Efficient Algorithms for Solution of the Adjoint Compressible Navier-Stokes Equations with Applications

Richard P. Dwight *, Joël Brezillon, Daniel B. Vollmer
German Aerospace Center (DLR), Lilienthalplatz, 7, Braunschweig, D-38108 Germany.

Abstract

The complete discrete adjoint equations for an unstructured finite volume compressible Navier-Stokes solver are discussed with respect to the memory and time efficient evaluation of their residuals, and their solution. It is seen that application of existing iteration methods for the non-linear equation - suitably adjointed - have a property of guaranteed convergence provided that the non-linear iteration is well behaved. For situations where this is not the case, a stabilization method based on the Recursive Projection Method is developed. The resulting adjoint solver is applied to 3D optimization of a turbulent wing-fuselage configuration, as well as to goal-oriented adaptation using an error representation formula.

Key words: discrete adjoint, recursive projection method, eigensystem analysis, aerodynamic optimization, error estimation

1. Introduction

The adjoint equations are enjoying increasing importance in the field of computational aerodynamics, as emphasis shifts from the modelling of physical phenomena to their control and optimization. These applications introduce a class of minimization problems to the field, where the extremal value of some functional of the flow solution in some parameter space is desired. The parameterization of three dimensional aircraft geometries can involve many hundreds of design parameters however [1], and all efficient solution approaches rely on gradients of the functional in the design space. The gradients themselves can only be efficiently evaluated using adjoint methods [2].

As optimization becomes more widely used the question of the accuracy of the flow solver becomes more important. In particular quantitative bounds on the error in functionals are essential for establishing whether a given optimization is physically meaningful. Recently several techniques have been developed which represent the error in a functional as a product of an adjoint field and a residual [3,4], giving a rapid means of establishing accuracy and identifying how it may be improved.

But solving the adjoint equations is a problem at least as hard as solving the original flow equations and - as will be seen - often significantly harder. Although it is possible to apply the same methods used for the non-linear problem, Section 4, often these do not converge due to small discrepancies between the adjoint and the original problem. It then becomes necessary to apply some stabilization to the iteration, Section 5.

We forego a detailed description of the compressible Navier-Stokes equations and their unstructured finite volume discretization in the DLR *TAU*-Code, referring the interested reader to [5,6], and note only that the method uses the Jameson-Schmitt-Turkel (JST) flux with scalar artificial dissipation [7] and the one-equation Spalart-Allmaras turbulence model with Edwards modification [8]. The resulting semi-discrete system is written simply

$$\frac{\mathrm{d}W}{\mathrm{d}t} + R\left(W, \alpha\right) = 0, \tag{1}$$

where W is the vector of unknown flow variables, α is some arbitrary scalar parameter of the flow problem or geometry, t is the time and R is denoted the *residual*. Given this we are interested in formulating and solving linear systems of the form

$$\frac{\partial R}{\partial W}\,\Theta = -\frac{\partial R}{\partial \alpha}, \tag{2}$$

* Corresponding author.
Email addresses: richard.dwight@dlr.de (Richard P. Dwight), joel.brezillon@dlr.de (Joël Brezillon), daniel.vollmer@dlr.de (Daniel B. Vollmer).
URL: http://www.maths.man.ac.uk/~rdwight (Richard P. Dwight).

for unknowns Θ, the *primal* equation, and more especially

$$\left[\frac{\partial R}{\partial W}\right]^T \Lambda = -\left[\frac{\partial I}{\partial W}\right]^T, \tag{3}$$

for unknowns Λ and scalar *cost function* $I(W, \alpha)$, the *adjoint* equation. We denote $\partial R/\partial W$ the *Jacobian* and the term *linear problem* will refer to either (2) or (3). Given the solution to one of these, the derivative of I with respect to α may be written

$$\frac{\mathrm{d}I}{\mathrm{d}\alpha} = \frac{\partial I}{\partial \alpha} + \left[\frac{\partial I}{\partial W}\right]^T \Theta = \frac{\partial I}{\partial \alpha} + \Lambda^T \frac{\partial R}{\partial \alpha}, \tag{4}$$

whereby evaluating this expression for m different α and n different I requires either m solutions of (2) or n solutions of (3). For typical aerodynamic optimization problems $m \gg n$, as we are concerned with a few output forces, and the parameterization of 3d geometries, and the adjoint becomes significantly more efficient.

2. Approximation of the Jacobian

The convective terms within (1) are discretized using the JST scheme, the numerical Riemann flux across a face $\{ij\}$ with normal vector n_{ij} being written,

$$\begin{aligned}\hat{f}^c_{ij} = &\frac{1}{2}\left(f^c(W_i) + f^c(W_j)\right)\cdot n_{ij} \\ &- \frac{1}{2}\left|\lambda^c_{ij}\right|\left\{\varepsilon^{(2)}_{ij}(W_j - W_i) - \varepsilon^{(4)}_{ij}(L_j(W) - L_i(W))\right\},\end{aligned} \tag{5}$$

where f^c is the exact convective flux, $|\lambda^c|$ is the maximum local convective eigenvalue, $\varepsilon^{(2,4)}$ control the switching between 1st- and 3rd-order dissipation, and

$$L_i(W) = \sum_{k\in\mathcal{N}(i)} (W_k - W_i), \tag{6}$$

where $\mathcal{N}(i)$ is the set of all immediate neighbours of grid point i. A complete linearization of this flux may be found in [6], and covers three full pages. More problematic than this is that its stencil includes not only immediate, but also next-neighbouring points, and the Jacobian therefore has next-neighbour fill-in.

In previous work in 2d [9] this Jacobian was stored explicitly. Restarted GMRES was applied with an ILU(4) preconditioner as a solution method. This scheme provided an adjoint solution in approximately 5% of the time required for the non-linear field, but required a factor of 11 times more memory than the standard solver. This approach is no longer tenable in 3d as the memory requirements given in Table 1 show. This table was generated empirically by running the sequential codes for one 2d and one 3d turbulent test case with an LU-SGS smoothed 3W multigrid cycle. The discrepancy between the two sets of results is due to the much increased number of faces in 3d. Notable is the memory usage of the original non-linear code; we believe 1.2 million points in 1GB of memory is unusually efficient for an unstructured solver. On the other hand, mere storage of the full Jacobian in 3d (without the additional memory needed by ILU and GMRES) multiplies the memory costs of the code by a factor of 13.5. Even though the approach reduces evaluation of the linear residual to a matrix-vector multiplication, this is a factor 1.7 times more expensive to evaluate than the non-linear residual due to the memory bandwidth bottleneck.

To alleviate these deficiencies the following device is introduced: by making the assumption that the ε and $|\lambda|$ are constant, a simplification of the adjoint residual is possible [9,10]. Treat L as an independent variable, so that the derivative and adjoint of (5) may be written respectively

$$\frac{\mathrm{d}\hat{f}^c}{\mathrm{d}W} = \frac{\partial \hat{f}^c}{\partial W} + \frac{\partial \hat{f}^c}{\partial L}\frac{\partial L}{\partial W}, \tag{7}$$

$$\left[\frac{\mathrm{d}\hat{f}^c}{\mathrm{d}W}\right]^T = \left[\frac{\partial \hat{f}^c}{\partial W}\right]^T + \left[\frac{\partial L}{\partial W}\right]^T \left[\frac{\partial \hat{f}^c}{\partial L}\right]^T, \tag{8}$$

whereby all matrices on the right-hand sides of these equations have immediate neighbour fill-in only. Further $\partial \hat{f}^c/\partial L$ is a multiple of the identity at each grid face and $\partial L/\partial W$ is trivial. If $\partial L/\partial W$ is calculated on-the-fly, memory requirements are only slightly greater than those required for a single Jacobian with immediate neighbour fill-in.

Also the matrices may be stored very naturally on the edges and nodes of the grid, given which the adjoint residual may be evaluated in two loops over all edges by introducing an intermediate variable Λ^* as follows:

$$\Lambda^* = \left[\frac{\partial \hat{f}^c}{\partial L}\right]^T \Lambda, \tag{9}$$

$$\left[\frac{\mathrm{d}\hat{f}^c}{\mathrm{d}W}\right]^T \Lambda = \left[\frac{\partial \hat{f}^c}{\partial W}\right]^T \Lambda + \left[\frac{\partial L}{\partial W}\right]^T \Lambda^*. \tag{10}$$

Since viscous and turbulence diffusion fluxes are used which involve only gradients normal to grid faces (the so-called Thin Shear-Layer discretization), the remaining terms of the Jacobian contain only immediate neighbour information and may be formed exactly. The performance of the resulting method is given in the third column of Table 1, of particular importance being the factor 4 increase in memory, which is considered acceptable. If it happens to be unacceptable there remains the option of computing the Jacobian on-the-fly, giving a code of similar memory requirements to the original, but with a linear residual three times more expensive than the non-linear residual.

The assumption of constant dissipation coefficients necessarily affects the accuracy of the adjoint solution, however this error was systematically investigated in [9] together with several other Jacobian approximations, and found to be insignificant for the purposes of gradient evaluation and optimization. A theoretical rationale for this result was provided in [6]. Note also that - depending on the form of the

Table 1
The memory requirements and run-time of the standard code and two linearized versions of the code for representative 2d (hybrid 28k points) and 3d (hybrid 120k points) grids.

	Standard *TAU*	Full Jac. [a]	Reduced Jac. [a]
Memory (Bytes) 2d	12M	100M	44M
Factor increase 2d	×1.0	×8.3	×3.7
Points in 1GB 2d	2.3×10^6	280×10^3	640×10^3
Time res. eval 2d	×1.0	×1.3	×0.7
Time Jac. eval 2d	-	×26.2	×2.6
Memory (Bytes) 3d	100M	1350M	400M
Factor increase 3d	×1.0	×13.5	×4.0
Points in 1GB 3d	1.2×10^6	89×10^3	300×10^3
Time res. eval 3d	×1.0	×1.7	×0.6
Time Jac. eval 3d	-	×28.0	×3.0

[a] Complete code, using same iterative procedure as standard *TAU*.

ε - by choosing suitable intermediate variables it may possible to treat the complete dissipation operator in a similar manner, so that no constant coefficient approximation must be made. This step was considered unnecessary given the results of [9].

3. Parallelization of the Adjoint Code

Parallelization of the adjoint residual code does not fit easily into the pattern established by parallelization of the original code. This latter follows a domain decomposition approach, the global grid is divided into n non-overlapping point sets $\mathcal{K} = \{\mathcal{K}_1, \mathcal{K}_2, \ldots, \mathcal{K}_n\}$, each of which is supplemented by all points which immediately neighbour a point in the set and are not in the set themselves:

$$\hat{\mathcal{K}}_m = \{i \in \mathcal{K} \backslash \mathcal{K}_m : \exists\, j \in \mathcal{K}_m \text{ s.t. } i \in \mathcal{N}(j)\}, \tag{11}$$

to give domains $\mathcal{D}_m = (\mathcal{K}_m, \hat{\mathcal{K}}_m)$. Variables defined at points in $\hat{\mathcal{K}}_m$ are copied from their native domain after they have been updated there, and before they are needed in $\mathcal{D}_m$. This parallel copying operation is the only parallel operation needed on variables defined at grid points. The variables defined at $\hat{\mathcal{K}}_m$ are never modified within $\mathcal{D}_m$.

To illustrate the difficulty with parallelization of adjoint consider the code sample

$$y_i := x_j - x_i, \tag{12}$$

evaluated on domain $\mathcal{D}_m$ and (borrowing the notation of algorithmic differentiation, see e.g. [2]) its adjoint

$$\bar{x}_i := -\bar{y}_i, \quad \bar{x}_j := \bar{y}_i. \tag{13}$$

If $i, j \in \mathcal{K}_m$ there is no difficulty, but if $i \in \mathcal{K}_m$ and $j \in \hat{\mathcal{K}}_m$ ($j \in \mathcal{K}_l$ say), then the code requires that a point in $\hat{\mathcal{K}}_m$ be updated. There are two possible approaches: either transfer the second operation of (13) to $\mathcal{D}_l$, or perform it within $\mathcal{D}_m$ and communicate the result "backwards" to $\mathcal{D}_l$. In most circumstances the former is more natural, but in situations where the stencil is not symmetric, e.g. where there is no corresponding $y_j := x_i - x_j$ to (12) above, the latter can be simpler. For the second approach it should also be carefully noted that a single point may receive contributions from multiple neighbouring domains during one "backward" communication. In any case, each such occurrence must be explicitly handled if the parallelized code is to be correct.

As an example, in the present effort $\partial \hat{f}^c / \partial W$ is not transposed in memory as would be required for evaluation of $R_j^{\text{adj}} := [\partial \hat{f}_i^c / \partial W_j]^T \Lambda_i$ local to $\mathcal{D}_l$. Rather R_j^{adj} is evaluated on $\mathcal{D}_m$ and communicated. The complete resulting parallel adjoint code has negligibly more communication than the original code and therefore scales similarly with number of processors.

4. Application of Existing Fixed-Point Iterations

Efficient solution methods for the non-linear problem (1) have been an important topic of research for some time, and significant experience has been gained in this area. Most methods currently in use employ some form of *fixed-point iteration* (FPI), a class of methods including Runge-Kutta, multigrid and implicit schemes, which may all be written

$$M(W^{n+1} - W^n) = -R(W^n), \tag{14}$$

for some *preconditioning* matrix M. Given the level of advancement of these techniques, it is worth considering whether they can be applied to the solution of the corresponding linear equations. In fact by rearranging (14), and linearizing about $\tilde{W}$, where $R(\tilde{W}) = 0$, we have

$$W^{n+1} = (I - M^{-1}A)W^n + g(\tilde{W}) + \mathcal{O}\|W^n - \tilde{W}\|^2, \tag{15}$$

where A is the Jacobian evaluated at the exact solution and $g(\cdot)$ is some function independent of n. The relation to the linearized problem is immediately apparent, and informed by this we consider the FPI

$$M(\Theta^{n+1} - \Theta^n) = \frac{\partial R}{\partial \alpha} - A\Theta^n, \tag{16}$$

or rearranged:

$$\Theta^{n+1} = (I - M^{-1}A)\Theta^n + M^{-1}\frac{\partial R}{\partial \alpha}. \tag{17}$$

The coefficients of W^n and Θ^n in these two iterations are identical, so we can conclude that - for a sufficiently converged non-linear iteration - the errors reduce at the same rate; i.e. the asymptotic convergence behaviour is identical. However this is only true if A is the exactly formulated Jacobian based on a highly converged non-linear solution. The rate of convergence itself is given by the dominant eigenvalue of $(I - M^{-1}A)$.

To achieve the same for the adjoint equation consider the FPI

$$M^T(\Lambda^{n+1} - \Lambda^n) = \left[\frac{\partial I}{\partial W}\right]^T - A^T\Lambda^n, \tag{18}$$

so that

$$\Lambda^{n+1} = (I - M^{-T}A^T)\Lambda^n + M^{-T}\left[\frac{\partial I}{\partial W}\right]^T. \tag{19}$$

where M^{-T} denotes the inverse transpose of M. Since any real matrix and its transpose have identical eigenspectra, the asymptotic convergence rate of (19) will be the same as that of (17).

Further, by requiring that $\Theta^0 = \Lambda^0 = 0$, and expanding (17) and (19) to the Nth iteration we have respectively

$$\Theta^{N+1} = -\sum_{n=0}^{N}(I - M^{-1}A)^n M^{-1}\frac{\partial R}{\partial \alpha}, \tag{20}$$

$$\Lambda^{N+1} = -\sum_{n=0}^{N}(I - M^{-T}A^T)^n M^{-T}\left[\frac{\partial I}{\partial W}\right]^T, \tag{21}$$

given which it quickly follows that

$$\left[\Lambda^{N+1}\right]^T \frac{\partial R}{\partial \alpha} = \left[\frac{\partial I}{\partial W}\right]^T \Theta^{N+1}, \tag{22}$$

i.e. that the primal and adjoint results for $\mathrm{d}I/\mathrm{d}\alpha$ will be identical not only after the equations have fully converged, but also at every intermediate step.

These two properties: identical asymptotic convergence, and a relation on the transient convergence, are very desirable. The first effectively provides a guarantee that, given convergence of the non-linear problem, the linear problem will also converge. Ideally this would mean that the level of intervention required from a user of the code to run the linear code subsequent to the non-linear code is minimal; CFL numbers and such do not need to be adjusted. The second property, which holds to working accuracy, aids development of the adjoint code considerably by providing a means of verifying that the Jacobian and FPI have been correctly adjointed. For example this test revealed that the Laplacian residual smoothing present in the Runge-Kutta algorithm was not quite symmetric, as might have been expected.

In the present work the approximately factored implicit LU-SGS scheme [6,11] is adjointed. Written

$$(\tilde{D} + \tilde{L})\tilde{D}^{-1}(\tilde{D} + \tilde{U})x = b, \tag{23}$$

where $\tilde{D}$, $\tilde{L}$ and $\tilde{U}$ are the block diagonal, lower triangular and upper triangular parts respectively of an approximate 1st-order Jacobian $\tilde{A}$. The LHS matrix clearly identifies with M. The corresponding adjoint iteration is therefore

$$(\tilde{D} + \tilde{U})^T\tilde{D}^{-T}(\tilde{D} + \tilde{L})^T x = b, \tag{24}$$

where the order of the forward and backward sweeps has been reversed by the application of the transpose. Similarly the adjoint of linear multigrid may be found by writing the iteration in matrix notation and transposing. As a result, the transpose of the original interpolation operator becomes the adjoint prolongation operator, and the transpose of the prolongation the adjoint interpolation. Also if the relaxation steps are performed before the sweep in the original method, they are performed after the sweep in the adjoint method. The adjointing of Runge-Kutta is somewhat more involved [12], but follows the same basic pattern of writing the iteration in matrix notation and transposing.

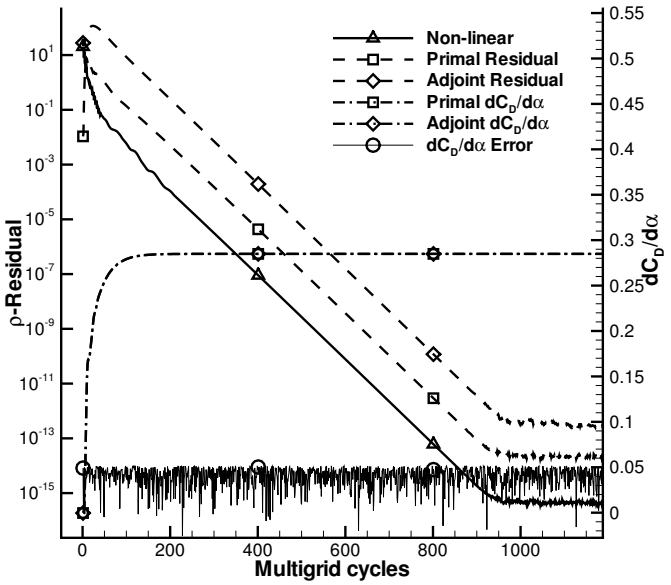

Fig. 1. Convergence of non-linear, primal and adjoint codes for an NACA0012 test case. Also shown are transients of $\mathrm{d}C_D/\mathrm{d}\alpha$.

In all cases it is necessary to work in the same variables as the non-linear iteration (which is in conservative flow variables), if the iterations are to be consistent. If the Jacobian is formulated in terms of derivatives with respect to primitive variables it must first be transformed. The memory and CPU requirements of the adjoint FPIs are identical to their primal counterparts. Also of interest is that all the standard non-adjointed iterative methods perform very poorly on the adjoint problem. For the untransposed Runge-Kutta method on a simple adjoint test case, it was necessary to reduce the CFL number to 1/4 of its original value to achieve stability. This effect was not as significant for multigrid, likely due to the similarity of the original and adjoint formulations.

A numerical demonstration of the two properties derived here is shown in Figure 1 using LU-SGS and a 3W multigrid cycle for a transonic NACA0012 aerofoil. Plotted are the convergence of the non-linear, linear and adjoint codes, together with the gradient $\mathrm{d}C_D/\mathrm{d}\alpha$ from the primal and adjoint (where here α is the angle-of-attack), as well as the difference between these two gradients. Note that the asymptotic convergence of all codes is identical and the error in the two versions of $\mathrm{d}C_D/\mathrm{d}\alpha$ rests at the truncation error of the floating point arithmetic used. The monitoring of such gradients is also a useful way for a user of the code to estimate the progress of convergence, corresponding to monitoring of forces in the non-linear case.

5. The Recursive Projection Method

Despite the guarantees regarding convergence provided by the theory of the previous section, there are regularly situations in which it is possible to obtain a reasonably converged solution of the non-linear equations, but not of the corresponding linear equations. This can occur for three reasons, all of which violate the parity of (14) and (16). Either a) the non-linear solution is not sufficiently converged, or b) there is a discrepancy between the linear and non-linear problem due to some approximation of the Jacobian, or c) the FPI applied to the non-linear problem does not converge asymptotically itself.

All three cases appear regularly in practice. An engineer may reasonably consider a computation converged when the integrated forces that interest her no longer vary significantly, though this may occur prior to the asymptotic regime. With regard to b), as already described, the Jacobian is simplified to obtain a scheme with reasonable memory costs. Although the simplification is very slight, it is not difficult to find a situation in which the non-linear problem converges and the linear diverges, simply by increasing the CFL number until the iteration is right on its stability boundary for the non-linear problem.

However experience suggests the most significant is the third case - particularly for problems involving complex geometries - where minor unsteady physical phenomena are almost always present and reflected in the FPI due to the pseudo-time marching approach, leading to eventual stalling or limit cycles in the convergence. Treating these cases strictly correctly would require an unsteady computation which may be an order of magnitude more expensive than a stationary calculation, and in practice the latter tends to converge to an acceptable level anyway. However the lack of true asymptotic convergence often leads to instability in the linear problem.

In an effort to understand and mitigate these phenomenas, we consider the Recursive Projection Method (RPM), originally developed by Schroff and Keller in 1993 for the purposes of bifurcation analysis and stabilization of unstable procedures [13,14]. Since then it has been applied in aerodynamics for convergence acceleration [15,16], and stabilization of linear frequency domain solvers [17].

The idea is to regard the transient solution of the linear problem as a sum of eigenvectors of the relaxation operator $\Phi = (I - M^{-1}A)$. The application of Φ to an approximate solution then corresponds to a product of each eigenvector with its corresponding eigenvalue. Divergence of the iteration implies that there is at least one eigenvalue of Φ with modulus greater than unity. Assuming that the number of such eigenvalues is small, and that the space spanned by their eigenvectors is known, call it $\mathcal{P}$, then it must be possible to solve the projection of the problem onto this low dimensional subspace using some expensive but stable method, while solving the projection onto the complimentary subspace $\mathcal{Q}$ using the original FPI iteration, which is known to be stable there.

Newton-Raphson is typically used on $\mathcal{P}$. The space of dominant eigenvectors is determined as the calculation progresses, by applying the principle that the difference between successive applications of the FPI on $\mathcal{Q}$ form a power iteration on the dominant eigenvalues of Φ restricted to $\mathcal{Q}$.

In detail: consider the relaxation operator of (17) written

$$N(x) = (I - M^{-1}A)x + M^{-1}b, \tag{25}$$

where x and b are the unknown and RHS respectively. Let V be an orthonormal basis of $\mathcal{P}$, then orthogonal projection operators onto $\mathcal{P}$ and $\mathcal{Q}$ may be written respectively $P = VV^T$ and $Q = I - VV^T$. Further define $x_P = Px$, $x_Q = Qx$. Then the RPM iteration may be written

$$x_Q^{n+1} = QN(x^n), \tag{26}$$

$$x_P^{n+1} = x_P^n + (I - P\Phi P)^{-1}\left[PN(x^n) - x_P^n\right], \tag{27}$$

where

$$x^{n+1} = x_P^{n+1} + x_Q^{n+1}. \tag{28}$$

The "derivative" term in the Newton iteration (27) may be rearranged as follows:

$$(I - P\Phi P)^{-1} = V(I - V^T\Phi V)^{-1}V^T \tag{29}$$

$$= V(I - H)^{-1}V^T, \tag{30}$$

where H is a square matrix of size dimension of $\mathcal{P}$, whose inversion is cheap if $\mathcal{P}$ is of low dimension. The inversion is performed only once for each basis, using LU factorization [18]. Also the term $PN(x^n)$ in the Newton iteration does not require an additional residual evaluation, as $N(x^n)$ is already available from (26). The projection of this vector onto $\mathcal{P}$ is the only mesh-size dependant part of the Newton iteration (excluding the one-time evaluation of H), which is therefore extremely cheap.

To determine the basis itself consider the sequence

$$K = \left[\Delta x_Q^n, \Delta x_Q^{n-1}, \ldots, \Delta x_Q^{n-k+1}\right], \tag{31}$$

where $\Delta x_Q^n = x_Q^{n+1} - x_Q^n$ are readily available from the iteration on $\mathcal{Q}$. For a general non-linear N a Taylor expansion of $QN(x^n)$ gives

$$x_Q^{n+1} = QN(x^n) \tag{32}$$

$$= Q\left(N(x^{n-1}) + \frac{\partial N}{\partial x}\Delta x^n + \mathcal{O}(\Delta x^n)^2\right), \tag{33}$$

$$= Q\left(x^n + \Phi\Delta x^n + \mathcal{O}(\Delta x^n)^2\right), \tag{34}$$

$$= x_Q^n + Q\Phi Q\Delta x^n + Q\Phi P\Delta x^n + \mathcal{O}(\Delta x^n)^2, \tag{35}$$

where $Q\Phi P \approx 0$ because $\mathcal{P}$ is an approximately invariant subspace of Φ, so

$$\Delta x_Q^n = x_Q^{n+1} - x_Q^n \tag{36}$$

$$= Q\Phi Q\Delta x^{n-1} + \mathcal{O}(\Delta x_Q^n)^2 + \mathcal{O}(\Delta x_P^n)^2. \tag{37}$$

In our case N is linear, so all higher-order derivatives in the Taylor expansion vanish and $Q\Phi P = 0$, so

$$\Delta x_Q^n = Q\Phi Q \Delta x^{n-1}. \tag{38}$$

Therefore K is the k dimensional Krylov space for $Q\Phi Q$ seeded with Δx_Q^0. Asymptotically this subspace will tend to contain the dominant k eigenvectors of $Q\Phi Q$, as they are the components of Δx_Q^0 most amplified by repeated application of the operator.

Via QR factorization an orthogonal basis $\bar{Q}$ for K is obtained

$$K = \bar{Q}\bar{R}, \tag{39}$$

whereby column pivoting is used such that the sequence of diagonal elements $|\bar{R}_{ii}|$ of the upper triangular matrix $\bar{R}$ is decreasing. Additional vectors for V are then chosen on the basis of the *Krylov Acceptance Ratio*, κ. For the largest i satisfying

$$\frac{|\bar{R}_{ii}|}{|\bar{R}_{i+1i+1}|} > \kappa, \tag{40}$$

the first i columns of $\bar{Q}$ are added to V. Since these vectors lie in $\mathcal{Q}$ they are already orthogonal to the existing V, and further orthogonalization is not necessary.

The user defined constant κ controls the accuracy, and indirectly the size, of the basis. If κ is very large the largest eigenmodes must dominate the others by a large margin. This corresponds to many applications of the operator N, and hence many iterations, but also implies that the power iteration will be more highly converged, and the resulting basis therefore more accurate. Unfortunately κ is problem dependant, as the convergence of the power iteration depends on the distribution of the dominant eigenvectors. Based on our experience we choose κ in the range $1 \times 10^3 - 1 \times 10^4$.

Note that eigenvalues of real matrices have either zero real part or occur in complex conjugate pairs. If (40) is to be able to identify these pairs, which grow at identical rates, then the dimension k of the Krylov subspace must be at least 3. Similarly if there are n eigenvalues of Φ of *identical* modulus then k must be at least $n+1$. This latter situation has not been observed in practice for $n > 2$.

The storage of V and K dominate the memory requirements of the method, so it is advantageous to keep k small. In practice the number of dominant eigenvalues treatable by the Newton method is limited by the storage of V, rather than the CPU cost. In the following we set $k = 3$ and place an upper limit of 20 on the size of V.

Since $\mathcal{P}$ contains the dominant eigenvalues of Φ it is possible to solve an eigenvalue problem on this low dimensional subspace to obtain estimates for its eigenvalues and eigenvectors, which may allow more detailed analysis of the FPI with respect to a specific problem on a specific grid. The eigenvalues of Φ restricted to $\mathcal{P}$ are given by

$$P\Phi P\bar{\Omega} = \bar{\Omega}\bar{\Xi}. \tag{41}$$

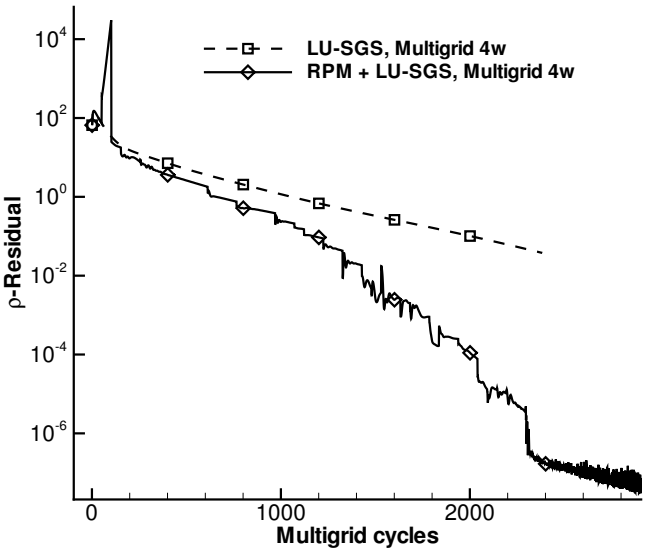

Fig. 2. Convergence acceleration of turbulent adjoint problem for an RAE2822 with RPM.

Rearranging $P\Phi P$ as in (29) we have

$$H\Omega = \Omega\Xi, \tag{42}$$

where Ω has k columns consisting of eigenvectors, and Ξ is the diagonal $k \times k$ matrix of eigenvalues. The sought dominant eigenvectors of Φ are then

$$\bar{\Omega} = V\Omega. \tag{43}$$

The algorithm is first applied to an adjoint problem on an RAE2822 aerofoil with adjointed turbulence model. Figure 2 shows convergence of the case with LU-SGS smoothed multigrid and the same FPI applied with RPM. In this case the original iteration is stable, so the dominant eigenmodes identified by RPM are stable modes with eigenvalues close to 1, which dominate the asymptotic convergence. Hence RPM operates as a convergence acceleration algorithm. To emphasize this effect, κ was set to 100, as a result of which a basis is found very quickly, but is fairly inaccurate. This can be seen in the spike right at the beginning of the convergence: initially the two curves are identical, then RPM finds a first basis vector whose low accuracy causes divergence until a second improved vector is found. RPM then recovers rapidly giving superlinear convergence. Depending on the level of residual required RPM is 2 to 10 times faster than the original method in terms of CPU time as well as multigrid cycles.

However we are more concerned with problems that do not converge initially, and we consider the very difficult case of the DLR-F6 wing-body configuration, which will also be used for optimization in Section 6. At the free stream conditions considered this case has a large region of separated flow in the junction between the upper surface of the wing and the fuselage. Further there is separation along almost the entire length of the wing, shortly before the trailing edge. All these features are well resolved by the non-linear calculation on the mesh used.

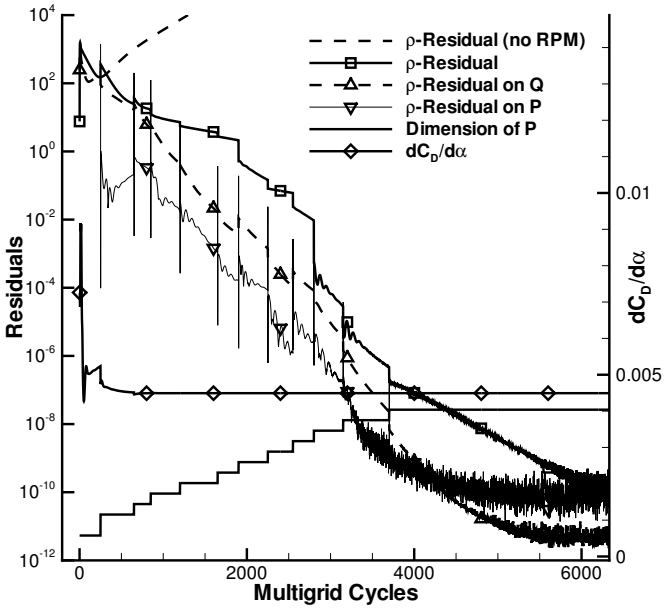

Fig. 3. Stabilization of DLR-F6 adjoint computation with RPM.

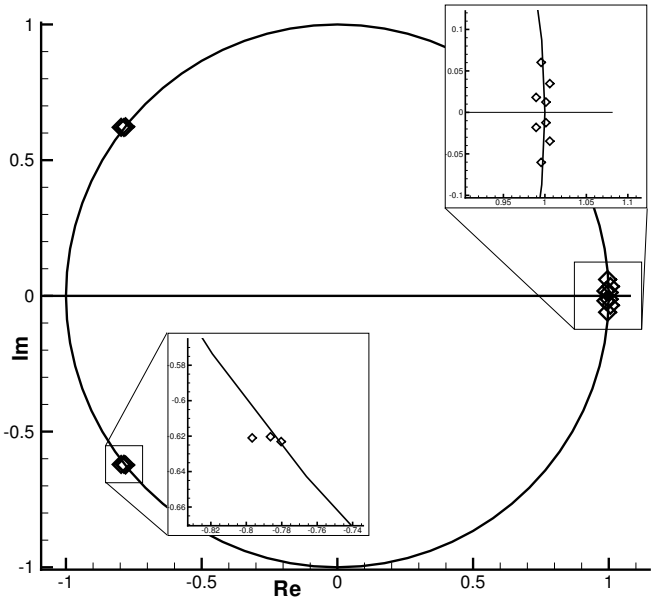

Fig. 4. Dominant eigenvalues of LU-SGS smoothed multigrid applied to the DLR-F6, determined with RPM.

For the adjoint problem on this geometry, both transposed LU-SGS and Runge-Kutta diverge for all CFL numbers, so that the only recourse is to a stabilized iteration. RPM was applied with $k = 3$ and $\kappa = 1 \times 10^3$. The QR factorization of the Krylov space was evaluated every 50 multigrid cycles to determine if a new basis vector had been found. The results are shown in Figure 3, where the residual of the problem is also shown restricted to $\mathcal{P}$ and $\mathcal{Q}$ so that the convergence of the Newton iteration is visible. Also plotted is the dimension of V (without axis), revealing when the basis updates were made.

The corresponding eigenvalues, plotted in Figure 4, show that in total there were 8 unstable modes. Examining the associated eigenvectors shows that they are all large in the regions of separation, and particularly in the recirculating corner flow. The eigensystem analysis has therefore allowed the conclusive identification of the adjoint convergence difficulties with the separation in the flow.

A major advantage of the RPM is that its implementation requires no modification of the flow solver, requiring as it does only application of solver FPI iterations to given input vectors. This makes it possible to implement the entire algorithm in Matlab [16], calling the external flow solver each time $N(x)$ is required. In contrast to the use of GMRES as a stabilizer, RPM does not require a priori knowledge of the size of the unstable space, which is expanded as necessary.

As for all Krylov methods, RPM does not offer grid independent convergence, and is therefore likely to scale well with problem size only in combination with a FPI containing multigrid. Further, as problem complexity increases, the number of unstable modes is also likely to increase, resulting in a larger basis with its associated costs. Parallelization of the algorithm is complicated by the presence of the QR factorization. While distributed memory QR algorithms exist, they are complex, and often have onerous restrictions. For example the ScaLAPACK implementation [19] requires that all but one domain have the same dimension, which is unlikely to be compatible with existing domain decomposition and load-balancing code. The alternative is a modification of RPM with a basis local to each domain, updated either independently or globally. An investigation of this issue is underway.

6. Gradient-Based Optimization

The first application of the adjoint method in aerodynamics was to flow control [20,21], and the discrete adjoint method developed in the previous sections is applied here in a similar context. The problem considered is drag minimization at constant lift of the DLR-F6 wing-body configuration. We use a similar strategy to that already described in [9,22]. The on-flow is at Mach 0.75, at a relatively low Reynolds number of 3×10^6 and initial angle-of-attack of 0°. The surface of the computational grid is shown in Figure 5 and is entirely structured, a structured layer of constant thickness extends from this in order to resolve the boundary layer, the stacks are then topped with pyramids and the remainder of the domain is filled with tetrahedra [23]. The final mesh is coarse, with a total of 120 thousand points, however it captures well the large region of separated flow already mentioned, which appears in the corner between the upper surface of the wing and the fuselage. The memory requirements and relative performance of the adjoint solver for this case were given in Table 1. The adjoint solver is applied with adjointed 3W multigrid smoothed LU-SGS and the RPM outer iteration. Due to robustness problems only the adjoint mean-flow equations were solved while the eddy-viscosity was frozen. The gradi-

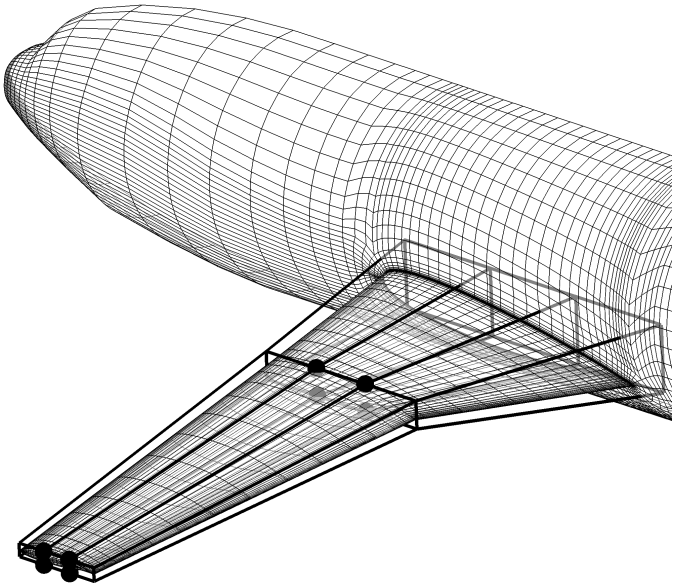

Fig. 5. F6 wing-fuselage showing parameterization of the wing with a free-form deformation box.

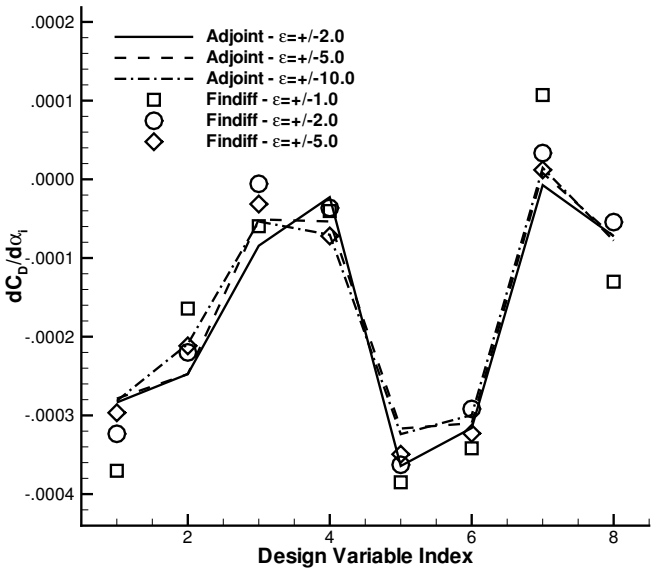

Fig. 6. Comparison of gradients evaluated with finite difference and adjoint for various step sizes.

ents thus obtained were used to drive a conjugate gradient algorithm.

The parameterization of the wing is also shown in Figure 5. The dark lines show the position of a free-form deformation control box [24], the vertical positions of the 8 vertices marked with a circle are the design variables. Note that since the bounding box passes inside the fuselage, the wing-body junction also varies, and this is accounted for by the geometry and grid generation process. The lift constraint is met by modifying the angle-of-attack in the so-called *target-lift* mode of the solver. Because we wish to minimize the drag at the target lift C_L^*, rather than at the preexisting lift C_L, the objective function must be modified to consider the constraint consistently [25],

$$I = C_D - \frac{(\mathrm{d}C_D/\mathrm{d}\alpha)}{(\mathrm{d}C_L/\mathrm{d}\alpha)}\left(C_L - C_L^*\right), \tag{44}$$

so gradients of C_L are also needed.

The extremely low number of design variables, and the coarse grid, allow the adjoint optimization to be compared with an optimization using gradients obtained using central finite differences:

$$\frac{\mathrm{d}I}{\mathrm{d}\alpha} \approx \frac{W(\alpha+\epsilon) - W(\alpha-\epsilon)}{2\epsilon}, \tag{45}$$

whereby each gradient of I with respect to all 8 design parameters requires $2 \times 8 = 16$ non-linear solutions. If a typical geometry parameterization with ≈ 50 parameters were to be used, this calculation would become impractical. The computation of the adjoint gradients also includes a finite difference component, due to the dependence of R and I on the computational grid X, which introduces terms $\partial R/\partial X$ and $\partial I/\partial X$ in (4). Since a explicit linearization of these terms is not presently available in the code, they are also evaluated using central differencing, at a cost of two mesh deformations per design variable. Hence increasing the number of design variables here incurs relatively minor additional costs, see e.g. [1].

To examine the errors in the gradient produced due to the approximations of the Jacobian, the gradients of C_D with respect to each design variable were computed using central finite differences with various step sizes ϵ, for the initial geometry. The adjoint gradients were calculated similarly, using various step sizes for the evaluation of $\partial R/\partial X$. The results are shown in Figure 6 where it is apparent that the two sets of gradients agree well, and that the adjoint gradients are less sensitive to the step size - as might be expected.

Given the successful gradient validation we proceed to the optimization, the convergence of which is shown in Figure 7. The horizontal axis shows the number of calls to the flow solver (both linear and non-linear), thereby approximately representing the computational effort required. Symbols indicate gradient evaluations; ϵ was set to 2.0 for both adjoint and finite difference gradients. As the scale of the design space is initially unknown the first gradient is used to estimate the step in the design space necessary to produce a 2% reduction in C_D. Given this step the adjoint gradient actually produces a 1.94% reduction while the FD gradient gives a 2.03% reduction, increasing our confidence in both gradients. After 34 solver calls the adjoint is unable to reduce the drag further, giving a final reduction of 42 counts. The FD optimization requires three times as many iterations and reduces the drag by 47 counts. The optimizations took in total 24 and 42 hours respectively on 16 processors, the adjoint being run sequentially due to RPM. However as the time required for the FD gradient evaluation increases proportionally to the number of design variables, the gap in CPU time will widen dramatically for practical problems with hundreds of design vari-

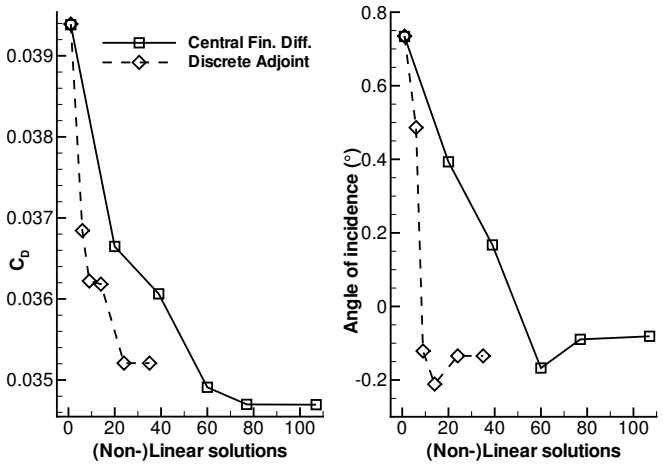

Fig. 7. Convergence of the F6 drag-minimization optimization with finite-difference gradients and adjoint gradients.

ables.

The discrepancy in the final solutions of the two optimizations is likely a result of the adjoint not delivering gradients that are as accurate as the central finite-differences. This in turn is likely a result of the frozen eddy-viscosity in the adjoint, and will be the subject of further investigation.

7. Adjoint Error Estimation

One of the main advantages of unstructured CFD codes is their ability to locally refine the mesh, allowing improvement of flow feature resolution automatically. For this purpose adaptation indicators have been developed sensitive to strong gradients and vorticity, so-called *feature-based* indicators. However while engineers are interested in these features *qualitatively*, they are often *more* interested in the behaviour of functionals of the flow field, such as C_D, *quantitatively*. The demands on accuracy made by quantitative outputs is typically much higher than for qualitative outputs, and therefore adaptation is often more necessary for the latter. Unfortunately good resolution of flow features does not necessarily imply improvement in functional accuracy. In fact, using feature-based indicators can lead to *apparent* mesh convergence that results in erroneous values for these functionals. A way to improve this situation is to use the adjoint problem to obtain information about where discretization errors in the solution have an impact on the functional in question.

Pierce and Giles [3,26] use the adjoint problem for error *correction* of integral functionals in a structured finite volume context. Their correction term is made up of pointwise contributions which makes it an appealing choice for an adaption indicator. In a series of papers, Venditti and Darmofal [4,27] develop a correction and refinement strategy for unstructured finite volume methods using a discrete adjoint solver. This is derived via a Taylor series expansion of the residual on a coarse mesh to an embedded fine mesh, where then many terms are approximated to avoid most computations on the fine mesh.

A similar adaption strategy has been implemented in *TAU*. The indicator is evaluated as follows:

(i) Obtain non-linear and adjoint solutions on the initial mesh.
(ii) Globally refine the mesh and linearly interpolate the non-linear solution onto it.
(iii) Compute a single residual R on the fine mesh, and sum its contributions to coarse grid cells on the original mesh with volume weighting.
(iv) An optional smoothing of the residuals may be performed [28], but was unnecessary here.
(v) The adaptation indicator ϵ_i is calculated point-wise as $\epsilon_i = R_i^T \Lambda_i$.

The evaluation of the residual on the *fine* mesh is essential as it is zero on the initial mesh by construction.

Given ϵ_i, those edges of the grid whose end-points satisfy

$$|\epsilon_i| \geq \sigma \frac{\epsilon^t}{|\epsilon^G|} \text{ where } \epsilon^G = \sum_i \epsilon_i, \tag{46}$$

are marked for adaptation [29]. Here σ is the standard deviation of the ϵ_i, and ϵ^t is the prescribed uncertainty one wishes to achieve in the functional. The quantity ϵ^G is a global error estimate. Finally the mesh is adapted to these criteria using a module [30].

This algorithm is applied to 2d inviscid flow about a transonic NACA0012 at a Mach number of 0.85 and angle of attack of 2°. Results are compared against the baseline of global grid adaption, as well as the feature-based adaptation already mentioned. The comparison proceeds on the basis of error in C_L, an accurate estimate for which is obtained by Richardson extrapolation from the results of global adaptation.

The feature-based adaptation was set up to introduce approximately 30% more points into the mesh at each adaptation step, whereas the goal-oriented procedure flagged as many elements of the mesh for refinement as it deemed necessary to achieve the prescribed error tolerance. This amount was usually well in excess of that produced by the feature-based adaptation, which is desirable considering the relatively high computational cost of ϵ_i.

The convergence of all methods is shown in Figure 8, whereby the horizontal axis gives number of mesh points, aiming to represent the computational cost of each individual flow calculation shown. Immediately apparent is the surprising result that feature-based adaptation does not appear to converge at all. The goal-oriented adaptation gives good results for all error tolerances. After three adaptation steps all meshes provide a lift coefficient that is within 0.0003 of the exact value. Surprisingly, the mesh with the lowest requested accuracy yields the best results for the first two adaptation cycles.

The mesh after the third adaptation for $\epsilon_t = 0.001$ can be seen in Figure 9 together with the third feature-adapted mesh, which has a similar number of points, but appears much less dense due to the large number of points "wasted"

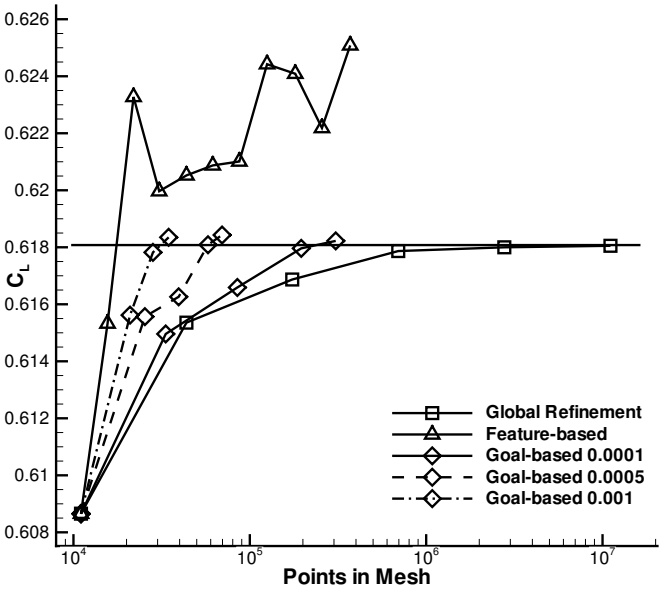

Fig. 8. Convergence of C_L w.r.t. grid size for various adaptation strategies. Estimated exact solution is marked with a horizontal line.

in the wake, in resolving the discontinuities excessively, and far above the profile in the main shock. In particular the upper surface of the aerofoil – where numerical dissipation contributes to the position of the shock, and thereby strongly influences the lift – is completely unrefined. Given this the lack of convergence seen in Figure 8 is less surprising.

In contrast the error-adapted mesh concentrates points close the to aerofoil, resolving the upper and lower surfaces well. It also resolves the shocks over a broader area, and refines the (critical) trailing edge at every iteration, a consequence of a singularity in the adjoint solution.

These satisfying results are none the less preliminary. The terms used for the adaption indicator can also be used to compute a correction term for the functional, namely ϵ^G, which may allow a more accurate solution on a given grid. This and the further development of the adaption criteria are to be the subject of future work.

8. Conclusions

Our immediate goal in the investigation of the discrete adjoint method was to develop an solver which is at least as robust and efficient as the non-linear solver, as only then can it reliably be applied to practical engineering problems in optimization and adaptation. The algorithms described in this paper go a long way towards achieving this goal, in particular the memory requirements and time per residual evaluation are acceptable. The adjoints of the non-linear solution algorithms allow robust and efficient convergence, at least in cases which behave well non-linearly. For the cases that behave poorly RPM offers a means of obtaining a solution, provided there are not more than 10 to 20 unstable modes in the problem. However there remain cases

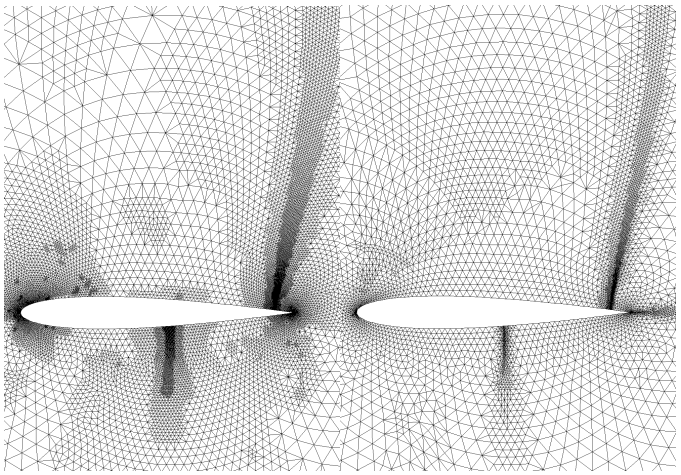

Fig. 9. Grid for transonic NACA0012 with goal- and feature-based adaptation, left and right respectively. The two grids have similar size.

that are presently insoluble with reasonable computer resources. Some contain too many unstable modes, others diverge before RPM has the chance to find a basis (for example with an extremely large eigenvalue appearing in the turbulence equations). The tackling of these problems represents the next challenge. It is envisaged that use of eigensystem analysis will aid the design of FPIs especially suited as preconditioners to RPM and Krylov methods. The parallelization of these algorithms is also necessary for practical applications.

With respect to gradient-based optimization and error estimation we feel that both fields offer considerable potential; the former in the integration of its increasingly efficient and maturing algorithms into the aircraft design process, the latter in terms of new approaches and the systematic study of existing solvers with respect to accuracy.

Acknowledgments

We are very grateful to Stefan Görtz for sharing his extensive insight into and experience of the recursive projection method.

References

[1] J. Brezillon, O. Brodersen, R. Dwight, A. Ronzheimer, J. Wild, Development and application of a flexible and efficient environment for aerodynamic shape optimisation, in: Proceedings of the ONERA-DLR Aerospace Symposium (ODAS), Toulouse, 2006.

[2] A. Griewank, Evaluating Derivatives, Principles and Techniques of Algorithmic Differentiation, Number 19 in Frontiers in Appl. Math. SIAM, Philadelphia, 2000, ISBN 08-987-1451-6.

[3] M. Giles, N. Pierce, Improved lift and drag estimates using adjoint Euler equations, in: American Institute of Aeronautics and Astronautics, Paper AIAA-1999-3293, 1999.

[4] D. Venditti, D. Darmofal, Adjoint error estimation and grid adaption for funtional outputs: Application to quasi-one-dimensional flow, Journal of Computational Physics 164 (2000) 204–227.

[5] T. Gerhold, M. Galle, O. Friedrich, J. Evans, Calculation of complex 3D configurations employing the DLR TAU-Code, in: American Institute of Aeronautics and Astronautics, Paper AIAA-97-0167, 1997.

[6] R. Dwight, Efficiency improvements of RANS-based analysis and optimization using implicit and adjoint methods on unstructured grids, Ph.D. thesis, School of Mathematics, University of Manchester (2006).

[7] A. Jameson, W. Schmidt, E. Turkel, Numerical solutions of the Euler equations by finite volume methods using Runge-Kutta time-stepping schemes, in: AIAA Paper, AIAA-81-1259, 1981.

[8] J. Edwards, S. Chandra, Comparison of eddy-viscosity transport turbulence models for three-dimensional shock-seperated flowfields, AIAA Journal 34 (4) (1996) 1–16.

[9] R. Dwight, J. Brezillon, Effect of various approximations of the discrete adjoint on gradient-based optimization, in: Proceedings of the 44th AIAA Aerospace Sciences Meeting and Exhibit, Reno NV, AIAA-2006-0690, 2006.

[10] D. Mavriplis, Personal communication (2004).

[11] S. Yoon, A. Jameson, An LU-SSOR scheme for the Euler and Navier-Stokes equations, AIAA Journal 26 (1988) 1025–1026.

[12] M. Giles, On the iterative solution of adjoint equations, Automatic Differentiation: From Simulation to Optimization (2001) 145–152.

[13] G. Schroff, H. Keller, Stabilization of unstable procedures: The Recursive Projection Method, SIAM Journal of Numerical Analysis 30 (4) (1993) 1099–1120.

[14] H. Keller, RPM: A remedy for instability, in: D. Estep, S. Tavener (Eds.), Collected Lectures on the Preservation of Stability under Discretization, SIAM Proceedings in Applied Mathematics 109, 2002, pp. 185–196.

[15] J. Möller, Aspects of the Recursive Projection Method applied to flow calculations, Ph.D. thesis, NADA, KTH, Stockholm, Sweden, ISBN 91-7283-940-6, TRITA-NA-0444 (2005).

[16] S. Görtz, J. Möller, Evaluation of the Recursive Projection Method for efficient unsteady turbulent CFD simulation, ICAS 2004.

[17] M. Campobasso, M. Giles, Stabilization of a linear flow solver for turbomachinery aeroelasticity by means of the recursive projection method, AIAA Journal 42 (9) (2004) 1765–1774.

[18] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, D. Sorensen, LAPACK Users' Guide, 3rd Edition, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1999.

[19] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, R. C. Whaley, ScaLAPACK Users' Guide, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1997.

[20] O. Pironneau, On optimum design in fluid mechanics, Journal of Fluid Mechanics 64 (1974) 97–110.

[21] A. Jameson, Aerodynamic design via control theory, Journal of Scientific Computing 3 (1988) 233–260.

[22] J. Brezillon, R. Dwight, Discrete adjoint of the Navier-Stokes equaitons for aerodynamic shape optimization, in: Evolutionary and Deterministic Methods for Design, EUROGEN, 2005.

[23] J. Wild, Acceleration of aerodynamic optimization based on RANS-equations by using semi-structured grids, in: Design Optimization International Conference, Athens, 2004.

[24] A. Ronzheimer, Shape based on freeform deformation in aerodynamic design optimization, in: ERCOFTAC Design Optimization International Conference, Athens, 2004.

[25] J. Reuther, J. Alonso, M. Rimlinger, D. Sanders, A. Jameson, Constrained multipoint aerodynamic shape optimization using an adjoint formulation and parallel computers, Journal of Aircraft 36 (1999) 51–60.

[26] N. Pierce, M. Giles, Adjoint recovery of superconvergent functionals from PDE approximations, SIAM Review 42 (2) (2000) 247–264.

[27] D. Venditti, D. Darmofal, Anisotropic grid adaptation for functional outputs: Application to two-dimensional viscous flows, Journal of Computational Physics 187 (2003) 22–46.

[28] J.-D. Müller, M. Giles, Solution adaptive mesh refinement using adjoint error analysis, in: American Institute of Aeronautics and Astronautics, Paper AIAA-2001-2550, 2001.

[29] H.-J. Kim, K. Nakahashi, Output-based error estimation and adaptive mesh refinement using viscous adjoint method, in: American Institute of Aeronautics and Astronautics, Paper AIAA-2006-1395, 2006.

[30] T. Alrutz, M. Rütten, Investigation of vortex breakdown over a pitching delta wing applying the DLR TAU-Code with full, automatic grid adaptation, Paper 5162, AIAA (2005).