Computational Fluid Dynamics 2002, Proceedings of The Second International Conference on Computational Fluid Dynamics, ICCFD, Sydney, Australia, 15-19 July, 2002 Springer 2003, pp82-87 ISBN 3-540-00793-3

A Discrete Sensitivity Solver for an Industrial CFD Code via Automatic Differentiation

David W. F. Standingford[1] and Shaun A. Forth[2]

[1] BAE SYSTEMS Advanced Technology Centre, Sowerby Building, PO Box 5, FPC 267 Filton, Bristol BS34 7QW, UK, email: David.Standingford@baesystems.com
[2] Applied Mathematics & Operational Research, ESD, Cranfield University (RMCS Shrivenham), Swindon SN6 8LA, UK, email: S.A.Forth@rmcs.cranfield.ac.uk

Abstract. We report on development and validation of a discrete sensitivity solver for the BAE SYSTEMS/Airbus UK CFD code FLITE3D. We used the Odyssée automatic differentiation (AD) tool to create a discrete forward sensitivity version of FLITE3D. Validation is via comparing sensitivities of integrated forces (lift, drag, side force) with respect to (w.r.t) angle-of-attack (AoA) α, calculated using the sensitivity solver and central-differencing. Validation, w.r.t. changes in the surface and field mesh, is performed by setting sensitivities of all mesh related quantities as if rotated by an infinitesimal angle $\Delta\alpha$. Such sensitivities correspond to those calculated w.r.t AoA α. We investigate calculating discrete sensitivities under 2 approximations regarding so-called mesh sensitivities, concluding that they are not applicable to FLITE3D. We present results of a wing geometry optimisation using forward sensitivities.

1 Introduction

Industrial use of CFD is now routine and the current challenge is to incorporate large scale CFD codes into the engineering design cycle via design optimisation. The associated use of gradients is non-trivial and is the subject of the monograph [8]. Aerodynamic designs may have 100's of design variables making the cost of finite–differencing prohibitive. This prompted interest in adjoint methods which compute gradients in time proportional to the (small) number of design objectives [5]. Such methods may be implemented at the continuous level (e.g. [1]) in which adjoint PDEs are formulated, discretised and solved. This requires mathematical analysis, both initially and as the CFD solver is updated, and may be perceived by industry as hard to maintain. Discrete adjoint methods, in which the discretised flow solver is adjoined, are therefore of interest. A discrete adjoint solver may be generated and maintained by hand–coding [9], again with maintenance problems, or via automatic differentiation (AD) [6] tools. We are using the AD tool Odyssée [4] to develop both forward and adjoint sensitivity solvers for the BAE SYSTEMS/Airbus UK compressible flow solver FLITE3D. Here we report on development and validation of the forward sensitivity solver.

2 The Flite3D CFD Package

The CFD package FLITE3D consists of a number of separate programs.

Surface mesh For geometric parameters $\underline{\beta}$ generates surface mesh $\boldsymbol{x_S}$.
Volume mesh Generate volume mesh $\boldsymbol{x}$ using advancing front algorithm.
Solver preprocessor Produces agglomeration data for multigrid acceleration and edge–based data for the solver.
Flow Solver Uses unstructured, node–based finite volumes with second–order spatial accuracy, Runge–Kutta (RK) time integration and multigrid acceleration to solve the compressible Euler equations. It is similar to [3], by default uses a pseudo-Laplacian numerical dissipation, and features many complex boundary conditions associated with aeronautical design.

A 2-D mesh, close to a geometric boundary, is depicted in Fig. 1. Away from

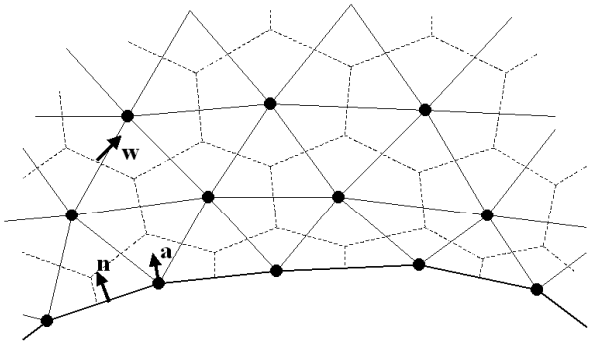

Fig. 1. Flite3D grid showing: actual mesh (unbroken lines); node points (black dots); and dual mesh (dashed lines) defining the flow–solver control–volumes.

the boundary, an area-weighted normal to a typical control volume is labelled $\boldsymbol{w}$. Control volumes for boundary vertices are cut by the boundary. An area weighted normal for a boundary face is denoted $\boldsymbol{n}$ and it contributes to several control–volumes. A unit point normal for a boundary vertex is denoted $\boldsymbol{a}$.

Neglecting RK time-stepping and multigrid, FLITE3D's solver is described by (1–3). For the flow–solver (3), $\boldsymbol{P}_L(\boldsymbol{u}^n, \boldsymbol{x}) = -\Delta t/\boldsymbol{V}$ is the diagonal matrix of negative time steps $-\Delta t_i^n$ divided by control volumes V_i for each cell i in the mesh. In (3) wall zero–mass–flux boundary conditions are applied in 2 stages [3]. Before residual evaluation, the momentum $\boldsymbol{\rho u}_j$ for boundary vertices j is *chopped* removing its normal component $\boldsymbol{\rho u}_j \rightarrow \boldsymbol{\rho u}_j - (\boldsymbol{a}_j \cdot \boldsymbol{\rho u}_j)\boldsymbol{a}_j$. After residual evaluation, the corresponding momentum residuals $\boldsymbol{R}_{Mi}$ are similarly chopped $\boldsymbol{R}_{Mi} \rightarrow \boldsymbol{R}_{Mi} - (\boldsymbol{a} \cdot \boldsymbol{R}_{Mi})\boldsymbol{a}_i$ to ensure stability.

$$\left.\begin{array}{l} \textbf{Surface Mesh Generator} \\ \text{Given design parameters } \underline{\beta} \\ \text{Surface mesh generation } \boldsymbol{x_S} = \boldsymbol{X_S}(\underline{\beta}) \\ \text{Calculate point and face surface normals } \boldsymbol{a}(\boldsymbol{x_S}, \underline{\beta}), \boldsymbol{n}(\boldsymbol{x_S}) \end{array}\right\} \quad (1)$$

Volume Mesh Generator
Volume Mesh generation $\boldsymbol{x} = \boldsymbol{X}(\boldsymbol{x_S})$
Calculate Volume Mesh normals $\boldsymbol{w}(\boldsymbol{x}, \boldsymbol{x_S})$ (2)

Flow Solver
Initialise flow variables $\boldsymbol{u}^0$
`Do` $n = 0, n_{max} - 1$
on surface set momentum $\rho\boldsymbol{u}$ s.t. $\rho\boldsymbol{u}.\boldsymbol{a} = 0$
$\boldsymbol{r}^n = \boldsymbol{R}(\boldsymbol{u}^n, \boldsymbol{x}, \boldsymbol{w}, \boldsymbol{n})$
on surface set momentum residual $\boldsymbol{R}_M$ s.t. $\boldsymbol{R}_M.\boldsymbol{a} = 0$
`If (converged) Exit Do Loop`
$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n - \boldsymbol{P}_L(\boldsymbol{u}^n, \boldsymbol{x})\boldsymbol{r}^n$
Aerodynamic Coefficients $[C_L, C_D, C_S] = C(\boldsymbol{u}^n, \boldsymbol{x}, \boldsymbol{n})$
`EndDo` (3)

3 Forward Sensitivities via AD Tool Odyssée

From the flow solver we extracted 75 subroutines, comprising 21,000 lines of code (including comments), concerned with calculating normals ($\boldsymbol{w}$, $\boldsymbol{n}$, $\boldsymbol{a}$), flow residuals and conservative variable updates. Twelve of these routines were individually differentiated, either by hand or with Odyssée's aid. The remaining 63 were differentiated exclusively by Odyssée. Forty–three, concerning the flow solver, were differentiated in one application of Odyssée taking 20 minutes CPU time on a COMPAQ Alpha XP1000 workstation (RAM: 1GB, clock–speed 667 MHz). In total 38,500 lines of differentiated Fortran code were produced. To ease code-maintenance, the C pre–processor and makefiles are used to automate nearly all differentiation.

When differentiating we chose free-stream Mach number M, AoA α, lateral angle of incidence β, together with all mesh points $\boldsymbol{x_S}$, $\boldsymbol{x}$ as independent variables. Sensitivities of lift C_L, drag C_D and side force C_S coefficients may then be calculated.

We validated the AD generated sensitivity solver first w.r.t. changes in AoA α and then w.r.t. changes in the surface $\boldsymbol{x_S}$ and volume $\boldsymbol{x}$ meshes.

3.1 Angle-of-Attack (AoA) Sensitivities

As is standard, flows of speed U at incidence α are calculated by rotating the far-field velocity $u = U\cos(\alpha)$, $v = U\sin(\alpha)$. To validate our sensitivity solver we calculated $dC_L/d\alpha$ using both AD code and central-differencing (CD). Convergence criteria were, at iteration n, $|C_L^i - C_L^{i-10}| \leq 10^{-5} \times \max\left(1, |C_L^i|\right)$ for $i = n$ and $i = n - 10$. For the sensitivity solver we ensured,

$$\left|\frac{dC_L^i}{d\alpha} - \frac{dC_L^{i-10}}{d\alpha}\right| \leq 10^{-5} \times \max\left(1, \left|\frac{dC_L^i}{d\alpha}\right|\right),$$

for $i = n, n - 10$. Identical convergence criteria were used for all further calculations. Results are in Table 1, and show good agreement of the two techniques. Table 2(a), gives AD results for a succession of finer meshes.

Table 1. AoA α sensitivities for M6 wing, AoA $\alpha = 5^o$, Mach no. 0.84 calculated by AD and approximated by CD (perturbations $\pm\Delta\alpha$) on 14,148 vertex mesh.

Method	$\Delta\alpha$	$dC_L/d\alpha$	$dC_D/d\alpha$	$dC_S/d\alpha$
AD	$\to 0$	6.5743×10^{-2}	1.1317×10^{-2}	9.8631×10^{-4}
CD	0.1	6.5729×10^{-2}	1.1318×10^{-2}	9.8373×10^{-4}
CD	0.05	6.5736×10^{-2}	1.1322×10^{-2}	9.8329×10^{-4}
CD	0.01	6.5887×10^{-2}	1.1321×10^{-2}	9.6541×10^{-4}

Table 2. Grid convergence of force sensitivities to AoA α calculated by 3 forward mode AD enabled techniques.

(a) No approximation and freestream flow derivatives set by rotation.

Mesh pts.	$dC_L/d\alpha$	$dc_d/d\alpha$	$dc_s/d\alpha$
28,244	7.3186×10^{-2}	1.1625×10^{-2}	1.8982×10^{-3}
47,177	7.3712×10^{-2}	1.1495×10^{-2}	2.0702×10^{-3}
70,102	7.4098×10^{-2}	1.1752×10^{-2}	2.0775×10^{-3}

(b) No approximation and derivatives of all grid quantities set by rotation.

Mesh pts.	$dC_L/d\alpha$	$dc_d/d\alpha$	$dc_s/d\alpha$
28,244	7.3186×10^{-2}	1.1626×10^{-2}	1.8980×10^{-3}
47,177	7.3711×10^{-2}	1.1495×10^{-2}	2.0702×10^{-3}
70,102	7.4098×10^{-2}	1.1752×10^{-2}	2.0776×10^{-3}

(c) Approximated by neglecting volume mesh sensitivities.

Mesh pts.	$dC_L/d\alpha$	$dC_D/d\alpha$	$dC_S/d\alpha$
28,244	2.1455×10^{-1}	2.6997×10^{-2}	3.9774×10^{-3}
47,177	2.1970×10^{-1}	2.6838×10^{-2}	4.4372×10^{-3}
70,102	2.2079×10^{-1}	2.7964×10^{-2}	4.2330×10^{-3}

3.2 Differentiation with respect to all Mesh Quantities

To validate the sensitivity solver to changes in surface and volume meshes, we set sensitivities of all mesh points as for a rotation α about the y (wing span–wise) axis. This unconventional approach calculates AoA sensitivities as if by rotating the mesh by the infinitesimal $\Delta\alpha \to 0$. We calculated $d(\boldsymbol{w}, \boldsymbol{n}, \boldsymbol{a})/d\alpha$ using differentiated versions of volume, surface and point surface normal calculation routines. We zeroed far-field flow derivatives and the resulting sensitivities of Table 2(b) show excellent agreement with Table 2(a).

4 Eliminating Volume Mesh Sensitivities

In Sect. 3.2 we used sensitivities of all mesh related quantities in calculating surface mesh sensitivities. For design sensitivities of solution algorithm (1-3),

this necessitates differentiating both surface (1) and volume (2) mesh generators. We would anticipate improved efficiency if we only differentiated the surface grid generation and flow solver stages. This approach is motivated by noting that in continuous sensitivity formulations there are no volume mesh sensitities [2].

4.1 Neglecting Volume Mesh Normals w

We set all sensitivities of volume mesh normals $\boldsymbol{w}$ to zero, retaining sensitivies of the surface $\boldsymbol{n}$ and point surface $\boldsymbol{a}$ normals of Fig. 1. Results are in Table 2(c). We might expect grid convergence to the results of tables 2(a,b). Instead, sensitivities $dC_L/d\alpha$, $dC_D/d\alpha$ and $dC_S/d\alpha$ converge, but to different values.

Others [2], have obtained a consistent continuous sensitivity solver from a discrete (AD) sensitivity solver on neglecting variation of interior mesh points but, crucially, by also augmenting the boundary conditions. Extending their analysis to the Euler equations is straightforward.

Consider the zero mass flux boundary condition,

$$\boldsymbol{\rho u} \cdot \boldsymbol{a} = 0. \tag{4}$$

Application of an AD tool to (4)'s discretisation produces code equivalent to,

$$\frac{\partial \boldsymbol{\rho u}}{\partial \alpha} \cdot \boldsymbol{a} + \boldsymbol{\rho u} \cdot \frac{\partial \boldsymbol{a}}{\partial \alpha} = 0. \tag{5}$$

Now consider $\boldsymbol{\rho u}(\boldsymbol{x}, \alpha)$ to be a function of $\boldsymbol{x}$ and α, as when deriving a continuous sensitivity solver. Differentiating (4) gives,

$$\frac{\partial \boldsymbol{\rho u}}{\partial \alpha} \cdot \boldsymbol{a} + \boldsymbol{\rho u} \cdot \frac{\partial \boldsymbol{a}}{\partial \alpha} + \sum_{i=1}^{3} \left(\mathbf{grad}(\rho u_i) \cdot \frac{\partial \boldsymbol{x}_S(\alpha)}{\partial \alpha} \right) a_i = 0, \tag{6}$$

with $\boldsymbol{x}_S$ the location of the boundary mesh point. Obviously (6) differs from (5) only in its final term. We have hand-coded this term, and a similar modification of the surface pressure integration for C_L, C_D and C_S, into our sensitivity solver. This was facilitated by FLITE3D's option of a second–order, high resolution AUSM solver [7] which necessarily calculates gradients of momentum and pressure fields. The resulting sensitivity solver did not converge. From (3) and [3], we see that to maintain stability of the flow solver we must enforce $\boldsymbol{R}_M.\boldsymbol{a} = 0$ where $\boldsymbol{R}_M$ is the momentum residual at a boundary vertex. Comparison with (6) indicates the need for the gradient of the momentum residual to enforce the appropriate condition in the differentiated solver. Such a quantity is not calculated within FLITE3D and it would not seem sensible to pursue this further.

4.2 Incomplete Sensitivities

Mohammadi and Pironneau [8, p146-153] considered use of *incomplete sensitivities*, arguing, for example that the boundary condition (4), which on differentiation is (5), be approximated as $\boldsymbol{u}.\nabla \boldsymbol{n} = 0$. We implemented this by using

only our differentiated force calculation routine after convergence of the flow solver. The results (not shown) were seriously in error with $dC_L/d\alpha$ of incorrect sign. This indicates that incomplete sensitivities, at best, must be used under restricted conditions (e.g. fixed wing leading and trailing edges).

5 Optimisation

We employ a quadratic programming method to minimise the drag C_D of an ONERA M6 wing at a fixed lift coefficient $C_L = 0.5$. We employ a single design variable: the linear twist about the leading-edge, θ. The AoA, α is also an input. The optimal design (θ=5.9^o,α=9.7^o,C_L=0.5,C_D=0.0509) is achieved in 12 steps by supplying the sensitivity matrix J=($dC_L/d\alpha$, $dC_D/d\alpha$; $dC_L/d\theta$, $dC_D/d\theta$) at each stage. This represents a drag saving of 2.8%

Conclusions

Using the AD tool Odyssée we have developed forward sensitivity version of the industrial unstructured mesh flow solver FLITE3D. Sensitivities w.r.t. changes in AoA and the field and volume mesh have been validated. Techniques to eliminate the need for volume mesh sensitivites have been unsuccessful in line with previous work [1,9]

Acknowledgement

We thank the UK Department of Trade and Industry and Murray Cross of Airbus UK for supporting this project under the CAST program. We thank Laurent Hascoët and INRIA for use of the Odyssée AD tool.

References

1. W. K. Anderson, V. Venkatakrishnan: Computers and Fluids **28** (1999)
2. J. Borggaard, A. Verma: SIAM Journal on Scientific Computing **22** (2000)
3. P. I. Crumpton, P. Moinier, M. B. Giles: 'An Unstructured Algorithm for High Reynolds Number Flows on Highly-Stretched Grids.' In: *10th International Conference on Numerical Methods for Laminer and Turbulent Flow, 1997.*
4. C. Faure, Y. Papegay: 'Odyssée User's Guide. Version 1.7' Rapport Technique RT–0224, INRIA, Sophia-Antipolis, France (1998)
5. M. B. Giles, N. A. Pierce: An introduction to the adjoint approach to design. *Flow, Turbulence and Combustion*, to appear.
6. A. Griewank: *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation* (SIAM, Philadelphia 2000)
7. M. S. Liou and C. J. Steffen: Journal of Computational Physics **107** (1993)
8. B. Mohammadi, O. Pironneau: *Applied Shape Optimization for Fluids.* (Oxford Science Publications 2001)
9. E. J. Nielsen and W. K. Anderson: AIAA Journal **37** (1999)