Computer Physics Communications 147 (2002) 720–723

Computer Physics Communications

www.elsevier.com/locate/cpc

Computing sensitivities of the electrostatic potential by automatic differentiation

H.M. Bücker [*], B. Lang, A. Rasch, C.H. Bischof

Institute for Scientific Computing, Aachen University of Technology, D-52056 Aachen, Germany

Abstract

Given a computer model for the electrostatic potential in an L-shaped region with media of different dielectric permeabilities in two subregions, we are interested in the robustness of the simulation by identifying the rate of change of the potential with respect to a change in the permeabilities. Such sensitivity analyses, assessing the rate of change of certain model outputs implied by varying certain model inputs, can be carried out by computing the corresponding partial derivatives. In large-scale computational physics, the underlying computer model is typically available as a complicated computer code in a high-level programming language such as Fortran, C, or C++. To obtain accurate and efficient derivatives of functions given in this form, we use a technique called automatic or algorithmic differentiation. Unlike numerical differentiation based on divided differences, derivatives generated by automatic differentiation are free of truncation error. Here, the automatic differentiation tool Adifor is used to transform the given computer model—implemented with the general purpose finite element package SEPRAN—into a new computer code computing the derivatives of the electrostatic potential with respect to the dielectric permeabilities. In doing so, we automatically translate 400,000 lines of Fortran 77 into a new program consisting of 600,000 lines of Fortran 77. We compare our approach with a traditional approach based on numerical differentiation and quantify its advantages in terms of accuracy and computational efficiency. © 2002 Elsevier Science B.V. All rights reserved.

PACS: 89.80; 02.70; 07.05.T

Keywords: Sensitivity analysis; Automatic differentiation; Algorithmic differentiation; Derivatives; Adifor; SEPRAN

1. Introduction

Complicated computer models are increasingly becoming a vital part of many scientific investigations affecting computational physicists in areas ranging from materials science and soft matter to quantum physics and astrophysics, just to name a few. With this noticeable and rapid shift toward highly complex simulation codes, methodological developments to verify and validate computer codes are also gaining importance. One way to obtain information about a complicated computer model's behavior is to determine the change of its output, $\mathbf{y}$, caused by specific changes in its input, $\mathbf{x}$. Such a sensitivity analysis is traditionally carried out by running the computer code over and over again with slightly perturbed inputs. This approach is equivalent to a numerical approximation of the derivative $\partial\mathbf{y}/\partial\mathbf{x}$. While this numerical differentiation approach is conceptually simple, it may consume enormous computing time and is also prone to truncation error.

[*] Corresponding author.
E-mail address: buecker@sc.rwth-aachen.de (H.M. Bücker).

An alternative for the computation of derivatives of functions given in the form of complicated computer models is a general technique called automatic or algorithmic differentiation (AD). Here, a given computer program is automatically translated into another program capable of evaluating not only the original function 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.

In this note, we apply the AD tool Adifor [1], developed at Argonne National Laboratory and Rice University, to the large general purpose finite element package SEPRAN [2], developed at "Ingenieursbureau SEPRA" and Delft University of Technology. More precisely, given a computer code using SEPRAN routines for the numerical solution of the electrostatic potential problem in an L-shaped region consisting of two subregions with media of different dielectric permeabilities, we compute the derivatives of the electrostatic potential with respect to the two permeabilities. In this example, constant potentials were prescribed for the lower and upper boundaries of the region.

In Section 2, the technique of automatic differentiation is briefly explained and the process of applying the Adifor tool to SEPRAN is sketched. In Section 3, the approach using automatic differentiation is compared in terms of time, storage, and accuracy to a traditional numerical divided difference approach. The findings are summarized in Section 4.

2. Automatic differentiation of 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++; see the book by Griewank [3] and the proceedings of AD workshops [4–6] for details on this technique. In automatic differentiation the program is treated 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 stepwise derivatives to yield the derivatives of the whole program. For instance, given a statement like

$$\texttt{z} \leftarrow \texttt{x} * \texttt{sin(y)}$$

in the original computer program, the AD-generated program contains an additional statement of the form

$$\texttt{g\_z} \leftarrow \texttt{sin(y)} * \texttt{g\_x} + \texttt{x} * \texttt{cos(y)} * \texttt{g\_y}.$$

Here, new objects `g_z`, `g_x`, and `g_y` containing the derivative values are associated with the objects `z`, `x`, and `y`, respectively. Properly initializing the derivative objects results in a code capable of computing the original function and its derivatives.

This mechanical process can be automated, and several AD tools are available augmenting a given code with statements for the computation of derivatives. Thus, AD requires little human effort and produces accurate derivatives.

A common misconception is that the AD technology is only applicable to small codes. In this note we show that AD scales up to complex codes: we apply AD to the SEPRAN package consisting of approximately 800,000 lines of Fortran 77. SEPRAN is intended to be used for the numerical solution of second-order elliptic and parabolic partial differential equations in two and three dimensions. It enables simulation in various scientific areas [7–10] ranging from fluid dynamics to 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.

In the present study, we use the SEPRAN package to compute the electrostatic potential ϕ in an L-shaped geometry consisting of two regions

$$R_1: \quad 0 \leqslant x \leqslant 2 \quad \text{and} \quad 0 \leqslant y \leqslant 1,$$
$$R_2: \quad 0 \leqslant x \leqslant 1 \quad \text{and} \quad 1 \leqslant y \leqslant 2,$$

with homogeneous media of dielectric permeabilities, ϵ_1 and ϵ_2, respectively. We are interested in the derivatives $\partial\phi/\partial\epsilon_1$ and $\partial\phi/\partial\epsilon_2$.

After specifying the dependent (output) variable associated to ϕ and the independent (input) variables associated to ϵ_1 and ϵ_2, the AD tool Adifor is applied to a SEPRAN subset of 400,000 lines, generating a Fortran 77 program with roughly 600,000 lines computing the desired derivatives. The details of this process are beyond the scope of this note.

3. Comparison with divided differences

Given ϵ_1 and ϵ_2 one can easily use SEPRAN to compute the electrostatic potential ϕ. From an abstract point of view, the corresponding code implements a function f taking ϵ_1 and ϵ_2 as input and producing the output ϕ; that is

$$\phi = f(\mathbf{x}), \quad \text{where } \mathbf{x} = (\epsilon_1, \epsilon_2).$$

Calling the AD-generated code not only evaluates f but also the derivatives of f with respect to $\mathbf{x}$ at the same given input data. One of the derivatives computed by this "differentiated" version of SEPRAN is displayed in Fig. 1 demonstrating that the largest sensitivity of ϕ with respect to ϵ_2 is found to be at the boundary of the two media. Throughout this note, for simplicity, all quantities have been properly scaled and dimensions omitted.

On a Hewlett Packard V-class server with 240 MHz PA8200 processors, the differentiated version takes 2.36 times as long to compute as the original SEPRAN code, and the storage requirement increases by a factor of 2.88. These factors are reasonable given that the differentiated SEPRAN code not only computes the original function but also two partial derivatives.

A conceptual advantage of AD-generated derivatives is the absence of any truncation error. In AD, there is no need for experimenting with different step sizes to balance truncation error and round-off error.

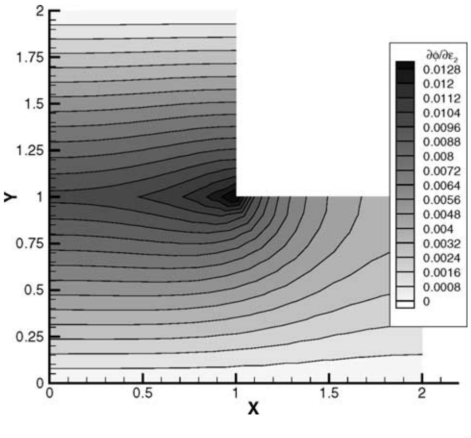

Fig. 1. The derivatives $\partial\phi/\partial\epsilon_2$ at $\epsilon_1 = 1.0$ and $\epsilon_2 = 10.0$.

To compare the AD-generated derivatives with those based on divided differences, let

$$\Delta\epsilon_1 = \left\| \frac{\partial\phi(\epsilon_1, \epsilon_2)}{\partial\epsilon_1} - \frac{\phi(\epsilon_1 + h, \epsilon_2) - \phi(\epsilon_1, \epsilon_2)}{h} \right\|_\infty$$

denote the difference of $\partial\phi/\partial\epsilon_1$ obtained from AD, given by the first term on the right-hand side, and the value computed by a first-order forward difference with step size h. Similarly, $\Delta\epsilon_2$ denotes the corresponding difference for $\partial\phi/\partial\epsilon_2$.

In Table 1, the influence of the step size h on the accuracy of the derivatives computed by divided differences is given. Decreasing h from 10^{-2} decreases the truncation error of the divided differences so that $\Delta\epsilon_1$ and $\Delta\epsilon_2$ also decrease. Decreasing h further leads to an increase of the round-off error in the divided differences so that $\Delta\epsilon_1$ and $\Delta\epsilon_2$ also increase. Note that the "best" step sizes for $\Delta\epsilon_1$ and $\Delta\epsilon_2$ are different. These general features of divided difference approximations are insensitive to changes in the mesh used in the simulation whereas the specific values $\Delta\epsilon_i$, as well as the best step sizes, depend mildly on the actual mesh. The data in Fig. 1 and Table 1 were obtained with a mesh containing 202 nodes and 352 elements.

In practice, the simulation does not yield the exact solution $\phi(\mathbf{x})$, but only an approximation $\phi(\mathbf{x}) + \delta(\mathbf{x})$. Thus, the AD-generated derivative $\phi'(\mathbf{x}) + \delta'(\mathbf{x})$ differs from the exact value by $\delta'(\mathbf{x})$.

In comparison to divided differences, the derivatives generated by AD are not only more accurate but their computation also requires less time because the implementation of divided differences here demands calling the original code at least three times to obtain approximations of derivatives with respect to two vari-

Table 1
Comparison of accuracy of derivatives obtained from divided differences using a step size h and from AD

h	$\Delta\epsilon_1$	$\Delta\epsilon_2$
10^{-2}	0.00022876125	0.00001154521
10^{-3}	0.00002291012	0.00000115539
10^{-4}	0.00000229137	0.00000011553
10^{-5}	0.00000022877	0.00000001141
10^{-6}	0.00000002321	0.00000000443
10^{-7}	0.00000001924	0.00000002605
10^{-8}	0.00000022175	0.00000032113
10^{-9}	0.00000533564	0.00000191681
10^{-10}	0.00004025372	0.00001957613

ables whereas the corresponding factor is 2.36 for AD. Run-time and memory factors for AD are below the expected value of 3 because some intermediate computations do not depend on ϵ_1 and ϵ_2, or do not influence the final output ϕ. Such computations are *automatically* recognized during the AD process, and no derivative code is produced for them.

4. Concluding remarks

Rather than using the traditional numerical approach based on divided differences, we implement the sensitivity analysis of an electrostatic potential problem via automatic differentiation. The derivatives produced by this approach are accurate up to machine precision. Their computation is also shown to be more efficient than derivatives based on divided differences. Furthermore, the study demonstrates that the technology of automatic differentiation is not only applicable to small codes but scales up to computer models consisting of hundreds of thousands of lines of code. With the rapid increase of the complexity of computer models in today's scientific investigations we expect that the importance of techniques similar to the approach presented here will grow in the near future.

Finally, note that derivatives play a crucial role not only in sensitivity analysis but in computational science in general. Examples where automatic differentiation is used include nonlinear systems of equations, ordinary and partial differential equations, differential-algebraic equations, and multidisciplinary design optimization.

Acknowledgements

The authors thank Jakob W. Risch for his instrumental role during the initial phase of this project which is 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.

References

[1] C. Bischof, A. Carle, P. Khademi, A. Mauer, ADIFOR 2.0: Automatic differentiation of Fortran 77 programs, IEEE Comput. Sci. Engrg. 3 (3) (1996) 18–32.

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

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

[4] M. Berz, C. Bischof, G. Corliss, A. Griewank (Eds.), Computational Differentiation: Techniques, Applications, and Tools, SIAM, Philadelphia, 1996.

[5] G. Corliss, A. Griewank, C. Faure, L. Hascoët, U. Naumann (Eds.), Automatic Differentiation of Algorithms: From Simulation to Optimization, Springer, 2002.

[6] A. Griewank, G. Corliss (Eds.), Automatic Differentiation of Algorithms, SIAM, Philadelphia, 1991.

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

[8] G. Segal, C. Vuik, F. Vermolen, A conserving discretization for the free boundary in a two-dimensional Stefan problem, J. Comput. Phys. 141 (1) (1998) 1–21.

[9] P. van Keken, D.A. Yuen, L. Petzold, DASPK: A new high order and adaptive time-integration technique with applications to mantle convection with strongly temperature- and pressure-dependent rheology, Geophys. Astrophys. Fluid Dynam. 80 (1995) 57–74.

[10] C. Vuik, A. Segal, F.J. Vermolen, A conserving discretization for a Stefan problem with an interface reaction at the free boundary, Comput. Visualization Sci. 3 (1/2) (2000) 109–114.