Report no. 01/15

Algorithm Developments for Discrete Adjoint Methods

Michael B. Giles
giles@comlab.ox.ac.uk
Mihai C. Duta
mcd@comlab.ox.ac.uk
Jens-Dominik Müller
jdm@comlab.ox.ac.uk
Oxford University Computing Laboratory, Oxford OX1 3QD

Niles A. Pierce
niles@caltech.edu
Applied and Computational Mathematics, California Institute of Technology, Pasadena, CA 91125, USA

This paper presents a number of algorithm developments for adjoint methods using the 'discrete' approach in which the discretisation of the non-linear equations is linearised and the resulting matrix is then transposed. With a new iterative procedure for solving the adjoint equations, exact numerical equivalence is maintained between the linear and adjoint discretisations. The incorporation of strong boundary conditions within the discrete approach is discussed, as well as a new application of adjoint methods to linear unsteady flow in turbomachinery.

Oxford University Computing Laboratory
Numerical Analysis Group
Wolfson Building
Parks Road
Oxford, England OX1 3QD

August, 2001

1 Introduction

There is a long history of the use of adjoint equations in optimal control theory [26]. In fluid dynamics, the first use of adjoint equations for design was by Pironneau [33], but within the field of aeronautical computational fluid dynamics, the use of adjoint equations for design optimisation has been pioneered by Jameson [19, 20, 22] for the potential flow, Euler and Navier-Stokes equations. The complexity of the applications within these papers has also progressed from 2D airfoil optimisation, to 3D wing design and finally to complete aircraft configurations [21, 34, 35]. A number of other research groups have also developed adjoint CFD codes [24, 39, 4, 3, 8] using the same 'continuous' approach in which the first step is to linearise the original partial differential equations. The adjoint p.d.e. and appropriate boundary conditions are then formulated, and finally the equations are discretised. While this minimises the memory requirements and the CPU cost per iteration, it requires one to develop an appropriate iterative solution procedure, and this may not give as good a convergence rate as the original nonlinear code. In addition, the debugging and validation of the adjoint code is complicated by the lack of a suite of benchmark testcases.

The alternative 'discrete' approach, which we use, takes a discretisation of the Navier-Stokes equations, linearises the discrete equations and then uses the transpose of the linear operator to form the adjoint problem. This approach has been developed by Elliott [11, 10], Anderson [32, 1], Mohammadi [29] and Kim [23]. The main advantage of this approach, in our opinion, is that the code development becomes a more straightforward process. The linearisation of the nonlinear discrete equations can either be performed manually or by automatic differentiation software and the linear code can be validated by direct comparison with the nonlinear code. Similarly, since the adjoint code is obtained by transposing the linear operator, it must yield exactly the same values for the linearised objective function, and so can be validated against the linear code.

For an excellent review of research on both continuous and discrete adjoint design methods, see the paper by Newman *et al* [31].

In this paper we contribute to the development and understanding of discrete adjoint methods in five respects:

- Discussion of the implementation of the adjoint code in a way which minimises the memory and CPU requirements, and can be automated using automatic differentiation tools;
- Development of an adjoint multigrid iteration procedure with preconditioned timestepping which maintains exact equivalence between the linear and adjoint codes at all times during the evolution of their respective solutions;
- A detailed discussion of the imposition of strong boundary conditions and the inclusion of viscous stresses in objective functions and the consequence for the formulation of the adjoint code;
- Development of a harmonic adjoint code which is the counterpart of a linear unsteady code for a single frequency of unsteadiness, and which has applications in

turbomachinery blade design for reduced vibration due to forced response;

- A numerical investigation indicating the potential for problems with strong shocks.

This research forms part of the development of the HYDRA suite of codes. The foundation is a nonlinear code which approximates the Reynolds-averaged Navier-Stokes equations on unstructured hybrid grids, using an edge-based discretisation. The solution procedure uses Runge-Kutta time-marching accelerated by Jacobi preconditioning and multigrid [30], with dual-timestepping for unsteady flows.

The second code in the suite is for the linear analysis of unsteady flows. This is based on a linearisation of the unsteady flow equations around the steady-state flow conditions calculated by the nonlinear code. Due to linearity, unsteady periodic flows can be decomposed into a sum of harmonic terms, each of which can be computed independently. Thus, the linear harmonic code considers just one particular frequency of unsteadiness, resulting in a formulation in which the purpose is to compute a complex flow solution which represents the amplitude and phase of the unsteady flow. This is explained in greater detail later in this paper.

The third code is the steady adjoint code, which again is based on a linearisation of the flow equations around the nonlinear steady-state flow conditions. The fourth code, which is an extension of the third, is the adjoint counterpart of the linear harmonic code. It is the development of these codes that is the subject of this paper.

2 Discrete adjoint formulation

We start by considering the discrete nonlinear Euler equations with a weak imposition of boundary conditions on solid walls through the specification of zero mass flux through faces on the surface. If the far-field boundary conditions are also imposed through far-field fluxes then the discrete system of equations which is solved is of the form

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

Here U is the vector of flow field variables, α represents one or more design variables which control the geometry of the airfoil or wing (and hence the grid coordinates) and $R(U)$ represents the discrete flux residuals which are driven to zero by the iterative solution process.

If there is just one design variable, then linearising the steady-state equations with respect to a change in that design variable yields

$$Lu = f,$$

where

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

The corresponding perturbation in a nonlinear objective function $J(U, \alpha)$ is

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

where

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

In the adjoint approach, this same quantity can be obtained by evaluating

$$\widetilde{J} = v^T f + \frac{\partial J}{\partial \alpha},$$

where the adjoint solution v satisfies the equation

$$L^T v = g.$$

The equivalence of this formulation comes from the following identity.

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

If there are many design variables (each giving rise to a different vector f) and only one objective (yielding a single vector g), then the benefit of the adjoint approach is that the objective sensitivity $\tilde{J}$ can be obtained following a single evaluation of v instead of separate evaluations of u for each f.

3 Implementation of adjoint discretisation

In the implementation, the linear operator L is split into two parts,

$$L\,u = C\,u + D\,u. \tag{3.1}$$

The first part represents the convective fluxes due to a Galerkin finite element discretisation. The second part represents the smoothing fluxes (to which the viscous fluxes are added later for the Navier-Stokes equations). The operator D can be further broken down into the product of two operators,

$$D\,u = V\,G\,u,$$

where G computes the gradient and a pseudo-Laplacian of u at each node, in addition to u itself.

The corresponding adjoint operator is

$$L^T v = C^T v + D^T v,$$

with

$$D^T v = G^T\,V^T v,$$

indicating that the adjoint gradient routine is applied *after* the adjoint smoothing routine, which at first seems counter-intuitive.

At an even more detailed level, the action of each of the operators C, V and G is computed by a loop over all edges in the unstructured grid. Therefore, taking Cu as

an example, we can express it as a sum of elemental edge matrices whose only non-zero entries corresponds to the two nodes at either end of the edge,

$$C\,u = \sum_e C_e u.$$

The adjoint version of this is simply

$$C^T v = \sum_e C_e^T v,$$

corresponding to a similar loop over all edges.

For the convective fluxes, it is easy to compute the edge product $C_e^T v$ directly without explicitly forming the matrix C_e. The transposed gradient operator G^T is also easily formulated. The product $V^T v$ presents greater difficulties. Elliott [11, 10] precomputed and stored the non-zero entries in the elemental matrices V_e, and then evaluated the matrix-vector products $V_e^T v$. However, the storage of these matrices for each edge requires a substantial amount of memory. Anderson [1] avoided the memory cost by recomputing the matrices during each iteration, but this greatly increases the CPU cost.

To minimise both the memory and CPU requirements, it is necessary to calculate the edge product $V_e^T v$ directly, as with $C_e^T v$. The difficulty is in working out how best to do this. One approach is to use AD (Automatic Differentiation) software such as Odyssée [12], ADIFOR [5, 7] or TAMC [13]. In forward mode, AD software takes the original nonlinear code and then uses the basic rules of linearisation to construct the code to evaluate $V_e u$. In reverse mode, it produces the code to calculate $V_e^T v$; 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 [16].

Mohammadi used Odyssée to generate much of his adjoint code [29] but a lot of hand-coding was still required. In our work we have written the adjoint code manually, but following many of the techniques of automatic differentiation. To simplify the expressions for the partial derivatives, we chose to use the primitive variables (density, velocity and pressure) as our working variables, rather than the usual conservative variables. The equations are still in conservative form so this choice of working variables has no effect on the final solution.

The memory requirements for the adjoint code are 20-30% greater than for the non-linear code, depending on the grid that is used. The CPU cost per iteration is only 10-20% greater than for the nonlinear code, with the increased cost of evaluating the adjoint residuals partially offset by the fact that the Jacobian for the preconditioning remains fixed.

Another important point concerns the evaluation of the term f, which is the source term for the linear perturbation equations, and also appears in the linearised 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 Taylor series expansion method [37] used by Anderson and co-workers [2]. The essence of the idea is that

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

In this equation, $R(U,\alpha)$ has been taken to be a complex analytic function, and the notation $\mathcal{I}\{\ldots\}$ denotes the imaginary part of a complex quantity. The equation itself is an immediate consequence of a Taylor series expansion. The convergence to the limiting value is second order in ϵ so numerical evaluation with $\epsilon < 10^{-8}$ yields double precision accuracy. In practice, we use $\epsilon = 10^{-20}$. Unlike the usual finite difference approximation of a linear sensitivity, there is no cancellation effect from the subtraction of two quantities of similar magnitude, and therefore 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 the intrinsic functions `min,max,abs`.

We have also found this complex variable method to be extremely helpful during program development. Because we have also written a linear perturbation code, we have used it verify that each of the linear flux subroutines is consistent with the original nonlinear flux subroutines, by checking the identity

$$Lu = \lim_{\epsilon \to 0} \frac{\mathcal{I}\left\{R(U+\mathrm{i}\epsilon u, \alpha)\right\}}{\epsilon},$$

for arbitrary choices of u. The l.h.s. is computed by the linear flux routines, and the r.h.s. is computed by applying the complex variable method to the nonlinear flux routines. Having performed these checks, we then verified that the adjoint flux routines were consistent with the linear routines by checking that the identity $u^T(L^T v) = v^T(Lu)$ holds for any u, v.

If one were developing an adjoint code without first writing a linear perturbation code, then these two steps could be combined into one to compare the adjoint routines to the nonlinear flux routines to check for consistency.

4 Adjoint Solution Procedure

An important issue is how best to solve the adjoint equations. The eigenvalues of the adjoint matrix L^T are the same as those of the linear matrix L, and therefore one is guaranteed to get the same convergence rate when using Krylov subspace iteration methods such as GMRES, as used by Anderson [32, 1]. On the other hand, if one uses standard time-marching methods with multigrid, as are commonly used to solve the nonlinear equations, it is not necessarily the case that the iterative convergence rate for the adjoint solver will match that of the linear solver.

We have analysed this for our time-marching method which uses Jacobi preconditioning with partial updates of the numerical smoothing fluxes (and the viscous fluxes for the Navier-Stokes equations) at selected stages in the Runge-Kutta iteration [19].

One full step of the M-stage procedure for the linear equations can be expressed as

$$\begin{aligned} u^{(0)} &= u^n \\ d^{(m)} &= \beta_m D\, u^{(m-1)} + (1-\beta_m)\, d^{(m-1)} \\ u^{(m)} &= u^{(0)} + \alpha_m P \left(f - C\, u^{(m-1)} - d^{(m)} \right) \\ u^{n+1} &= u^{(M)} \end{aligned}$$

where $\beta_1 = \alpha_5 = 1$, P is the Jacobi preconditioning matrix and C and D are again the convective and diffusive matrices whose sum is the linear matrix L, as in Equation (3.1).

The outcome of this analysis [14] is that if the adjoint equations are solved using the following M-stage iterative procedure,

$$\begin{aligned} \tilde{v}^{(M)} &= P^H \left(g - L^H\, v^n \right) \\ \tilde{d}^{(M)} &= -\alpha_M\, \tilde{v}^{(M)} \\ \tilde{v}^{(m)} &= P^H \left(-\alpha_{m+1} C^H \tilde{v}^{(m+1)} + \beta_{m+1} D^H \tilde{d}^{(m+1)} \right) \\ \tilde{d}^{(m)} &= -\alpha_m \tilde{v}^{(m)} + (1-\beta_{m+1})\, \tilde{d}^{(m+1)} \\ v^{n+1} &= v^n + \sum_{m=1}^{M} \alpha_m \tilde{v}^{(m)} \end{aligned}$$

then the value of the linearised objective function from the linear and adjoint codes is not only identical once they have each converged to the final steady state, but it is also identical after each Runge-Kutta timestep. Note that this iteration uses the transpose of the Jacobi preconditioning matrix, and works "backwards" from $m = M$ to $m = 1$. If partial updating of the dissipative fluxes is not used, then it can be shown that this reduces to the standard Runge-Kutta method, but with the transposed preconditioner. However, with the use of partial updating, which is commonly employed to lower the CPU cost, it requires quite a lengthy analysis to determine this form for the adjoint iteration.

Furthermore, the analysis also extends to the use of multigrid, and shows that the key here is that the restriction operator for the adjoint code must be the transpose of the prolongation operator for the linear code, and *vice versa*, and the number of pre-smoothing iterations for the adjoint code must equal the number of post-smoothing iterations for the linear code, and *vice versa*. Provided these two conditions are satisfied, the linear and adjoint codes produce identical values for the functional after the same number of multigrid cycles.

This result is important for two reasons. The first is that it guarantees that the adjoint code converges, and that it does so with the same rate of convergence as the linear code, which is itself equal to the asymptotic rate of convergence of the nonlinear code. Thus the adjoint code benefits from the wealth of experience and fine tuning of iterative procedures for nonlinear codes. The second reason is that it provides another validation check on the correct implementation of the adjoint code. If the linear and

adjoint codes do not produce identical values for the functional after one timestep, it indicates a programming error.

5 Strong boundary conditions

Although it is possible to solve the Euler equations with solid wall boundary conditions imposed weakly by specifying zero mass flux through the wall faces, it is more common when there are grid nodes on the wall to use strong boundary conditions and force the normal component of the velocity at surface nodes to be zero. In doing so, the normal component of the momentum equation flux residual is discarded. Similarly, in discretising the Navier-Stokes equations, the entire velocity at the surface nodes is set to zero, and all components of the momentum residual are discarded. Thus in both cases the equations which are solved are actually of the form

$$\begin{aligned}(I-B)\,R(U) &= 0,\\ B\,U &= 0.\end{aligned}$$

Here I is the identity matrix and B is a projection matrix which in the case of the Euler equations extracts the normal component of the boundary velocity, and in the case of the Navier-Stokes equations extracts the entire boundary velocity. The presence of the term $(I - B)$ reflects the discarding of the appropriate flux residual components, to be replaced by the strong boundary conditions $BU = 0$.

When considering linear perturbations to these equations, we obtain

$$\begin{aligned}(I-B)\,(Lu - f) &= 0,\\ B\,u &= b,\end{aligned}$$

where b is a boundary velocity which is zero for the Navier-Stokes equations but non-zero for the Euler equations due to a rotation in the surface normal.

These two equations can be combined to form

$$((I-B)L + B)\,u = (I-B)f + b, \tag{5.1}$$

and the appropriate adjoint equation is then found by transposing the linear operator, noting that B is symmetric, to obtain

$$\left(L^T(I-B) + B\right)v = g. \tag{5.2}$$

At this point it is convenient to decompose both v and g into orthogonal components as

$$\begin{aligned}v &= (I-B)v + Bv = v_{\|} + v_{\perp},\\ g &= (I-B)g + Bg = g_{\|} + g_{\perp}.\end{aligned}$$

Pre-multiplying Equation (5.2) by $(I-B)$ shows that $v_{\|}$ satisfies the adjoint equations

$$\begin{aligned}(I-B)L^T v_{\|} &= g_{\|},\\ B\, v_{\|} &= 0.\end{aligned}$$

These are the equations which are solved iteratively by the adjoint code. Then, once $v_{\|}$ has been computed, $v_{\perp}$ is calculated in a post-processing step using an equation obtained by pre-multiplying Equation (5.2) by B:

$$v_{\perp} = g_{\perp} - BL^T v_{\|}. \tag{5.3}$$

Having computed $v_{\|}$ and $v_{\perp}$, the linearised functional is given by

$$\begin{aligned}\widetilde{J} &= v^T\left((I-B)f + b\right) + \frac{\partial J}{\partial \alpha}\\ &= v_{\|}^T f + v_{\perp}^T b + \frac{\partial J}{\partial \alpha}.\end{aligned}$$

This shows that $v_{\perp}$ gives the sensitivity of the functional to the boundary condition b which arises from the rotation of the boundary normal in the case of inviscid flows.

Note that $v_{\perp}$ does *not* correspond to the normal momentum component of the analytic adjoint solution at the boundary. Hence for visualisation purposes, it is desirable to replace $v_{\perp}$ by the analytic boundary condition,

$$v_{\perp}^{\text{analytic}} = h$$

whcih would normally be employed using a "continuous" formulation. Here h is zero everywhere except on the solid wall, where it corresponds to the sensitivity of the functional to the addition of momentum on the surface. In the case of a lift functional, for example, the element of h at a surface node n is

$$h_n = \begin{pmatrix} 0 \\ \vec{\jmath} \\ 0 \end{pmatrix},$$

with $\vec{\jmath}$ being the unit vector in the lift direction.

6 Residual contributions to the functional

If the functional of interest is a force, such as lift or drag, we have to include the surface momentum residuals, which are discarded in imposing the strong boundary conditions, in order to have a complete force balance. Indeed, for viscous calculations, it is the tangential component of these residuals which corresponds to the viscous shear stress. i.e. one defines the surface shear stress to have the value which is necessary to make the tangential momentum residual equal to zero.

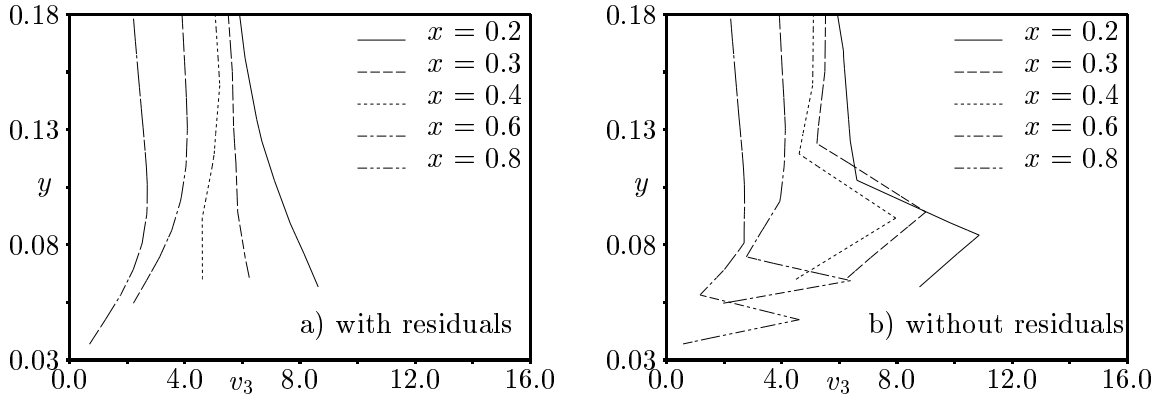

Figure 1: Variation in third adjoint component in y-direction for a subsonic NACA0012 test case, with and without residual contributions to the functional

The nonlinear functional is thus of the form

$$J = J_p(U) + h^T B\, R(U), \tag{6.1}$$

where J_p corresponds to the force due to the pressure distribution on the body and h is again the vector which takes the component of the discarded momentum residuals in the selected force direction (e.g. the direction normal to the freestream in the case of lift).

The corresponding linearised functional is

$$\widetilde{J} = g_p^T u + h^T BLu + \frac{\partial J}{\partial \alpha}, \tag{6.2}$$

where

$$g_p^T \equiv \frac{\partial J_p}{\partial U}, \tag{6.3}$$

and so we obtain

$$g = g_p + L^T Bh. \tag{6.4}$$

Fortunately, the second term in this equation can be computed in a pre-processing step using the adjoint flux routines.

The inclusion of the extra term makes a dramatic improvement to the quality of the adjoint solution near the surface, as illustrated in Figure 1 for a subsonic NACA0012 test case to be discussed later in more detail. To understand why it makes such a difference, it is important to remember that the adjoint variables correspond to the linearised effect of mass, momentum and energy sources on the functional of interest. Therefore, it is helpful to consider what happens in the linearised flow calculation when normal momentum is added close to a wall, as shown in Figure 2.

The effect of the momentum addition on the far field flow solution will be negligible. Therefore, with a conservative treatment, through the inclusion of the discarded

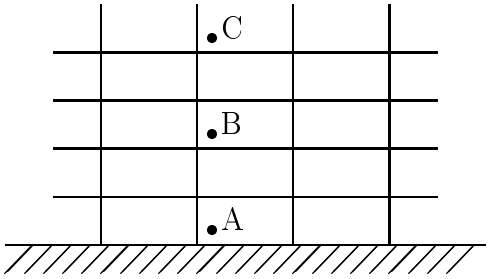

Figure 2: Diagram showing three possible locations of momentum injection close to a wall.

momentum residuals, the linear code will correctly predict that the change in the lift is equal and opposite to the addition of normal momentum, regardless of the location of the momentum addition. On the other hand, without the inclusion of the discarded residuals, the addition of momentum at point A, right next to the wall, will have zero effect on the functional because it will contribute solely to the momentum residuals at surface nodes. Similarly, addition at point B will have some effect on the residuals at nearby surface points; if these are not included in the functional then the influence on the functional must be incorrect. Only at point C, well away from the surface, will the effect on the surface residuals be very small and so the effect on the functional is correctly captured without the inclusion of the discarded residuals.

7 Harmonic adjoint

In analysing unsteady flow in turbomachinery, it is now common to use linearised Euler and Navier-Stokes methods which treat the unsteadiness as a linear perturbation to a nonlinear mean flow [17, 28, 27, 38, 18].

For forced response problems, in which the unsteadiness is due to periodic unsteady inflow or outflow boundary conditions, the original nonlinear unsteady discrete equations may be written as

$$M\frac{dU}{dt} + R(U) = 0,$$

where M is a block-diagonal mass matrix. Expressing $U(t)$ as the sum of a steady part plus a small amplitude perturbation

$$U(t) = \overline{U} + \widetilde{u}(t), \quad \|\widetilde{u}\| \ll \|\overline{U}\|$$

and linearising the equations gives

$$M\frac{d\widetilde{u}}{dt} + L\widetilde{u} = \widetilde{f},$$

where

$$L \equiv \frac{\partial R}{\partial U},$$

and $\widetilde{f}$ is zero except at the inflow and outflow boundary nodes where it gives the residual perturbations due to the incoming disturbances.

By the principle of linear superposition, the periodic input $\widetilde{f}(t)$ can be decomposed into the sum of a number of harmonic terms each of which can be written as the real part of a complex quantity of the form

$$\widetilde{f}(t) = \mathcal{R}\left\{e^{i\omega t}\widehat{f}\right\}.$$

Making a similar decomposition for the response $\widetilde{u}(t)$ yields the complex harmonic equations

$$(i\omega M + L)\,\widehat{u} = \widehat{f}.$$

In the case of unsteadiness due to the periodic vibration of the blades, the grid nodes all oscillate with the blades. Therefore, the nonlinear equations are best written as

$$M(x)\,\frac{dU}{dt} + R(U, x, \dot{x}) = 0,$$

to emphasise that the mass matrix and residuals depend on the grid coordinates, and the cell residual has additional flux terms due to the motion of the grid. Performing the same steps of linearisation and harmonic substitution then yields the same equations as before, with M and L being based on the undisturbed grid coordinates and flow, but with $\widehat{f}$ defined as

$$\widehat{f} = -\frac{\partial R}{\partial x}\,\widehat{x} - \mathrm{i}\,\omega\,\frac{\partial R}{\partial \dot{x}}\,\widehat{x}$$

due to the linearised motion of the grid.

One important engineering concern is the level of vibration caused by the incoming wakes. To determine this, one needs to compute a surface integral known as the "worksum". Following the theory of Lagrangian mechanics, this is the virtual work associated with the displacement of a particular natural mode of vibration of the blade. Numerically, it requires the computation of an inner product of the form

$$\widehat{g}^H\widehat{u},$$

where the superscript H denotes the complex conjugate transpose. The vector g is nonzero everywhere except at the grid nodes on the surface of the blade where it corresponds to the vibration mode being considered.

The adjoint alternative is to evaluate the inner product

$$\widehat{v}^H\,\widehat{f},$$

where the adjoint variables $\widehat{v}$ satisfy the adjoint equation

$$(i\omega M + L)^H\widehat{v} = \widehat{g}.$$

The implementation of this harmonic adjoint analysis is extremely similar to the usual steady adjoint analysis. The main differences are the coupled computation of the real and imaginary components of the complex variables $\widehat{v}$, and the use of phase-lagged periodic boundary conditions [9].

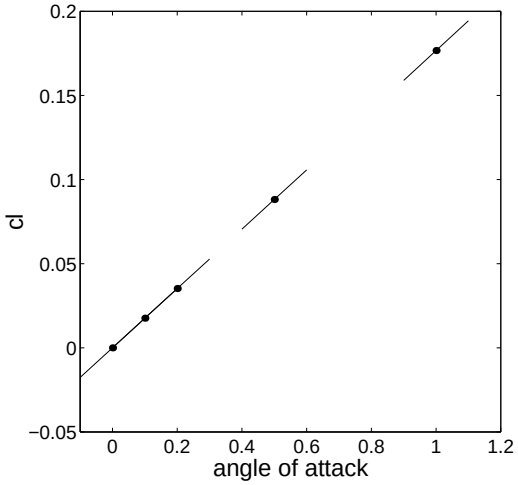

Figure 3: C_l vs. angle of attack for a NACA 0012 profile at $M = 0.68$.

8 Validation cases

8.1 Inviscid flow over NACA 0012 airfoil

The first two test cases consider steady inviscid flow over a NACA 0012 airfoil. The circles in Figure 3 show the lift coefficient obtained from the nonlinear code plotted against angle of attack at a freestream Mach number of 0.68. The angle of attack variation is achieved by rotating the airfoil as well as the points on and near the airfoil surface. Doing this in a linearised sense gives the geometric perturbations required for the source terms in the linear code and the functional in the adjoint code. The lines in Figure 3 are the lift slope obtained from the linear and adjoint codes, with the base flow in each case being the nonlinear flow conditions at the angle of attack at the mid-point of the line. The agreement between the nonlinear and linear/adjoint results looks quite good. To quantify this, Table 1 shows the nonlinear, linear and adjoint sensitivities at 0° angle of attack. The different nonlinear sensitivities are obtained by finite difference approximation over different intervals. There is perfect agreement between the linear and adjoint sensitivities, and the agreement with the nonlinear sensitivities is within the range one would expect give the errors inherent in finite difference approximation of the nonlinear sensitivities.

An interesting situation arises at higher Mach numbers at which there are strong shocks. Figure 4 shows the Mach contours for the NACA 0012 at an angle of attack of 1° and an increased Mach number of 0.85. There are now two shocks, with the maximum local Mach number reaching approximately 1.45 on the supersonic side of the suction surface shock. The circles in Figure 5 show the nonlinear lift coefficients over a limited range of angles of attack. The line in this figure is a linear regression least-square fit of the nonlinear data. The results indicate a peculiar lack of smoothness in the nonlinear data; this is shown more clearly in Figure 6 which plots the difference between the

	α	$\partial c_L/\partial\alpha$
nonlinear	0.0–0.1	0.17778
	0.0–0.2	0.176174
	0.0–0.5	0.1760996
	0.0–1.0	0.1764888
linear	0.0	0.1756657
adjoint	0.0	0.1756657

Table 1: Sensitivity of the lift to angle of attack for a NACA 0012 profile at $M = 0.68$ around 0° angle of attack.

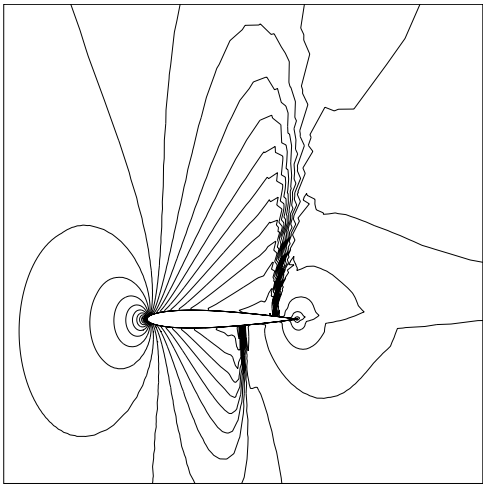

Figure 4: Mach contours for NACA 0012 at $M = 0.85$.

nonlinear data and the linear regression.

The key point is that there is no physical justification for the loss of smoothness. It appears to be a purely numerical artifact that is probably related to the displacement of the shock as the angle of attack changes. Therefore, the slope of the linear regression line is probably the best representation of the true lift slope. However, the linear/adjoint codes give lift slopes that correspond to the local derivative of the nonlinear data. Figure 7 plots the difference between the linear/adjoint slopes and the slope of the linear regression, showing a large discrepancy around 1.17° where the local derivative of the nonlinear data differs significantly from the linear regression value. Figure 8 plots the number of multigrid cycles required to converge the nonlinear code to a very tight tolerance. Interestingly, the number of cycles increases substantially around 1.17°. This suggests that the linearisation matrix may be almost singular, which could be related to the fact that small changes in the angle of attack produce larger changes in the lift than one would otherwise expect.

This observation of limitations with the application of linear methods to flows with strong shocks may be primarily of academic interest, and not of engineering concern.

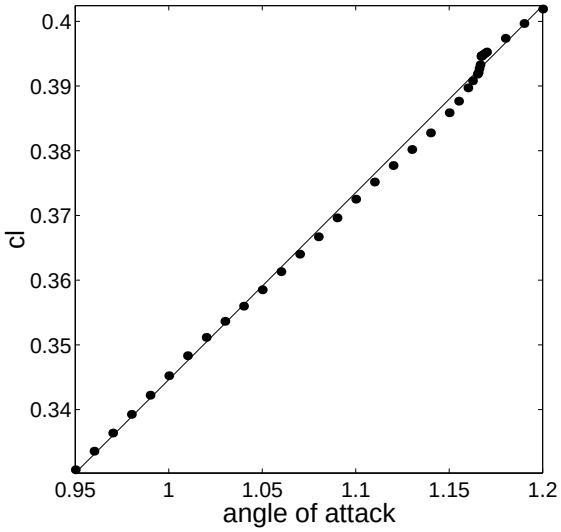

Figure 5: C_l vs. angle of attack for NACA 0012 at $M = 0.85$.

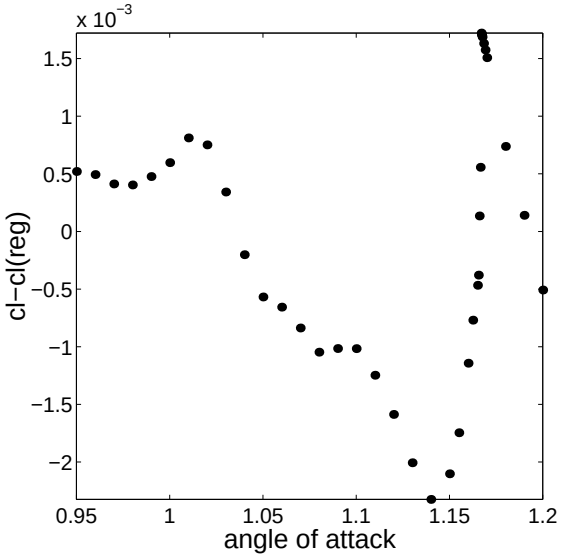

Figure 6: $C_l - C_l$(regression) for NACA 0012 at $M = 0.85$.

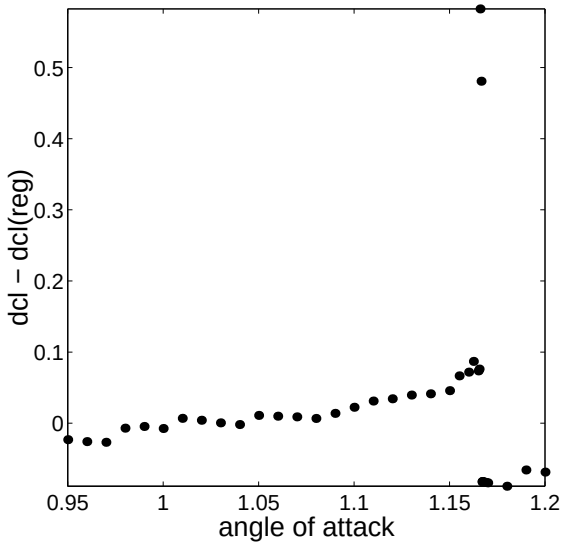

Figure 7: $dC_l/d\alpha$(linear) $-$ $dC_l/d\alpha$(regression) for NACA 0012 at $M = 0.85$.

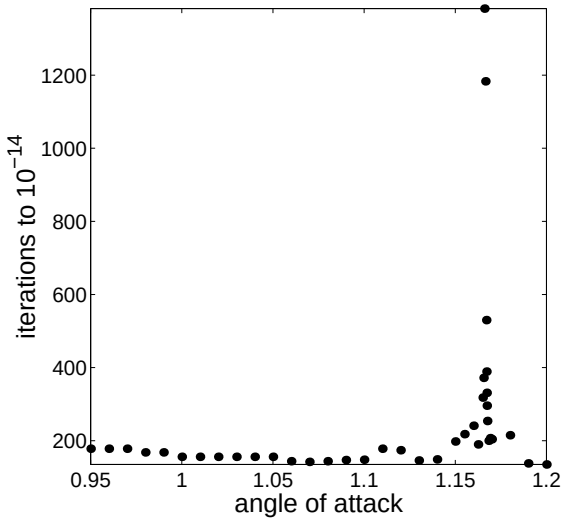

Figure 8: Number of multigrid cycles for nonlinear calculations for NACA 0012 at $M = 0.85$.

	α	$\partial c_L/\partial\alpha$
nonlinear	1.40–2.40	0.1815048
	1.90–2.40	0.1772154
	2.15–2.40	0.1742684
	2.40–2.65	0.1713676
	2.40–2.90	0.1644908
	2.40–3.40	0.1398736
linear	2.40	0.1680554
adjoint	2.40	0.1680554

Table 2: Sensitivity of the lift for a RAE 2822 profile at $M = 0.725$, $\mathrm{Re} = 6.5 \times 10^6$ around 2.4° angle of attack.

Most aeronautical applications do not have such strong normal shocks, and with weaker shocks with a peak normal Mach number of less than 1.3 we have not observed a similar phenomenon. However, it may be necessary to look more closely at the issue of linearised shock displacement, and to use more numerical smoothing at shocks to obtain the correct linear sensitivity [25].

8.2 Turbulent flow over RAE 2822 airfoil

Figure 9 presents the Mach contours for the Reynolds-averaged flow over the RAE 2822 airfoil at angle of attack $\alpha = 2.4°$, freestream Mach number $M = 0.725$ and Reynolds number $\mathrm{Re} = 6.5 \times 10^6$. The turbulence is modeled using a Spalart-Allmaras single equation model. The circles in Figure 10 show the sensitivity of the variation in the lift coefficient with changes in the angle of attack. The lines correspond to the lift slopes computed by the linear and adjoint codes, which are again in perfect agreement with each other. There is no evidence of any lack of smoothness in the nonlinear lift predictions, and the linear/adjoint codes give lift slopes which are in very good agreement with the nonlinear results. This is quantified in Table 2 using finite differences to estimate the nonlinear lift slope at $\alpha = 2.4°$.

Figure 11 shows the convergence histories for the non-linear, linear and adjoint codes for the RAE 2822 testcase at $\alpha = 2.4°$. As expected, they all exhibit the same asymptotic convergence rate.

8.3 Turbomachinery wake interaction

The linear harmonic code has been validated against a number of test cases. Figure 12 shows results for unsteady wake interaction with a 2D cascade of flat plate airfoils. Real and imaginary components of the complex pressure jump across one blade are compared with the results from the LINSUB [40] which implements the analytic theory of Smith [36].

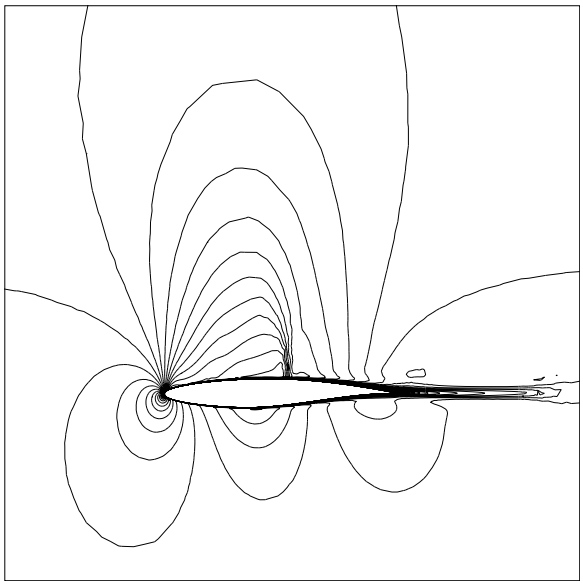

Figure 9: Mach contours for a RAE 2822 profile at $M = 0.725$, $\mathrm{Re} = 6.5 \times 10^6$.

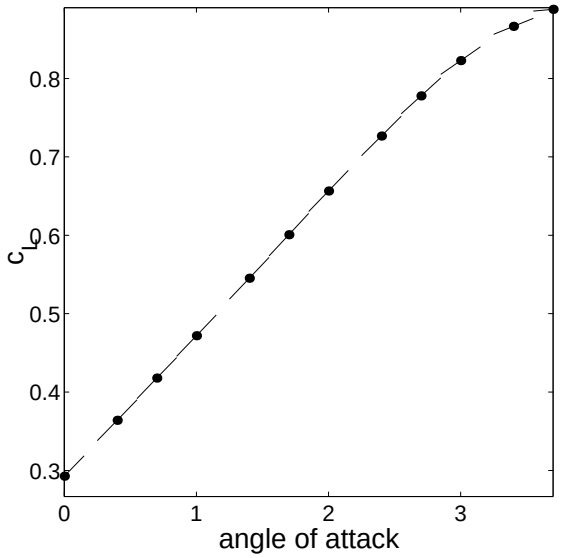

Figure 10: Lift vs. angle of attack for a RAE 2822 profile at $M = 0.725$, $\mathrm{Re} = 6.5 \times 10^6$.

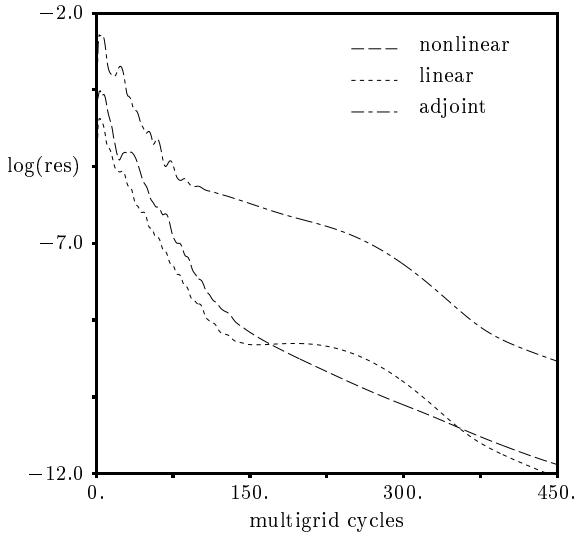

Figure 11: Convergence histories for the nonlinear, linear and adjoint codes for a RAE 2822 profile at $M = 0.725$, $\mathrm{Re} = 6.5 \times 10^6$.

Figure 13 demonstrates the variation in the bending mode worksum as the wake pitch is changed (representing modifications both the inter-blade phase angle and the frequency of the unsteadiness). The results show that the linear harmonic and adjoint harmonic codes produce identical values for the worksum, and these are in good agreement with the analytic values produced by LINSUB.

For further validation cases, and an example of the usefulness of the adjoint method for design of blades with reduced forced response, see Duta *et al* [9, 6].

9 Conclusions

In this paper we have presented a number of algorithm developments concerned with the formulation and solution of adjoint Euler and Navier-Stokes equations using the discrete approach. These include the treatment of strong boundary conditions and the associated adjoint boundary conditions for lift and drag functionals, as well as a Runge-Kutta time-marching scheme that ensures exact equivalence with a linear perturbation code throughout the convergence process. This property guarantees the same asymptotic convergence rate for nonlinear, linear and adjoint solvers, as well as being very useful during code validation.

Acknowledgements

This research has been supported by the Engineering and Physical Sciences Research Council under grants GR/K91149 and GR/L95700, and by Rolls-Royce plc (technical monitor: Leigh Lapworth) DERA (technical monitor: John Calvert), and BAESystems

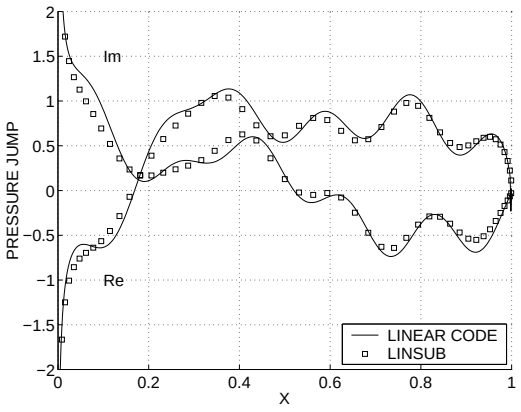

Figure 12: Real and imaginary components of the flat plate pressure jump due to wake interaction.

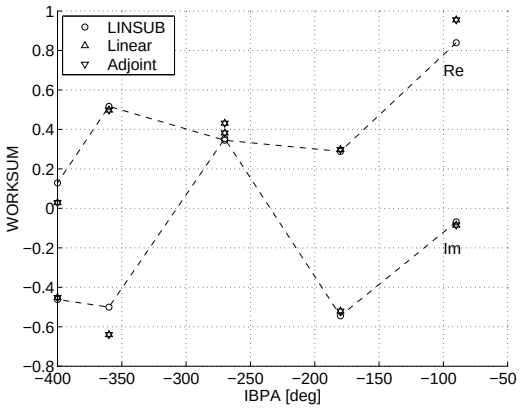

Figure 13: Bending mode worksum components due to wake interaction, versus interblade phase angle associated with wake pitch.

plc (technical monitor: David Standingford).

We also acknowledge the contributions of P. Moinier, M.S. Campobasso, L. Lapworth and M. West to the development of the HYDRA codes.

References

[1] W.K. Anderson and D.L. Bonhaus. Airfoil design on unstructured grids for turbulent flows. *AIAA J.*, 37(2):185–191, 1999.

[2] W.K. Anderson and E. Nielsen. Sensitivity analysis for Navier-Stokes equations on unstructured grids using complex variables. *AIAA J.*, 39(1):56–63, 2001.

[3] W.K. Anderson and V. Venkatakrishnan. Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation. *Comput. & Fluids*, 28(4-5):443–480, 1999.

[4] O. Baysal and M. Eleshaky. Aerodynamic design optimization using sensitivity analysis and computational fluid dynamics. *AIAA J.*, 30(3):718–725, 1992.

[5] C.H. Bischof, A. Carle, P.D. Hovland, P. Khademi, and A. Mauer. ADIFOR 2.0 User's Guide (Revison D). Technical Report 192, Mathematics and Computer Science Division, Argonne National Laboratory, 1998. See `www.mcs.anl.gov/adifor`.

[6] M.S. Campobasso, M. Duta, and M.B. Giles. Adjoint methods for turbomachinery design. ISABE Conference, Bangalore, 2001.

[7] A. Carle, M. Fagan, and L.L. Green. Preliminary results from the application of automated code generation to CFL3D. AIAA Paper 98-4807, 1998.

[8] A. Dadone and B. Grossman. Progressive optimization of inverse fluid dynamic design problems. *Comput. & Fluids*, 29(1), 2000.

[9] M. Duta, M.B. Giles, and M.S. Campobasso. The harmonic adjoint approach to unsteady turbomachinery design. ICFD Conference, Oxford, 2001.

[10] J. Elliott. *Aerodynamic optimization based on the Euler and Navier-Stokes equations using unstructured grids.* PhD thesis, MIT Dept. of Aero. and Astro., 1998.

[11] J. Elliott and J. Peraire. Practical 3D aerodynamic design and optimization using unstructured meshes. *AIAA J.*, 35(9):1479–1485, 1997.

[12] C. Faure and Y. Papegay. Odyssée User's Guide version 1.7. Technical Report RT-0224, INRIA, Sophia-Antipolis, 1998. See `www.inria.fr/safir/SAM/Odyssee/odyssee.html`.

[13] R. Giering and T. Kaminski. Recipes for adjoint code construction. *ACM Trans. Math. Software*, 24(4):437–474, 1998.

[14] M.B. Giles. On the use of Runge-Kutta time-marching and multigrid for the solution of steady adjoint equations. Technical Report NA00/10, Oxford University Computing Laboratory, 2000. See `www.comlab.ox.ac.uk/oucl/work/mike.giles/`.

[15] M.B. Giles and N.A. Pierce. Analytic adjoint solutions for the quasi-one-dimensional Euler equations. *J. Fluid Mech.*, 426:327–345, 2001.

[16] A. Griewank. *Evaluating derivatives : principles and techniques of algorithmic differentiation*. SIAM, 2000.

[17] K.C. Hall, W.S. Clark, and C.B. Lorence. A linearized Euler analysis of unsteady transonic flows in turbomachinery. *J. Turbomachinery*, 116:477–488, 1994.

[18] D. Hoyniak and W.S. Clark. Aerodynamic damping predictions using a linearized Navier-Stokes analysis. ASME Paper 99-GT-207, 1999.

[19] A. Jameson. Aerodynamic design via control theory. *J. Sci. Comput.*, 3:233–260, 1988.

[20] A. Jameson. Optimum aerodynamic design using control theory. In M. Hafez and K. Oshima, editors, *Computational Fluid Dynamics Review 1995*, pages 495–528. John Wiley & Sons, 1995.

[21] A. Jameson. Re-engineering the design process through computation. *J. Aircraft*, 36:36–50, 1999.

[22] A. Jameson, N. Pierce, and L. Martinelli. Optimum aerodynamic design using the Navier-Stokes equations. *J. Theor. Comp. Fluid Mech.*, 10:213–237, 1998.

[23] H.-J. Kim, D. Sasaki, S. Obayashi, and K. Nakahashi. Aerodynamic optimization of supersonic transport wing using unstructured adjoint method. Proceedings of the ICCFD conference, Kyoto, 2000.

[24] V.M. Korivi, A.C. Taylor III, and G.W. Hou. Sensitivity analysis, approximate analysis and design optimization for internal and external viscous flows. AIAA Paper 91-3083, 1991.

[25] D.R. Lindquist and M.B. Giles. Validity of linearized unsteady Euler equations with shock capturing. *AIAA J.*, 32(1):46, 1994.

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

[27] S.R. Manwaring, D.C. Rabe, C.B. Lorence, and Wadia A.R. Inlet distortion generated forced response of a low-aspect-ratio transonic fan. *J. Turbomachinery*, 119(4):665–676, 1997.

[28] J.G. Marshall and M.B. Giles. Some applications of a time-linearised Euler method to flutter and forced response in turbomachinery. In *Proceedings of the 8th ISUAAT Stockholm*, 1997.

[29] B. Mohammadi and O. Pironneau. Mesh adaption and automatic differentiation in a CAD-free framework for optimal shape design. *Internat. J. Numer. Methods Fluids*, 30(2):127–136, 1999.

[30] P. Moinier, J.-D. Müller, and M.B. Giles. Edge-based multigrid and preconditioning for hybrid grids. AIAA Paper 99-3339, 1999.

[31] J.C. Newman, A.C. Taylor, R.W. Barnwell, P.A. Newman, and G. J.-W. Hou. Overview of sensitivity analysis and shape optimization for complex aerodynamic configurations. *J. Aircraft*, 36(1):87–96, 1999.

[32] E. Nielsen and W.K. Anderson. Aerodynamic design optimization on unstructured meshes using the Navier-Stokes equations. *AIAA J.*, 37(11):957–964, 1999.

[33] O. Pironneau. On optimum design in fluid mechanics. *J. Fluid Mech.*, 64:97–110, 1974.

[34] J. Reuther, A. Jameson, J.J. Alonso, M.J. Remlinger, and D. Saunders. Constrained multipoint aerodynamic shape optimisation using an adjoint formulation and parallel computers, part 1. *J. Aircraft*, 36(1):51–60, 1999.

[35] J. Reuther, A. Jameson, J.J. Alonso, M.J. Remlinger, and D. Saunders. Constrained multipoint aerodynamic shape optimisation using an adjoint formulation and parallel computers, part 2. *J. Aircraft*, 36(1):61–74, 1999.

[36] S.N. Smith. Discrete frequency sound generation in axial flow turbomachines. University of Cambridge, Department of Engineering, 1971.

[37] W. Squire and G. Trapp. Using complex variables to estimate derivatives of real functions. *SIAM Rev.*, 10(1):110–112, 1998.

[38] K. Sreenivas and D.L. Whitfield. Time- and frequency-domain numerical simulation of linearized Euler equations. *AIAA J.*, 36(6):968–975, 1998.

[39] S. Ta'asan, G. Kuruvila, and M.D. Salas. Aerodynamic design and optimization in one shot. AIAA Paper 92-0025, 1992.

[40] D. S. Whitehead. Classic two-dimensional methods. In M. Platzer and F. O. Carta, editors, *Aeroelasticity in Axial-Flow Turbomachines, AG-298*, volume 1. AGARD, 1987.