
Flow, Turbulence and Combustion **65:** 393–415, 2000.

An Introduction to the Adjoint Approach to Design

MICHAEL B. GILES[1] and NILES A. PIERCE[2]
[1]*Computing Laboratory, Oxford University, Wolfson Building, Parks Road, Oxford OX1 3QD, U.K.*
[2]*Applied Mathematics, California Institute of Technology, Pasadena, CA 91125, U.S.A.*

Received 13 December 1999; accepted in revised form 2 February 2000

Abstract. Optimal design methods involving the solution of an adjoint system of equations are an active area of research in computational fluid dynamics, particularly for aeronautical applications. This paper presents an introduction to the subject, emphasising the simplicity of the ideas when viewed in the context of linear algebra. Detailed discussions also include the extension to p.d.e.'s, the construction of the adjoint p.d.e. and its boundary conditions, and the physical significance of the adjoint solution. The paper concludes with examples of the use of adjoint methods for optimising the design of business jets.

Key words: computational fluid dynamics, adjoint p.d.e., design.

1. Introduction

There is a long history of the use of adjoint equations in optimal control theory [31]. In fluid dynamics, the first use of adjoint equations for design was by Pironneau [37], but within the field of aeronautical computational fluid dynamics, the use of adjoint equations has been pioneered by Jameson, who used his knowledge of optimal control theory to develop what he calls optimal design methods. The term 'optimal' refers to the fact that one is trying to find the geometry which minimises some objective function subject to a set of constraints. In a sequence of papers by himself [24–26] and with Reuther and other co-authors [28, 39, 40] Jameson developed the adjoint approach for potential flow, the Euler equations and the Navier–Stokes equations. The complexity of the applications within these papers also progressed from 2D airfoil optimisation, to 3D wing design and finally to complete aircraft configurations [27, 41, 42],

A number of other research groups have developed adjoint CFD codes for design optimisation [3, 4, 6, 8, 22, 29, 30, 44]. An overview of recent developments in adjoint design methods is provided elsewhere [23]. Of particular interest is the work of Elliott [9, 11] and Anderson [2, 34] on unstructured grids using the 'discrete' adjoint approach, and the work of Mohammadi [32, 33] in using automatic differentiation software to create the adjoint code from an original CFD code; both of these approaches will be discussed further in this paper.

Considering the importance of design to aeronautical engineering, and indeed to all of engineering, it is perhaps surprising that the development of adjoint CFD

codes has not been more rapid in the decade since Jameson's first papers appeared. In part, this may be due to some of the limitations of the adjoint approach, which will be discussed later in this paper. However, it seems likely that part of the reason is its complexity, both in the mathematical formulation of the adjoint p.d.e. and boundary conditions in the 'continuous' approach favoured by Jameson, and in the creation of the adjoint CFD code in the 'discrete' approach.

In this paper we aim to address some of these difficulties. The adjoint theory is presented firstly in the context of linear algebra, in which it is most easily understood. This is the basis for the discrete adjoint CFD approach in which one works with the algebraic equations that come from the discretisation of the original fluid dynamic equations.

The paper then treats the extension to p.d.e.'s as used in Jameson's continuous adjoint approach in which the adjoint p.d.e. is formulated and then discretised. The emphasis in the present review is on the construction of the adjoint p.d.e. and its boundary conditions, the physical significance of the adjoint solution, and the manner in which geometric perturbations are introduced.

The paper concludes with a discussion of the pros and cons of the two approaches, the discrete and the continuous, and examples of the use of adjoint methods to optimise business jet designs.

2. Discrete Adjoint Approach

2.1. Linearised objective function

The goal of aerodynamic design optimisation is the minimisation (or maximisation) of an objective function that is a nonlinear function of a set of discrete flow variables. For example, the lift may be expressed as $L(U)$ where U is the set of all flow variables at discrete grid points arising from an approximate solution of the Euler equations, and L is a scalar function which approximates the appropriate weighted integral of pressure over the surface of an aircraft.

In design optimisation, the question of interest is: what is the perturbation in L due to a perturbation in the geometry, and hence the flow field? If u is the perturbation in the flow field, then the linearised perturbation in the lift is

$$g^T u \equiv \frac{\partial L}{\partial U} u.$$

Therefore, the goal is to evaluate the quantity $g^T u$ where u satisfies the appropriate linearised flow equations.

2.2. Duality and adjoint variables

Suppose one wishes to evaluate the quantity $g^T u$ given that u satisfies the linear system of equations

$$Au = f,$$

for some given matrix A and vector f. The dual form is to evaluate $v^T f$ where the adjoint solution v satisfies the linear system of equations

$$A^T v = g.$$

Note the use of the transposed matrix A^T, and the interchange in the roles of f and g.

The equivalence of the two forms is easily proved as follows:

$$v^T f = v^T A u = (A^T v)^T u = g^T u.$$

Given a single f and a single g, nothing would be gained (or lost) by using the dual form. Exactly the same value for the linear objective function would be obtained with exactly the same computational effort. However, suppose now that we want the value of the objective function for p different values of f, and m different values of g. The choice would be to do *either* p different primal calculations *or* m different dual calculations. When the dimension of the system is very large, the cost of the vector dot products is negligible compared to solving the linear systems of equations, and therefore the dual (or adjoint) approach is much cheaper when $m \ll p$.

2.3. PHYSICAL INTERPRETATION

It is possible to work with adjoint variables and regard them as a purely mathematical construct, but they do have physical significance.

One way of looking at them is that they give the influence of an arbitrary source term f on the functional of interest,

$$\underset{\text{source term}}{Au = f} \quad \longrightarrow \quad \underset{\text{functional perturbation}}{v^T f}$$

Another is that they are the value of the objective function corresponding to the appropriate Green's function. To see this, we define $f^{(i)}$ to be a vector whose elements are zero apart from the ith which is unity. The corresponding solution $u^{(i)}$ given by

$$Au^{(i)} = f^{(i)}$$

is the discrete equivalent of a Green's function and

$$v^T f^{(i)} = v_i = g^T u^{(i)}.$$

Thus, the ith component of the adjoint variables is equal to the value of the objective function when the solution is equal to the ith Green's function.

2.4. DUALITY FORMULATION FOR ADJOINT DESIGN

Given a set of design variables, α, which control the geometry of the airfoil, wing or aircraft being designed, and a set of flow variables at discrete grid points, U, the aim is to minimise a scalar objective function $J(U, \alpha)$. This minimisation is subject to the constraint that the discrete flow equations and boundary conditions are all satisfied. These may be expressed collectively as

$$N(U, \alpha) \equiv N(U, X(\alpha)) = 0,$$

where X is the vector of grid point coordinates which depends on α. Using techniques such as the 'method of springs' [38] or variants on transfinite interpolation [45, 40], the grid deforms smoothly as changes in the design variables modify the surface geometry. Hence, $\partial X/\partial\alpha$ is usually non-zero at both interior and surface grid points.

For a single design variable, we can linearise about a base solution U_0 to get

$$\frac{\mathrm{d}J}{\mathrm{d}\alpha} = \frac{\partial J}{\partial U}\frac{\mathrm{d}U}{\mathrm{d}\alpha} + \frac{\partial J}{\partial \alpha},$$

subject to the constraint that the flow sensitivity $\mathrm{d}U/\mathrm{d}\alpha$ satisfies the linearised flow equations

$$\frac{\partial N}{\partial U}\frac{\mathrm{d}U}{\mathrm{d}\alpha} + \frac{\partial N}{\partial \alpha} = 0.$$

By defining

$$u = \frac{\mathrm{d}U}{\mathrm{d}\alpha}, \quad A = \frac{\partial N}{\partial U},$$

$$g^T = \frac{\partial J}{\partial U}, \quad f = -\frac{\partial N}{\partial \alpha},$$

we can convert this into the standard form

$$\frac{\mathrm{d}J}{\mathrm{d}\alpha} = g^T u + \frac{\partial J}{\partial \alpha},$$

subject to

$$Au = f.$$

The direct sensitivity of the objective function to perturbations in the design variables is easy to evaluate. The term $g^T u \equiv v^T f$ can be computed either by the direct approach, solving $Au = f$, or by the adjoint approach, solving $A^T v = g$. For a single design variable there would be no benefit in using the adjoint approach, but for multiple design variables, each has a *different* f, but the *same* g, so the adjoint approach is computationally much more efficient.

2.5. ALTERNATIVE LAGRANGE VIEWPOINT

In the presentation above, we have used the terminology of duality, coming from the mathematics of vector spaces, linear algebra and linear programming. An alternative description arises using the terminology of Lagrange multipliers associated with constrained minimisation. In this framework, the adjoint variables are Lagrange multipliers, usually written as λ, and are introduced into an augmented objective function

$$I(U, \alpha) = J(U, \alpha) - \lambda^T N(U, \alpha),$$

to enforce the satisfaction of the discrete flow equations. Considering general perturbations $\mathrm{d}U$ and $\mathrm{d}\alpha$ gives

$$\mathrm{d}I = \left(\frac{\partial J}{\partial U} - \lambda^T \frac{\partial N}{\partial U}\right) \mathrm{d}U + \left(\frac{\partial J}{\partial \alpha} - \lambda^T \frac{\partial N}{\partial \alpha}\right) \mathrm{d}\alpha.$$

If λ^T is chosen to satisfy the adjoint equation

$$\frac{\partial J}{\partial U} - \lambda^T \frac{\partial N}{\partial U} = 0 \implies \left(\frac{\partial N}{\partial U}\right)^T \lambda = \left(\frac{\partial J}{\partial U}\right)^T,$$

then

$$\mathrm{d}I = \left(\frac{\partial J}{\partial \alpha} - \lambda^T \frac{\partial N}{\partial \alpha}\right) \mathrm{d}\alpha,$$

and thus $\mathrm{d}I/\mathrm{d}\alpha$ is obtained.

The final equations are exactly the same as those derived by considering duality; it is really only the description of the mathematics which differs. In aeronautical CFD, most people follow Jameson in adopting the Lagrange multiplier viewpoint for design optimisation because of its connection to constrained optimisation and optimal control theory. On the other hand, we prefer the duality viewpoint because it seems more natural for other uses of adjoint variables, such as error analysis [17, 35, 36, 46], which do not involve constrained optimisation.

2.6. NONLINEAR OPTIMISATION

Returning to the design problem, the aim is to find the set of design variables α which minimise the nonlinear objective function $J(U, \alpha)$, where U is an implicit function of a through the flow equations

$$N(U, \alpha) = 0.$$

These nonlinear flow equations and the corresponding linear adjoint equations are both large systems which are usually solved by an iterative procedure.

There are two principal schools of thought as to the best method for marching the design variables to a local minimum. In the first approach, a simple steepest descent algorithm is employed,

$$\Delta\alpha = -\varepsilon \frac{\mathrm{d}J}{\mathrm{d}\alpha},$$

where ε controls the step size. The advantage of this method is that partially-converged flow and adjoint solutions may be used to evaluate the gradients as long as these gradients are properly smoothed (preconditioned) prior to updating α [26]. As a result, the cost per design cycle is relatively low.

In the second approach, approximations to the Hessian matrix of second derivatives

$$\frac{\mathrm{d}^2 J}{\mathrm{d}\alpha_i \, \mathrm{d}\alpha_j},$$

are used to speed convergence via a quasi-Newton procedure such as BFGS [18]. This method therefore requires more accurate flow and adjoint solutions, which must generally be converged fully during each design iteration. As a result, the cost of each design cycle is significantly increased.

The relative efficiency and robustness of the partially and fully-converged approaches is still subject to debate. We have been unable to find any reference which presents a clear quantitative comparison of the two approaches, but the anecdotal evidence is that the partially-converged approach yields the lowest total computational time.

2.7. Limitations of the adjoint approach

2.7.1. *Constraints*

Engineering design applications often have a set of constraints which must be satisfied, in addition to the discrete flow equations. Some of these may be geometric, such as airfoil design in which the length of the chord and the area of the airfoil are fixed. Others may depend on the flow variables, such as wing design in which one wishes to minimise the drag but keep the lift fixed.

Geometric constraints are easily incorporated by modifying the search direction for the design variables to ensure that the geometric constraints are satisfied. It is the constraints which depend on the flow which pose a problem. If the constraint is taken to be 'hard' and so must be satisfied at all stages of the optimisation procedure, then we need to know both the value of the constraint function, which we shall label $J_2(U(\alpha), \alpha)$, and its linear sensitivity to the design variables. The latter requires a second adjoint calculation; the addition of more flow-based hard constraints would require even more adjoint calculations. This type of constraint therefore undermines the computational cost benefits of the adjoint approach. If

the number of hard constraints is almost as large as the number of design variables, then the benefit is entirely lost.

To avoid this, the alternative is to use 'soft' constraints via the addition of penalty terms in the objective function, e.g. $J(U)+\lambda(J_2(U))^2$. The value of λ controls the extent to which the optimal solution violates the constraint $J_2(U,\alpha)=0$. The larger the value of λ, the smaller the violation, but it also worsens the conditioning of the optimisation problem and hence increases the number of steps to reach the optimum.

2.7.2. *Least-Squares Problems*

In the direct linear perturbation approach one evaluates each of the linear flow sensitivities $\mathrm{d}U/\mathrm{d}\alpha_i$, one by one, by solving the linearised flow equations corresponding to a unit perturbation in a single design variable. From these one can then calculate the linear sensitivity of the objective function to each of the design variables, but the total cost is proportional to the number of the design variables, making the adjoint approach much cheaper.

However, if the objective function is of a least-squares type,

$$J(U)=\frac{1}{2}\sum_n (p_n(U)-P_n)^2,$$

then

$$\frac{\mathrm{d}J}{\mathrm{d}\alpha_i}=\sum_n \frac{\partial p}{\partial U}\frac{\mathrm{d}U}{\mathrm{d}\alpha_i}(p_n(U)-P_n),$$

and so

$$\frac{\mathrm{d}^2 J}{\mathrm{d}\alpha_i \mathrm{d}\alpha_j}\approx\sum_n\left(\frac{\partial p}{\partial U}\frac{\mathrm{d}U}{\mathrm{d}\alpha_i}\right)\left(\frac{\partial p}{\partial U}\frac{\mathrm{d}U}{\mathrm{d}\alpha_j}\right),$$

assuming that $p_n(U)-P_n$, is small. Thus, the direct linear perturbation approach also gives the approximate Hessian matrix, leading to very rapid convergence for the optimisation iteration. By contrast, the adjoint approach provides no information on the Hessian, so optimisation methods such as BFGS which build up an approximation to the Hessian take more steps to converge than the direct linear perturbation approach for least-squares applications. It is important to keep in mind, however, that for large numbers of design variables, the adjoint approach may still be more efficient, since the cost of each step is significantly higher when the sensitivities are evaluated directly.

2.7.3. *Limitations of Gradient-Based Optimisation*

The adjoint approach is only helpful in the context of gradient-based optimisation and such optimisation has its own limitations. Firstly, it is only appropriate when

the design variables are continuous. For design variables which can take only integer values (e.g. the number of engines on an aircraft) stochastic procedures such as simulated annealing and genetic algorithms are more suitable. Secondly, if the objective function contains multiple minima, then the gradient approach will generally converge to the nearest local minimum without searching for lower minima elsewhere in the design space. If the objective function is known to have multiple local minima, and possibly discontinuities, then again a stochastic search method may be more appropriate.

2.8. Implementation issues

In concept, the discrete adjoint approach is relatively straightforward. The linear algebra derivation is easy to grasp, and there is the attractive feature that the gradient of the objective function with respect to the design variables is exactly the same as would be obtained by the direct linear perturbation method.

Nonetheless, the practical implementation of this approach can be challenging. The nonlinear flow solver often solves the steady-state equations, $R(U) = 0$, by a time-marching iterative solution of

$$\frac{\mathrm{d}U}{\mathrm{d}t} + R(U) = 0.$$

Linearising the steady-state equations gives $Lu = f$, where

$$L \equiv \frac{\partial R}{\partial U}, \quad u \equiv \frac{\partial U}{\partial \alpha}, \quad f \equiv -\frac{\partial R}{\partial \alpha}.$$

Following a direct approach, the linear perturbation equations could also be solved by marching to steady-state the equations

$$\frac{\mathrm{d}u}{\mathrm{d}t} + Lu = f.$$

Similarly, the adjoint equations $L^T v = g$, can be solved by time-marching

$$\frac{\mathrm{d}v}{\mathrm{d}t} + L^T v = g.$$

The fact that L and L^T have the same eigenvalues means that the asymptotic convergence of the time-marching iteration in both cases will be identical, and will be equal to the asymptotic convergence rate of the nonlinear flow solver.

Let us turn now to the construction of the product $L^T v$. When approximating the Euler equations on an unstructured grid, the residual vector $R(U)$ can be expressed as a sum of contributions from each edge of the grid, with each edge contributing only to the residuals at the nodes at either end of the edge. Symbolically, we can write this as

$$R \equiv \sum_e R_e(U).$$

Linearisation gives

$$Lu = \sum_e L_e u, \quad L_e \equiv \frac{\partial R_e}{\partial U},$$

where L_e is a sparse matrix whose only non-zero elements have row and column numbers both matching one or other of the two nodes at either end of the edge. Therefore,

$$L^T v = \sum_e L_e^T v.$$

At the programming level, this product involves exactly the same loop over all of the edges as for the original nonlinear flow discretisation. In principle, one could compute the non-zero elements of the matrix L_e and then form the product $L_e^T v$. However, it is more efficient to calculate the product directly without explicitly constructing the matrix. A common objection to the discrete approach is the memory overhead that is incurred if the linearised matrix is pre-computed and stored to reduce the total number of operations. By forming the product directly, this memory overhead can be avoided while maintaining an operation count that is not substantially greater than that of the original nonlinear solver.

When approximating the Navier–Stokes equations on an unstructured grid, the residual vector can sometimes be expressed symbolically as

$$R \equiv \sum_e R_e(U, DU),$$

where the vector DU represents the numerical approximation to the flow solution gradient at the grid nodes at either end of the edge. When linearised, this becomes

$$Lu \equiv Au + VDu,$$

in which the matrices A, V, D can each be expressed as a sum of extremely sparse elemental matrices as described above for the Euler equations. The discrete adjoint operator for the Navier–Stokes equations is then

$$L^T v \equiv A^T v + D^T V^T v,$$

indicating that the adjoint gradient subroutine responsible for D^T must be applied *after* the viscous subroutine responsible for V^T. At first this seems counter-intuitive, but the mathematics is quite clear.

Working out the mathematical expressions for $L_e^T v$ and determining the best method for implementing the product is relatively easy for the inviscid fluxes of the Euler equations. This process is far more arduous for the viscous fluxes in the Navier–Stokes equations and for characteristic smoothing fluxes for the Euler equations. An alternative is to use AD (Automatic Differentiation) software such as Odyssée [12, 13] or ADIFOR and ADJIFOR [5, 7] to generate the Fortran code

to compute the product $L_e^T v$. In forward mode, AD software takes the original code which computed $R_e(U)$ and then uses the basic rules of linearisation to construct the code to evaluate $L_e u$. In reverse mode, it produces the code to calculate $L_e^T v$; it may seem that this is a much harder task but in fact it is not. Furthermore, there are theoretical results which guarantee that the number of floating point operations is no more than three times that of the original nonlinear code [20].

A final point concerns the evaluation of the term f, which is the source term for the direct perturbation equations and is in the objective function in the adjoint approach. Again, forward mode AD software could be used, but a very much simpler alternative is to use the 'complex variable method' [43] used by Anderson and co-workers [1]. The essence of the idea is that

$$\lim_{\varepsilon \to 0} \frac{I\{R(U, \alpha + i\varepsilon)\}}{\varepsilon} = \frac{\partial R}{\partial \alpha}.$$

In this equation, $R(U, \alpha)$ has been taken to be a complex analytic function, and the notation $I\{\ldots\}$ denotes the imaginary part of a complex quantity. The equation itself is an immediate consequence of a Taylor series expansion. The key is that this can be evaluated numerically using $\varepsilon = 10^{-20}$. Unlike the usual finite difference approximation of a linear sensitivity, there is no subtraction of two quantities which are almost equal; therefore there is no unacceptable loss of accuracy due to machine rounding error. Applying this technique to a FORTRAN code requires little more than replacing all `REAL*8` declarations by `COMPLEX*16`, and defining appropriate complex analytic versions of certain intrinsic functions.

We have found this complex variable method to be extremely effective. We have also used it to verify the correctness of our handcoded adjoint calculations by checking the identity $u^T(L^T v) = v^T(Lu)$, with the product $L^T v$ being computed using the adjoint code, and the product

$$Lu = \lim_{\varepsilon \to 0} \frac{I\{R(U + i\varepsilon u, \alpha)\}}{\varepsilon},$$

being computed using the complex variable method.

3. Continuous Adjoint Approach

3.1. Duality and the adjoint P.D.E.

Duality in the case of p.d.e.'s is a natural extension of duality in the linear algebra formulation. Using the notation (V, U) to denote an integral inner product over some domain Ω,

$$(V, U) \equiv \int_\Omega V^T U \, \mathrm{d}x,$$

suppose that one wants to evaluate the functional (g, u), where u is the solution of the p.d.e.

$$Lu = f,$$

on the domain Ω subject to homogeneous boundary conditions on the boundary $\partial\Omega$.

Using the adjoint formulation, the identical functional takes the form (v, f) where v is the solution of the adjoint p.d.e.

$$L^* v = g,$$

plus appropriate homogeneous adjoint b.c.'s. The adjoint operator L^* is defined by the identity

$$(V, LU) = (L^* V, U),$$

which must hold for all functions V, U satisfying the respective homogeneous boundary conditions. Given the definitions, the proof of the equivalence of the two forms of the problem is trivial

$$(v, f) = (v, Lu) = (L^* v, u) = (g, u).$$

3.2. EXAMPLES

To illustrate the construction of the adjoint operator and boundary conditions, let us consider the one-dimensional convection-diffusion equation

$$Lu \equiv \frac{\mathrm{d}u}{\mathrm{d}x} - \varepsilon \frac{\mathrm{d}^2 u}{\mathrm{d}x^2}, \quad 0 < x < 1,$$

subject to the homogeneous boundary conditions $u(0) = u(1) = 0$.

Using integration by parts, for any twice-differentiable function v we have

$$\begin{aligned}
(v, Lu) &= \int_0^1 v \left(\frac{\mathrm{d}u}{\mathrm{d}x} - \varepsilon \frac{\mathrm{d}^2 u}{\mathrm{d}x^2} \right) \mathrm{d}x \\
&= \int_0^1 u \left(-\frac{\mathrm{d}v}{\mathrm{d}x} - \varepsilon \frac{\mathrm{d}^2 v}{\mathrm{d}x^2} \right) \mathrm{d}x + \left[vu - \varepsilon v \frac{\mathrm{d}u}{\mathrm{d}x} + \varepsilon u \frac{\mathrm{d}v}{\mathrm{d}x} \right]_0^1 \\
&= \int_0^1 u \left(-\frac{\mathrm{d}v}{\mathrm{d}x} - \varepsilon \frac{\mathrm{d}^2 v}{\mathrm{d}x^2} \right) \mathrm{d}x + \left[-\varepsilon v \frac{\mathrm{d}u}{\mathrm{d}x} \right]_0^1 .
\end{aligned}$$

For the integral term to equal the inner product (g, u) in the adjoint identity, we need to define the adjoint operator to be

$$L^* v = -\frac{\mathrm{d}v}{\mathrm{d}x} - \varepsilon \frac{\mathrm{d}^2 v}{\mathrm{d}x^2},$$

Table I. Various operators and their adjoints.

Operator	Adjoint
$\dfrac{\mathrm{d}u}{\mathrm{d}x} - \varepsilon\dfrac{\mathrm{d}^2u}{\mathrm{d}x^2}$	$-\dfrac{\mathrm{d}v}{\mathrm{d}x} - \varepsilon\dfrac{\mathrm{d}^2v}{\mathrm{d}x^2}$
$\nabla\cdot(k\nabla u)$	$\nabla\cdot(k\nabla v)$
$\dfrac{\partial u}{\partial t} - \dfrac{\partial^2 u}{\partial x^2}$	$-\dfrac{\partial v}{\partial t} - \dfrac{\partial^2 v}{\partial x^2}$
$\dfrac{\partial u}{\partial t} + \dfrac{\partial u}{\partial x}$	$-\dfrac{\partial v}{\partial t} - \dfrac{\partial v}{\partial x}$

and to eliminate the boundary term the adjoint b.c.'s must be

$$v(0) = v(1) = 0.$$

Note the reversal in sign of the first derivative in the adjoint operator; this implies a reversal in the convection direction.

Table I lists a number of other differential operators and their adjoints. Note the changes of sign which occur due to the integration by parts. This produces a reversal of causality in time-varying problems so that, for example, the adjoint parabolic operator is well-posed only if one starts with 'initial data' at the final time and then integrates backwards in time towards the initial time of the original problem.

3.3. PHYSICAL INTERPRETATION

The physical significance of adjoint variables can again be understood by considering Green's functions and their effect on the inner product of interest.

The solution of the p.d.e. $Lu = f$ is

$$u(x) = \int_\Omega G(x, x')\, f(x')\, \mathrm{d}x',$$

where $G(x, x')$ is the Green's function. Therefore,

$$\begin{aligned}\int_\Omega g^T(x) u(x)\, \mathrm{d}x &= \int_\Omega \int_\Omega g^T(x) G(x, x')\, f(x')\, \mathrm{d}x\, \mathrm{d}x' \\ &= \int_\Omega v^T(x')\, f(x')\, \mathrm{d}x',\end{aligned}$$

where

$$v^T(x') = \int_\Omega g^T(x) G(x, x')\, \mathrm{d}x.$$

Thus, the adjoint variables at a particular point correspond to the functional evaluated using the Green's function for the same point.

3.4. BOUNDARY TERMS

So far, we have assumed that the original problem has homogeneous b.c.'s and the objective function consists only of an inner product over the whole domain and not a boundary integral. More generally, boundary integral terms in the primal objective function lead to inhomogeneous b.c.'s for the adjoint, while inhomogeneous b.c.'s for the primal problem lead to boundary terms in the adjoint functional [15]. The general form of the adjoint identity is

$$(V, LU)_\Omega + (C^*V, BU)_{\partial\Omega} = (L^*V, U)_\Omega + (B^*V, CU)_{\partial\Omega}$$

for all functions U, V, with the notation $(.,.)_{\partial\Omega}$ denoting an inner product over the boundary. B and C are both boundary operators (possibly involving normal derivatives) given in the definition of the original problem. B^* and C^* are the corresponding adjoint boundary operators which can be found by integration by parts.

Using this general adjoint identity, it follows immediately that

$$(v, f)_\Omega + (C^*v, f_2)_{\partial\Omega} = (g, u)_\Omega + (g_2, Cu)_{\partial\Omega}$$

when

$$\begin{aligned} Lu &= f \text{ in } \Omega, \quad \text{and} \quad Bu = f_2 \text{ on } \partial\Omega, \\ L^*v &= g \text{ in } \Omega, \quad \text{and} \quad B^*v = g_2 \text{ on } \partial\Omega. \end{aligned}$$

There are some restrictions on what can be imposed as b.c.'s and objective functions. The analysis is complicated (see [15, 28] for details) but it reveals that on a solid surface, the boundary integral term in the objective function must be a weighted integral of the linear perturbation in the pressure when using the Euler equations. Similarly, for the Navier–Stokes equations it must be a weighted integral of the linear perturbation in the normal and tangential forces on the surface, and either the heat flux or the surface temperature (depending whether one is specifying the surface temperature or adiabatic conditions, respectively).

3.5. GEOMETRIC EFFECTS

Perhaps the most complicated part of the continuous approach to design is the manner in which design variable perturbations produce the source term f for the linearised p.d.e. and the inhomogeneous term f_2 for the linearised b.c.'s.

We will outline two approaches, both of which use curvilinear coordinates (ξ, η) in two dimensions. Writing the Euler equations in their usual vector form as

$$\frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = 0,$$

when transformed to the curvilinear coordinates they become

$$\frac{\partial}{\partial\xi}\left(F\frac{\partial y}{\partial\eta}-G\frac{\partial x}{\partial\eta}\right)+\frac{\partial}{\partial\eta}\left(-F\frac{\partial y}{\partial\xi}+G\frac{\partial x}{\partial\xi}\right)=0.$$

In the approach used by Jameson, the curvilinear coordinates correspond to grid lines of a structured grid, with the airfoil surface being defined as $\eta = 0$ [26]. A small perturbation $\tilde{\alpha}$ to a design parameter produces changes such as

$$\begin{aligned} F &\longrightarrow F+\frac{\partial F}{\partial U}\frac{\mathrm{d}U}{\mathrm{d}\alpha}\tilde{\alpha},\\ \frac{\partial x}{\partial\eta} &\longrightarrow \frac{\partial x}{\partial\eta}+\frac{\partial^2 x}{\partial\eta\partial\alpha}\tilde{\alpha}. \end{aligned}$$

Terms *not* depending on $\tilde{\alpha}$ all cancel, and terms depending on $\tilde{\alpha}^2$ are neglected. Hence, we get the linearised equations

$$\begin{aligned} &\frac{\partial}{\partial\xi}\left(\left(A\frac{\partial y}{\partial\eta}-B\frac{\partial x}{\partial\eta}\right)u\right)+\frac{\partial}{\partial\eta}\left(\left(-A\frac{\partial y}{\partial\xi}+B\frac{\partial x}{\partial\xi}\right)u\right)\\ &\quad=-\frac{\partial}{\partial\xi}\left(F\frac{\partial^2 y}{\partial\eta\partial\alpha}-G\frac{\partial^2 x}{\partial\eta\partial\alpha}\right)-\frac{\partial}{\partial\eta}\left(-F\frac{\partial^2 y}{\partial\xi\partial\alpha}+G\frac{\partial^2 x}{\partial\xi\partial\alpha}\right), \end{aligned}$$

where

$$A=\frac{\partial F}{\partial U},\quad B=\frac{\partial G}{\partial U},\quad u=\frac{\mathrm{d}U}{\mathrm{d}\alpha}.$$

The boundary condition on an inviscid wall is that there is no flow normal to the surface $\eta = 0$. This remains true as α changes but one needs to consider the linearised perturbation to the unit normal, which eventually leads to the inhomogeneous boundary term f_2.

For complex geometries, it is often not possible to generate structured grids in which the surface corresponds to $\eta = \text{const}$. Instead, one can generalise the above approach by defining

$$\begin{aligned} x(\xi,\eta) &= \xi+\tilde{\alpha}X(\xi,\eta),\\ y(\xi,\eta) &= \eta+\tilde{\alpha}Y(\xi,\eta), \end{aligned}$$

so that $(x, y) \equiv (\xi, \eta)$ when $\tilde{\alpha} = 0$, and $X(\xi, \eta)$, $Y(\xi, \eta)$ are smooth functions matching the surface deformation so that the surface remains fixed in (ξ, η) coordinates as α changes. This leads to the linearised equation

$$\frac{\partial}{\partial\xi}(Au)+\frac{\partial}{\partial\eta}(Bu)=-\tilde{\alpha}\frac{\partial}{\partial\xi}\left(F\frac{\partial Y}{\partial\eta}-G\frac{\partial X}{\partial\eta}\right)-\tilde{\alpha}\frac{\partial}{\partial\eta}\left(-F\frac{\partial Y}{\partial\xi}+G\frac{\partial X}{\partial\xi}\right).$$

This equation can then be approximated using an unstructured grid in the (ξ, η) domain, which is the same as the (x, y) domain for the unperturbed geometry.

Boundary conditions are handled in the same way as in Jameson's treatment, taking account of the perturbation to the unit normal as the surface geometry changes.

3.6. Other issues

With the continuous adjoint approach, after linearising the original flow equations and integrating by parts to obtain the adjoint formulation of the problem, there is then total freedom as to how one discretises the adjoint p.d.e. Indeed, without making recourse to the discrete approach, where the adjoint implementation is fixed by the primal discretisation, there is even some ambiguity as to how one should implement the inviscid adjoint fluxes for the Euler equations. In principle, the adjoint discretisation may be developed without regard for the discretisation of the nonlinear flow problem. Of course the standard issues or accuracy, stability and convergence remain critical to the success of the iterative solution process.

When considering shocked Euler flows, then in the analytic formulation, the shocks need to be treated as discontinuities across which the Rankine–Hugoniot shock jump relations are enforced [16]. This treatment leads to the result that the adjoint variables are continuous across the shock and that an additional adjoint boundary condition must be imposed along the length of the shock. Imposing such a b.c. would be complicated, as it would require the automatic identification of the shock location in the nonlinear flow calculation. Quasi-1D results have demonstrated that the continuous implementation naturally leads to satisfaction of the adjoint boundary condition at the shock [16]. In practice, researchers using the continuous adjoint approach do not enforce this b.c., and their results indicate no difficulties as a consequence.

The observations about the limitations of the discrete adjoint approach apply equally to the continuous adjoint approach. There is one additional point that needs to be made regarding the optimisation process. The continuous adjoint approach yields a discrete approximation to the gradient of the analytic objective function with respect to each of the design variables. This will not be exactly equal to the gradient of the discrete approximation to the objective function. Therefore, there is a slight inconsistency between the discrete objective function and the computed gradient. As a result, the optimisation process will fail to converge further once the solution is near a local minimum.

4. Relative Advantages of Two Approaches

In the previous two sections we have gone through, in some detail, the formulation of the discrete and continuous adjoint approaches currently in use by different researchers. The difference between the two approaches is shown schematically in Figure 1. In both cases one ends up with a set of discrete adjoint equations. In the fully-discrete approach one starts by discretising the nonlinear p.d.e.; these equations are then linearised and transposed. In the continuous adjoint approach,

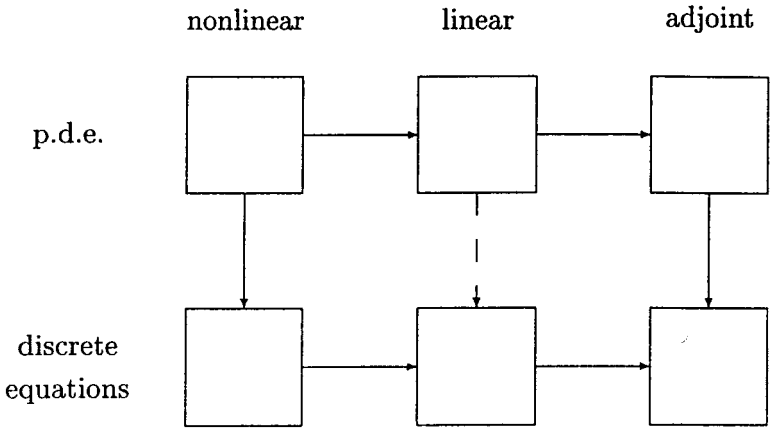

Figure 1. Alternative approaches to forming discrete adjoint equations.

the discretisation is the final step, after first linearising and forming the adjoint problem. One could even follow an intermediate path, linearising the original equations, discretising them and then taking the transpose. In principle, if each of the steps is performed correctly, and all of the solutions are sufficiently smooth (e.g. no shocks) then in the limit of infinite grid resolution all three approaches should be consistent and converge to the correct analytic value for the gradient of the objective function.

However, there are important conceptual differences between the different approaches, and for finite resolution grids there will be differences in the computed results. Here we attempt to summarise what we see as being the advantages and disadvantages of the two approaches. This assessment is based on our joint experience in developing an adjoint Navier–Stokes code using the discrete approach, and the experience of the second author in working with Jameson to develop an adjoint Navier–Stokes code by the continuous approach [28].

The advantages of the fully-discrete approach are:

- The exact gradient of the discrete objective function is obtained.
  This ensures that the optimisation process can converge fully. It also provides a convenient check on the correctness of the programming implementation; with the continuous approach one does not know whether a slight disagreement is a consequence of the inexact gradient or a possible programming error.
- Creation of the adjoint program is conceptually straightforward.
  In the future this should enable the almost automatic creation of adjoint programs using AD software. This benefit includes the iterative solution process since the transposed matrix has the same eigenvalues as the original linear matrix and so the same iterative solution method is guaranteed to converge.

On the other hand, the advantages of the continuous approach are:

- The physical significance of adjoint variables and the role of adjoint b.c.'s is much clearer.
  Only by constructing the adjoint flow equations can one develop a good understanding of the nature of adjoint solutions, such as the continuity at shocks, the logarithmic singularity at a sonic point in quasi-1D flows but not in 2D or 3D (in general) and the inverse square-root singularity along the stagnation streamline upstream of an airfoil in 2D [15].
- The adjoint program is simpler and requires less memory.
  Because one is free to discretise the adjoint p.d.e. in any consistent way, the adjoint code can be much simpler. However, our experience has been that even when following a continuous approach, it is advantageous to consult the discrete formulation so as to choose an appropriate discretisation for the continuous adjoint equations. It is also generally the case that continuous adjoint solvers require less memory than the fully-discrete codes, but this difference is not substantial if pre-computation and storage of the linearised matrix is avoided when implementing the discrete method.

It remains an open question as to which approach is better when there are non-linear discontinuities such as shocks. For quasi-1D Euler calculations, for which we have derived the analytic solution of the adjoint equations [16], both approaches give numerical results which converge to the analytic solution. For the discrete approach, this follows because the integrated pressure can be proved to be predicted with second-order accuracy [14]. The linearised discretisation should therefore yield perturbations to the integral of pressure that are at least first-order accurate. The discrete adjoint formulation, which is constructed using this linearised operator, must therefore behave correctly to first order at the shock. For the continuous approach, in the absence of explicit enforcement of the correct adjoint b.c. at the shock, the correct asymptotic behaviour can be explained as the effect of numerical smoothing, given that the correct analytic solution is the only smooth solution at the shock [16].

In 2D and 3D there is no proof of second-order accuracy for quantities such as lift and drag, and there is a discontinuity in the gradient of the adjoint variables at the location of the shock. Therefore it remains an open question as to whether either approach will give a consistent approximation to the gradient of the objective function in the limit of infinite grid resolution. However, practical results for applications with weak shocks suggest that any inconsistency must be small.

Although we have aimed to be objective in our assessment of the relative advantages of the two approaches, it should be noted that we are advocates of the discrete approach. An advocate of the continuous approach may place a different emphasis on the above observations and hence reach a different conclusion. Certainly, both methods have performed well in practice, and it remains to be seen whether either

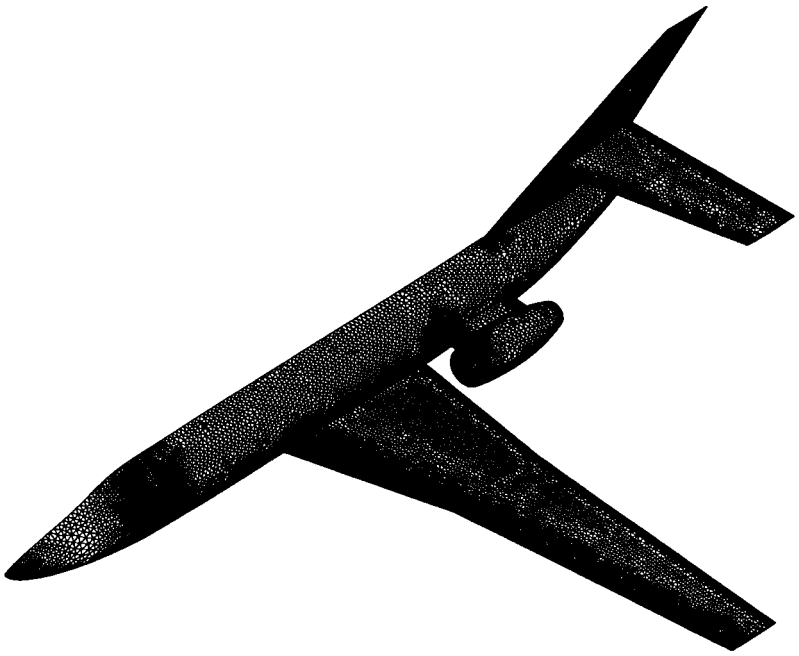

Figure 2. Initial surface grid for aircraft wing design [10].

approach will demonstrate compelling advantages over the other in terms of design performance. Ultimately, the final choice may always remain, to some extent, a matter of personal taste.

5. Applications

Results from a paper by Elliot and Peraire [10] show the use of a discrete adjoint implementation for design optimisation on unstructured grids. The main application considered is the wing optimisation of a business jet for which the surface grid of the baseline configuration is shown in Figure 2.

Simple algebraic functions are used to define six design perturbation modes for the wing surface; care was taken to ensure compatible perturbations to grid points on the fuselage. A linearised version of the method of springs is used to create the grid deformations in the interior. The implementation is based on the discrete adjoint approach, using BFGS optimisation, and both multigrid and parallel computing to reduce the execution time.

The objective function is the mean-square deviation from a target pressure distribution corresponding to a 'clean wing' in the absence of the rear-mounted engine nacelle and pylon. Two design iterations are taken, decreasing the objective function by 75%. Figures 3 and 4 show the evolution of the wing geometry and pressure distributions, respectively.

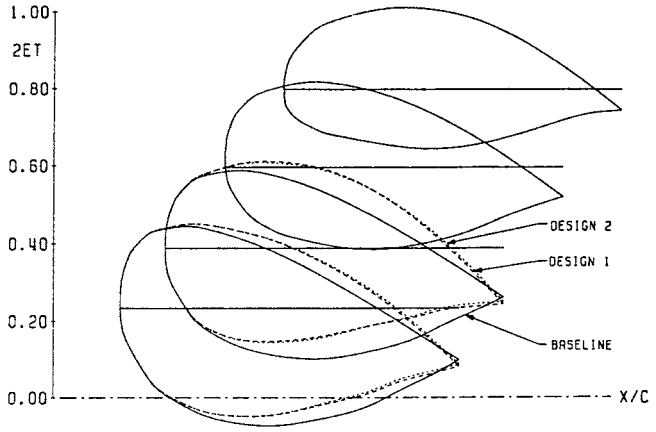

Figure 3. Evolution of the wing geometry during design [10].

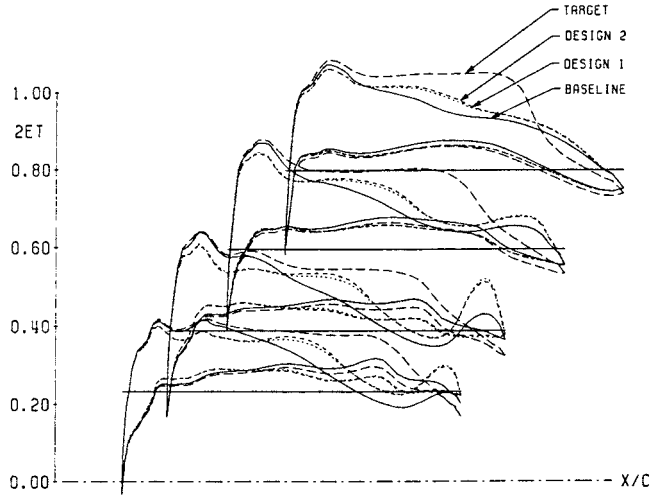

Figure 4. Evolution of the pressure distribution on the wing [10].

Another example of the adjoint approach to design is provided by the work of Reuther and co-authors [42], who perform a transonic multipoint wing design for a business jet configuration of the type shown in Figure 5. Here, the objective is to minimize drag for several flight conditions simultaneously.

This work employs a continuous adjoint formulation on a structured multiblock mesh using parallel multigrid flow and adjoint solvers. The wing surface is parameterised with 18 Hicks–Henne bump functions [21] at each of five span stations and a total of 30 constraints are imposed on maximum thickness, spar thickness, leading edge bluntness and trailing edge angle. The results were obtained after five design iterations using the optimization package NPSOL [19] during which the interior grid points were perturbed using WARP-MB [41].

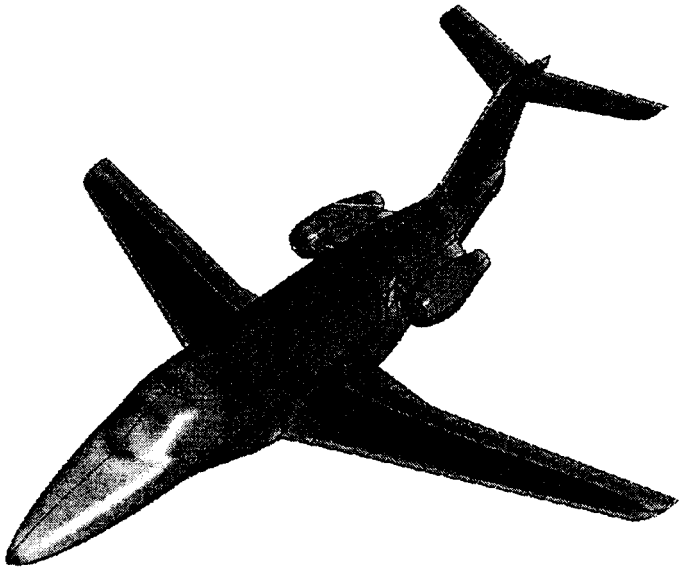

Figure 5. Business jet configuration [42].

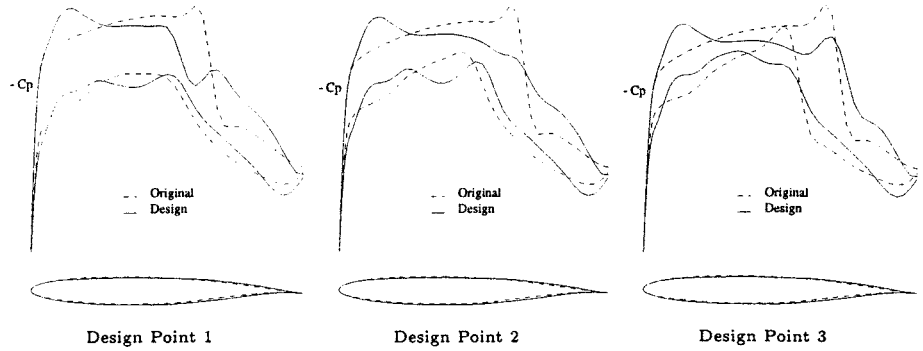

Figure 6. Multipoint drag minimisation at fixed lift. Pressure distributions at the $z = 0.475$ span station for the three design points described in Table II [42].

The initial configuration was designed for cruise at $M = 0.8$ and $C_L = 0.3$ and the three new design points are summarised in Table II. The original and designed pressure distributions are displayed at a single span station for each of the three design points in Figure 6. The shock strength has been substantially reduced in all cases, leading to the drag reductions described in Table II. While a single point design would achieve lower drag at the specified cruise conditions, the multipoint design has the advantage of maintaining better off-design performance [42].

Table II. Multipoint drag reduction.

Mach	C_L	Original C_D^*	Design C_D^*
0.81	0.35	1.00257	0.85413
0.82	0.30	1.00000	0.77915
0.83	0.25	1.08731	0.76836

*Drag coefficients normalised relative to original drag at central design point [42].

6. Conclusions

The development of design environments is currently a major focus of research in computational engineering. As part of this effort, adjoint methods offer the ability to efficiently compute linear design sensitivities when there are a large number of design variables.

In reviewing the fundamental theory, we began with the linear algebra perspective from which these ideas are most easily understood. For conceptual as well as pragmatic reasons, we believe that the 'discrete' numerical implementations which follow this approach have a number of advantages over those based on the alternate 'continuous' approach. On the other hand, a sound grasp of the adjoint p.d.e. theory is essential to understanding the physical significance of the adjoint variables and their behaviour at key points in the flow field, such as at shocks.

It is hoped that this overview of the theory and of a number of important implementation issues will help others to develop adjoint techniques as an integral part of engineering design systems. Although the focus of this paper has been on aeronautical design, the ideas are equally relevant to any area of engineering design involving large numbers of continuous design variables.

References

1. Anderson, K., Newman, J., Whitfield, D. and Nielsen, E., Sensitivity analysis for the Navier–Stokes equations on unstructured grids using complex variables. AIAA Paper 99-3294 (1999).
2. Anderson, W.K. and Bonhaus, D.L., Airfoil design on unstructured grids for turbulent flows. *AIAA J.* **37**(2) (1999) 185–191.
3. Anderson, W.K. and Venkatakrishnan, V., Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation. AIAA Paper 97-0643 (1997).
4. Baysal, O. and Eleshaky, M.E., Aerodynamic design optimization using sensitivity analysis and computational fluid dynamics. *AIAA J.* **30**(3) (1992) 718–725.
5. Bischof, C., Carle, A., Corliss, G., Griewank, A. and Hoveland, P., ADIFOR: Generating derivative codes from Fortran programs. *Scientific Programming* **1**(1) (1992) 11–29.
6. Cabuk, H., Shung, C.H. and Modi, V., Adjoint operator approach to shape design for internal incompressible flow. In: Dulikravich, G.S. (ed.), *Proceedings 3rd International Conference on Inverse Design and Optimization in Engineering Sciences* (1991) pp. 391–404.
7. Carle, C., Fagan, M. and Green, L.L., Preliminary results from the application of automated code generation to CFL3D. AIAA Paper 98-4807 (1998).

8. Dadone, A. and Grossman, B., CFD design problems using progressive optimization. AIAA Paper 99-3295 (1999).
9. Elliott, J., Aerodynamic optimization based on the Euler and Navier–Stokes equations using unstructured grids. Ph.D. Thesis, MIT, Department of Aeronautics and Astronomy (1998).
10. Elliott, J. and Peraire, J., Aerodynamic design using unstructured meshes. AIAA Paper 96-1941 (1996).
11. Elliott, J. and Peraire, J., Practical 3D aerodynamic design and optimization using unstructured meshes. *AIAA J.* **35**(9) (1997) 1479–1485.
12. Faure, C., Splitting of algebraic expressions for automatic differentiation. In: Griewank, A. (ed.), *Proceedings of the Second SIAM International Workshop on Computational Differentiation*. SIAM, Philadelphia, PA (1996).
13. Gilbert, J., Le Vey, G. and Masse, J., La différentiation automatique de fonctions représentées par des programmes. INRIA Rapport de Recherche 1557 (1991).
14. Giles, M.B., Analysis of the accuracy of shock-capturing in the steady quasi-1D Euler equations. *Comput. Fluid Dynamics J.* **5**(2) (1996) 247–258.
15. Giles, M.B. and Pierce, N.A., Adjoint equations in CFD: Duality, boundary conditions and solution behaviour. AIAA Paper 97-1850 (1997).
16. Giles, M.B. and Pierce, N.A., On the properties of solutions of the adjoint Euler equations. In: Baines, M. (ed.), *Numerical Methods for Fluid Dynamics VI*. ICFD (1998) pp. 1–16.
17. Giles, M.B. and Pierce, N.A., Improved lift and drag estimates using adjoint Euler equations. AIAA Paper 99-3293 (1999).
18. Gill, P.E., Murray, W. and Wright, M.H., *Practical Optimization*. Academic Press, New York (1981).
19. Gill, P.E., Murray, W., Saunders, M.A. and Wright, M.H., *User's Guide for NPSOL (Version 4.0). A FORTRAN Package for Nonlinear Programming*. Department of Operations Research, TR SOL86-2, Stanford University, Stanford, CA (1986).
20. Griewank, A., On automatic differentiation. In: *Mathematical Programming '88*. Kluwer Academic Publishers, Dordrecht (1989) pp. 83–108.
21. Hicks, R.M. and Henne, P.A., Wing design by numerical optimization. *J. Aircraft* **15** (1978) 407–412.
22. Huffman, W.P., Melvin, R.G., Young, D.P., Johnson, F.T., Bussoletti, J.E., Bieterman, M.B. and Himes, C.L., Practical design and optimization in computational fluid dynamics. AIAA Paper 93-3111 (1993).
23. Newman III, J.C., Taylor III, A.C., Barnwell, R.W., Newman, P.A. and Hou, G.J.-W., Overview of sensitivity analysis and shape optimization for complex aerodynamic configurations. *J. Aircraft* **36**(1) (1999) 87–96.
24. Jameson, A., Aerodynamic design via control theory. *J. Sci. Comput.* **3** (1988) 233–260.
25. Jameson, A., Optimum aerodynamic design using CFD and control theory. AIAA95-1729-CP (1995).
26. Jameson, A., Optimum aerodynamic design using control theory. In: Hafez, M. and Oshima, K. (eds), *Computational Fluid Dynamics Review*, Annual Book Series. John Wiley & Sons, New York (1995) pp. 495–528.
27. Jameson, A., Re-engineering the design process through computation. *J. Aircraft* **36**(1) (1999) 36–50.
28. Jameson, A., Pierce, N.A. and Martinelli, L., Optimum aerodynamic design using the Navier–Stokes equations. AIAA Paper 97-0101 (1997).
29. Korivi, V.M., Taylor III, A.C. and Hou, G.W., Sensitivity analysis, approximate analysis and design optimization for internal and external viscous flows. AIAA Paper 91-3083 (1991).
30. 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 (1995).

31. Lions, J.L., *Optimal Control of Systems Governed by Partial Differential Equations*. Springer-Verlag, Berlin (1971). Translated by S.K. Mitter.
32. Mohammadi, B., Optimal shape design, reverse mode of automatic differentiation and turbulence. AIAA Paper 97-0099 (1997).
33. Mohammadi, B., Practical applications to fluid flows of automatic differentiation for design problems. VKI Lecture Series 1997-05 on Inverse Design (1997).
34. Nielsen, E. and Anderson, W.K., Aerodynamic design optimization on unstructured meshes using the Navier–Stokes equations. AIAA Paper 98-4809 (1998).
35. Pierce, N.A. and Giles, M.B., Adjoint recovery of superconvergent functionals from approximate solutions of partial differential equations. Technical Report NA98/18, Oxford University Computing Laboratory (1998).
36. Pierce, N.A. and Giles, M.B., Adjoint recovery of superconvergent functionals from PDE approximations. *SIAM Rev.*, in press.
37. Pironneau, O., On optimum design in fluid mechanics. *J. Fluid Mech.* **64** (1974) 97–110.
38. Rausch, R.D., Batina, J.T. and Yang, H.T.Y., Three-dimensional time-marching aeroelastic analyses using an unstructured-grid Euler method. *AIAA J.* **31**(9) (1993) 1626–1633.
39. Reuther, J. and Jameson, A., Control based airfoil design using the Euler equations. AIAA Paper 94-4272-CP (1994).
40. 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 (1996).
41. Reuther, J., Jameson, A., Alonso, J.J., Remlinger, M.J. and Saunders, D., Constrained multipoint aerodynamic shape optimisation using and adjoint formulation and parallel computers, Part 1. *J. Aircraft* **36** (1) (1999) 51–60.
42. Reuther, J., Jameson, A., Alonso, J.J., Remlinger, M.J. and Saunders, D., Constrained multipoint aerodynamic shape optimisation using and adjoint formulation and parallel computers, Part 2. *J. Aircraft* **36**(1) (1999) 61–74.
43. Squire, S. and Trapp, G., Using complex variables to estimate derivatives of real functions. *SIAM Rev.* **40**(1) (1998) 110–112.
44. Ta'asan, S., Kuruvila, G. and Salas, M.D., Aerodynamics design and optimization in one shot. AIAA Paper 92-0025 (1992).
45. Thompson, J.F., Warsi, Z.U.A. and Mastin, C.W., *Numerical Grid Generation, Foundations and Applications*. Elsevier, Amsterdam (1985).
46. Veditti, D. and Darmofal, D., A multilevel error estimation and grid adaptive strategy for improving the accuracy of integral outputs. AIAA Paper 99-3292 (1999).