Delayed Propagation of Derivatives in a Two-dimensional Aircraft Design Optimization Problem

H. Martin Bücker, Arno Rasch, Emil Slusanschi, Christian H. Bischof [a]

[a]Institute for Scientific Computing, Aachen University, 52056 Aachen, Germany
{buecker, rasch, slusanschi, bischof}@sc.rwth-aachen.de

In computational fluid dynamics (CFD), the use of high-performance computers is often indispensable, simply to manage the high request for computation time and storage. We consider a particular two-dimensional aircraft design problem and report on the use of automatic differentiation (AD) to obtain accurate derivatives rather than approximations based on numerical differentiation. More precisely, we apply the AD system ADIFOR to the large-scale CFD solver TFS and quantify the computational savings of a technique starting the derivative computations in a delayed manner over a black box AD approach.

En dynamique des fluides (CFD), l'utilisation d'ordinateurs performants est souvent indispensable, en raison des besoins énormes en temps de calcul et en stockage de données. Nous considérons un problème de conception d'avion en deux dimension. Nous utilisons une technique de différentiation automatique (AD) pour obtenir des dérivées précises, plutt que des approximations basées sur des dérivées numériques. Plus précisément, nous appliquons le système de AD appelée ADIFOR au programme de CFD TFS et quantifions les gains en temps de calcul d'une technique initiant les calculs de dérivées d'un manière progressive, en comparaison d'une approche AD de type "boîte noire".

1 Introduction

High-performance numerical simulations are among the indispensable tools used in the design and analysis of flight vehicles. We consider a particular computational fluid dynamics (CFD) simulation with the long-term aim to create an airfoil with the highest possible lift-over-drag ratio. A crucial preprocessing step of this design optimization problem consists in efficiently evaluating the derivatives of the lift-over-drag ratio with respect to some given geometric parameters characterizing the geometry of the underlying airfoil. Here, efficiency of the derivative computation is of major importance because the evaluation of the objective function includes the solution of the Navier-Stokes equations, representing a hard computational problem on its own. Therefore, one has to be careful when adding further computational work by the evaluation of derivatives.

In a previous work [1], we used a technique called automatic differentiation (AD), whose basics are summarized in Sect. 2, to obtain derivatives for a two-dimensional airfoil design optimization problem and demonstrated the advantages of the AD approach over a numerical approach in terms of accuracy and reliability. In this note, we extend our previous work [1] sketched in Sect. 3 by delaying the derivative computations resulting in a significant increase of the efficiency of the black box AD approach. The new approach is detailed in Sect. 4.

2 Automatic differentiation

When a function is given in the form of a computer code, automatic differentiation (AD) may be used to generate another computer program capable of evaluating the function *and* its derivatives. The basic idea of AD is to treat a computer program as the composition of a large sequence of elementary functions and intrinsics whose derivatives are known. Then, these derivatives are accumulated using the chain rule. From a computer science point of view, AD represents a program transformation not preserving the semantics of a given program. There are AD tools mechanically transforming computer codes by augmenting a given computer program with statements for the computation of derivatives.

When using an AD tool a user has to specify the output variables whose derivatives are sought, called *dependent* variables, and the input variables with respect to which one is differentiating, called *independent* variables. More details on AD are given in [2–6]. The AD technology is not only applicable to small computer codes but also scales to large programs of several hundred thousand lines of code.

Automatic differentiation is not to be confused with symbolic differentiation implemented in computer algebra packages like Maple. In contrast to symbolic differentiation, AD is applicable to large computer codes with loops, conditional branches, and subroutine calls. AD generates a program for evaluating derivatives rather than a mathematical formula. Yet the results obtained by an AD-generated code are accurate up to machine precision due to the absence of any numerical approximation scheme such as divided differencing.

3 Differentiating the TFS Program

In the present work, we mainly use the automatic differentiation system ADIFOR [7] to obtain derivatives in an aerodynamic application. The underlying computer simulations is carried out with the CFD solver TFS, developed at the Aerodynamics Institute, Aachen University. The TFS program [8,9] is capable of solving the Navier–Stokes equations of a compressible ideal gas in two or three space dimensions with a finite volume formulation. The spatial discretization of the convective terms follows a variant of the advective upstream splitting method (AUSM+) proposed by Liou [10,11]. The viscous stresses are centrally discretized to second-order accuracy. The implicit method employed in this work uses a relaxation-type linear solver whereas the explicit method relies on a multigrid scheme. The TFS code consists of about $25,000$ lines of Fortran 77 source code. Applying ADIFOR to TFS results in an AD-augmented program referred to as TFS.AD hereafter. The size of the automatically generated AD-code, TFS.AD, depends on the choice of the derivatives being computed. Typically, TFS.AD together with the original code consists of about $45,000$ lines of Fortran code.

After the successful application of ADIFOR to TFS, running TFS.AD results in accurate derivatives of the flow field with respect to various flow parameters, like the angle of attack, the Mach number, or the pressure; see the previous work [12,13] for details. In a more recent study [1], we are interested in the derivatives with respect to some geometric shape parameters. In this particular two-dimensional aircraft design optimization problem, the function to be differentiated consists of a sequence of three programs: The first program written in MATLAB generates an airfoil from given eight scalar geometric shape parameters $\xi_1, \xi_2, \ldots, \xi_8$; a Fortran 77 program is used in a second step to generate a grid around the airfoil; finally, the solver TFS is used to compute the flow field around the airfoil.

In [1], we use ADIFOR as well as ADiMat [14], a recent AD tool for MATLAB programs, to get derivatives of the entire sequence of programs in a black box fashion. AD is shown to reliably compute accurate derivatives whereas numerical differentiation based on divided differencing turns out to be inadequate. The reason for the latter is that, in this particular problem, it is extremely hard, if not impossible, to determine a step size for divided differencing so that the derivative values are meaningful. Compared to the original simulation program, the storage of the AD-generated program increases by a factor of 8.3, while the execution time increases by a factor of 16.5. Recall that eight scalar derivatives are computed and, thus, the approach reported in [1] is feasible to get accurate derivatives, but expensive in terms of execution time. In the following section, we show how to improve the efficiency of the black box AD approach.

4 Delaying the derivative propagation

When optimizing the geometry of an airfoil, and implicitly solving the two-dimensional Navier-Stokes equations, the efficiency of the derivative computation is crucial for the overall efficiency of the solution of the design optimization problem. For small problems in two-dimensions, the black box AD approach described in the previous section is sufficient. However, when more complex and realistic problems in three dimensions are considered, the computation of accurate derivatives should be more efficient.

The black box AD approach in [1] is to compute, from the very beginning, the original function, the lift-over-drag ratio, together with the derivatives with respect to the geometric parameters $\xi_1, \xi_2, \ldots, \xi_8$. The efficiency can be improved by delaying the derivative computations. That is, the original code TFS is started to compute the lift-over-drag ratio and, after a suitable number of iteration steps, TFS is stopped to continue the computation with TFS.AD, where the parts of the code associated to the original simulation are started using the data from the last iteration of the TFS run. Thus, the iteration of the original function is computed as in the original simulation while the iteration of the derivatives is started not until the original function is already converged to a certain precision. The theory behind that approach is outlined in [2] and is also applied in [15–19].

We consider the following steady state flow problem. The grid consists of four blocks totaling $17,428$ points. The transonic flow conditions are described by a Mach number of 0.74 and a Reynolds number of $2.4 \cdot 10^6$. The angle of attack is set to 1.5593 degrees, because a lift coefficient $c_L = 0.5$ is desired for the initial geometry of the airfoil. On the airfoil, a no slip condition and an adiabatic wall is assumed, the flow variables on the far field boundary are computed with one-dimensional linearized characteristics. Throughout the following the iteration steps of the implicit method used to compute the flow field are called "(local) time steps". All the computations are performed in double precision. The stopping criterion of the original simulation for the lift-over-drag ratio is as follows. The changes in the values of the lift and drag coefficients, c_L and c_D, respectively, are checked every 100 time steps. The simulation is considered to be converged if both of these changes are smaller than 10^{-4}. Using this stopping criterion, the original simulation converges after 1700 time steps. Notice that the values of c_L and c_D continue to vary in the range of 10^{-5}, but the engineers consider the precision to be sufficient.

The convergence history of a subset of the derivative computations is depicted in Figure 1. The figure shows, on a logarithmic scale, the derivatives $\partial c_D / \partial \xi_1$. Recall that these quantities are among the crucial ingredients for the derivatives of c_L / c_D with respect to $\xi_1, \xi_2, \ldots, \xi_8$. The derivatives of the lift-over-drag ratio with respect to the geometric parameters are checked for convergence every 100 time steps and are considered to be converged if there is no change larger than 10^{-2} from one time step to the next. It is worth noting that all these derivatives computed with the approaches described in the following, were consistent with each other, meaning that they were the same within the given limit of 10^{-2}. The only difference is the time taken to compute these derivatives.

Let k denote the number of time steps the original TFS is run before switching to TFS.AD to start the derivative computations. Then, in Figure 1, the convergence history of the derivative of the drag coefficient with respect to the first geometry parameter, $\partial c_D / \partial \xi_1$, is given for three different

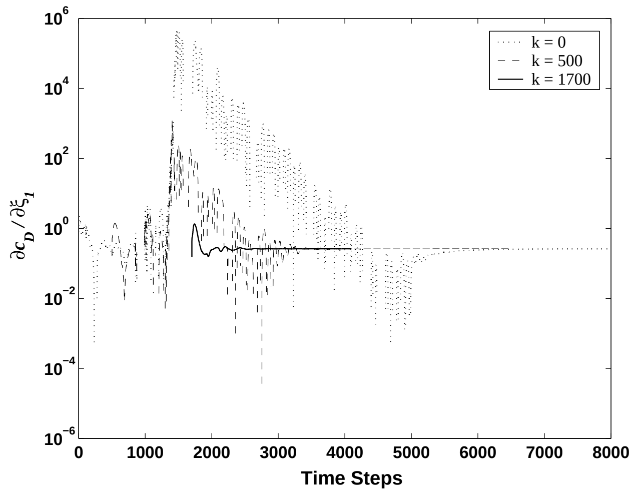

Figure 1. Convergence history when starting the derivative computations after $k = 0$, $k = 500$ and $k = 1700$ time steps.

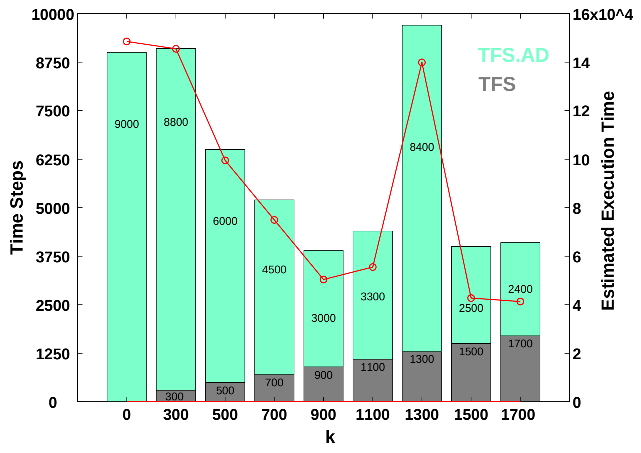

Figure 2. Performance of the approach delaying the derivative computations by k time steps.

values of k. The first curve, $k = 0$, describes the behavior of the derivative that is computed from the beginning together with the original function, i.e., a black-box-like AD approach. The second curve represents the delayed computation of the derivatives after the original function has already iterated for $k = 500$ time steps and has a precision of about 10^{-2}. Finally the third curve, depicts the delay of $k = 1700$ time steps, where the original function has converged to 10^{-4}. The figure demonstrates the strong influence of the parameter k on the convergence behavior of $\partial c_D/\partial \xi_1$. All curves converge to the same value but the convergence tends to be increased when increasing the delay of starting the derivative computations.

In order to better understand the degree of the improvement in efficiency that the delayed employment of the AD-enhanced code provides, a series of experiments is conducted in which the computation of derivatives is started after k time steps of the original function. Figure 2 represents the corresponding results. At the bottom of each column, the number of time steps representing the computation of the original function without any derivatives is presented as a TFS bar, labeled with the respective number of time steps. Above each TFS bar, there is a TFS.AD bar, representing the number of time steps that are performed by the AD-augmented code, in order to satisfy the convergence criterion. For example, the first experiment illustrated by the leftmost column represents the computation of the derivatives from the beginning, together with the original function, i.e., $k = 0$. The rightmost experiment is composed of $k = 1700$ time steps of TFS and then another 2400 time steps of TFS.AD.

As the figure shows, the number of time steps required for the computation of the desired derivatives decreases as the number of initial original TFS time steps increases, with the exception of the experiment where $k = 1300$. Recall that the execution time of a single time step of TFS.AD is approximately 16.5 times that of a single time step of TFS. Therefore, the optimal efficiency is obtained with a minimal number of TFS.AD time steps. An estimate of the execution time of the approach starting the derivative computation after k TFS time steps is given by $k+16.5\cdot\ell$, where ℓ is the number of TFS.AD time steps. This estimated execution time, given in the corresponding number of TFS time steps, is depicted in the right of Figure 2 showing that the best combination is obtained in the last experiment corresponding to $k = 1700$, with the smallest overall execution time of $41,300$.

The use of the delayed computation of derivatives starting from a “converged” original function, results in a significant increase in efficiency with respect to the execution time and in this way accelerates the optimization process that makes use of the derivatives. Here an important issue is the convergence behavior of the original function. If the function is too far away from convergence then a large increase of the number of time-consuming time steps of the AD-enhanced program may occur as in the experiment with $k = 1300$ as shown in Figure 2. Therfore, it is important to run the original simulation until it is properly converged.

5 Concluding remarks

Automatic differentiation turns out to be a crucial ingredient to efficiently solving a particular aircraft design optimization problem. For the problem at hand, automatic differentiation reliably computes accurate derivative values whereas numerical differentiation is inadequate because of the difficulties involved in finding an appropriate step size. Note that this technique is not only applicable to fairly small codes but scales to large programs like the computational fluid dynamics solver TFS consisting of about $25,000$ lines of Fortran 77.

For the computation of accurate derivatives, a black box application of automatic differentiation is shown to be feasible. However, the efficiency of a black box approach can be significantly improved by delaying the derivative computations. More precisely, the original simulation is run for a certain number of time steps before switching to the derivative computations. That is, the derivative-enhanced code is started with the values from the last iteration of the original code. Thus, the original function is computed as without any modifications while the time-consuming derivative computations are only started when the function is converged. In this way, the derivative computations are likely to converge faster than in a black box approach of automatic differentiation.

6 Acknowledgments

The authors would like to thank Jakob Risch for his notable contribution during the initial phase of this project and Andre Vehreschild for providing the ADiMat tool. We would also like to thank the Aerodynamics Institute, Aachen University, for making available the source code of the flow solver TFS. In particular, Matthias Meinke and Ehab Fares deserve special recognition. 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, Germany.

References

1. C. H. Bischof, H. M. Bücker, B. Lang, A. Rasch, and E. Slusanschi. Efficient and accurate derivatives for a software process chain in airfoil shape optimization. *Future Generation Computer Systems*, 2002. To appear.
2. A. Griewank. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. SIAM, Philadelphia, 2000.
3. L. B. Rall. *Automatic Differentiation: Techniques and Applications*, volume 120 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1981.
4. George Corliss, Christèle Faure, Andreas Griewank, Laurent Hascoët, and Uwe Naumann, editors. *Automatic Differentiation of Algorithms: From Simulation to Optimization*. Springer, New York, 2002.
5. A. Griewank and G. Corliss. *Automatic Differentiation of Algorithms*. SIAM, Philadelphia, 1991.
6. M. Berz, C. Bischof, G. Corliss, and A. Griewank. *Computational Differentiation: Techniques, Applications, and Tools*. SIAM, Philadelphia, 1996.
7. C. Bischof, A. Carle, P. Khademi, and A. Mauer. ADIFOR 2.0: Automatic differentiation of Fortran 77 programs. *IEEE Computational Science & Engineering*, 3(3):18–32, 1996.
8. E. Fares, M. Meinke, and W. Schröder. 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.
9. E. Fares, M. Meinke, and W. Schröder. 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.
10. M. S. Liou and Ch. J. Steffen. A new flux splitting scheme. *Journal of Computational Physics*, 107:23–39, 1993.
11. M. S. Liou. A sequel to AUSM: AUSM$^+$. *Journal of Computational Physics*, 129:164–182, 1996.
12. H. M. Bücker, B. Lang, A. Rasch, and C. H. Bischof. Sensitivity Analysis of an Airfoil Simulation Using Automatic Differentiation. In M. H. Hamza, editor, *Proceedings of the IASTED International Conference on Modelling, Identification, and Control, Innsbruck, Austria, February 18–21, 2002*, pages 73–76, Anaheim, CA, USA, 2002. ACTA Press.
13. H. M. Bücker, B. Lang, A. Rasch, and C. H. Bischof. Computation of Sensitivity Information for Aircraft Design by Automatic Differentiation. In P. M. A. Sloot, C. J. K. Tan, J. J. Dongarra, and A. G. Hoekstra, editors, *Computational Science – ICCS 2002, Proceedings of the International Conference on Computational Science, Amsterdam, The Netherlands, April 21–24, 2002. Part II*, volume 2330 of *Lecture Notes in Computer Science*, pages 1069–1076, Berlin, 2002. Springer.
14. C. H. Bischof, H. M. Bücker, B. Lang, A. Rasch, and A. Vehreschild. Combining source transformation and operator overloading techniques to compute derivatives for MATLAB programs. In *Proceedings of the Second IEEE International Workshop on Source Code Analysis and Manipulation (SCAM 2002)*, pages 65–72, Los Alamitos, CA, USA, 2002. IEEE Computer Society.
15. B. Mohammadi and O. Pironneau. *Applied Shape Optimization for Fluids*. Oxford University Press, Oxford, 2001.
16. A. Carle, L. L. Green, C. H. Bischof, and P. A. Newman. Applications of automatic differentiation in CFD. In *Proceedings of the 25th AIAA Fluid Dynamics Conference, Colorado Springs, CO, USA, June 20–23, 1994*, AIAA Paper 94–2197, 1994.
17. L. L. Green, P. A. Newman, and K. J. Haigler. Sensitivity derivatives for advanced CFD algorithm and viscous modeling parameters via automatic differentiation. *Journal of Computational Physics*, 125(2):313–324, 1996.
18. Shaun A. Forth and Trevor P. Evans. Aerofoil optimisation via AD of a multigrid cell-vertex Euler flow solver. In Corliss et al. [4], chapter 17, pages 153–160.
19. A. Carle, M. Fagan, and L. L. Green. Preliminary results from the application of automated adjoint code generation to CFL3D. In *Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, MO, USA, September 2–4, 1998*, AIAA Paper 98–4807, 1998.