Preprint
UCRL-JC-137728

Sensitivity Analysis Using Parallel ODE Solvers and Automatic Differentiation in C: SensPVODE and ADIC

S. L. Lee, P. D. Hovland

This article was submitted to
3rd International Conference/Workshop on Automatic Differentiation: From Simulation to Optimization.
Nice, France
June 19-23, 2000

U.S. Department of Energy

Lawrence
Livermore
National
Laboratory

September 15, 2000

Sensitivity Analysis Using Parallel ODE Solvers and Automatic Differentiation in C: SensPVODE and ADIC

Steven L. Lee
Paul D. Hovland

ABSTRACT PVODE is a high-performance ordinary differential equation solver for the types of initial value problems (IVPs) that arise in large-scale computational simulations. Often, one wants to compute sensitivities with respect to certain parameters in the IVP. We discuss the use of automatic differentiation (AD) to compute these sensitivities in the context of PVODE. Results on a simple test problem indicate that the use of AD-generated derivative code can reduce the time to solution over finite difference approximations.

1 Background

In complicated, large-scale computational simulations, the governing equations can often be spatially discretized and then numerically solved as a system of ordinary differential equation (ODE) or differential-algebraic equation (DAE) initial value problems. PVODE [BH99] and IDA [HT99] are powerful, parallel codes for solving these types of ODEs and DAEs, respectively. The codes are written in C and use MPI to achieve parallelism and portability. Typically, the equations contain parameter values (e.g., chemical reaction rates) that are not precisely known. In analyzing the simulations, the scientist would like to know which parameters are most influential in affecting the behavior of the simulation. Such sensitivity information is useful because it identifies which parameters will require precise measurements if the simulation results are to be made more accurate. This article summarizes preliminary work in which automatic differentiation (AD) is being used with PVODE to create a solver that computes sensitivity information for ODE systems.

In computing sensitivities for ODEs, one is interested in solving

$$y'(t) = f(t, y, p), \quad y(t_0) = y_0(p), \quad y \in \mathbf{R}^N, \quad p \in \mathbf{R}^m, \tag{1.1}$$

where the solution vector $y(t)$ depends upon an additional vector of parameters p, and the *sensitivities* are defined as

$$s_i(t) = \frac{\partial y(t,p)}{\partial p_i},\ i = 1, \cdots, m.$$

One approach for computing these sensitivities is to apply AD techniques to the entire PVODE solver. However, PVODE is a variable-stepsize, variable-order solver and, for this situation, Eberhard and Bischof [EB99] have demonstrated that AD may compute unexpected derivative values unless an a posteriori correction is applied. In contrast to such a "black-box" approach, it is often superior to couple the use of AD with some insight into the computational requirements of the problem. To do this, we formally differentiate the original ODE (1.1) with respect to each component p_i of p. Thus, we obtain the sensitivity ODEs

$$s_i'(t) = \frac{\partial f}{\partial y} s_i(t) + \frac{\partial f}{\partial p_i},\ s_i(t_0) = \frac{\partial y_0(p)}{\partial p_i},\ i = 1, \cdots, m. \qquad (1.2)$$

The initial sensitivity vector $s_i(t_0)$ is either all zeros (if p_i occurs only in f), or has nonzeros according to how $y_0(p)$ depends on p_i. The time integration of $y'(t)$ and each $s_i'(t)$ can be accomplished by solving an ODE system of size $N(m+1)$, where

$$Y = \begin{pmatrix} y(t) \\ s_1(t) \\ \vdots \\ s_m(t) \end{pmatrix} \quad \text{and} \quad F(t,Y,p) = \begin{pmatrix} f(t,y,p) \\ \frac{\partial f}{\partial y} s_1(t) + \frac{\partial f}{\partial p_1} \\ \vdots \\ \frac{\partial f}{\partial y} s_m(t) + \frac{\partial f}{\partial p_m} \end{pmatrix}.$$

The new ODE-sensitivity IVP to be solved is simply

$$Y'(t) = F(t,Y,p), \quad Y(t_0) = Y_0(p), \qquad (1.3)$$

and each $s_i'(t)$ can be evaluated by computing $\frac{\partial f}{\partial y} s_i(t) + \frac{\partial f}{\partial p_i}$ via AD, or by approximating their sum via finite differences.

In general, the sensitivities of the ODE problem can be solved for in a variety of ways. For example, the ODE problems can be decoupled: compute and store the solution $y(t)$ in advance; then use interpolation, along with AD or finite differences, to evaluate $s_i'(t)$ wherever needed by an ODE solver. If the same ODE solver is used for (1.1) and (1.2), the effort needed to integrate them is often comparable since the ODEs have the same Jacobian matrix $\frac{\partial f}{\partial y}$ and therefore the same stiffness properties. For a comprehensive review of methods for computing sensitivity information in ODE systems, see [RKD83].

SensPVODE [LHB00] is a variant of PVODE that *simultaneously* computes the solution and the sensitivities in the augmented ODE system (1.3). Also, for many large-scale applications, implicit time integration methods are required. Several papers describe how to modify Newton's method for efficiently solving the nonlinear systems that arise at each timestep [FTB97, MP96]. Also, we note that the sensitivity ODEs (1.2) are linear in $s_i(t)$, even if the original ODE (1.1) is nonlinear. This observation is significant in the next section as we discuss the need to properly scale the sensitivities that we compute.

2 Scaled Sensitivities Using Finite Differences

Several observations motivate our modifications to the sensitivity ODEs (1.2). First, the units for the ODE solution, $[y(t)]$, and the units for the sensitivity vectors, $[s_i(t)]$, do not match. This mismatch in units can lead to scaling problems, especially when using finite difference methods. Fortunately, the issue is easily remedied. In particular, the sensitivity vectors have units of $[y]/[p_i]$. For $y(t)$ and the sensitivities to share the same units, the linearity of the sensitivity ODEs (1.2) allows us to multiply the sensitivities by their respective parameter values to obtain the *scaled* sensitivity ODEs

$$w_i'(t) = \frac{\partial f}{\partial y} w_i(t) + \overline{p}_i \frac{\partial f}{\partial p_i}, \tag{2.4}$$

where

$$w_i(t) = \overline{p}_i s_i(t),$$

and $\overline{p}_i$ is a nonzero constant that is dimensionally consistent with p_i. Typically $\overline{p}_i = p_i$. In general, the scale factor $\overline{p}_i$ can be any nonzero multiple of p_i, and this can sometimes be used to create a well-scaled problem for the ODE variables and sensitivities.

To improve the accuracy of estimating the scaled sensitivity derivatives in (2.4), SensPVODE has an option that applies centered differences to each term separately:

$$\frac{\partial f}{\partial y} w_i \approx \frac{f(t, y + \delta_y w_i, p) - f(t, y - \delta_y w_i, p)}{2\,\delta_y} \tag{2.5}$$

and

$$\overline{p}_i \frac{\partial f}{\partial p_i} \approx \frac{f(t, y, p + \delta_i \overline{p}_i e_i) - f(t, y, p - \delta_i \overline{p}_i e_i)}{2\,\delta_i}. \tag{2.6}$$

As is typical for finite differences, the proper choice of perturbations δ_y and δ_i is a delicate matter. Our recommended value for δ_y and δ_i takes into account several problem-related features: the relative ODE error tolerance RTOL, the machine unit roundoff $\epsilon_{\mathsf{machine}}$, and the weighted root-mean-square (RMS) norm of the scaled sensitivity w_i. We then define

$$\delta_i = \sqrt{\max(\mathrm{RTOL}, \epsilon_{\mathsf{machine}})} \quad \text{and} \quad \delta_y = \frac{1}{\max(||w_i||, 1/\delta_i)}. \tag{2.7}$$

The terms $\epsilon_{\text{machine}}$ and $1/\delta_i$ are included as divide-by-zero safeguards in case RTOL $= 0$ or $||w_i|| = 0$. Roughly speaking (i.e., if the safeguard terms are ignored), δ_i gives a $\sqrt{\text{RTOL}}$ relative perturbation to parameter i, and δ_y gives a unit weighted RMS norm perturbation to y. Of course, the main drawback of this approach is that it requires four evaluations of $f(t, y, p)$.

A less costly technique for estimating scaled sensitivity derivatives is also based on centered differences. However, it uses the formula

$$w_i' = \frac{\partial f}{\partial y} w_i + \overline{p_i} \frac{\partial f}{\partial p_i} \approx \frac{f(t, y + \delta w_i, p + \delta \overline{p}_i e_i) - f(t, y - \delta w_i, p - \delta \overline{p}_i e_i)}{2\,\delta} \tag{2.8}$$

in which

$$\delta = \min(\delta_i, \delta_y).$$

If $\delta_i = \delta_y$, a Taylor series analysis shows that the sum of (2.5) and (2.6) and the value of (2.8) are equivalent to within $O(\delta^2)$. However, the latter approach is half as costly, since it requires only two evaluations of $f(t, y, p)$. To take advantage of this savings, it may also be desirable to use the latter formula when $\delta_i \approx \delta_y$. In [LHB00], we explore the possibility of allowing SensPVODE to select the finite difference formula based on how closely δ_i and δ_y agree.

In summary, the sensitivity version of PVODE is equipped with a variety of finite difference formulas for approximating the scaled sensitivity derivatives. However, for some problems, finite differences do not work. Typically, difficulties arise in applications where the solution components are very badly scaled. In addition to failure or accuracy problems, finite differences may be inefficient for functions $f(t, y, p)$ that are expensive to evaluate. Such shortcomings motivate the need for an efficient, exact, and (preferably) automated process for computing sensitivity derivatives within SensPVODE.

3 Scaled Sensitivities Using AD

Automatic differentiation must be nearly as easy to use as finite differences, or it will only be used when finite differences fail, if at all. Previous work [Cor92, LP99, FMM98, ABG+00, Ger00] has demonstrated that it is possible to automate the AD process by exploiting the existence of well-defined interfaces for the user's function implementing $f(t, y, p)$. This makes it easy to identify the independent and dependent variables and to properly initialize the AD-generated code.

Applying AD is complicated by the fact that the user's function is implemented in C with MPI parallelism [GLS94]. We are therefore adding support for MPI to the ADIC [BRM97] automatic differentiation tool, building on earlier work by Hovland [Hov97, HB98]. The use of C poses challenges from the standpoint of automation. PVODE, like many other numerical toolkits, allows the user to pass around application-specific data in a user-defined `struct`. As part of the AD process, it may be necessary to associate derivatives with some of the variables in this structure. To avoid aliasing problems, this generally implies changing the type of these variables [BRM97]. Thus, all code (not just the function) must be modified to use this new datatype. Our initial approach has been to circumvent this problem through the use of two data structures, one with derivatives and one without, copying data back and forth as necessary. To eliminate the overhead of copying, we plan to use a single data structure. This will necessitate applying ADIC to automatically modify the user code to use the new datatype.

4 Experimental Results

We applied SensPVODE to a simple test case, a two-species diurnal kinetics advection-diffusion system in two space dimensions. The PDEs can be written as

$$\frac{\partial c_i}{\partial t} = K_h \frac{\partial^2 c_i}{\partial x^2} + V \frac{\partial c_i}{\partial x} + \frac{\partial}{\partial y}\left(K_v(y) \frac{\partial c_i}{\partial y} \right) + R_i(c_1, c_2, t) \quad (i = 1, 2),$$

where the superscripts i are used to distinguish the chemical species. The reaction terms are given by

$$\begin{aligned} R_1(c_1, c_2, t) &= -q_1 c_1 c_3 - q_2 c_1 c_2 + 2 q_3(t) c_3 + q_4(t) c_2, \quad \text{and} \\ R_2(c_1, c_2, t) &= q_1 c_1 c_3 - q_2 c_1 c_2 - q_4(t) c_2; \end{aligned}$$

and $K_v(y) = K_0 \exp(y/5)$. The scalar constants for this problem are $K_h = 4.0 \times 10^{-6}$, $V = 10^{-3}$, $K_0 = 10^{-8}$, $q_1 = 1.63 \times 10^{-16}$, $q_2 = 4.66 \times 10^{-16}$, and $c_3 = 3.7 \times 10^{16}$. The diurnal rate constants are

$$\begin{aligned} q_i(t) &= \exp[-a_i / \sin \omega t] \quad \text{for } \sin \omega t > 0, \\ q_i(t) &= 0 \quad \text{for } \sin \omega t \leq 0, \end{aligned}$$

where $i = 3$ and 4, $\omega = \pi/43200$, $a_3 = 22.62$, and $a_4 = 7.601$. The time interval of integration is $[0, 86400]$, representing 24 hours measured in seconds.

The problem is posed on the square $0 \leq x \leq 20$, $30 \leq y \leq 50$ (all in km), with homogeneous Neumann boundary conditions. The PDE system is treated by central differences on a uniform mesh, with simple polynomial initial profiles. See [LHB00] for more details. For the purpose of sensitivity analysis, we identify the following 8 parameters associated with this problem: $p_1 = q_1$, $p_2 = q_2$, $p_3 = c_3$, $p_4 = a_3$, $p_5 = a_4$, $p_6 = K_h$, $p_7 = V$, and $p_8 = K_0$. In solving for (say) 5 sensitivities, we are computing the ODE solution together with the scaled sensitivities with respect to the first 5 parameters; that is, $y(t)$ and $w_1(t), \ldots, w_5(t)$.

In the numerical experiments that follow, we allowed the number of sensitivities to vary from 1 to 8. In computing the scaled sensitivity derivatives, we compared the use of AD against the finite difference strategies described in Section 2. Two centered difference strategies were examined: separate evaluations, based on the sum of (2.5) and (2.6); and a combined evaluation, given by (2.8). A forward difference method was also tested in which $\frac{\partial f}{\partial y} w_i(t)$ and $\overline{p}_i \frac{\partial f}{\partial p_i}$ are each approximated by forward differences. The results are summarized in Figures 4.1 and 4.2.

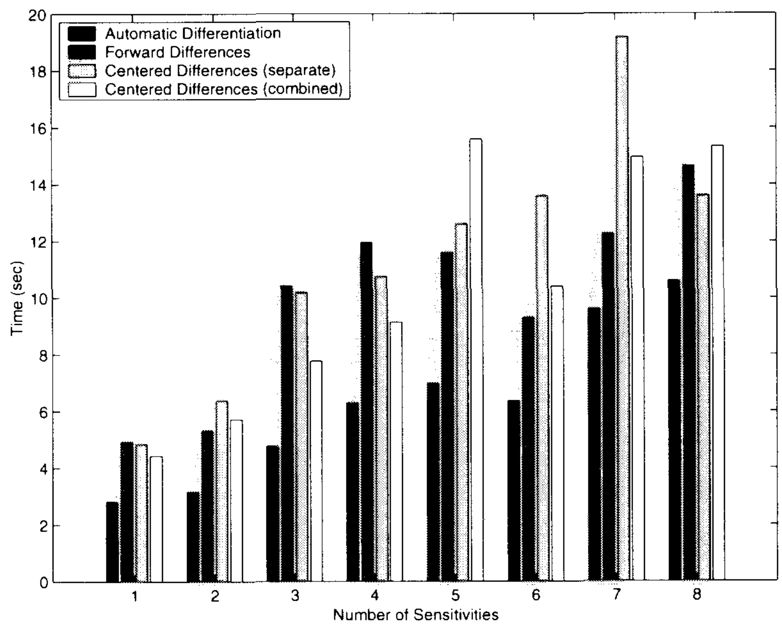

FIGURE 4.1. Comparison of performance for various derivative-computation strategies. Results are the average of three runs on 4 processors of an SGI Origin 2000.

Although the present framework for using AD includes some inefficiencies such as the copying of data, Figure 4.1 shows that AD is still markedly faster than each of the three finite difference methods. As shown in Figure 4.2, this advantage can be attributed primarily to the reduced number of time steps. The increased accuracy of the analytic derivatives provided by AD results in larger time steps in the variable-stepsize, variable-order solver.

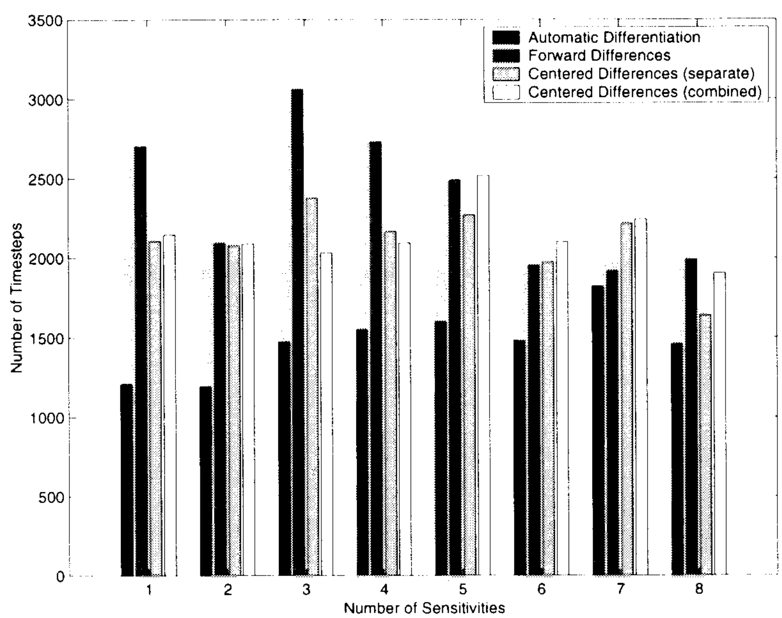

FIGURE 4.2. Number of timesteps for various derivative-computation strategies. Results are the average of three runs on 4 processors of an SGI Origin 2000.

5 Conclusions and Future Work

SensPVODE provides an efficient and easy-to-use mechanism for computing the sensitivities for simulations that use the PVODE parallel ODE solver. Results for a simple problem indicate that derivatives computed using AD provide performance superior to finite difference approximations. We plan to examine whether this performance advantage holds for more complex problems, and how well this advantage scales with respect to the number of processors used.

Future work also includes developing a mechanism that eliminates the need to copy data from one structure to another, while preserving the ease of use of the current implementation. This issue is related to those faced in the use of AD with other numerical toolkits such as PETSc and TAO [ABG+00], and we therefore hope to benefit from lessons learned in those projects. In addition, the algorithms used by SensPVODE require the solution of linear systems with multiple right-hand side vectors [LHB00, MP96]. A similar situation arises when one differentiates through a linear or nonlinear solver [BB98, STG+94, HNRS98]. Thus, we expect to leverage other work [BBH00] in the development of block solvers for systems with multiple right-hand sides. All of these developments should increase the efficiency of sensitivity computations using SensPVODE and ADIC. Finally, we note that the SensPVODE package is available for general distribution. Interested users should contact Alan Hindmarsh (alanh@llnl.gov) and Steven Lee (slee@llnl.gov).

Acknowledgments

This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48 and was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract W-31-109-Eng-38.

We thank Gail Pieper for proofreading a draft manuscript and Peter Brown, Alan Hindmarsh, and Linda Petzold for valuable advice regarding sensitivity analysis of ODEs and DAEs.

6 References

[ABG+00] Jason Abate, Steve Benson, Lisa Grignon, Paul Hovland, Lois McInnes, and Boyana Norris. Integrating automatic differentiation with object-oriented toolkits for high-performance scientific computing. Technical Report ANL/MCS-P820-0500, Mathematics and Computer Science Division, Argonne National Laboratory, 2000. Submitted to AD2000.

[BB98] M.C. Bartholomew-Biggs. Using forward accumulation for automatic differentiation of implicitly-defined functions. *Computational Optimization and Applications*, 9:65–84, 1998.

[BBH00] Christian Bischof, Martin Bücker, and Paul Hovland. On combining computational differentiation and toolkits for parallel scientific computing. Technical Report ANL/MCS-P797-0200, Mathematics and Computer Science Division, Argonne National Laboratory, 2000. To appear in Proceedings of EuroPar 2000.

[BH99] G. D. Byrne and A. C. Hindmarsh. PVODE, an ODE solver for parallel computers. *Int. J. High Perf. Comput. Appl.*, 13:354–365, 1999.

[BRM97] Christian Bischof, Lucas Roh, and Andrew Mauer. ADIC — An extensible automatic differentiation tool for ANSI-C. *Software – Practice and Experience*, 27(12):1427–1456, 1997.

[Cor92] George F. Corliss. ADIFOR case study : VODE + ADIFOR. Technical Memorandum ANL/MCS-TM-168, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, 1992.

[EB99] P. Eberhard and C. Bischof. Automatic differentiation of numerical integration algorithms. *Mathematics of Computation*, 68:717–731, April 1999.

[FMM98] M. C. Ferris, M. Mesnier, and J. J. Moré. NEOS and Condor: Solving optimization problems over the Internet. Preprint ANL/MCS-P708-0398, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, 1998.

[FTB97] W. F. Feehery, J. E. Tolsma, and P. I. Barton. Efficient sensitivity analysis of large-scale differential-algebraic equations. *Appl. Numer. Math.*, 25:41–54, 1997.

[Ger00] Michael Gertz, 2000. Personal communication.

[GLS94] William Gropp, Ewing Lusk, and Anthony Skjellum. *Using MPI – Portable Parallel Programming with the Message Passing Interface*. MIT Press, Cambridge, 1994.

[HB98] Paul Hovland and Christian Bischof. Automatic differentiation of message-passing parallel programs. In *Proceedings of the First Merged International Parallel Processing Symposium and Symposium on Parallel and Distributed Processing*, Los Alamitos, CA, 1998. IEEE Computer Society Press.

[HNRS98] Paul Hovland, Boyana Norris, Lucas Roh, and Barry Smith. Developing a derivative-enhanced object-oriented toolkit for scientific computations. In *Proceedings of the SIAM Workshop on Object Oriented Methods for Inter-operable Scientific and Engineering Computing*, pages 129–137. SIAM, Oct 1998.

[Hov97] Paul D. Hovland. *Automatic Differentiation of Parallel Programs*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL, May 1997.

[HT99] Alan C. Hindmarsh and Allan G. Taylor. User documentation for IDA, a differential-algebraic equation solver for sequential and parallel computers. Technical Report UCRL-MA-136910, Lawrence Livermore National Laboratory, 1999.

[LHB00] Steven L. Lee, Alan C. Hindmarsh, and Peter N. Brown. User documentation for SensPVODE, a variant of PVODE for sensitivity analysis. Technical Report UCRL-MA-140211, Lawrence Livermore National Laboratory, 2000.

[LP99] Shengtai Li and Linda Petzold. Design of new DASPK for sensitivity analysis. Technical report, University of California at Santa Barbara, 1999.

[MP96] T. Maly and L. Petzold. Numerical methods and software for sensitivity analysis of differential-algebraic systems. *Applied Numerical Mathematics*, 20:57–79, February 1996.

[RKD83] H. Rabitz, M. Kramer, and D. Dacol. Sensitivity analysis in chemical kinetics. *Ann. Rev. Phys. Chem.*, 34:419–461, 1983.

[STG+94] L. Sherman, A. Taylor, L. Green, P. Newman, G. Hou, and V. Korivi. First- and second-order aerodynamic sensitivity derivatives via automatic differentiation with incremental iterative methods. In *Proceedings of the 5th AIAA/NASA/-USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, AIAA 94-4262, pages 87–120. American Institute of Aeronautics and Astronautics, 1994.

This work was performed under the auspices of the U.S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.