On the Use of a Differentiated Finite Element Package for Sensitivity Analysis*

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

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

Abstract. Derivatives are ubiquitous in various areas of computational science including sensitivity analysis and parameter optimization of computer models. Among the various methods for obtaining derivatives, automatic differentiation (AD) combines freedom from approximation errors, high performance, and the ability to handle arbitrarily complex codes arising from large-scale scientific investigations. In this note, we show how AD technology can aid in the sensitivity analysis of a computer model by considering a classic fluid flow experiment as an example. To this end, the software tool ADIFOR implementing the AD technology for functions written in Fortran 77 was applied to the large finite element package SEPRAN. Differentiated versions of SEPRAN enable sensitivity analysis for a wide range of applications, not only from computational fluid dynamics.

1 Introduction

In assessing the robustness of a computer code, or to determine profitable avenues for improving a design, it is important to know the rate of change of the model output that is implied by changing certain model inputs. Derivatives are one way to implement such a sensitivity analysis. Traditionally, divided differences are employed in this context to approximate derivatives, leading to results of dubious quality at often great computational expense. Automatic differentiation (AD), in contrast, is an alternative for the evaluation of derivatives providing guaranteed accuracy, ease of use, and computational efficiency. Note that derivatives play a crucial role not only in sensitivity analysis but in numerical computing in general. Examples include the solution of nonlinear systems of equations, stiff ordinary differential equations, partial differential equations, differential-algebraic equations, and multidisciplinary design optimization, to name just a few. Therefore, the availability of accurate and efficient derivatives is often indispensable in computational science.

* 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," Aachen University of Technology, Germany.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 795–801, 2001.

In this note we give an answer to the following question. Given an arbitrarily complicated computer program in a high-level programming language such as Fortran, C, or C++, how do we get accurate and efficient derivatives for the function implemented by the computer program? We will argue that the answer is to apply automatic differentiation. Although AD is a general technique applicable to programs written in virtually any high-level programming language [1, 4,5,6], we will assume in this note that the function for which derivatives are desired is written in Fortran 77, as it is the case for the package SEPRAN [8]. Developed at "Ingenieursbureau SEPRA" and Delft University of Technology, SEPRAN is a large general purpose finite element code intended to be used for the numerical solution of second order elliptic and parabolic partial differential equations in two and three dimensions. It is employed in a wide variety of engineering applications [3,9,10,11,12,13,14] including structural mechanics and laminar or turbulent flow of incompressible liquids.

In Sect. 2, we describe the basic principles behind the AD technology as well as the application of an AD tool to SEPRAN leading to a differentiated version of SEPRAN called SEPRAN.AD hereafter. The simulation of a classic fluid flow experiment, namely the flow over a 2D backward facing step, is taken as a simple, yet illustrative, example for carrying out numerical experiments in Sect. 3. We show how a SEPRAN user benefits from the preprocessed code SEPRAN.AD in that it provides—with no more effort than is required to run SEPRAN itself—a set of derivatives that is accurate and consistent with the numerical simulation. Finally, we point out that the functionality contained in differentiated versions of SEPRAN allows the sensitivity analysis of a wide range of potential SEPRAN applications, not only from computational fluid dynamics.

2 Automatic Differentiation and SEPRAN

Automatic differentiation is a powerful technique for accurately evaluating derivatives of functions given in the form of a high-level programming language, e.g., Fortran, C, or C++. The reader is referred to the recent book by Griewank [5] and the proceedings of AD workshops [1,4,6] for details on this technique. In automatic differentiation the program is treated as a—potentially very long—sequence of elementary statements such as binary 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 that augment a given code C to a new code C.AD such that, in addition to the original outputs, C.AD also computes the derivatives of some of these output variables with respect to selected inputs. This way AD requires little human effort and produces derivatives that are accurate up to machine precision.

The AD technology is not only applicable for small codes but scales up to large codes with several hundreds of thousand lines; see the above-mentioned proceedings and the references given therein. We applied automatic differentiation

to the general purpose finite element package SEPRAN consisting of approximately 400,000 lines of Fortran 77. The package enables simulation in various scientific areas ranging from fluid dynamics, structural mechanics to electromagnetism. Analyses of two-dimensional, axisymmetric and three-dimensional steady state or transient simulations in complex geometries are supported. Examples include potential problems, convection-diffusion problems, Helmholtz-type equations, heat equations, and Navier-Stokes equations.

We used the ADIFOR tool [2] to generate SEPRAN.AD, the differentiated version. ADIFOR (**A**utomatic **DI**fferentiation of **FOR**tran) implements the AD technology for Fortran 77 codes. The details of this process will be presented elsewhere. In general, a user of an AD tool needs to perform the following steps:

1. As a preprocessing step, "dirty" legacy code needs certain manual massaging to produce "clean" code conforming to the language standard. Notice that SEPRAN is programmed in an almost clean way so that only small changes to the original code had to be done by hand, examples being several instances where different routines interpret the same memory as holding either double precision real data or single precision complex data. This non-standard technique is sometimes employed in order to save memory, and it is not detected by current Fortran compilers because their view of the program is restricted to one routine or file at a time. ADIFOR, by contrast, does a global data flow analysis and immediately detects this kind of inconsistency.
2. The user indicates the desired derivatives by specifying the dependent (output) and independent (input) variables. This is typically done through a control file.
3. The tool is then applied to the clean code to produce augmented code for the additional computation of derivatives. We applied ADIFOR 2.1 to SEPRAN (approximately 400,000 lines of code including comments) to obtain SEPRAN.AD (roughly 600,000 lines of code including comments). Note that the global analysis enables ADIFOR to decide whether the work done in a routine is relevant to the desired derivative values. Therefore only a subset of the routines is actually augmented.
4. A small piece of code (driver code) is constructed that calls the generated routines made available by SEPRAN.AD.
5. The generated derivative code and the driver code is compiled and linked with supporting libraries.

Upon successful completion of these steps, derivatives are available by simply calling the corresponding routines from SEPRAN.AD, the differentiated version, rather than from SEPRAN, the original code.

Once the differentiated code is available, it enables sensitivity analysis of different problems (e.g., flow around obstacles, flow over a backward facing step, etc.) with respect to the specified input and output variables. If other variables are to be considered then steps 2 through 5 of the above procedure are repeated, which requires only little human interaction. (There is a slightly more sophisticated way to do it, which even avoids repeating steps 2 and 3.) Note that step 1 is the only step that might need substantial human effort and is done only once.

The above discussion demonstrates the ease of use and the versatility of the AD technology.

3 Results

In the numerical experiments reported in this section, a simulation of a classic fluid flow experiment, namely the flow over a 2D backward facing step [7], is taken as a sample problem. The goal of this note is not to concentrate on the values of the flow field but to give the reader an impression of the improved functionality of the differentiated version SEPRAN.AD as compared to SEPRAN. In this standard benchmark problem for incompressible fluids, a stationary flow over a backward facing step is considered. We carried out numerical experiments at Reynolds numbers around 50 with no-slip boundary conditions at the upper and lower walls of the pipe, a parabolic inflow in horizontal direction, and a parallel outflow.

Given the maximal horizontal velocity component v_0 of the inflow, the density ρ, and the viscosity μ, one can easily use SEPRAN to compute the velocity v and the pressure p at any point in the pipe. From an abstract point of view, the corresponding code implements a function f taking v_0, ρ, and μ as input and producing the output v and p; that is

$$\begin{pmatrix} v \\ p \end{pmatrix} = f(v_0, \rho, \mu) \ .$$

Invoking the corresponding SEPRAN code evaluates f at a given input.

Suppose that we are interested in evaluating the derivatives of some outputs of f with respect to some of its inputs at the same point where f itself is evaluated. For instance, an engineer might be interested in the rate of change of the pressure p with respect to the inflow velocity v_0, i.e., $\partial p/\partial v_0$. A numerical approach would make use of divided differences to approximate the derivative. For the sake of simplicity, we only consider first-order forward divided differences such as

$$\frac{\partial p(v_0, \rho, \mu)}{\partial v_0} \approx \frac{p(v_0 + h, \rho, \mu) - p(v_0, \rho, \mu)}{h} \ , \tag{1}$$

where h is a suitably chosen step size. An advantage of the divided difference approach is its simplicity; that is, the corresponding function is evaluated in a black-box fashion. The main disadvantage of divided differences is that the accuracy of the approximation depends crucially on a suitable step size h. Unfortunately, an optimal or even near-optimal step size is often not known a priori. Therefore, the program is usually run several times to find a reasonable step size. Note that there is a complementary influence of truncation and cancellation error to the overall accuracy of the method: on the one hand, the step size should be as small as possible to decrease the approximation error that would be present even if infinite-precision arithmetic were to be used. On the other hand, the step size must not be too small to avoid cancellation of significant digits when using finite-precision arithmetic in the evaluation of (1).

The above problem of determining a step size is a conceptual disadvantage in the divided difference approach and also applies to higher-order derivatives. Automatic differentiation, on the contrary, does not involve any truncation error. Derivatives produced by AD are exact up to machine precision. To demonstrate the difference in accuracy between AD and divided differences, we formally define

$$\mathrm{diff}(p, v_0) := \left\| \frac{\partial p(v_0, \rho, \mu)}{\partial v_0} - \frac{p(v_0 + h, \rho, \mu) - p(v_0, \rho, \mu)}{h} \right\|_\infty , \tag{2}$$

where the first term on the right-hand side is the value computed by automatic differentiation. Hence, $\mathrm{diff}(p, v_0)$ is a measure of the difference of the numerical accuracy of the derivatives of p with respect to v_0 obtained from automatic differentiation and divided differences.

For the backward facing step example, the difference between the derivative values generated by AD and divided differences using varying step sizes h is shown in Tab. 1.

Table 1. Comparison of the accuracy of derivatives obtained from divided differences using a step size h and automatic differentiation.

h	$\mathrm{diff}(v, v_0)$	$\mathrm{diff}(v, \rho)$	$\mathrm{diff}(v, \mu)$	$\mathrm{diff}(p, v_0)$	$\mathrm{diff}(p, \rho)$	$\mathrm{diff}(p, \mu)$
10^{-2}	0.002189	0.001134	11.774571	0.002310	0.001087	5.259458
10^{-3}	0.000218	0.000111	2.039314	0.000230	0.000107	0.996281
10^{-4}	0.000043	0.000042	0.217868	0.000032	0.000028	0.107945
10^{-5}	0.000277	0.000251	0.021579	0.000304	0.000326	0.010897
10^{-6}	0.002078	0.003096	0.002766	0.002146	0.001811	0.002294
10^{-7}	0.029861	0.038406	0.027861	0.020655	0.023987	0.028521
10^{-8}	0.197591	0.260977	0.213695	0.155814	0.193424	0.184808
10^{-9}	5.313513	3.374881	3.746390	1.622882	2.115335	3.093727
10^{-10}	25.566379	20.481873	27.184625	21.904384	14.420520	24.604476

Here, the definition (2) is extended to derivatives other than $\partial p/\partial v_0$ in a straight forward fashion. The derivatives of the pressure and the velocity fields are evaluated at $(v_0, \rho, \mu) = (1.0, 1.0, 0.01)$. The table demonstrates the dependence of the divided difference approach from the step size. In all columns of the table, the difference values first decrease with decreasing step size and then increase again, and the optimum step size depends on the particular derivative. For instance, $\mathrm{diff}(p, v_0)$ is minimal for $h = 10^{-4}$ whereas the minimum of $\mathrm{diff}(p, \mu)$ is at $h = 10^{-6}$ indicating the need for finding different suitable step sizes when differentiating with respect to v_0 and μ.

In contrast to divided differences, there is no need for experimenting with step sizes at all when applying automatic differentiation because there is no truncation error. Using AD, the accurate derivative values of p and v with respect to all three input parameters, together with the function values, were obtained

with a single call to the differentiated version SEPRAN.AD. This computation required roughly 3.3 seconds and 95 MB of memory, compared to 1.2 seconds and 26 MB for one run of SEPRAN. Note that using divided differences for approximating the derivatives with respect to three variables requires at least a total of four SEPRAN calls. Thus AD, in addition to providing more reliable results, also takes less time than divided differences. We finally mention that SEPRAN.AD needs additional memory to store the three derivatives. So, the above mentioned increase of a factor of 3.7 is moderate.

4 Concluding Remarks

The technique of automatic differentiation is proved to be an efficient way to obtain accurate derivatives of functions given in the form of a computer program written in any high-level language such as Fortran, C, or C++. The technique scales up to large simulation codes that are used today as a crucial part in a broad variety of scientific and engineering investigations. We applied automatic differentiation to the general purpose finite element package SEPRAN consisting of approximately 400,000 lines of Fortran 77. The resulting differentiated version is produced in an automated way by augmenting the original version by additional statements computing derivatives. For a classic fluid flow experiment, we showed the improved functionality including its ease of use. Moreover, we compared the values obtained from automatic differentiation with those produced by numerical differentiation based on divided differences. The latter approach is a sensitive approximation process inherently involving the choice of a suitable step size. On the contrary, there is no concept of a step size in automatic differentiation because it accumulates derivatives of known elementary operations, finally leading to exact derivatives. For the numerical fluid flow experiment, we also showed that automatic differentiation is more efficient in terms of execution times than divided differences while only moderately increasing storage requirement.

Besides the basic features presented in this note, automatic differentiation and the software tools implementing the technology offer even more functionality. One of the highlights of automatic differentiation is the fact that a particular way to accumulate the final derivatives, the so-called reverse mode, can deliver the gradient of a scalar-valued function at a cost proportional to the function evaluation itself. That is, its cost is *independent from the number of unknowns*, whereas the cost for divided differences is roughly proportional to the gradient's length. For purposes different from mere sensitivity analysis, derivatives of arbitrary order and directional derivatives can also be obtained with similar techniques.

References

[1] M. Berz, C. Bischof, G. Corliss, and A. Griewank. *Computational Differentiation: Techniques, Applications, and Tools.* SIAM, Philadelphia, 1996.

[2] 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.

[3] E. G. T. Bosch and C. J. M. Lasance. High accuracy thermal interface resistance measurement using a transient method. *Electronics Cooling Magazine*, 6(3), 2000.

[4] G. Corliss, A. Griewank, C. Faure, L. Hascoët, and U. Naumann, editors. *Automatic Differentiation 2000: From Simulation to Optimization*. Springer, 2001. To appear.

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

[6] A. Griewank and G. Corliss. *Automatic Differentiation of Algorithms*. SIAM, Philadelphia, 1991.

[7] G. Segal. *SEPRAN Standard Problems*. Ingenieursbureau Sepra, Leidschendam, NL, 1993.

[8] G. Segal. *SEPRAN Users Manual*. Ingenieursbureau Sepra, Leidschendam, NL, 1993.

[9] G. Segal, C. Vuik, and F. Vermolen. A conserving discretization for the free boundary in a two-dimensional Stefan problem. *Journal of Computational Physics*, 141(1):1–21, 1998.

[10] A. P. van den Berg, P. E. van Keken, and D. A. Yuen. The effects of a composite non-Newtonian and Newtonian rheology on mantle convection. *Geophys. J. Int.*, 115:62–78, 1993.

[11] P. van Keken, D. A. Yuen, and L. Petzold. 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, 1995.

[12] P. E. van Keken, C. J. Spiers, A. P. van den Berg, and E. J. Muyzert. The effective viscosity of rocksalt: implementation of steady-state creep laws in numerical models of salt diapirism. *Tectonophysics*, 225:457–476, 1993.

[13] N. J. Vlaar, P. E. van Keken, and A. P. van den Berg. Cooling of the Earth in the Archaean: consequences of pressure-release melting in a hot mantle. *Earth Plan. Sci. Lett.*, 121:1–18, 1994.

[14] C. Vuik, A. Segal, and F. J. Vermolen. A conserving discretization for a Stefan problem with an interface reaction at the free boundary. *Computing and Visualization in Science*, 3(1/2):109–114, 2000.