the adjoint equation and the boundary conditions. Further details concerning the approach as well as the development for other cost functions have been presented in Refs. 4, 6, 15, 35, 38, and 45. The adjoint Eq. (19), along with its boundary conditions [Eqs. (20–22)], must be discretized and solved.

In the current implementation, a cell-centered, central-difference stencil that mimics the flux balancing used for the flow solution is used. Because this choice of discretization differs from the one obtained if the discrete flow equation Jacobian matrix were actually transposed to form the adjoint system, the gradients obtained by the present method will not be exactly

equal to the gradients calculated by finite differencing of the discrete flow solution. However, as the mesh is refined, these differences should vanish.[31]

The discretized adjoint system is solved on the multiblock domain in a fashion identical to that used for the flow solution, namely with an explicit multistage Runge–Kutta-like algorithm accelerated by residual smoothing and multigridding. Interblock communication is again handled through a double halo that allows for the full transfer of information across boundaries, except for the stencil of support of the implicit residual smoothing.

Multiblock Mesh Variations

The expression for δI in Eq. (23) eliminates the need for multiple flow solutions to determine the gradient of I with respect to an arbitrary number of design variables. The variation in the cost function derived from independent changes in the design variables can now be explicitly calculated as long as the mesh metric variation δQ_{ij}, δS_x, and δS_y are known in all blocks in the mesh. These mesh metric variations can be computed easily using finite differences. However, if this method is used, the need to regenerate the mesh for independent perturbations to the design variables arises. Unfortunately, for complex three-dimensional configurations, elliptic or hyperbolic partial differential equations must often be solved in an iterative fashion to obtain acceptably smooth meshes. These iterative mesh generation procedures can be computationally expensive, and, in a worst-case scenario, their cost approaches that of the flow solution. Thus, despite the use of an adjoint solver to eliminate the dependence of the gradient on the variations in the flow variables, the use of finite difference methods for obtaining metric variations in combination with an iterative mesh generator could lead to overall computational costs that are still strongly correlated with the number of design variables. Even in the case where the mesh regeneration cost is reasonable, automatic multiblock mesh generation is by no means a trivial task. In fact, no method currently exists that allows this process to be accomplished without user interaction for complex three-dimensional configurations.

Alternatively, one can use an analytic mapping procedure to bypass the difficulties in the calculation of the mesh metric variations. Our earlier works[4,14,15,33,34,45] took advantage of this approach for the automatic design of simple geometric configurations. The technique is fully automatic, produces smooth consistent meshes, and eliminates the need to use finite differences for the variations in the mesh metric terms. Because the mapping function fully determines the entire mesh for a given surface shape, this analytic relationship may be directly differentiated to obtain the required information without actual remeshing. However, an analytic mapping method requires the topology of the geometry to be built directly into the formulation, and it is feasible for only relatively simple configurations. Nevertheless, within these limitations, it has proven to be highly effective.[33,34,45]

For calculations involving complex three-dimensional geometries, the use of an analytic mapping procedure is not feasible and an alternative procedure for automatic mesh generation had to be found. An option we have explored in this and earlier works[6,39,42] is the use of finite differencing with an analytic mesh perturbation scheme. In this approach, a high-quality multiblock mesh appropriate for the flow solver is first generated by any available procedure prior to the start of the design. The examples shown in Ref. 56 use meshes created using the Gridgen software developed by Pointwise, Inc.[51] This initial mesh becomes the basis for all subsequent meshes that are obtained by analytic perturbations.

For single-block meshes, our original work had used the simple strategy of moving all points in the mesh along coordinate lines normal to the moving surface. The amount of motion was attenuated by the normalized arc length along each index line. This approach was satisfactory for design problems in which a single surface was allowed to move. For complex geometries (such as wing–body or nacelle–pylon intersections) there is usually a need for two intersecting surfaces to be deformed simultaneously. In this situation, the coordinate direction normal to the moving surface becomes ill-defined and the original method needed to be modified in a way that resembles transfinite interpolation (TFI).[52] Unlike TFI, where there is no prior knowledge of the interior mesh, the perturbation algorithm developed here (WARP3D) makes use of the relative distributions of the interior points in the initial mesh.

The WARP3D algorithm has been modified from its initial form[6] and is now a three-stage procedure[39] that is applied to the blocks in the multiblock mesh that require it. The first stage shifts the internal mesh points to produce an interim block that is determined entirely by the new locations of the eight corner points of the block. The second stage corrects the perturbations resulting from the first stage by determining the distance each of the 12 edges of the stage 1 block needs to be moved to attain the final desired edge locations. Finally, with both corner and edge point motion accounted for, the third stage corrects the internal points for the relative motion of the six faces.

Because our flow solver and design algorithm assume a point-to-point match between adjacent blocks in the mesh, each block may be independently perturbed by WARP3D, provided that modified surfaces are treated continuously across block boundaries. The entire method of perturbing an existing mesh to create a new one is outlined in the following algorithm (WARP–MB).

1) All block faces that are directly affected by the design variables (active faces) are explicitly perturbed.

2) All edges that are in contact with an active face, either in the same or in an adjacent block, are implicitly perturbed with an arc-length attenuation method.

3) All inactive faces that either include an implicitly perturbed edge or abut to an active face are implicitly perturbed with a quasi-three-dimensional form of WARP3D.

4) WARP3D is used on each block that has one or more explicitly or implicitly perturbed faces to determine the final location of the interior points.

Note that much of the mesh, especially away from the surfaces, will not require mesh perturbations and, thus, may remain fixed throughout the entire design process. Close to the design surfaces, many blocks will either contain an active face or touch a block that contains an active face, either through an edge or a corner. As the design variations affect the active faces, the preceding scheme ensures that the entire mesh remains attached along block boundaries. Added complexity is needed to accomplish the second step in the procedure because the connectivity of the various edges and corners of the blocks must be somehow known to the program. Currently, pointers to and from a set of master edges and master corners are constructed in a preprocessing step. During the design calculation, deflections to any edges or corners are transferred to these master edges and corners which, in turn, communicate the changes to all coincident edges and corners.

Because this mesh perturbation algorithm is analytic, it would be possible to obtain analytic expressions for the variations in the metric terms required for Eq. (23). The use of this approach was shown in previous work.[4] However, because the mesh perturbation algorithm used in this work was significantly more complex, and because it was found that the computational cost of repeatedly using the block perturbation algorithm was reasonable (less than 5% of the total design calculation time), finite differences were used to calculate the mesh metric variations δQ_{ij}.

Design Variables and Constrained Optimization

To complete the design method, the remaining tasks are to choose appropriate design variables and impose geometric constraints. As discussed earlier, in some of our previous work where an analytic mesh mapping strategy was used, each point

of the surface mesh served as an independent design variable. This technique combined with smoothing and a projection into feasible space has proven to be highly effective for single-block design cases.[4,14,15,33,34,38,45] In cases where an analytic mapping becomes impractical, the use of the location of the surface mesh points as independent design variables becomes unreasonable because the number of these points is usually very large. In this work we rely on the alternative strategy of using Hicks–Henne functions described in an earlier section. This choice has the benefit of reducing the number of design variables and allowing for greater user control over the design process.

To implement geometric constraints, the multiblock approach was coupled to the constrained nonlinear optimization package of Gill et al.[53] NPSOL implements a sequential quadratic programming (SQP) method that requires relatively few function and gradient evaluations. Search directions for both primal and dual variables (the design variables and Lagrange multipliers for the nonlinear constraints) are computed by solving a quadratic program (QP). The QP Hessian is a quasi-Newton approximation to the Lagrangian, and the QP constraints are linearizations of the original constraints (including all linear constraints, which are satisfied throughout). The step length is chosen to reduce an augmented Lagrangian merit function involving the primal and dual variables. SQP methods and other optimization strategies are discussed in Gill et al.[54] A complete treatment of the NPSOL algorithm and its merit function are given by Gill et al.[53,55]

Because aircraft configurations are composed of separate geometry entities (wings, fuselages, nacelles, etc.), on which constraints must be imposed, it was considered advantageous to use an underlying set of unintersected geometry entities as starting input to the design process. The user-specified design variables are allowed to act independently on any of these geometry entities. Linear and nonlinear geometric constraints are also evaluated on these primary geometry entities. During the design procedure, changes to the original mesh surfaces are obtained using a two-step process. Firstly, the modified geometry entities are intersected to construct a set of perturbed parametric patches that represent the complete configuration. Secondly, by evaluating these perturbed patches at prespecified (u, v) locations, the associated x, y, z coordinates of the surface points in the perturbed multiblock mesh are determined. The (u, v) locations are computed and stored in a preprocessing step using the initial geometry entities and multiblock mesh. Once the surface mesh points have been updated, the volume mesh may be perturbed via the WARP–MP procedure and either the solution or the gradient can be calculated.

The important feature of this approach is that a set of simple geometry entities lies at the core of the entire design process. This technique retains the typical way in which aerodynamic vehicles are defined, and provides strict control over how wing/body and other surface intersections are treated. Furthermore, because the chosen design variables act directly upon the geometry entities, at the end of the design process these entities may be output for further analysis. In the current implementation, the input geometry entities are restricted to those defined by networks of discrete point locations. However, in the future, CAD entities such as NURBS surfaces will also serve in this role, thereby allowing the I/O of the aerodynamic surface optimization to interface directly with a CAD database.

Domain Decomposition and Parallel Implementation

The main strategies that are used to accomplish the parallelization of the design code are a domain decomposition model, a single program multiple data strategy, and the message passing interface (MPI) library for message passing. The choice of MPI was determined by the requirement that the resulting code be easily portable to different parallel computing platforms as well as to homogeneous and heterogeneous networks of workstations while still achieving high computational efficiency.

As the previous sections have indicated, obtaining the desired parallelization by domain decomposition entails treatment of the four separate parts of the design method: the solution of the flow equations, the solution of the adjoint equations, the calculation of the mesh perturbations, and the calculation of the gradient integral formulas. No attempt is made to parallelize the constrained SQP optimization algorithm or the calculation of the changes to the underlying geometry entities. This approach inherently assumes that the determination of the step sizes and search directions provided by the optimization algorithm is computationally insignificant when compared with the other elements of the design process. Experimental evidence shows this statement to be true for typical numbers of design variables (100–300).

Because the flow and adjoint equations are to be solved using exactly the same efficient numerical algorithms, the same parallelization techniques used for the flow equations apply to the solution of the adjoint equations. Therefore, all details of the parallel implementation corresponding to these first two parts of the program are identical. Furthermore, because the mesh perturbation algorithm WARP3D also works on a block-by-block basis, the communication necessary to maintain mesh consistency can be addressed by the same domain decomposition strategies that are used for the state and costate fields. The essential details of WARP3D and its parallel implementation can be found in Ref. 39.

The message-passing strategy is based on the lists of pointers to and from halo cells described in a previous section. These lists result from the decomposition of the blocks in the mesh into a specified number of processors. The decomposition is achieved by assigning complete blocks to individual processors. To preserve high parallel efficiency, the issue of load balancing is taken into account by carefully distributing a comparable amount of work (computation and communication) to each of the processors in the calculation. Although each processor in the calculation will typically contain more than one block, the data to be sent from one processor to another is lumped into a single message, regardless of the number of blocks in each processor. Communication is achieved using asynchronous MPI constructs to minimize contention in the communication network.

Reference 43 includes further refinements to the message-passing strategy that enhance the parallel efficiency of the method. It also describes in detail the important issues in the parallelization of the method, as well as the parallel performance in distributed memory computers with various levels of networking sophistication.

Summary: Complete Multiblock Design Algorithm (SYN87-MB)

With all the necessary components defined for the multiblock adjoint-based design, we now proceed to outline the complete algorithm.

1) Decompose the multiblock mesh into an appropriate number of processors and create lists of pointers for the communication of the block halo cells.

2) Solve the flowfield governing equations [Eqs. (7–13)] for each design point.

3) Solve the adjoint equations [Eq. (19)] subject to the boundary condition [Eq. (20)] for each design point.

4) For each of the n design variables, repeat the following: perturb the design variable by a finite step to modify the geometry entities; reintersect the geometry entities to form perturbed parametric patches; use the WARP–MB mesh perturbation algorithm to translate changes in the modified parametric patches to all points in the multiblock mesh; calculate all of the delta metric terms $\delta Q_{i,j}$ within those blocks that were perturbed by finite differencing; and integrate Eq. (23) to obtain δI for those blocks that contain nonzero $\delta Q_{i,j}$,

and for each design point, to determine the gradient component.

5) Calculate the search direction using NPSOL and perform a line search (possibly with multiple function calls).

6) Return to step 2 if a minimum has not been reached.

Practical applications of the design algorithm can be found in Ref. 56.

References

[1]Young, D. P., Huffman, W. P., Melvin, R. G., Bieterman, M. B., Hilmes, C. L., and Johnson, F. T., "Inexactness and Global Convergence in Design Optimization," AIAA Paper 94-4286, Sept. 1994.

[2]Reuther, J., Cliff, S., Hichs, R., and Van Dam, C. P., "Practical Design Optimization of Wing/Body Configurations Using the Euler Equations," AIAA Paper 92-2633, 1992.

[3]Reuther, J., "Practical Aerodynamic Optimization of Airfoils and Wings," M.S. Thesis, Univ. of California, Davis, CA, 1991.

[4]Reuther, J., and Jameson, A., "Aerodynamic Shape Optimization of Wing and Wing-Body Configurations Using Control Theory," AIAA Paper 95-0123, Jan. 1995.

[5]Gallman, J., Reuther, J., Pfeiffer, N., Forrest, W., and Bernstorf, D., "Business Jet Wing Design Using Aerodynamic Shape Optimization," AIAA Paper 96-0554, Jan. 1996.

[6]Reuther, J., Jameson, A., Farmer, J., Martinelli, L., and Saunders, D., "Aerodynamic Shape Optimization of Complex Aircraft Configurations via an Adjoint Formulation," AIAA Paper 96-0094, Jan. 1996.

[7]Elliott, J., and Peraire, J., "3-D Aerodynamic Optimization on Unstructured Meshes with Viscous Effects," AIAA Paper 97-1849, June 1997.

[8]Lions, J. L., *Optimal Control of Systems Governed by Partial Differential Equations*, Springer–Verlag, New York, 1971, Translated by S. K. Mitter.

[9]Hicks, R. M., Murman, E. M., and Vanderplaats, G. N., "An Assessment of Airfoil Design by Numerical Optimization," NASA TM X-3092, July 1974.

[10]Hicks, R. M., and Henne, P. A., "Wing Design by Numerical Optimization," *Journal of Aircraft*, Vol. 15, 1978, pp. 407–412.

[11]Reuther, J., Van Dam, C. P., and Hicks, R., "Subsonic and Transonic Low-Reynolds-Number Airfoils with Reduced Pitching Moments," *Journal of Aircraft*, Vol. 29, 1992, pp. 297–298.

[12]Oyama, A., Obayashi, S., Nakahashi, K., and Nakamura, T., "Transonic Wing Optimization Using Genetic Algorithm," AIAA Paper 97-1854, June 1997.

[13]Reuther, J., "Aerodynamic Shape Optimization Using Control Theory," Ph.D. Dissertation, Univ. of California, Davis, CA, June 1996.

[14]Reuther, J., and Jameson, A., "Control Theory Based Airfoil Design for Potential Flow and a Finite Volume Discretization," AIAA Paper 94-0499, Jan. 1994.

[15]Jameson, A., and Reuther, J., "Control Theory Based Airfoil Design Using the Euler Equations," AIAA Paper 94-4272, Sept. 1994.

[16]Baysal, O., and Eleshaky, M. E., "Aerodynamic Design Optimization Using Sensitivity Analysis and Computational Fluid Dynamics," *AIAA Journal*, Vol. 30, No. 3, 1992, pp. 718–725.

[17]Baysal, O., and Eleshaky, M. E., "Airfoil Shape Optimization Using Sensitivity Analysis on Viscous Flow Equations," *Journal of Fluids Engineering*, Vol. 15, 1993, pp. 75–84.

[18]Burgreen, G. W., and Baysal, O., "Three-Dimensional Aerodynamic Shape Optimization of Wings Using Sensitivity Analysis," AIAA Paper 94-0094, Jan. 1994.

[19]Lacasse, J. M., and Baysal, O., "Design Optimization of Single- and Two-Element Airfoils on Multiblock Grids," AIAA Paper 94-4273, Sept. 1994.

[20]Korivi, V. M., Taylor, A. C., III, and Newman, P. A., "Aerodynamic Optimization Studies Using a 3-D Supersonic Euler Code with Efficient Calculation of Sensitivity Derivatives," AIAA Paper 94-4270, Sept. 1994.

[21]Taylor, A. C., III, Hou, G. W., and Korivi, V. M., "Sensitivity Analysis, Approximate Analysis, and Design Optimization for Internal and External Viscous Flows," AIAA Paper 91-3083, Sept. 1991.

[22]Korivi, V. M., Taylor, A. C., III, Hou, G. W., Newman, P. A., and Jones, H. E., "Sensitivity Derivatives for Three-Dimensional Supersonic Euler Code Using Incremental Iterative Strategy," *Proceedings of the AIAA 11th Computational Fluid Dynamics Conference*, AIAA, Washington, DC, 1993, pp. 1053, 1054.

[23]Ta'asan, S., Kuruvila, G., and Salas, M. D., "Aerodynamic Design and Optimization in One Shot," AIAA Paper 92-0025, Jan. 1992.

[24]Kuruvila, G., Ta'asan, S., and Salas, M. D., "Airfoil Optimization by the One-Shot Method," *AGARD-VKI Lecture Series, Optimum Design Methods in Aerodynamics*, von Kármán Inst. for Fluid Dynamics, Rhode Saint Genese, Belgium, 1994.

[25]Pironneau, O., *Optimal Shape Design for Elliptic Systems*, Springer–Verlag, New York, 1984.

[26]Pironneau, O., "Optimal Shape Design for Aerodynamics," *AGARD-VKI Lecture Series, Optimum Design Methods in Aerodynamics*, von Kármán Inst. for Fluid Dynamics, Rhode Saint Genese, Belgium, 1994.

[27]Huffman, W. P., Melvin, R. G., Young, D. P., Johnson, F. T., Bussoletti, J. E., Bieterman, M. B., and Hilmes, C. L., "Practical Design and Optimization in Computational Fluid Dynamics," AIAA Paper 93-3111, July 1993.

[28]Lewis, J., and Agarwal, R., "Airfoil Design via Control Theory Using the Full-Potential and Euler Equations," The Forum on CFD for Design and Optimization, (IMECE 95), San Francisco, CA, Nov. 1995.

[29]Huan, J., and Modi, V., "Design of Minimum Drag Bodies in Incompressible Laminar Flow," The Forum on CFD for Design and Optimization, (IMECE 95), San Francisco, CA, Nov. 1995.

[30]Hafez, M., and Matsuzawa, T., "Treatment of Boundary Conditions in Optimum Shape Design Using Adjoint Equation for Compressible Flows," AIAA Paper 97-2078, June 1997.

[31]Anderson, W. K., and Venkatakrishnan, V., "Aerodynamic Design Optimization on Unstructured Grids with a Continuous Adjoint Formulation," AIAA Paper 97-0643, Jan. 1997.

[32]Jameson, A., "Optimum Aerodynamic Design Using CFD and Control Theory," AIAA Paper 95-1729, June 1995.

[33]Jameson, A., "Aerodynamic Design via Control Theory," *Journal of Scientific Computing*, Vol. 3, 1988, pp. 233–260.

[34]Jameson, A., "Automatic Design of Transonic Airfoils to Reduce the Shock Induced Pressure Drag," *Proceedings of the 31st Israel Annual Conference on Aviation and Aeronautics* (Tel Aviv), 1990, pp. 5–17.

[35]Reuther, J., and Jameson, A., "A Comparison of Design Variables for Control Theory Based Airfoil Optimization," 6th International Symposium on Computational Fluid Dynamics, Lake Tahoe, Nevada, Sept. 1995.

[36]Reuther, J., and Jameson, A., "Supersonic Wing and Wing-Body Shape Optimization Using an Adjoint Formulation," The Forum on CFD for Design and Optimization, (IMECE 95), San Francisco, CA, Nov. 1995.

[37]Jameson, A., and Alonso, J. J., "Automatic Aerodynamic Optimization on Distributed Memory Architectures," AIAA Paper 96-0409, Jan. 1996.

[38]Jameson, A., *Optimum Aerodynamic Design Using Control Theory, Computational Fluid Dynamics Review 1995*, Wiley, New York, 1995.

[39]Reuther, J., Alonso, J. J., Rimlinger, M. J., and Jameson, A., "Aerodynamic Shape Optimization of Supersonic Aircraft Configurations via an Adjoint Formulation on Parallel Computers," AIAA Paper 96-4045, Sept. 1996.

[40]Jameson, A., "Re-Engineering the Design Process Through Computation," AIAA Paper 97-0641, Jan. 1997.

[41]Jameson, A., Pierce, N., and Martinelli, L., "Optimum Aerodynamic Design Using the Navier-Stokes Equations," AIAA Paper 97-0101, Jan. 1997.

[42]Reuther, J., Jameson, A., Alonso, J. J., Rimlinger, M. J., and Saunders, D., "Constrained Multipoint Aerodynamic Shape Optimization Using an Adjoint Formulation and Parallel Computers," AIAA Paper 97-0103, Jan. 1997.

[43]Reuther, J., Alonso, J. J., Vassberg, J. C., Jameson, A., and Martinelli, L., "An Efficient Multiblock Method for Aerodynamic Analysis and Design on Distributed Memory Systems," AIAA Paper 97-1893, June 1997.

[44]Baysal, O., and Pandya, M., "Three-Dimensional Viscous ADI Algorithm and Strategies for Shape Optimization," AIAA Paper 97-1853, June 1997.

[45]Jameson, A., "Optimum Aerodynamic Design via Boundary Control," *AGARD-VKI Lecture Series, Optimum Design Methods in Aerodynamics*, von Kármán Inst. for Fluid Dynamics, Rhode Saint Genese, Belgium, 1994.

[46]Jameson, A., "Aerodynamic Design Methods," International Workshop on Solution Techniques for Large-Scale CFD Problems, Montreal, Canada, Sept. 1994.

[47]Lorentzen, L., "Aerodynamic Shape Optimization Based on Control Theory," Aeronautical Research Inst. of Sweden, FFA TN 1997-42, June 1997.

[48]Jameson, A., Schmidt, W., and Turkel, E., "Numerical Solutions

of the Euler Equations by Finite Volume Methods with Runge-Kutta Time Stepping Schemes," AIAA Paper 81-1259, Jan. 1981.

[49]Jameson, A., "Multigrid Algorithms for Compressible Flow Calculations," *Lecture Notes in Mathematics*, edited by W. Hackbusch and V. Trottenberg, Vol. 1228, Springer–Verlag, 1986.

[50]Jameson, A., "Solution of the Euler Equations for Two Dimensional Transonic Flow by a Multigrid Method," *Applied Mathematics and Computations*, Vol. 13, 1983, pp. 327–356.

[51]Steinbrenner, J. P., Chawner, J. R., and Fouts, C. L., "The GRIDGEN 3D Multiple Block Grid Generation System," Flight Dynamics Laboratory, Wright Research and Development Center, Wright–Patterson AFB, OH, July 1990.

[52]Thompson, J. F., Warsi, Z. U. A., and Mastin, C. W., *Numerical Grid Generation, Foundations and Applications*, Elsevier, New York, 1985.

[53]Gill, P. E., Murray, W., Saunders, M. A., and Wright, M. H., "User's Guide for NPSOL (Version 4.0). A FORTRAN Package Nonlinear Programming," Dept. of Operations Research, TR SOL86-2, Stanford Univ., Stanford, CA, 1986.

[54]Gill, P. E., Murray, W., and Wright, M. H., *Practical Optimization*, Academic, 1981.

[55]Gill, P. E., Murray, W., Saunders, M. A., and Wright, M. H., "Some Theoretical Properties of an Augmented Lagrangian Merit Function," *Advances in Optimization and Parallel Computing*, edited by P. M., Pardalos, North-Holland, Amsterdam, pp. 101–128.

[56]Reuther, J. J., Jameson, A., Alonso, J. J., Rimlinger, M. J., and Saunders, D., "Constrained Multipoint Aerodynamic Shape Optimization Using an Adjoint Formulation and Parallel Computers, Part 2," *Journal of Aircraft*, Vol. 36, No. 1, 1999, pp. 61–74.