APPLICATION OF AUTOMATIC DIFFERENTIATION TO GROUNDWATER TRANSPORT MODELS

C. H. Bischof,[1] G. J. Whiffen,[2] C. A. Shoemaker,[2] Alan Carle,[3] and A. A. Ross[1]

(1) Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, Illinois 60439
(2) School of Civil and Environmental Engineering
Cornell University
Ithaca, New York 14853
(3) Center for Research on Parallel Computation
Rice University
Houston, Texas 77251

Argonne Preprint MCS-P414-0194
to appear in the Proceedings of the International Conference on Computational Methods on Water Resources, Heidelberg, July 1994

Automatic differentiation (AD) is a technique for generating efficient and reliable derivative codes from computer programs with a minimum of human effort. Derivatives of model output with respect to input are obtained exactly. No intrinsic limits to program length or complexity exist for this procedure. Calculation of derivatives of complex numerical models is required in systems optimization, parameter identification, and systems identification. We report on our experiences with the ADIFOR (Automatic Differentiation in Fortran) tool on a two-dimensional groundwater flow and contaminant transport finite-element model, ISOQUAD, and a three-dimensional contaminant transport finite-element model, TLS3D. Derivative values and computational times for the automatic differentiation procedure are compared with values obtained from the divided differences and handwritten analytic approaches. We found that the derivative codes generated by ADIFOR provided accurate derivatives and ran significantly faster than divided-differences approximations, typically in a tenth of the CPU time required for the imprecise divided-differences method for both codes. We also comment on the benefit of automatic differentiation technology with respect to accelerating the transfer of general techniques developed for using water resource computer models, such as optimal design, sensitivity analysis, and inverse modeling problems, to field problems.

1 INTRODUCTION

Calculation of derivatives of computer models is necessary for a wide variety of procedures of interest to numerical modelers, for example, sensitivity analysis, optimization, and inverse problems. Several methods have traditionally been used to obtain derivatives of computer model output with respect to input. Analytical calculation can be used whenever it is reasonable to write analytic derivatives and then code them by hand (see, for example, *Chang et al.* [1992]). If a code is extremely complex, this approach may be infeasible. Symbolic differentiation programs often generate very large and inefficient derivative formulas. Writing an adjoint code to calculate derivatives is a well-defined procedure that calculates exact derivatives very efficiently (*Leitmann,* 1981). This method, however, also requires a potentially large amount of human effort.

One of the most common procedures used to obtain model derivatives of complex computer models is the divided-differences method. This method calculates derivatives by perturbing model input by small finite amounts and dividing the resulting model output perturbations by input perturbations (see, for example, *Gorelick et al.,* 1984). This procedure may fail to give accurate derivatives and is inefficient since it requires as many model simulations as there are input parameters of interest. In general, however, the divided-differences method does not require much human effort and, until recently, has been the only universally applicable approach available to obtain derivatives for very complex computer models.

This paper describes the application of automatic differentiation to obtain codes that evaluate derivatives of complex computer models efficiently, exactly, and with a minimum of human effort. Automatic differentiation is a method that produces a derivative code, given the model code and a list of parameters that are considered dependent and independent with respect to differentiation. The method produces a code that will evaluate derivatives exactly (up to machine precision), usually in much less time than the approximate finite-differences method. There are no inherent limits on program size or complexity.

We applied the automatic differentiation tool ADIFOR (Automatic Differentiation in Fortran) to a two-dimensional and a three-dimensional groundwater flow and contaminant transport finite-element model to demonstrate the method. The CPU times for automatic differentiation were much faster than for the divided-differences method for both models, and somewhat slower than handwritten optimized analytic derivative code (written by *Chang et al.* [1992]) for the two-dimensional model. In both cases automatic differentiation produced exact derivatives.

2 AUTOMATIC DIFFERENTIATION AND ADIFOR

Automatic differentiation techniques rely on the fact that every function, no matter how complicated, is executed on a computer as a (potentially very long) sequence of elementary operations such as additions, multiplications, and elementary functions such as sin and cos. By applying the chain rule,

$$\frac{\partial}{\partial t} f(g(t))|_{t=t_o} = (\frac{\partial}{\partial s} f(s)|_{s=g(t_o)})(\frac{\partial}{\partial t} g(t)|_{t=t_o}), \tag{1}$$

over and over again to the composition of those elementary operations, one can compute derivate information of f exactly and in a completely mechanical fashion.

There are two canonical ways of propagating derivatives in automatic differentiation (*Griewank*, 1989). In the forward mode, we propagate derivatives of intermediate values with respect to the input parameters. In the reverse mode, we maintain derivatives of the final result with respect to intermediate quantities. The reverse mode is closely related to the previously mentioned adjoint approach. As a (very rough) rule of thumb, the run time and storage requirements of the forward mode are linear in the number of model inputs of interest, while the run time of the reverse mode is linear in the number of model outputs of interest w.r.t. differentiation. The storage requirements of the reverse mode are harder to assess and may be substantial because one must remember or recompute every intermediate value that nonlinearly impacts the final result.

There have been various implementations of automatic differentiation, and an extensive survey was prepared by *Juedes* (1991). Recently, new approaches to automatic differentiation have been explored in the ADIFOR (*Bischof et al.,* 1992) and ODYSSEE (*Rostaing et al.,* 1993) projects. Both tools transform Fortran code, applying the rules of automatic differentiation, and generating new Fortran code that, when executed, computes derivatives without the overhead associated with "tape interpretation" schemes. We used the ADIFOR tool in our experiments, which, as we will describe shortly, mainly employs the forward mode. In contrast, ODYSSEE employs the reverse mode.

ADIFOR provides automatic differentiation for programs written in Fortran 77. Given a Fortran subroutine (or collection of subroutines) describing a "function," and an indication of which variables in parameter lists or common blocks correspond to "independent" and "dependent" variables with respect to differentiation, ADIFOR produces portable Fortran 77 code that allows the computation of the derivatives of the dependent variables with respect to the independent ones.

ADIFOR accepts almost all of Fortran 77, in particular, arbitrary calling sequences, nested subroutines, common blocks, and equivalences. The ADIFOR-generated code tries to preserve vectorization and parallelism in the original code and employs a consistent subroutine-naming scheme that allows for code tuning, the exploitation of domain-specific knowledge, and the exploitation of vendor-supplied libraries.

ADIFOR employs a hybrid forward/reverse mode approach to generating derivatives. For each assignment statement, it generates code for computing the partial derivatives of the result with respect to the variables on the right-hand side using the reverse mode approach, and then employs the forward mode to propagate overall derivatives. For example, the statement

$$\mathtt{y} = \mathtt{x(1)} * \mathtt{x(2)} * \mathtt{x(3)} * \mathtt{x(4)} * \mathtt{x(5)}$$

gets transformed into the code shown in Figure 1. Note that none of the common subexpressions $x(i) * x(j)$ is recomputed in the reverse mode section.

ADIFOR-generated code can be used in various ways (*Bischof and Hovland,* 1991): Instead of simply producing code to compute the Jacobian J, ADIFOR produces code to compute $J * S$, where the "seed matrix" S is initialized by the user. So if S is the identity, ADIFOR computes the full Jacobian, and if S is just a vector, ADIFOR computes the product of the Jacobian by a vector. "Compressed" versions of sparse Jacobians can be computed by exploiting the same graph-coloring techniques that are used for divided-differences

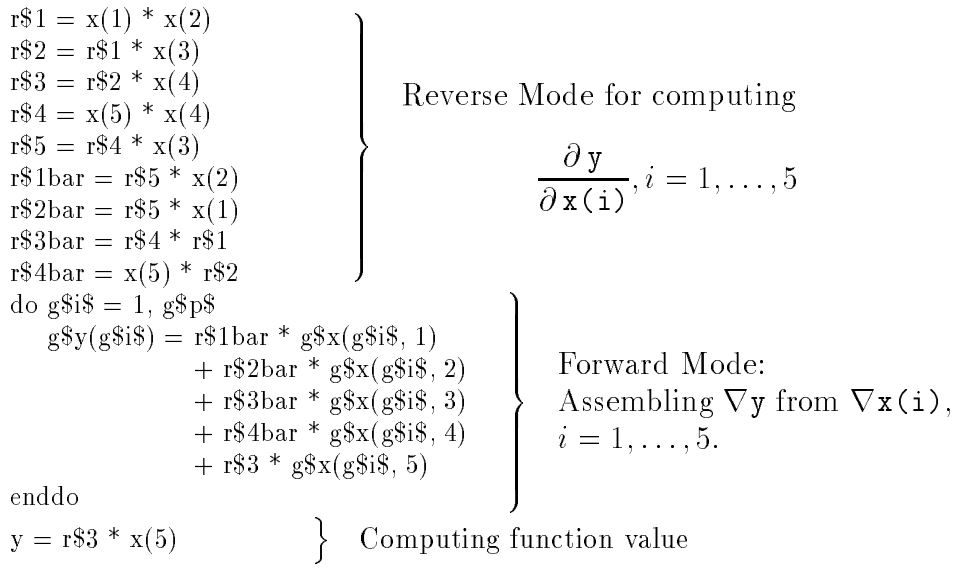

Figure 1: Sample of ADIFOR-generated Code

approximations of sparse Jacobians. In *Bischof et al.* [1994] it is also shown how one can use the flexibility of the ADIFOR interface to "stripmine" Jacobian computations and exploit parallelism to decrease turnaround time for derivative computations.

3 DIFFERENTIATED MODELS: TWO- AND THREE-DIMENSIONAL FLOW AND TRANSPORT

3.1 ISOQUAD: 2D Groundwater Flow and Transport Model

ISOQUAD is a two-dimensional (vertical dimension averaged) Galerkin finite-element model of groundwater transient flow and transport (see *Pinder and Frind* [1972], and *Pinder and Gray* [1977]). ISOQUAD is written in Fortran 77 and is on the order of two thousand lines of code. The model assumes the aquifer is confined but does allow for leakage. The model has been used extensively in optimal groundwater remediation design research (*Chang et al.*, 1992, and *Whiffen and Shoemaker*, 1993). Analytic expressions that can be coded for the derivatives of ISOQUAD are provided by *Chang et al.* (1992). These hand-coded, validated derivatives have been optimized for performance and allow a comparison and validation of ADIFOR-generated derivatives with divided-differences derivatives.

The model ISOQUAD is used in its implicit time-stepping mode, on a small mesh of 77 nodes for comparison of derivatives. The model has as input the pumping rates of wells located on m computational nodes. Outputs include contaminant concentrations and hydraulic head values at each of n active nodes for each time step in the simulation. Automatic differentiation was used to obtain the following derivatives:

$$\frac{\partial c_{t+1}}{\partial h_t} \quad \epsilon \ \Re^{n \times n} \tag{2}$$

$$\frac{\partial c_{t+1}}{\partial q_t} \quad \epsilon \ \Re^{n \times m} \tag{3}$$

$$\frac{\partial h_{t+1}}{\partial q_t} \quad \epsilon \ \Re^{n \times m}. \tag{4}$$

The vector $c_t \ \epsilon \ \Re^n$ is the value of the contaminant concentration at time t, the vector $h_t \ \epsilon \ \Re^n$ is the hydraulic head at time t, and the vector $q_t \ \epsilon \ \Re^m$ is the pumping rate at each computational node at time step t. The evaluation of all three derivatives, (2)–(4), together is considered one derivative evaluation. In this example, there are 126 independent variables, corresponding to a hydraulic head value and a contaminant concentration value at each of 63 active nodes.

ADIFOR-generated code produced derivatives that agreed with the validated handwritten code to the order of the machine precision, but executed in much less time than the (imprecise) divided-differences method. In particular, for a 77-element mesh, ADIFOR-generated code calculated derivatives (2)–(4) in about the time it would take to run the original simulation model 17 times. The same derivatives using the divided-differences approach require 126 simulations, one for each independent variable. Automatic differentiation code was somewhat slower than the optimized handwritten code by *Chang et al.* [1992], which requires about the same time as 5 simulations.

3.2 TLS3D: 3D Groundwater Transport Model

TLS3D is a model of the three-dimensional advection diffusion equation. TLS3D employs a Taylor least-squares finite-element procedure to solve the unsteady advection diffusion equation. The model uses a three-dimensional serendipity Hermite element for an eight-node hexahedron. For a complete description of the procedure and three-dimensional model, see *Park and Liggett* (1991). The code provided to the authors was written in Fortran 77 and is approximately 2300 lines in length.

The model is used on three-dimensional mesh ranging in size from 3 to 32 rectangular box elements to test the scalability of the code generated by ADIFOR. The model has as input the three-dimensional flow field; velocity vectors, v_t, for each simulation time step, t; and an initial contaminant concentration at each node, c_0. Model output is single-species contaminant concentrations at each active node, c_t, for each simulation time step t. Automatic differentiation was used to evaluate the derivative

$$\frac{\partial c_{t+1}}{\partial v_t} \quad \epsilon \ \Re^{n \times 3n}. \tag{5}$$

Again, automatic differentiation provided a code that calculated model derivatives exactly in much less time than the divided-differences method. Four different-sized discretizations consisting of 4, 7, 16, and 32 three-dimensional elements were used to compare derivative performances.

We obtained the following results on a SPARCstation 10 and a single node of the IBM SP1 parallel computer. The number of independent variables equals three times the number of nodes in the model. The columns labeled $f(x)$ show the run time (in seconds) of the simulation; the columns labeled $\frac{df(x(i))}{f(x)}$ show the average time required for the derivatives with respect to one independent variable, when the ADIFOR-generated code is used to generate derivatives with respect to 48 independent variables at a time.

# Elements (#independents)	SPARCstation 10		SP1 Node	
	$f(x)$	$\frac{df(x(i))}{f(x)}$	$f(x)$	$\frac{df(x(i))}{f(x)}$
4 (48)	2.0	0.20	0.27	0.17
7 (96)	4.5	0.21	0.51	0.21
16 (243)	30.9	0.20	4.72	0.15
32 (432)	77.9	0.19	13.26	0.14

We see that AD is more than 5 and 7 times faster than divided-differences approximations on the SPARCstation 10 and the SP1 node, respectively. No handcoded analytic derivatives of TLS3D are available for comparison.

4 CONCLUSIONS

We found that automatic differentiation can provide accurate and efficient derivative codes for the complex finite-element codes ISOQUAD and TLS3D. In particular, we found ADIFOR generated codes that calculated derivatives in approximately 13% of the time required by divided differences for the two-dimensional model ISOQUAD, and between 14% and 21% of the time required by divided differences for the three-dimensional model, TLS3D. A more detailed account of these results will be given in a forthcoming paper.

As a result of our investiations, we believe that automatic differentiation can facilitate model changes by providing a mechanism for generating accurate and efficient derivative codes for models with very little human effort. Automatic differentiation technology will greatly accelerate the transfer of general techniques developed for using water resource computer models, such as optimal design, sensitivity analysis, and inverse modeling problems to field problems. Automatic differentiation can also accelerate the rate at which algorithm and model development and testing can occur by providing exact sensitivity information that may not otherwise be available.

ACKNOWLEDGMENTS

Bischof's work was supported by the Office of Scientific Computing, U.S. Department of Energy, under Contract W-31-109-Eng-38, and by the National Aerospace Agency under Purchase Order L25935D. Primary support for Whiffen was in the form of a fellowship awarded by the Department of Energy Computational Science Graduate Fellowship Program. Shoemaker was funded in part by a grant from NSF (ASC 8915326) and the IBM Corporation. Carle was supported by the National Aerospace Agency under Cooperative Agreement No. NCCW-0027 and the National Science Foundation through NSF Cooperative Agreement No. CCR-9120008. Ross was supported by the Office of Scientific Computing, U.S. Department of Energy, under Contract W-31-109-Eng-38, and the National Science Foundation through NSF Cooperative Agreement No. CCR-8809615. TLS3D code and assistance was provided by Dr. Nam-Sic Park and Dr. James A. Liggett. The groundwater remediation optimal control model was modified from Fortran code provided by L. Chang.

References

[1] Ahlfeld, D. P., J. M. Mulvey, G. F. Pinder, and E. F. Wood, Contaminated Groundwater Remediation Design Using Simulation, Optimization, and Sensitivity Theory 1. Model Development, *Water Resour. Res., 24*(3), 431–441, 1988a.

[2] Ahlfeld, D. P., J. M. Mulvey, G. F. Pinder, and E. F. Wood, Contaminated Groundwater Remediation Design Using Simulation, Optimization, and Sensitivity Theory 2. Analysis of a Field Site, *Water Resour. Res., 24*(3), 443–452, 1988b.

[3] Averick B., J. Moré, C. Bischof, A. Carle, and A. Griewank, Computing large sparse Jacobian matrices using automatic differentiation. Preprint MCS–P348–0193, Mathematics and Computer Science Division, Argonne National Laboratory, 1993. Accepted for publication in SIAM Journal of Scientific Computing.

[4] Bischof, C. , A. Carle, G. Corliss, A. Griewank, and P. Hovland, ADIFOR: Generating derivative codes from Fortran programs. *Scientific Programming, 1*(1):11–29, 1992.

[5] Bischof, C., L. Green, K. Haigler, and T. Knauff, Parallel calculation of sensitivity derivatives for aircraft design using automatic differentiation, Preprint MCS-P419-0294, Mathematics and Computer Science Division, Argonne National Laboratory, 1994.

[6] Bischof, C., and P. Hovland, Using ADIFOR to compute dense and sparse Jacobians. ADIFOR Working Note #2, ANL/MCS–TM–158, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., 1991.

[7] Chang, L.-C., C. A. Shoemaker, and P. L.-F. Liu, Application of a constrained optimal control algorithm to groundwater remediation, *Water Resour. Res., 28*(12), 3157–3173, 1992.

[8] Gorelick, S. M., C. I. Voss, P. E. Grill, W. Murray, M. A. Saunders, and M. H. Wright, Aquifer reclamation design: The use of contaminant transport simulation combined with nonlinear programming, *Water Resour. Res., 20*(4), 415–427, 1984.

[9] Griewank, A., On automatic differentiation. In *Mathematical Programming: Recent Developments and Applications*, pages 83–108, Amsterdam, 1989. Kluwer Academic Publishers.

[10] Juedes, D., A taxonomy of automatic differentiation tools. In Andreas Griewank and George Corliss, editors, *Proceedings of the Workshop on Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, 315–329, SIAM, Philadelphia, 1991.

[11] Leitmann, G., *The Calculus of Variations and Optimal Control*, Vol. 20 of *Mathematical Concepts and Methods in Science and Engineering*, Plenum Press, New York, 1981.

[12] Park, N. -S., and J. A. Liggett, Application of Taylor-least squares finite element to three-dimensional advection-diffusion equation, *Int. J. Numer. Methods Fluids, 13*, 759–773, 1991

[13] Pinder, G. F., and Frind, Application of Galerkin's procedure to aquifer analysis, *Water Resour. Res.,* *8*(1), 108–120, 1972.

[14] Pinder, G. F., and W. G. Gray, *Finite Element Simulation in Surface and Subsurface Hydrology,* Academic Press, Orlando, Florida, 1977.

[15] Rostaing, N., S. Dalmas, and A. Galligo, Automatic differentiation in Odyssee. *Tellus,* *45a*(5):558–568, October 1993.

[16] Whiffen, G. J., and C. A. Shoemaker, Nonlinear weighted feedback control of groundwater Remediation under uncertainty, *Water Resour. Res.,* *29*(9), 3277–3289, 1993.