Automatic Differentiation of Algorithms: From Simulation to Optimization,
Springer-Verlag, New York, 2002, pp153-160. ISBN 0387953051



# Contents

# 17 Aerofoil Optimisation via AD of a Multigrid Cell-Vertex Euler Flow Solver

**Shaun A. Forth and Trevor P. Evans**

ABSTRACT We report preliminary results in the use of ADIFOR 2.0 to determine aerodynamic sensitivities of a 2-D airfoil with respect to geometrical variables. Meshes are produced with a hyperbolic interpolation technique. The flow field is calculated using the cell-vertex method of Hall, which incorporates local time-stepping, mesh sequencing and multigrid. We present results and timings using both Finite Differences (FD) and Automatic Differentiation (AD). We investigate the effect of starting the perturbed calculation for FD and the derivative calculation for AD from either the current or freestream conditions and highlight the need for careful implementation of convergence criteria.
We attempt to make a comparative study of AD and FD gradients in an aerofoil optimisation, using the DERA CODAS method from the perspective of DERA's eventual aim, 3D viscous optimisation of wing-body configurations.

## 17.1 CFD Design Process

We consider a generic CFD solution algorithm for given design parameters $\mathbf{p}$. We use the simultaneous update of flow and objective variables since the far-field boundary condition is set using the value of the lift [8].

**Algorithm 1:** A generic CFD algorithm:

1. A mesh generator $\mathbf{X}$ gives a set of discrete mesh points $\mathbf{x} = \mathbf{X}(\mathbf{p})$.
2. Initialise discrete flow-field variables $\mathbf{u}$ and design objectives $\mathbf{c}$.
3. Update $\mathbf{u}$ and $\mathbf{c}$ via an iteration, labelled $n$, of a numerical solution procedure $\mathbf{U}$ and $\mathbf{C}$ for the Euler or Navier-Stokes equations,

$$\mathbf{u}^{n+1} = \mathbf{U}\left(\mathbf{p}, \mathbf{x}, \mathbf{u}^n, \mathbf{c}^n\right), \tag{17.1}$$

$$\mathbf{c}^{n+1} = \mathbf{C}\left(\mathbf{p}, \mathbf{x}, \mathbf{u}^{n+1}, \mathbf{c}^n\right). \tag{17.2}$$

4. Repeat step 3 until the procedure converges.

The design optimisation process proceeds by calculating the gradient $\partial\mathbf{c}/\partial\mathbf{p}$ of the objective variables with respect to the design variables (we have used up to 17 design variables to date) and then using an SQP optimisation algorithm to modify the design variables. Accurate and efficient calculation of the gradient is important in developing an effective design algorithm.

### 17.1.1 Calculating gradients of the objectives

Obtaining the gradient $\partial \mathbf{c}/\partial \mathbf{p}$ is complicated by the implicit definition of the flow field variables as the solution of a discretised PDE via the iterative process of equations 17.1 and 17.2. In this chapter we consider four alternative ways to determine these gradients.

#### Piggy-Back AD (PB-AD)

In the *Piggy-Back AD (PB-AD)* approach [6] (also termed fully differentiated [1]), we simply differentiate through Algorithm 1 and obtain derivatives with respect to $\mathbf{p}$ of all variables as they are calculated. It has been shown [1] that, under mild conditions on the iteration functions $\mathbf{U}$ and $\mathbf{C}$, such a scheme converges to the correct derivatives $\partial \mathbf{c}/\partial \mathbf{p}$.

#### Two-Phase AD (2P-AD)

Post-differentiation as advocated in [1] involves: first converging the flow solver (Algorithm 1), then performing one iteration involving the derivatives, then finally directly solving the linear system associated with the differentiated flow solver for $\partial \mathbf{u}/\partial \mathbf{p}$ and $\partial \mathbf{c}/\partial \mathbf{p}$. This approach is generally impractical in the context of CFD due to the large size of the linear system. In other applications it can be very efficient, see for instance the hand-coded post-differentiation used Chapter 7 [3] of this volume. Instead we adopt an iterative approach termed *Two-Phase AD* (2P-AD) [6], first converging the flow field and objective variables and then switching on the PB-AD algorithm to obtain the derivatives from the converged flow solution. This has the advantage of not performing expensive AD iteration early in the calculation when the flow field is far from converged. We continue to update the flow field and objective variables when calculating the derivatives.

#### Freestream Start FD (FS-FD)

We use one-sided finite differences (divided differences) to approximate $\partial \mathbf{c}/\partial \mathbf{p}$ with both the flow solution and its finite-difference perturbation obtained from free-stream starts via Algorithm 1.

#### Converged Start FD (CS-FD)

Again we use one-sided finite differences and obtain the flow solution from a freestream start, but the calculation of the perturbed solution is initialised with the converged unperturbed flow solution.

### 17.1.2 Convergence criteria

For two-phase differentiation of the flow solver and finite differencing, we used the convergence condition

$$\left|\left|\mathbf{c}^{n+1} - \mathbf{c}^{n}\right|\right|_{\infty} < \mathrm{tol}_{c}, \tag{17.3}$$

where $\mathrm{tol}_c$ is the *solver tolerance*. In addition for AD derivatives we used

$$\left|\left|\frac{\partial \mathbf{c}}{\partial \mathbf{p}}^{n+1} - \frac{\partial \mathbf{c}}{\partial \mathbf{p}}^{n}\right|\right|_{\infty} < \mathrm{tol}_{\partial \mathbf{c}/\partial \mathbf{p}}, \tag{17.4}$$

where $\mathrm{tol}_{\partial \mathbf{c}/\partial \mathbf{p}}$ is the *derivative tolerance*. We did not apply convergence conditions on the flow field variables $\mathbf{u}$ or their derivatives $\partial \mathbf{u}/\partial \mathbf{p}$.

## 17.2 ADIFOR Applied to Cell Vertex Flow Solver

We have applied the four strategies of §17.1.1 to the cell-vertex finite-volume compressible Euler flow solver `mheuler` of Hall [7]. The solver uses explicit Lax-Wendroff time-stepping and features local time-stepping, mesh sequencing, and multigrid to accelerate convergence to steady state.

Automatic differentiation of the flow solver was performed using ADIFOR version 2.0 revision D [2]. ADIFOR is a source text translation tool that takes existing Fortran 77 code and produces new code consisting of the original source code augmented with code to calculate directional derivatives. Before using ADIFOR we had to perform some minor rewriting of the flow solver so that the nonlinear iteration process abstractly described by equations 17.1 and 17.2 could be placed in a separate subroutine. ADIFOR can be used to calculate an essentially arbitrary number of directional derivatives, the user merely setting the maximum to be used. However in the present work, we used the option `AD_SCALAR_GRADIENTS=true` to force ADIFOR to produce code for a single, scalar directional derivative. This ensures that we have no short loops within the differentiated code that the compiler cannot *unroll*. In any industrial scale computer program, there are some lines of code that are not globally differentiable. In `mheuler` the Fortran intrinsics `MAX` and `ABS` are used within the second order numerical dissipation required for stability of the flow solver. ADIFOR can log the number of times such a function is called at a point of potential non-differentiability, and the user has control over how this is done. We used the option `AD_EXCEPTION_FLAVOR=reportonce` to produce a list of such *exceptions* at the end of the calculation. In the future we shall use `AD_EXCEPTION_FLAVOR=performance` to improve performance once we are satisfied that points of non-differentiability encountered are unimportant.

### 17.2.1 Comparison of AD and FD directional derivatives

To compare the relative accuracy of the AD and FD directional derivatives, we calculated the inviscid flow about a NACA 64A airfoil section at a Mach number of 0.63 and zero incidence on a "C"-mesh of 25 points normal to the airfoil and 193 around the airfoil and wake cut. We determined the directional derivative associated with a perturbation to a spline controlling the shape on the upper airfoil surface near the leading edge. Assuming that the PB-AD results are correct, then Figure 17.1 shows how

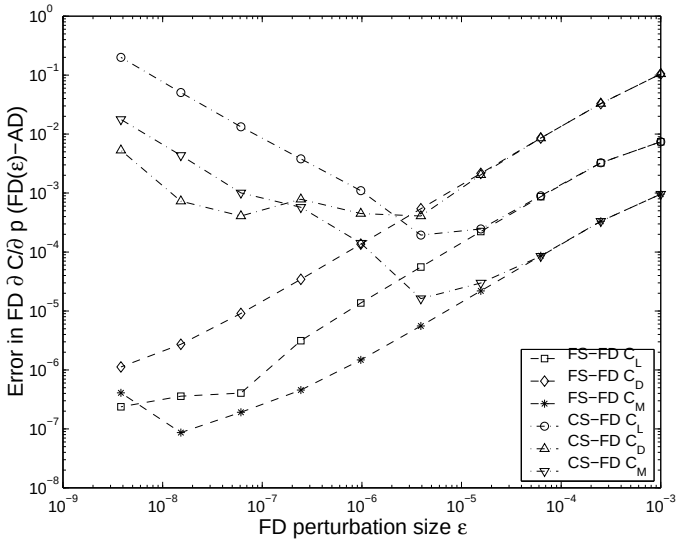


FIGURE 17.1. Error in Finite Difference (FS-FD and CS-FD) approximations to directional derivatives of the objective functions $C_L$, $C_D$ and $C_M$ as compared with AD (PB-AD).

the error (difference from PB-AD) for the two finite-difference approximations to this single directional derivative vary with the finite-difference perturbation size. Here the design objectives are: lift coefficient $C_L$, drag coefficient $C_D$, and pitching moment $C_M$. The convergence tolerances used were $\text{tol}_c = \text{tol}_{\partial \mathbf{c}/\partial \mathbf{p}} = 1 \times 10^{-10}$. The FS-FD results converge linearly to the PB-AD results as the FD step size is reduced until, at very small perturbation size, the error arising from the finite convergence criteria degrades the expected accuracy. This result validates the intrinsic accuracy of automatic differentiation by showing that the numerically approximate finite-difference gradient approaches the AD gradient as the FD step size is reduced. Similar convergence results may be seen in [3] elsewhere in this volume. The CS-FD results initially follow the FS-FD results but are then seen to diverge from the expected behaviour. This, and the slight differ-

| Method | Directional Derivatives of $C_L$ | $C_D$ | $C_M$ |
|---|---|---|---|
| PB-AD | 0.0916389175107 | -0.204563844953 | 0.0105117778674 |
| 2P-AD | 0.0916389172678 | -0.204563844468 | 0.0105117778962 |

TABLE 17.1. Directional derivatives of $C_L$, $C_D$ and $C_M$ computed by Piggy-Back AD and Two-Phase AD

| Method | CPU Time (s) | | | |
|---|---|---|---|---|
| | AD | FD perturbation size | | |
| | | $1 \times 10^{-3}$ | $1 \times 10^{-6}$ | $3.8 \times 10^{-9}$ |
| PB-AD | 1641.5 | | | |
| 2P-AD | 1051.2 | | | |
| FS-FD | | 420.5 | 422.7 | 422.6 |
| CS-FD | | 398.5 | 269.2 | 262.5 |

TABLE 17.2. CPU times corresponding to the calculations of Figure 17.1. 2P-AD and both FD times include 213.2s to obtain a converged flow-field.

ence between the 2P-AD and PB-AD derivatives (as seen in Table 17.1) is thought to be due to the lack of convergence criteria on the flow field variables (and their derivatives). In Table 17.2 we give CPU timings obtained on a COMPAQ Alpha 255-300 for the calculations of Figure 17.1 and Table 17.1. The second column gives the CPU times for the AD results, and columns 3 to 5 give the times for the FD results at 3 different perturbation sizes. For a single directional derivative FD appears more efficient than forward AD, though of inherently limited accuracy.

It is possible to obtain AD gradients with derivative tolerance $\text{tol}_{\partial \mathbf{c}/\partial \mathbf{p}} = 1 \times 10^{-5}$ in CPU times of $368s$ and $262.4s$ for PB-AD and 2P-AD respectively. These results are of a similar level of accuracy to the FD results shown in figure 17.1 and for a comparable CPU cost (c.f. Table 17.2).

## 17.3 AD within an Optimisation Design Method

We used the Euler solver `mheuler` of §17.2 in an aerodynamic optimisation test case making use of DERA's CODAS package [4]. We compared AD results with those obtained using FD. All the constrained optimisations performed with CODAS used a sequential quadratic algorithm [9].

### 17.3.1 Aerofoil drag minimisation

In this problem angle of incidence $\alpha$ and 16 cubic spline coefficients were used as design variables. The initial aerofoil geometry was that of an RAE2822 aerofoil. For fixed freestream Mach number of 0.8 and with lift coefficient $C_L$ constrained at 0.9, a minimal drag coefficient $C_D$ solution was sought. Figure 17.2 shows the final pressure coefficient distribution $C_P$

(top left), aerofoil shape (bottom left) with horizontal coordinates $X$ and airfoil displacement $Z$ scaled with the airfoil chord length (here denoted $C$). The right hand side of Figure 17.2 shows optimisation convergence histories (here `ALPHA` denotes $\alpha$ the airfoil angle of incidence) plotted against number of calls to the CFD flow solver. In this calculation all gradients were obtained using the Freestream Start FD approach. We see that the

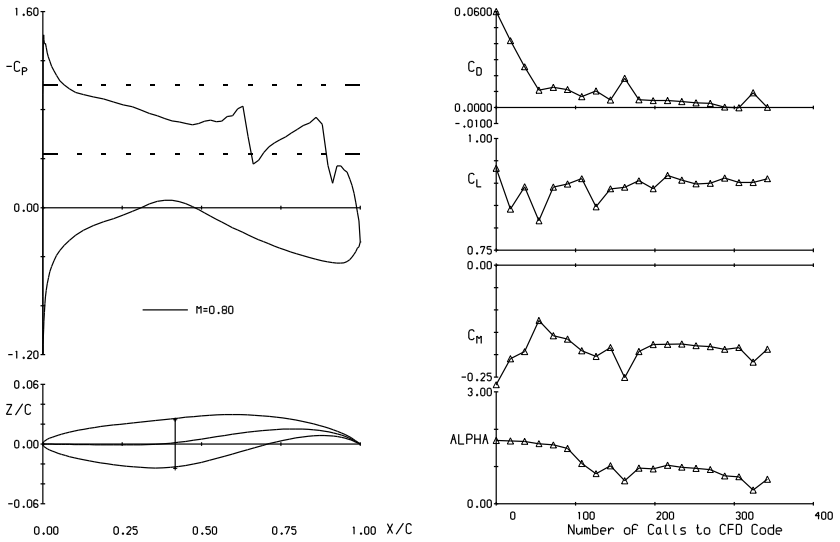


FIGURE 17.2. Finite difference drag optimisation of an RAE2822 aerofoil

drag $C_D$ has been minimised and, after some initial transient behaviour, the lift $C_L$ is close to its constrained value.

In Figure 17.3 we perform the same task using Two-Phase AD to obtain the gradients. Both techniques have converged to essentially the same solution.

The number of flow code iterations used here for FD was that currently employed by DERA in 2D optimisation studies. The number of iterations used for 2P-AD was chosen to give run times similar to those for FD. It was observed that gradient accuracy, compared to an earlier well-converged AD run, was similar for the illustrated FD and AD runs; and the expected resulting similarity in optimiser progress is clear from figures 17.2 and 17.3.

It seems then, that AD and FD have similar success in tasks such as this test case, when run times acceptable to industry are used. It was pleasing, though, that attention to both flow solver and grid generator convergence produced AD gradients with greater accuracy than that possible using FD, giving smoother optimiser performance late in the design process. It may be then that for more challenging search spaces, only AD will be able to produce a good design.

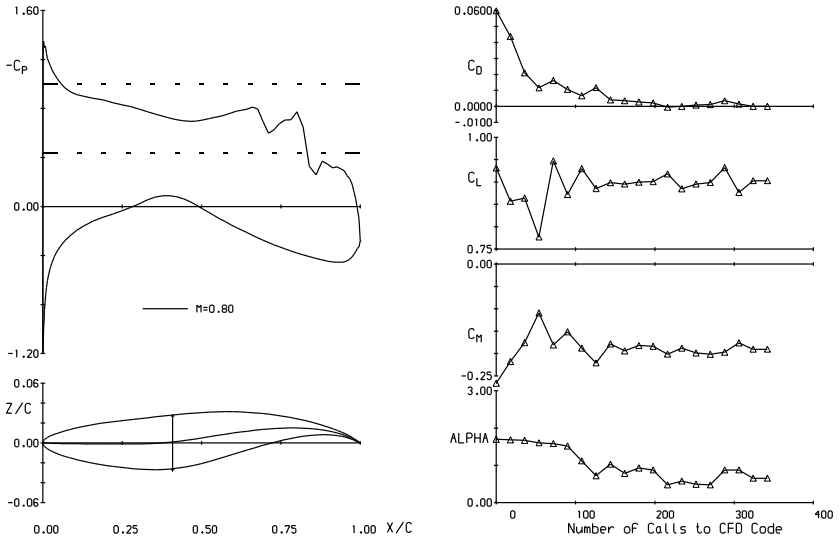


FIGURE 17.3. AD drag optimisation of RAE2822 aerofoil

## 17.4 Conclusions

We have confirmed that the forward mode of AD obtained using ADIFOR applied to our cell-vertex Euler flow solver obtains gradients consistent with those from finite differencing, but with greater accuracy. For the forward mode this greater accuracy comes at the expense of greater run times. However we have found that by adjusting the convergence criteria in the iterative solver for the AD gradients we obtain similar accuracy to finite differencing in similar run times. Although we only require objective values such as lift and drag coefficients within our optimisation techniques, we believe that careful control of the convergence of the flow-field itself is important for accuracy of both FD and AD gradients.

We have successfully used ADIFOR within an optimisation of an aerofoil shape to minimise drag for constant lift. For completeness we note that problems were encountered in the inverse design problem of matching a given pressure distribution and full details will be published elsewhere.

In our optimisation problem we have a large number of design variables (which control the shape of our airfoil) and only a small number of outputs (lift, drag, pitching moment). Consequently gradients should be calculated using the reverse or adjoint mode of AD since then run times will be proportional to the small number of outputs rather than to the large number of inputs (as in forward mode). Even for the 2-D optimisation problems we have been considering with 17 design variables and 3 outputs we would expect a 3 to 5 fold reduction in run times, allowing us to calculate more accurate gradients (if we need them) using AD in equivalent run times to

FD. For the 3-D wing design problems we are ultimately interested in we will be using in excess of 50 design variables and then reverse mode AD could be in excess of 10 times faster than FD. We shall be attacking such problems with the reverse mode of AD using INRIA's AD tool Odyssée [5] in the EU project AEROSHAPE.

*Acknowledgments:* The authors would like to thank the UK's Department of Trade and Industry for their funding of this work under the CARAD project.

Additionally the authors thank Mr. A. Gould of BAe Systems (SRC) for both technical support and his liaison with all parties, Dr. P. Betten of ANL for allowing Cranfield University use of ADIFOR and technical contributions from Dr. J. Pryce of Cranfield University (RMCS Shrivenham) and Dr. M. Bartholomew-Biggs, Prof. B. Christianson and Prof. A. Holdo all of the University of Hertfordshire.

[1] Michael C. Bartholomew-Biggs. Using forward accumulation for automatic differentiation of implicitly-defined functions. *Computational Optimization and Applications*, 9:65–84, 1998.

[2] Christian H. Bischof, Alan Carle, Paul D. Hovland, Peyvand Khademi, and Andrew Mauer. ADIFOR 2.0 user's guide (Revision D). Technical report, Mathematics and Computer Science Division Technical Memorandum no. 192 and Center for Research on Parallel Computation Technical Report CRPC-95516-S, 1998. See `www.mcs.anl.gov/adifor`.

[3] Bernard Cappelaere, David Elizondo, and Christèle Faure. Odyssée versus hand differentiation of a terrain modelling application. In George Corliss, Christèle Faure, Andreas Griewank, Laurent Hascoët, and Uwe Naumann, editors, *Automatic Differentiation: From Simulation to Optimization*, Computer and Information Science, chapter 7, pages 71–78. Springer, New York, 2001.

[4] John J. Doherty. Strategy for design optimisation. Customer Report DERA/MSS5/CR980389/1.0, Defence Evaluation & Research Agency, September 1998.

[5] Christèle Faure and Yves Papegay. Odyssée User's Guide. Version 1.7. Rapport technique RT–0224, INRIA, Sophia-Antipolis, France, Sept. 1998. See `www.inria.fr/RRRT/RT-0224.html`, and `www.inria.fr/safir/SAM/Odyssee/odyssee.html`.

[6] Andreas Griewank. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation.* Number 19 in Frontiers in Appl. Math. SIAM, Philadelphia, Penn., 2000.

[7] Michael G. Hall. Cell-vertex multigrid solution of the Euler equations for transonic flow past aerofoils. Technical Report 84116, Royal Aircraft Establishment, December 1984.

[8] Vamshi Mohan Koriva, Arthur C. Taylor III, Perry A. Newman, Gene W. Hou, and Henry E. Jones. An approximately factored incremental strategy for calculating consistent discrete aerodynamic sensitivity derivatives. *Journal of Computational Physics*, 113:336–346, 1994.

[9] J. J. Skrobanski. *Optimization Subject to Nonlinear Constraints.* PhD thesis, London University, 1986.

# Index