Computation of Sensitivity Information for Aircraft Design by Automatic Differentiation*

H. Martin Bücker, Bruno Lang, Arno Rasch, and Christian H. Bischof

Institute for Scientific Computing
Aachen University of Technology, D–52056 Aachen, Germany
{buecker, lang, rasch, bischof}@sc.rwth-aachen.de
http://www.sc.rwth-aachen.de

Abstract. Given a numerical simulation of the near wake of an airfoil, automatic differentiation is used to accurately compute the sensitivities of the Mach number with respect to the angle of attack. Such sensitivity information is crucial when integrating a pure simulation code into an optimization framework involving a gradient-based optimization technique. In this note, the ADIFOR system implementing the technology of automatic differentiation for functions written in Fortran 77 is used to mechanically transform a given flow solver called TFS into a new program capable of computing the original simulation and the desired derivatives in a simultaneous fashion. Numerical experiments of derivatives obtained from automatic differentiation and finite differences approximations are reported.

1 Introduction

The increasing aircraft traffic has led to a growing interest in maximizing take-off and landing frequencies. A detailed knowledge of the wake flow field is essential to estimate safe-separation distances between aircraft in take-off and landing. However, the complex flow field around an aircraft is still not completely understood and simulations are commonly used to advance the understanding of the underlying physical phenomena. At Aachen University of Technology, a team of engineers, mathematicians, and computer scientists is investigating the fluid-structure interaction at airplane wings to further advance scientific knowledge of the aerodynamics of cruise and high lift configurations. One of the projects aims at optimizing an airfoil with respect to certain design parameters. Traditionally, finding a suitable set of parameters is carried out by running the simulation code over and over again with perturbed inputs. However, this approach may consume enormous computing time and may also require an experienced user to select suitable sets of parameters just to achieve improvement, even without optimality. Here, numerical optimization techniques can help reduce the number

* This research is partially supported by the Deutsche Forschungsgemeinschaft (DFG) within SFB 401 "Modulation of flow and fluid–structure interaction at airplane wings," Aachen University of Technology, Germany.

P.M.A. Sloot et al. (Eds.): ICCS 2002, LNCS 2330, pp. 1069–1076, 2002.

of simulation runs, but in particular provide more goal-oriented computational support for a design engineer.

When embedding a pure simulation code in an optimization framework, a crucial ingredient to any gradient-based optimization algorithm is the derivative of the output of the simulation with respect to the set of design parameters. When the simulation is given in the form a high-level programming language such as Fortran, C, or C++, the derivatives can be computed by a technique called algorithmic or automatic differentiation (AD) [1]. Here, a given computer program is automatically transformed into another program capable of evaluating not only the original simulation but also derivatives of selected outputs with respect to selected inputs. In contrast to numerical differentiation, derivatives computed by AD are free of truncation error.

The aim of this note is to demonstrate the feasibility of accurate derivatives of a large-scale simulation in order to apply gradient-based optimization techniques. More precisely, AD is applied to a computational fluid dynamics code called TFS [2, 3] developed at the Aerodynamics Institute, Aachen University of Technology. This simulation code consists of 236 subroutines totaling approximately 24,000 lines of Fortran 77. Other references where automatic differentiation is applied to problems from computational fluid dynamics include [4–6]. The procedure for applying the ADIFOR [7] system implementing the AD technology is outlined in Sect. 3. The problems of a numerical differentiation approach based on finite differences are reported in Sect. 4. The discussion starts with a short review of the functionality of automatic differentiation in Sect. 2.

2 Automatic Differentiation

The term "Automatic Differentiation (AD)" comprises a set of techniques for automatically augmenting a given computer program with statements for the computation of derivatives. That is, given a computer program that implements a function

$$f(\mathbf{x}) = \big(f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})\big)^T \in \mathbb{R}^m \ ,$$

automatic differentiation generates another program that, at any point of interest $\mathbf{x} \in \mathbb{R}^n$, not only evaluates f at a point $\mathbf{x}$ but additionally evaluates its Jacobian

$$J(\mathbf{x}) := \begin{pmatrix} \frac{\partial}{\partial x_1} f_1(\mathbf{x}) & \ldots & \frac{\partial}{\partial x_n} f_1(\mathbf{x}) \\ \vdots & \ddots & \vdots \\ \frac{\partial}{\partial x_1} f_m(\mathbf{x}) & \ldots & \frac{\partial}{\partial x_n} f_m(\mathbf{x}) \end{pmatrix} \in \mathbb{R}^{m \times n}$$

at the same point.

AD technology is applicable whenever derivatives of functions given in the form of a high-level programming language, such as Fortran, C, or C++, are required. The reader is referred to the recent book by Griewank [1] and the proceedings of AD workshops [8–10] for details on this technique. The key idea of automatic differentiation is that any program can be viewed as a—potentially very long—sequence of elementary operations such as addition or multiplication,

for which the derivatives are known. Then the chain rule of differential calculus is applied over and over again, combining these step-wise derivatives to yield the derivatives of the whole program. This mechanical process can be automated, and several AD tools are available for transforming a given code into the new code that is called *differentiated code*. In this way, AD requires little human effort and produces derivatives that are accurate up to machine precision.

3 Applying the ADIFOR System to the TFS Code

At the Aerodynamics Institute, Aachen University of Technology, engineers are developing the TFS [2, 3] package for large-scale computational fluid dynamics. The simulation code consists of 236 subroutines totaling approximately 24,000 lines of Fortran 77. The TFS package is capable of simulating incompressible and compressible flows in two and three space dimensions on the basis of finite volume discretization on block-structured grids. For the present study, a version of TFS is taken to compute the two-dimensional flow field around a typical benchmark airfoil known as RAE 2822. A part of the grid of the underlying numerical simulation is depicted in Fig. 1 for the area close to the airfoil.

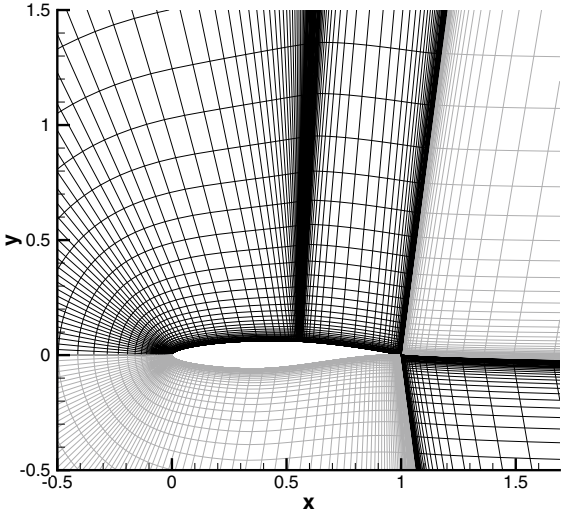

Fig. 1. Detail of a grid for airfoil RAE 2822 employing 4 blocks and 22356 nodes

In this note, we start from a given simulation of the Mach number, M, and the pressure, p, for a given angle of attack, α. That is, there is an implementation using TFS that, for any input α, computes the outputs M and p. From a

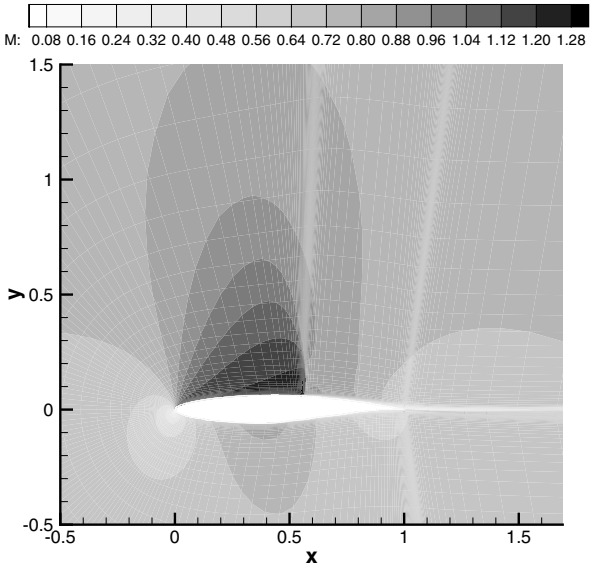

Fig. 2. The Mach number M of the flow field computed by TFS for $\alpha = 2.79°$

mathematical point of view, this TFS simulation evaluates some function

$$\begin{bmatrix} M(\alpha) \\ p(\alpha) \end{bmatrix} = f(\alpha) \ . \tag{1}$$

As an example of a TFS simulation, a plot of the Mach number M is depicted in Fig. 2 evaluated at $\alpha = 2.79°$.

Suppose that we are interested in the derivatives of the Mach number with respect to the angle of attack, i.e., $\partial M/\partial\alpha$. Since TFS is a computer program written in Fortran 77, these derivatives can be computed in a completely mechanical way by using any AD tool for Fortran 77. Notice that a list of available AD tools is currently being compiled at `http://www.autodiff.org`. In this note, the AD tool ADIFOR is applied to transform TFS into a differentiated version of TFS capable of computing $\partial M/\partial\alpha$ in addition to the original simulation of M.

As a preprocessing step, a few non-standard programming techniques are eliminated by hand in order to make TFS conforming to the Fortran 77 language standard. As a second step, a top-level routine in TFS implementing the function represented by (1) is identified and the program variables corresponding to $M$and α are indicated to the ADIFOR system. In our specific example, approximately 220 subroutines are fed into the ADIFOR system. ADIFOR recognizes about 100 subroutines as contributing to the derivatives and thus requiring additional code for the derivative computations. The code generated automatically by ADIFOR consists of approximately 20,000 lines of Fortran 77. Finally, a "driver" routine is implemented that initializes the derivative computations and

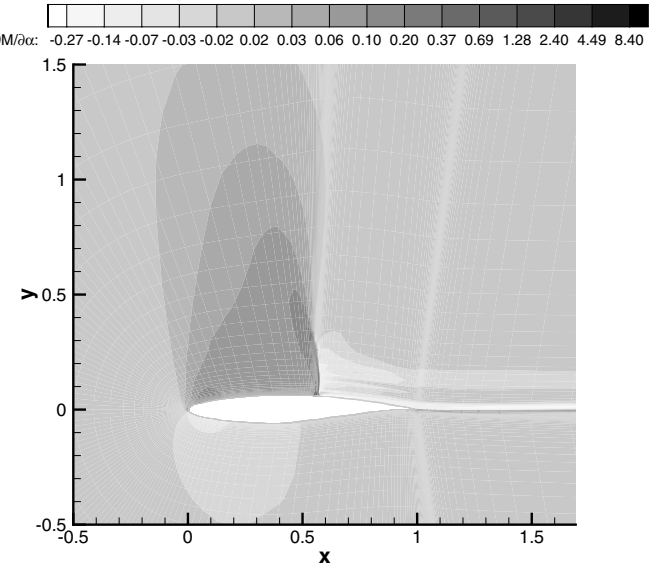

Fig. 3. AD-generated derivatives $\partial M/\partial\alpha$ evaluated at $\alpha = 2.79°$

invokes the differentiated top-level routine. This driver routine is then capable of computing the desired derivatives together with the original simulation. The AD-generated derivatives of the Mach number are depicted in Fig. 3. Here, the dark areas show the largest positive change of $M$with respect to changes in α. Note that the largest changes occur at the vertical shock and in the wake.

4 Problems with Divided Differences

Automatic differentiation is based on successively applying the chain rule to elementary operations leading to derivative values accurate up to machine precision. In contrast, a traditional alternative for the computation of derivatives is the numerical approximation by divided differences. For the sake of simplicity, we consider a first-order finite difference scheme where the derivative $\partial M/\partial\alpha$ of the Mach number $M(\alpha)$ is approximated by

$$\delta(h) := \frac{M(\alpha + h) - M(\alpha)}{h}$$

involving a step size h. Besides the truncation error, the crucial disadvantage of divided differences (DD) is the need to find a suitable step size.

For the TFS simulation, it turns out that an appropriate step size for the derivatives $\partial M/\partial\alpha$ is extremely hard to determine. When varying the step size h from 10^{-2} to 10^{-8}, the corresponding DD approximations $\delta(h)$ differ so significantly that a quantitative prediction for $\partial M/\partial\alpha$ using DD is not feasible. In

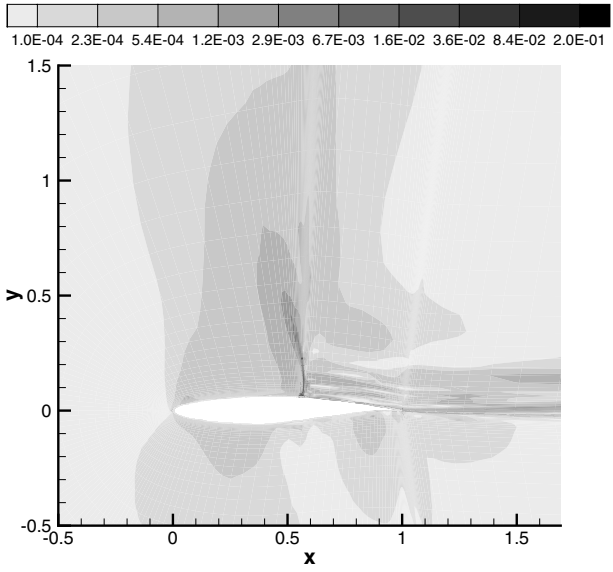

Fig. 4. The difference $|\delta(10^{-5}) - \delta(10^{-6})|/8.446$ at $\alpha = 2.79°$

Fig. 4, the difference of two DD approximations resulting from two different step sizes is given. More precisely, a normalized absolute difference of the DD approximations $\delta(10^{-5})$ and $\delta(10^{-6})$ is shown. The normalization is obtained from dividing the absolute difference by the largest absolute derivative value, 8.446, generated by AD. Recall from Fig. 3 that this scaling factor occurs in the region of the vertical shock.

The difference depicted in Fig. 4 is less than 10^{-4} in the outer region, meaning that, there, the two DD approximations $\delta(10^{-5})$ and $\delta(10^{-6})$ agree rather well. However, in the areas next to the vertical shock and the wake flow field the difference is large reaching a value around 0.2. Thus, the DD approximations of the derivatives are not accurate in these interesting areas.

5 Concluding Remarks

To use a large-scale flow field simulation within a Newton-type optimization framework, derivatives of the flow field with respect to simulation parameters are necessary. Automatic differentiation is a technique for transforming a given simulation code, written in a high-level programming language such as Fortran or C, into another computer program capable of computing not only the given simulation but also some user-specified derivatives. The technique is not only applicable to fairly small programs but scales to large computer codes such as computational fluid dynamics solvers. As opposed to numerical differentiation,

automatic differentiation does not involve any truncation error. Therefore, automatic differentiation is currently the only option whenever exact derivatives of functions given in the form of complex computer programs are required.

In this note, the automatic differentiation tool ADIFOR is applied to the TFS package. This simulation package consists of approximately 24,000 lines of Fortran 77. The flow field around the RAE 2822 airfoil is computed using TFS. The derivative of the Mach number with respect to the angle of attack is accurately computed using automatic differentiation whereas an approximation of the same derivatives using a divided difference approach results in reliable derivative values only at regions far from the airfoil. However, in the areas of interest, for instance in the vicinity of the shock, the derivative values produced by divided differences vary significantly with the actual step size being used and, in any case, do not deliver reliable derivative values. At best, divided differences give a qualitative prediction of the derivative field, but they do not permit a detailed understanding of the underlying phenomena.

Work is in progress to further increase the computational efficiency of the code generated by automatic differentiation. Moreover, the integration of the differentiated TFS code into several gradient-based optimization algorithms is currently under investigation.

Acknowledgments

The authors would like to thank Jakob Risch for his notable contribution during the initial phase of this project and Emil Slusanschi for completing the automatic differentiation framework. We would also like to thank the Aerodynamics Institute, Aachen University of Technology, for making available the source code of the flow solver TFS. In particular, Matthias Meinke and Ehab Fares deserve special recognition for their help with the RAE 2822 airfoil. This research is partially supported by the Deutsche Forschungsgemeinschaft (DFG) within SFB 401 "Modulation of flow and fluid–structure interaction at airplane wings," Aachen University of Technology, Germany.

References

1. Griewank, A.: Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation. SIAM, Philadelphia (2000)
2. Fares, E., Meinke, M., Schröder, W.: Numerical simulation of the interaction of flap side-edge vortices and engine jets. In: Proceedings of the 22nd International Congress of Aeronautical Sciences, Harrogate, UK, August 27–September 1, 2000. ICAS 0212 (2000)
3. Fares, E., Meinke, M., Schröder, W.: Numerical simulation of the interaction of wingtip vortices and engine jets in the near field. In: Proceedings of the 38th Aerospace Sciences Meeting and Exhibit, Reno, NV, USA, January 10–13, 2000. AIAA Paper 2000–2222 (2000)

4. Bischof, C., Corliss, G., Green, L., Griewank, A., Haigler, K., Newman, P.: Automatic differentiation of advanced CFD codes for multidisciplinary design. Journal on Computing Systems in Engineering **3** (1992) 625–638
5. Bischof, C., Green, L., Haigler, K., Knauff, T.: Parallel calculation of sensitivity derivatives for aircraft design using automatic differentiation. In: Proceedings of the 5th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization, AIAA 94-4261, American Institute of Aeronautics and Astronautics (1994) 73–84
6. Aubert, P., Di Césaré, N., Pironneau, O.: Automatic differentiation in C++ using expression templates and application to a flow control problem. Computing and Visualization in Science **3** (2001) 197–208
7. Bischof, C., Carle, A., Khademi, P., Mauer, A.: ADIFOR 2.0: Automatic differentiation of Fortran 77 programs. IEEE Computational Science & Engineering **3** (1996) 18–32
8. Griewank, A., Corliss, G.: Automatic Differentiation of Algorithms. SIAM, Philadelphia (1991)
9. Berz, M., Bischof, C., Corliss, G., Griewank, A.: Computational Differentiation: Techniques, Applications, and Tools. SIAM, Philadelphia (1996)
10. Corliss, G., Faure, C., Griewank, A., Hascoët, L., Naumann, U., eds.: Automatic Differentiation of Algorithms: From Simulation to Optimization. Springer (2002) (to appear).