# ALGORITHMIC DIFFERENTIATION OF DIFFERENT ALGORITHMS FOR THE SAME PROBLEM: A CASE STUDY*

Christian H. Bischof
H. Martin Bücker
Arno Rasch
Emil Slusanschi
Institute for Scientific Computing
RWTH Aachen University
D–52056 Aachen, Germany

**KEYWORDS**
Algorithmic/Automatic Differentiation, Sensitivity Analysis, Numerical Simulation, FLUENT, SEPRAN

## ABSTRACT

Large-scale numerical simulation is one of the basic blocks in computational science and engineering. Researchers in academia and industry rely on highly complex simulation codes that are able to simulate even the most complicated physical phenomena. Several approaches are possible in order to validate the results of a numerical simulation, such as examining the results obtained from different simulation packages. In addition one could analyze the impact of certain input parameters on the solution. Such sensitivities of the computed solution can be obtained by automatic differentiation, a technique for computing truncation error-free derivatives of functions given in the form of a computer program. For a standard flow problem, we examine the results obtained from the two simulation packages FLUENT and SEPRAN, and we compare the derivatives which are computed by automatic differentiation. We show that, although the two packages employ fundamentally different algorithms, the results obtained, i.e., the velocity fields and their respective sensitivities, are comparable.

## INTRODUCTION

How can you be sure that your computer simulation is correctly modeling your scientific or engineering problem? To what extent does the output of your computer simulation depend on the actual algorithm used to solve your problem? It is often difficult to obtain a consensus about the answers to such questions. A comparison of the same computational problem by different numerical simulations, probably implemented by different computer codes developed by different authors, is one way to give more insight into these issues. Such benchmarking of different codes, however, should be approached with care (Roache 1998). Another option to assess the behavior of a computer code is to consider the sensitivities of the output of a computer simulation with respect to its inputs. These sensitivities can be evaluated without truncation error by automatic differentiation. The idea behind this technique is to transform a given computer code into another code capable of evaluating the derivatives of selected output variables with respect to selected input variables. So, the algorithm used to implement a function, rather than the function itself, is differentiated. This is the reason why automatic differentiation (AD) is also called algorithmic differentiation. In the context of high-performance computing, where the underlying function is given by a large-scale simulation code, AD is often the only way to reliably evaluate derivatives.

Now, consider the scenario where AD is applied to different algorithms to solve the same computational problem. Here, it is not only interesting to study the differences in the original simulations but also to investigate how the AD-generated derivatives differ because they depend on the different algorithms used in the original simulations. Under certain circumstances, applying AD in a black box fashion can lead to surprising results because the automatic process is not only applied to the solution of the algorithm but also to the solution procedure itself (Eberhard and Bischof 1999; Bücker 2002).

To study the application of AD to different algorithms for the same problem, we consider a standard test problem in computational fluid dynamics (CFD), the flow over a backward-facing step, using the two multi-purpose simulation packages FLUENT (Fluent Inc. 1997) and SEPRAN (Segal 1993b). Both software packages are successfully used in academia and industry to simulate large-scale complicated scientific and engineering systems in various application areas. FLUENT and SEPRAN are first used to compute the flow field in this test problem. Then we apply the AD system ADIFOR (Bischof et al. 1996) to FLUENT and

*This research is partially supported by the Deutsche Forschungsgemeinschaft (DFG) within SFB 540 “Model-based experimental analysis of kinetic phenomena in fluid multi-phase reactive systems,” RWTH Aachen University, Germany.

*Proceedings 18th European Simulation Multiconference*
*Graham Horton (c) SCS Europe, 2004*
*ISBN 3-936150-35-4 (book) / ISBN 3-936150-36-2 (CD)*

SEPRAN to automatically generate two AD codes computing the derivatives of the velocity field with respect to the maximum inflow velocity, one of the input variables of the test problem.

This note is structured as follows. In the following section, some details on applying AD to FLUENT and SEPRAN are given. Thereafter, the specification of the standard test problem is described. Finally, the derivatives computed by the two AD codes are compared.

## AUTOMATIC DIFFERENTIATION OF FLUENT AND SEPRAN

In the present study we use the simulation packages SEPRAN and FLUENT. SEPRAN is a general-purpose finite element package developed at "Ingenieursbureau SEPRA" and Delft University of Technology. The package is employed in various scientific areas ranging from fluid dynamics to structural mechanics to electromagnetism (Bosch and Lasance 2000; Segal et al. 1998; van Keken et al. 1995) and consists of approximately 800,000 lines of mostly Fortran 77, including comments. In this project we use version 12/2002 of the SEPRAN simulation package.

FLUENT, developed by Fluent Inc., is one of the leading simulation packages for computational fluid dynamics (CFD). It uses a pressure-based finite volume method for incompressible and mildly compressible flows. In the present case study we use version 4.5.2 of the FLUENT code which is basically written in Fortran 77 with some additional Fortran 90 language elements that are mainly used for the dynamic memory management. The whole source code consists of more than 2 million lines Fortran, including comments. Furthermore there are approximately 50,000 lines of C code (including comments) primarily concerned with the graphical user interface and system calls. However, most of the C code is not mandatory for the computational part of FLUENT.

Automatic differentiation (AD) is a set of techniques for transforming a given computer program implementing some mathematical function into another program evaluating the original function and its derivatives for a given input. The AD technology is applicable whenever derivatives of functions given in the form of a high-level programming language, such as Fortran, C, C++, or MATLAB, are required. The key idea of AD is that every computer program, no matter how complicated, is 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 repeatedly, 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 to the new *differentiated* code. Note, that in contrast to numerical differentiation, derivatives obtained by AD are free from truncation error. The reader is referred to (Griewank 2000; Rall 1981) for a detailed introduction to this field.

Although automatic differentiation is in principle applicable to computer codes of arbitrary size, practical experiences have shown that automatically differentiating large simulation packages consisting of several hundreds of thousands of lines of code is still a challenging task. Non-standard programming techniques used in legacy codes often require manual modifications before a software tool implementing AD could perform the actual transformation. A detailed description of the manual "code massaging" process is beyond the scope of this paper. The reader is referred to (Bischof et al. 2003; Bischof et al. 2001) for a detailed description of applying AD to the SEPRAN package. We used the AD tool ADIFOR (Bischof et al. 1996), developed at Argonne National Laboratory and Rice University, for generating differentiated versions of FLUENT and SEPRAN.

## SIMULATING THE FLOW OVER A BACKWARD-FACING STEP

For comparing the differentiated versions of SEPRAN and FLUENT, we consider the simulation of an incompressible flow over a two-dimensional backward-facing step which is a common benchmark problem for CFD codes. A sample implementation of this problem is part of the SEPRAN distribution (Segal 1993a). The FLUENT tutorial guide (Fluent Inc. 1995) also contains a section on the backward-facing step problem.

In the following, we consider a two-dimensional rectangular domain of $35\,\mathrm{m} \times 5\,\mathrm{m}$ and a step height of $H = 1\,\mathrm{m}$ as depicted in Fig. 1. At the inlet, $x = 0$, a parabolic velocity profile with a maximum velocity $u_{\text{max}} = 1.0\,\mathrm{m/s}$ is defined. The scalar input parameter $u_{\text{max}}$ specifies the complete parabolic velocity profile in horizontal direction. The vertical velocity component at the inlet is zero. Furthermore, we assume a fluid density of $\rho = 1.0\,\mathrm{kg/m^3}$ and a dynamic viscosity of $\eta = 1.0 \times 10^{-2}\,\mathrm{kg/(m\,s)}$. Based on the step height, the Reynolds number is 100. In addition to the flow field $(u, v)$, where $u$ and $v$ denote the horizontal and vertical velocity components respectively, we are interested in the derivatives of $u$ and $v$ w.r.t. the maximum inflow velocity, $u_{\text{max}}$, i.e., $\partial u/\partial u_{\text{max}}$ and $\partial v/\partial u_{\text{max}}$.

Before comparing these derivatives which are to be computed by automatically differentiated versions of SEPRAN and FLUENT, we need to ensure that the values of $(u, v)$ resulting from FLUENT and SEPRAN simulations are comparable. The structured computational grid used for the FLUENT simulation consists of approximately 3,000 rectangular cells whereas an unstructured mesh consisting of 5,000 nodes and 10,000 triangle elements is used for the SEPRAN simulation. In order to compare the results obtained from SEPRAN and FLUENT, we extract linearly interpolated $(u, v)$-values at 4,000 sample points arranged in

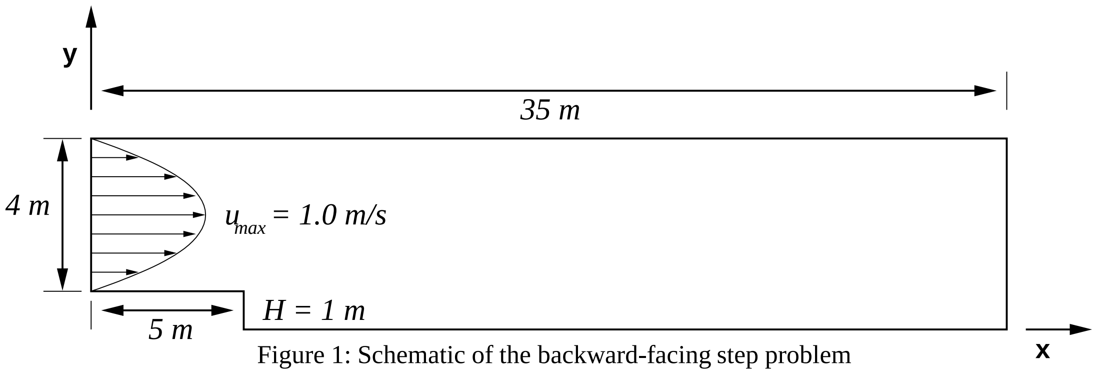

Figure 1: Schematic of the backward-facing step problem

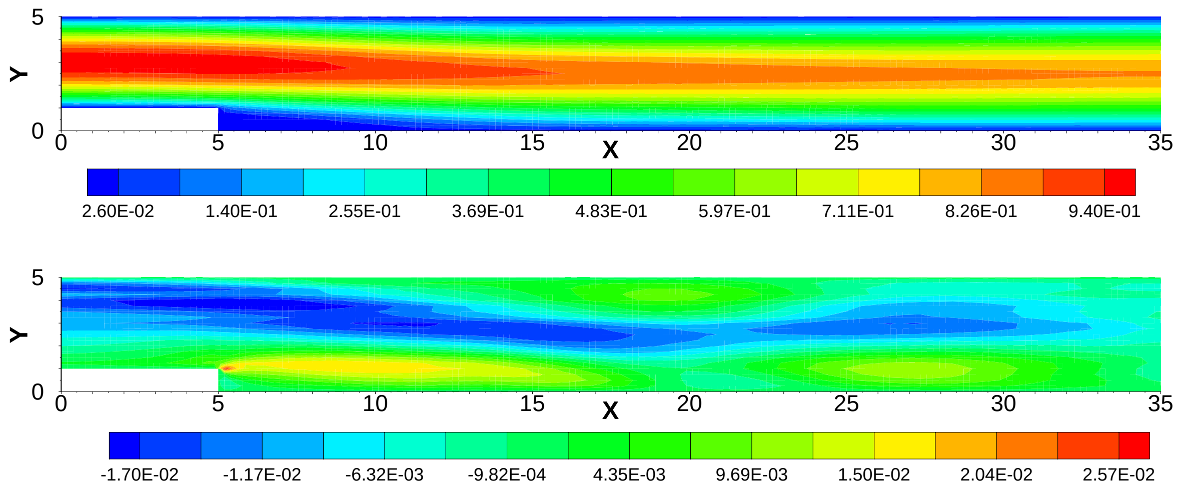

Figure 2: Horizontal velocity component, $u$, computed with FLUENT (top), and difference between the FLUENT and SEPRAN horizontal components of the velocity field, i.e., $u_{\mathrm{F}} - u_{\mathrm{S}}$ (bottom).

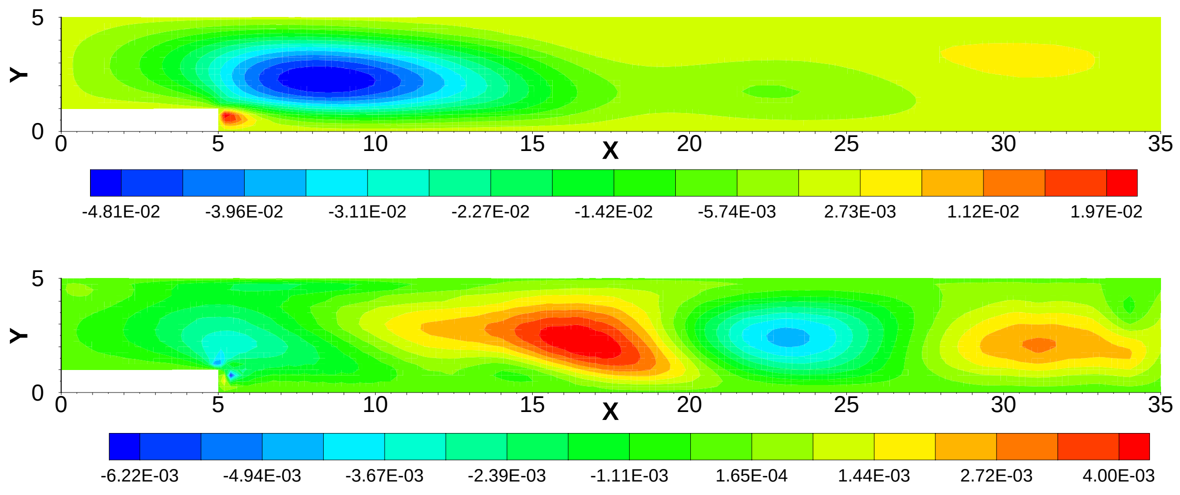

Figure 3: Vertical velocity component, $v$, computed with FLUENT (top), and difference between the FLUENT and SEPRAN vertical components of the velocity field, i.e., $v_{\mathrm{F}} - v_{\mathrm{S}}$ (bottom).

form of a regular grid covering the whole domain. All the plots presented in this note are based on this grid which enables the pointwise comparison of the results of the two simulations. Let $u_{\mathrm{F}}$ and $v_{\mathrm{F}}$ denote the interpolated values for $u$ and $v$ respectively, at the 4,000 sample points, based on the FLUENT solution. Similarly, $u_{\mathrm{S}}$ and $v_{\mathrm{S}}$ represent the corresponding values at the same points, where the SEPRAN solution is considered. In the upper plot of Fig. 2, the horizontal component of the velocity field, computed with FLUENT, $u_{\mathrm{F}}$, is depicted. Since the corresponding results computed by SEPRAN look almost identical to the FLUENT results, the lower plot of Fig. 2 shows the difference of $u$ computed by FLUENT, and $u$ computed by SEPRAN. More precisely, the values of $u_{\mathrm{F}} - u_{\mathrm{S}}$ are depicted in the lower plot of Fig. 2. In Fig. 3 the vertical velocity component, $v_{\mathrm{F}}$, and the difference of the vertical velocity components, $v_{\mathrm{F}} - v_{\mathrm{S}}$ are displayed. Note that for both components of the velocity vector, the differences of the two solutions found by SEPRAN and FLUENT are rather small, typically up to one order of magnitude, compared to the components of the velocity field. In comparison to the horizontal components, there are three separate clusters where the larger differences in the vertical components occur, located in the far flow field. However, from an engineering point of view, the recirculation area in the vicinity of the backward facing step is more interesting than the far flow field.

## DERIVATIVES OF THE FLOW FIELD

Now we compute the derivative of the velocity field w.r.t. the maximum inflow velocity, i.e., $\partial u/\partial u_{\mathrm{max}}$ and $\partial v/\partial u_{\mathrm{max}}$ using the differentiated versions of FLUENT and SEPRAN. In order to compare the results of these two differentiated simulation codes we interpolate values at certain points within the computational domain, as described in the previous section. Let $du_{\mathrm{F}}$ and $dv_{\mathrm{F}}$ denote the derivative values $\partial u/\partial u_{\mathrm{max}}$ and $\partial v/\partial u_{\mathrm{max}}$, respectively, computed by the differentiated version of FLUENT. In an analogous manner $du_{\mathrm{S}}$ and $dv_{\mathrm{S}}$ denote the corresponding derivative values obtained by the differentiated version of SEPRAN. In Fig. 4 the derivative values $du_{\mathrm{F}}$ are depicted in the upper plot. With the naked eye, the derivative values $du_{\mathrm{S}}$ are almost indistinguishable from $du_{\mathrm{F}}$. Similarly to the previous figures, the lower plot of Fig. 4 shows the difference, $du_{\mathrm{F}} - du_{\mathrm{S}}$ of the derivative values. Note that this difference is generally one order of magnitude below the derivative values. The derivatives of the vertical velocity component, computed by FLUENT and SEPRAN are compared in Fig. 5, where the upper and middle plot show the values for $dv_{\mathrm{F}}$ and $dv_{\mathrm{S}}$, respectively, and the lower plot shows the difference, $dv_{\mathrm{F}} - dv_{\mathrm{S}}$. It turns out that $dv_{\mathrm{F}}$ and $dv_{\mathrm{S}}$ differ in three areas around $x \approx 18$ , $x \approx 25$ , and $x \approx 33$ , which loosely correspond to the three clusters given in Fig. 3. In these clusters, the differences, $dv_{\mathrm{F}} - dv_{\mathrm{S}}$, are in the same order of magnitude as $dv_{\mathrm{F}}$ and $dv_{\mathrm{S}}$. In the remaining parts of the domain, $dv_{\mathrm{F}}$ and $dv_{\mathrm{S}}$ agree quite well. In particular, the extreme positive derivative values occurring behind the backward-facing step and the extreme negative values occurring in the area between $x \approx 10$ and $x \approx 15$ are computed similarly by the differentiated versions of both FLUENT and SEPRAN, so that the difference of $dv_{\mathrm{F}}$ and $dv_{\mathrm{S}}$ in those "interesting" parts is almost zero.

## SUMMARY

A commonly used technique for "validating" a given computer simulation is to compare various simulation codes, implementing different methods for solving the underlying numerical problems. Another option is to observe the sensitivities of the solution with respect to certain input parameters of the simulation. Such derivatives can be obtained by automatic differentiation, a technique for reliable and accurate computation of derivatives of functions given by a computer program.

In this study, the comparison of the numerical computations using two different simulation packages, FLUENT and SEPRAN, is described by considering the standard benchmark problem of an incompressible flow over a backward-facing step. For the specified flow problem, both simulation codes produce similar solutions, i.e., the resulting velocity fields are almost identical. Furthermore we investigate the derivatives of the flow field with respect to the maximum inflow velocity, which is a free input parameter of the simulation. These derivatives, computed by automatically differentiated versions of FLUENT and SEPRAN, show quite a good agreement with each other, although the two simulation packages employ fundamentally different algorithms for solving the flow problem. This demonstrates that automatic differentiation is a valuable tool for sensitivity analysis which can be successfully applied on large-scale simulation codes. This work suggests the assessment of the quality and robustness of numerical simulation codes not only by comparison of the results, but also by comparison of the corresponding sensitivities. The methods presented in this work provide a way towards "validating" numerical simulation codes, thus paving the way to reliable modeling of engineering problems.

## ACKNOWLEDGMENTS

This research is partially supported by the Deutsche Forschungsgemeinschaft (DFG) within SFB 540 "Model-based experimental analysis of kinetic phenomena in fluid multi-phase reactive systems," RWTH Aachen University, Germany.

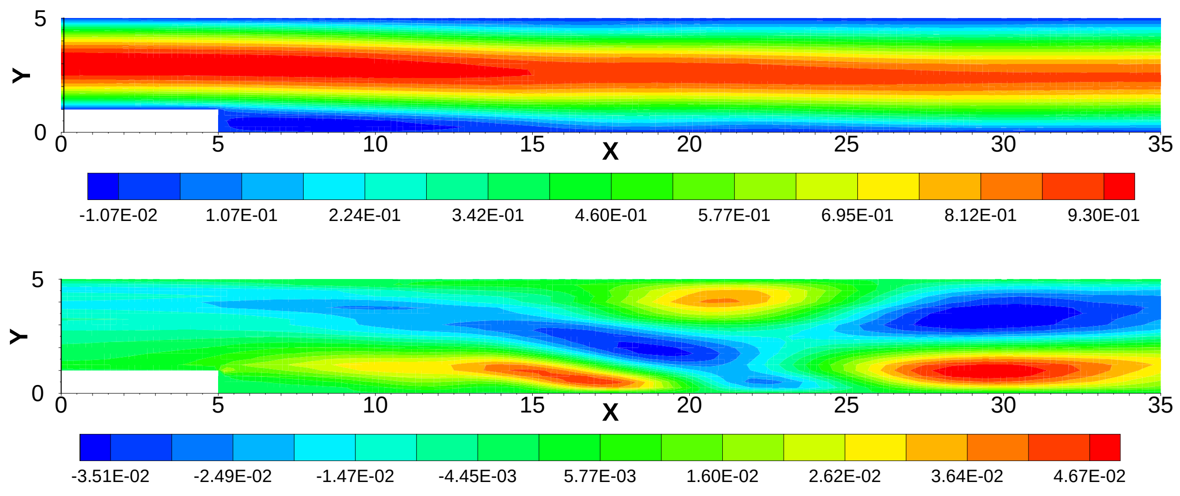

Figure 4: Derivative of $u$ w.r.t. $u_{\max}$, computed with FLUENT, i.e., $du_{\mathrm{F}}$ (top), and difference between the derivatives of the horizontal velocity components, computed by FLUENT and SEPRAN, i.e., $du_{\mathrm{F}} - du_{\mathrm{S}}$ (bottom).

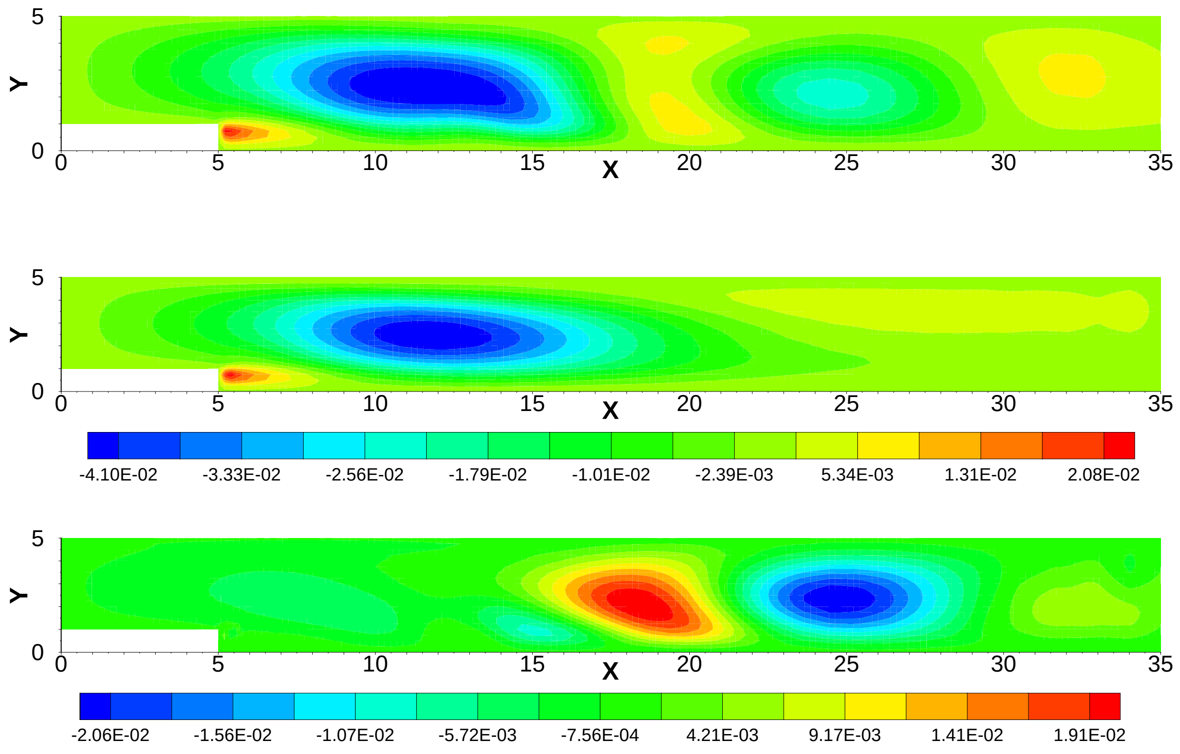

Figure 5: Comparison of the derivative of $v$ w.r.t. $u_{\max}$ computed by differentiated versions of FLUENT and SEPRAN: $dv_{\mathrm{F}}$ (top), $dv_{\mathrm{S}}$ (middle), $dv_{\mathrm{F}} - dv_{\mathrm{S}}$ (bottom).

## REFERENCES

Bischof C.; Carle A.; Khademi P.; and Mauer A. 1996. "ADIFOR 2.0: Automatic Differentiation of Fortran 77 Programs". *IEEE Computational Science & Engineering* 3, no. 3, 18–32.

Bischof C.H.; Bücker H.M.; Lang B.; Rasch A.; and Risch J.W. 2001. "On the Use of a Differentiated Finite Element Package for Sensitivity Analysis". In V.N. Alexandrov; J.J. Dongarra; B.A. Juliano; R.S. Renner; and C.J.K. Tan (eds.), "Computational Science – ICCS 2001, Proceedings of the International Conference on Computational Science, San Francisco, USA, May 28–30, 2001. Part I", Springer, Berlin, vol. 2073 of *Lecture Notes in Computer Science*, 795–801.

Bischof C.H.; Bücker H.M.; Lang B.; Rasch A.; and Risch J.W. 2003. "Extending the functionality of the general-purpose finite element package SEPRAN by automatic differentiation". *International Journal for Numerical Methods in Engineering* 58, no. 14, 2225–2238.

Bosch E.G.T. and Lasance C.J.M. 2000. "High accuracy thermal interface resistance measurement using a transient method". *Electronics Cooling Magazine* 6, no. 3, 26–32.

Bücker H.M. 2002. "Hierarchical Algorithms for Automatic Differentiation". Habilitationsschrift, Faculty of Mathematics, Computer Science, and Natural Sciences, Aachen University, Aachen.

Eberhard P. and Bischof C.H. 1999. "Automatic Differentiation of Numerical Integration Algorithms". *Mathematics of Computation* 68, 717–731.

*FLUENT Tutorial Guide*. Fluent Inc., Lebanon, NH, USA.

*FLUENT 4.4 User's Guide*. Fluent Inc., Lebanon, NH, USA.

Griewank A. 2000. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. SIAM, Philadelphia.

Rall L.B. 1981. *Automatic Differentiation: Techniques and Applications*, vol. 120 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin.

Roache P.J. 1998. *Verification and Validation in Computational Science and Engineering*. Hermosa Publishers, Albuquerque, New Mexico.

Segal G. 1993a. *SEPRAN Standard Problems*. Ingenieursbureau Sepra, Leidschendam, NL.

Segal G. 1993b. *SEPRAN Users Manual*. Ingenieursbureau Sepra, Leidschendam, NL.

Segal G.; Vuik C.; and Vermolen F. 1998. "A Conserving Discretization for the Free Boundary in a Two-Dimensional Stefan Problem". *Journal of Computational Physics* 141, no. 1, 1–21.

van Keken P.E.; Yuen D.A.; and Petzold L.R. 1995. "DASPK: a new high order and adaptive time-integration technique with applications to mantle convection with strongly temperature- and pressure-dependent rheology". *Geophysical & Astrophysical Fluid Dynamics* 80, 57–74.

## AUTHOR BIOGRAPHIES

**CHRISTIAN H. BISCHOF** is head of the Institute for Scientific Computing and of the Center for Computing and Communication of Aachen University. He received a Ph.D. in computer science from Cornell University in 1988. His interests lie in semantic transformation, especially automatic differentiation, high-performance computing, virtual reality, and problem solving environments.

**H. MARTIN BÜCKER** received the Dipl.-Ing. degree in electrical engineering, the Dipl.-Inform. degree in computer science, and the Ph.D. degree in electrical engineering from RWTH Aachen University in 1993, 1994, and 1997, respectively. He worked for the Central Institute for Applied Mathematics, Research Center Jülich, from 1993 to 1998, participating in the design of parallel algorithms for sparse matrix problems. Since 1998 he has been with the Institute for Scientific Computing, RWTH Aachen University, where he received his Habilitation in 2003. His current areas of research include high-performance computing, numerical linear algebra, and automatic differentiation.

**ARNO RASCH** received his Dipl.-Inform. in computer science from RWTH Aachen University in 2000. He is currently working as research assistant and PhD student at the Institute for Scientific Computing, RWTH Aachen University. His research interests include parallel computing, automatic differentiation, and engineering applications.

**EMIL-IOAN SLUSANSCHI** received his Dipl.-Ing. in computer science from University Politehnica of Bucharest at the Faculty of Automatic Control and Computers in 2000. He also received his Advanced Studies Diploma at the same University in 2001. He is currently working as research assistant and PhD student at the Institute for Scientific Computing, RWTH Aachen University. His research interests include parallel computing and parallel architectures, automatic differentiation, and engineering applications.