Derivative techniques for forward sensitivity analysis in time-dependent problems

Steven L. Lee [a] and Paul D. Hovland [b]

[a] *Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, P.O. Box 808, Livermore, CA 94551*

[b] *Mathematics and Computer Science Division, Argonne National Laboratory, 9700 S. Cass Ave., Argonne, IL 60439*

Abstract

The complex-step derivative approximation technique is a highly accurate and convenient method for computing directional derivatives within simulation codes. The method is similar to finite-difference approximations and automatic differentiation techniques in terms of ease of implementation and accuracy, respectively. We examine the performance and accuracy of finite-difference, automatic differentiation, and complex-step techniques in analyzing the solutions to physical systems modeled as initial-value problems in ordinary differential equations (ODEs). In particular, we investigate the use of these derivative techniques in computing the sensitivity of the ODE solutions with respect to various model parameters. In doing so, we identify the strengths and weaknesses of the derivative techniques and find that the derivify implementation of automatic differentiation can be a simple, accurate, and cost-effective means of computing directional derivatives for forward sensitivity analysis.

Key words: sensitivity analysis, time-dependent differential equations, forward mode, automatic differentiation, complex-step derivative approximation, finite differences

Email addresses: slee@llnl.gov (Steven L. Lee), hovland@mcs.anl.gov (Paul D. Hovland).

1 Introduction

Large-scale scientific simulations often model complex physical systems by formulating and solving them as initial-value problems in ordinary differential equations (ODEs) or differential-algebraic equations. Furthermore, these mathematical models usually contain scalar parameters related to important features of the simulation, such as chemical reaction rates, material opacities, and problem coefficients. The ability to compute the sensitivity of the simulation results with respect to the model parameters is valuable for several reasons. For example, sensitivity information can be used to rank the model parameters from most to least influential, design improved experiments, and quantify uncertainty in the simulation results [1]. Other applications include parameter estimation, optimization, and process sensitivity studies [2].

In this paper, we carry out sensitivity analysis experiments on partial differential equation (PDE) model problems that have been discretized in space using finite differences and solved using the code CVODES [3]. This ODE solver uses linear multistep methods for the time integration of the model equations and, if specified, also integrates the sensitivity equations that can be formally derived by differentiating the model problem with respect to selected parameters. For large or complicated ODE systems, a formal derivation of the latter equations can be problematic. The differentiation of a large system of equations can be a tedious and error-prone process, even with the possible assistance of symbolic differentiation software. Another option is the use of automatic differentiation software [4,5], a technique for obtaining exact derivatives to within rounding error. By default and mainly for user convenience, CVODES and similar sensitivity-capable codes [6–11] approximate the sensitivity equations internally using finite difference methods.

Recently, the complex-step derivative (CSD) approximation method [12] has been demonstrated to be effective for sensitivity analysis in computational fluid dynamics [13] and aerospace engineering applications [14–16], and in computing directional derivatives in simulations using pseudospectral methods [17]. This method has the accuracy of the analytic and automatic differentiation techniques yet also provides ease of implementation comparable to finite differences. The most direct implementation of CSD involves the use of complex numbers and complex-valued function evaluations in which perturbations are made to the imaginary components of the problem variables. In the aerospace studies, CSD was used to obtain derivatives of the entire simulation code with respect to a few aerodynamic parameters. For our purposes, we need only apply CSD techniques to obtain the sensitivity equations needed for use within the ODE solver. The adverse effects (if any) of using low-order finite-difference approximations for the sensitivity equations can be difficult to assess. To investigate this issue, we focus our comparison on the various derivative techniques and their relative accuracy and performance in computing sensitivities for time-dependent PDE problems.

2 Sensitivity Analysis for ODEs

In computing sensitivities for ODE initial-value problem (IVPs), let

$$\dot{y} = f(t, y, p), \quad y(t_0, p) = y_0(p), \quad y \in \mathbf{R}^N, \quad p \in \mathbf{R}^{N_p}, \tag{1}$$

where the solution vector y depends on time t and a vector of N_p scalar parameters in p. The sensitivities are then given as

$$s_i(t, p) = \frac{\partial y(t, p)}{\partial p_i}, \quad i = 1 \dots N_p,$$

which is the (time-dependent) first-order sensitivity of the solution with respect to each parameter. The ODE-IVP that describes the time-dependent behavior of the sensitivities is obtained by differentiating the original problem $\dot{y} = f(t, y, p)$ with respect to each parameter

$$\dot{s}_i = J s_i + \frac{\partial f}{\partial p_i}, \qquad s_i(t_0, p) = \frac{\partial y_0(p)}{\partial p_i}, \tag{2}$$

where $J = \frac{\partial f}{\partial y}$ is the Jacobian matrix for the original ODE system (1) and we note that the sensitivity ODEs are linear in s_i. By combining the solution, sensitivities, and ODEs, we get an augmented ODE system $\dot{Y} = F(t, Y, p)$ where

$$\dot{Y}(t) \equiv \begin{pmatrix} \dot{y}(t) \\ \dot{s}_1(t) \\ \vdots \\ \dot{s}_{N_p}(t) \end{pmatrix} \quad \text{and} \quad F(t, Y, p) \equiv \begin{pmatrix} f \\ J s_1 + \frac{\partial f}{\partial p_1} \\ \vdots \\ J s_{N_p} + \frac{\partial f}{\partial p_{N_p}} \end{pmatrix}. \tag{3}$$

The augmented Y is larger than y by a factor of $(1 + N_p)$ so, for large-scale systems, we typically consider computing sensitivities with respect to only a few (usually $N_s \ll N_p$) parameters. The right-hand side F involves the use of the system Jacobian matrix and partial derivatives and can be problematic to derive by hand. However, it can be readily estimated by finite differences or computed exactly by more advanced methods. In the next section, we focus on the use of finite differences, automatic differentation, and complex-step derivative approximation techniques for evaluating F and assess their respective accuracy and performance in solving model problems by using CVODES.

The time integration of the augmented solution-sensitivity ODE system in Eq. (3) can be accomplished by using either the Adams-Moulton or backward differentiation formula (BDF) as implemented in CVODES. Both formulas allow their step size and method order to vary

based on the behavior of the computed solutions. For brevity we restrict our discussion to BDFs and consider the use of the first-order backward Euler method

$$\frac{Y_n - Y_{n-1}}{h_n} = F(t_n, Y_n, p), \tag{4}$$

where $h_n = t_n - t_{n-1}$ is the current step size. In order to obtain the solution and sensitivities at time t_n, the nonlinear system

$$0 = G(Y_n) \equiv Y_n - h_n F(t_n, Y_n, p) - Y_{n-1} \tag{5}$$

must be solved for Y_n. In using a so-called staggered corrector method, a Newton iteration is used to solve first for y and then for the sensitivities s_1, s_2, and so on. The Newton iteration matrix $I - h_n J$ is the same for each linear system that is solved in updating the solution and the sensitivities at t_n. Note that since the Newton matrix remains the same within each time step, the preconditioner or linear system solver can be reused also. For higher-order BDFs, the Newton iteration matrix has the form $I - h_n \beta_0 J$, where β_0 is a scalar that depends on the order of the BDF method.

In order to assess convergence of iterative methods and monitor error estimates within CVODES, a weighted root-mean-square (WRMS) norm is used for all error-like quantities. The weights $W_{n,i}$ are based on the current solution Y_n and the relative and absolute error tolerances, RTOL and ATOL, specified by the user so that

$$W_{n,i} = \text{RTOL} \cdot |Y_{n,i}| + \text{ATOL}_i \tag{6}$$

for $i = 1, \ldots, (1+N_s)N$. Unless indicated otherwise, all norms in our discussions are WRMS norms. For complete details on CVODES, see [3,18].

3 Derivative Techniques

Diverse techniques are available for computing the directional derivatives required by CVODES in performing sensitivity analysis. The main criteria for comparison are their respective accuracy, cost, and ease of implementation, and we survey those aspects below.

3.1 Finite Differences

Finite-difference methods vary in their cost and accuracy in estimating sensitivity derivatives. One of the approximations provided by CVODES is

$$\frac{\partial f}{\partial y}s_i + \frac{\partial f}{\partial p_i} \approx \frac{f(t, y + \delta s_i, p + \delta pe_i) - f(t, y - \delta s_i, p - \delta pe_i)}{2\delta}, \tag{7}$$

in which a centered difference and two function evaluations are used to achieve estimates with $O(\delta^2)$ accuracy. Although the method is easy to implement, developing a robust heuristic for the proper selection of δ can be difficult. This difficulty is the main weakness of the technique. The current heuristic takes into account several problem-related features: the relative ODE error tolerance RTOL, the machine unit roundoff error $\epsilon_{\text{machine}}$, and the norm of s_i.

3.2 Automatic Differentiation

Automatic differentiation (AD) is a technique for augmenting computer subroutines with instructions for the computation of derivatives [19,20]. This technique combines rules for analytically differentiating the finite number of elemental functions in a programming language with the chain rule for differential calculus. The two principal approaches to implementing AD are operator overloading and compiler-based source transformation, each with its respective advantages and disadvantages. The two basic modes of AD are the forward mode and the reverse mode. The latter is preferable when the number of dependent variables is much smaller that the number of independent variables or when a vector is multiplied by the transpose of a Jacobian. We consider two implementations of automatic differentiation, derivify [21] and ADIC [4]. The derivify library is a simple, operator-overloading-based implementation of the forward mode that computes a single directional derivative at a time. ADIC is a source transformation tool that uses the forward mode to generate code that computes multiple directional derivatives simultaneously. The derivify library is easier to use than ADIC but does not perform as well for most problems.

3.3 Complex-Step Derivative Approximations

The use of complex variables for derivative approximation was originally devised by Lyness and Moler [22,23]. Recent work by Squire and Trapp [24] demonstrated its application for obtaining simple expressions for first-order derivatives that are highly accurate, robust, easy to implement, and achieved at a reasonable cost.

Table 1
Derivative techniques and relative merits

Technique	Accuracy	Features
Finite Differences	Approximate	Easy to implement
		Step-size dependence
Automatic Differentiation–operator overloading	Exact	Easy to use
		Programming-language dependent
Automatic Differentiation–source transformation	Exact	Harder to use
Complex-step Derivatives	Nearly Exact	Easy to implement
		Programming-language dependent

The basic formula is

$$\frac{\partial f}{\partial y}s_i + \frac{\partial f}{\partial p_i} \approx \frac{Im\left[f(t, y + j\delta s_i, p + j\delta pe_i)\right]}{\delta}, \tag{8}$$

where $j = \sqrt{-1}$ and the step size parameter δ is used to perturb the imaginary components of the solution y and the parameters p. Unlike finite-difference approximations, the perturbation δ can be made arbitrarily small (until underflow occurs) without roundoff error. Thus, the truncation error of the complex-step method can be reduced to almost zero.

The complex-step method is simple to implement in languages with an intrinsic complex data type, such as Fortran or C99. It can also be implemented in languages that support operator overloading through a complex class library. We consider one such class library, complexify [21], whose implementation and use closely resembles that of derivify. There are strong connections between the complex-step method and automatic differentiation, as observed by Martins et al. [25] and Griewank [26]. Lesk proposed implementing automatic differentiation directly by "overloading" the intrinsic functions for the complex data types in Fortran [27].

3.4 Summary

The tradeoffs of using the various derivative techniques are given in Table 1. The suitability of each technique depends greatly on the accuracy requirements of the application and on the programming language in which the simulation code is implemented.

4 Model Problems

To compare the various differentiation techniques for forward sensitivity analysis, we tested them on two PDEs—each with two parameters. The first test problem is a simplified version of a thermal wave problem [28]. The problem has been modified so that the parameters are physically meaningful: one parameter controls the speed of the wave, and the other controls the width of the wave front. The second problem is a two-dimensional chemical diurnal kinetics problem. It describes a simplified model of the transport, production, and loss of the oxygen singlet and ozone in the upper atmosphere.

4.1 1D Thermal Wave

The first example is a one-dimensional time-dependent nonlinear reaction-diffusion problem that is derived from [28,29]. The problem has a smooth analytic solution and models a steady-propagating wave front. The PDE can be written as

$$\frac{\partial w}{\partial t} = \frac{c\delta}{2}\left(\frac{\partial^2 w}{\partial x^2} + \frac{8}{\delta^2}w^2(1-w)\right), \tag{9}$$

where the boundary conditions are $w(x=-\infty,t)=1$ and $w(x=+\infty,t)=0$. The speed and width of the wave front are specified by c and δ, respectively, and the analytic solution is

$$w(x,t) = \frac{1}{2}\left(1-\tanh\left(\frac{x-ct}{\delta}\right)\right). \tag{10}$$

From differentiation, the analytic solution for the sensitivities are

$$\frac{\partial w(x,t)}{\partial c} = \frac{t}{2\delta}\,\mathrm{sech}^2\left(\frac{x-ct}{\delta}\right), \tag{11}$$

$$\frac{\partial w(x,t)}{\partial \delta} = \frac{(x-ct)}{2\delta^2}\,\mathrm{sech}^2\left(\frac{x-ct}{\delta}\right). \tag{12}$$

A system of N ODEs is obtained by discretizing the x-axis with $N+2$ grid points and replacing the second-order spatial derivative with a central difference approximation. Since the value of w is constant at the two endpoints, the semi-discrete equations for those points can be eliminated. The resulting system of ODEs can now be written with w_i as the approximation to $w(t,x_i)$, $x_i = i(\Delta x)$, and $\Delta x = 1/(N+1)$:

$$w_i'(t) = \frac{c\delta}{2}\frac{w_{i+1}-2w_i+w_{i-1}}{(\Delta x)^2} + \frac{8}{\delta^2}w_i^2(1-w_i). \tag{13}$$

The above equation holds for $i = 1, 2, \ldots, N$ with $w_0 = 1.0$ and $w_{N+1} = 0$.

The thermal wave problem is solved over the spatial domain $-20 \leq x \leq 20$ and integrated over the time interval $0 \leq t \leq 0.75$. The initial values are obtained by evaluating Eqs. (10)–(12) at time $t = 0$. The initial sensitivity values are strictly zero for (11) since $t = 0$, and a nonzero function of x for (12). The wavefront speed and width parameters are $c = 20$ and $\delta = 0.5$, respectively.

4.2 2D Diurnal Kinetics

The second example is a two-species diurnal kinetics advection-diffusion system in two spatial dimensions. The PDEs can be written as

$$\frac{\partial c_i}{\partial t} = K_h \frac{\partial^2 c_i}{\partial x^2} + V \frac{\partial c_i}{\partial x} + \frac{\partial}{\partial y} K_v(y) \frac{\partial c_i}{\partial y} + R_i(c_1, c_2, t) \quad (i = 1,\ 2), \tag{14}$$

where the subscripts i identify the chemical species, $K_v(y) = K_0 \exp(y/5)$, and the reaction terms are

$$R_1(c_1, c_2, t) = -q_1 c_1 c_3 - q_2 c_1 c_2 + 2q_3(t) c_3 + q_4(t) c_2 \tag{15}$$
$$R_2(c_1, c_2, t) = \quad q_1 c_1 c_3 - q_2 c_1 c_2 - q_4(t) c_2. \tag{16}$$

The scalar constants for this problem vary by many orders of magnitude: $q_1 = 1.63 \times 10^{-16}$, $q_2 = 4.66 \times 10^{-16}$, $K_0 = 10^{-8}$, $K_h = 4.0 \times 10^{-6}$, $V = 10^{-3}$, and $c_3 = 3.7 \times 10^{16}$. The diurnal rate constants are

$$q_i(t) = \exp\left[-a_i / sin(\omega t)\right] \quad \text{for } \sin(\omega t) > 0, \tag{17}$$
$$q_i(t) = 0 \qquad\qquad\qquad\quad \text{for } \sin(\omega t) \leq 0, \tag{18}$$

with $i = 3$ or 4, $\omega = \pi/(4.32 \times 10^4)$, $a_3 = 22.62$, $a_4 = 7.601$. Homogeneous Neumann boundary conditions are imposed, and the initial conditions are

$$c_1(x, z, 0) = 10^6 \alpha(x) \beta(y), \quad c_2(x, z, 0) = 10^{12} \alpha(x) \beta(y) \tag{19}$$
$$\alpha(x) = 1 - (0.1x - 1)^2 + (0.1x - 1)^4/2 \tag{20}$$
$$\beta(y) = 1 - (0.1y - 4)^2 + (0.1y - 4)^4/2. \tag{21}$$

The initial values for the sensitivities are strictly zero because the initial values in (19) are independent of the problem parameters.

The PDE system is discretized in space with central differencing to obtain an ODE system $\dot{y} = f(t, y, p)$ representing (14). The spatial domain is $0 \leq x \leq 20$, $30 \leq y \leq 50$. The time interval of integration is $[0, 4.32 \times 10^4]$, which represents 12 hours as measured in seconds.

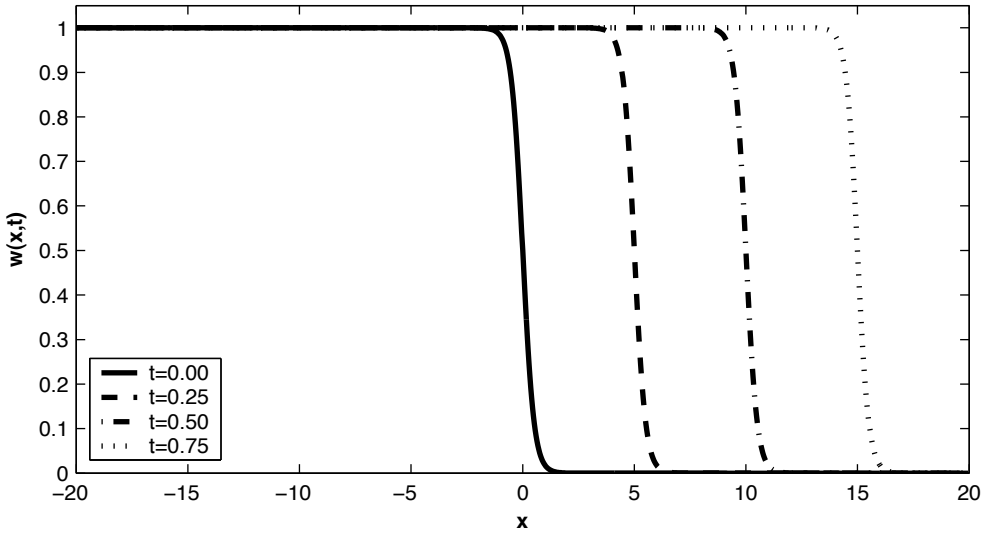

Fig. 1. Reference solutions for thermal wave problem.

5 Experimental Results

In this section we investigate the use of different derivative techniques as applied to the model problems in Section 4. Our first example is a simple but realistic problem in which the parameters values are physically meaningful and an analytic solution is available for accuracy comparisons. The second example is more challenging because it is a multicomponent PDE and the solutions, sensitivities, and parameters differ by many orders of magnitude. Both examples were compiled using `g++ -O -float-store` and executed on a 1.4GHz Pentium III with 1 gigabyte of memory. For our investigation, we focus on the accuracy and cost of computing the solution and sensitivities.

5.1 1D Thermal Wave Problem

Figure 1 shows some reference solutions for the thermal wave problem. Figures 2a and 2b give the reference solutions for the sensitivity with respect to wave speed and wavefront width, respectively. When applying the different derivative techniques, the solver statistics and accuracy are nearly identical. The most notable variation is in the run time. Of the methods, we see that the complex-step technique is the most costly, derivify and ADIC are roughly equal, and finite differences are typically cheapest. The various solver statistics are presented in Table 2 with respect to tighter relative and absolute ODE error tolerances. Several factors may be responsible for the increased cost in the use of complex arithmetic, including the choice of C++ compiler. In contrast, the operator overloading of derivify and the source code transformation approach of ADIC yield faster calculation of exact derivatives.

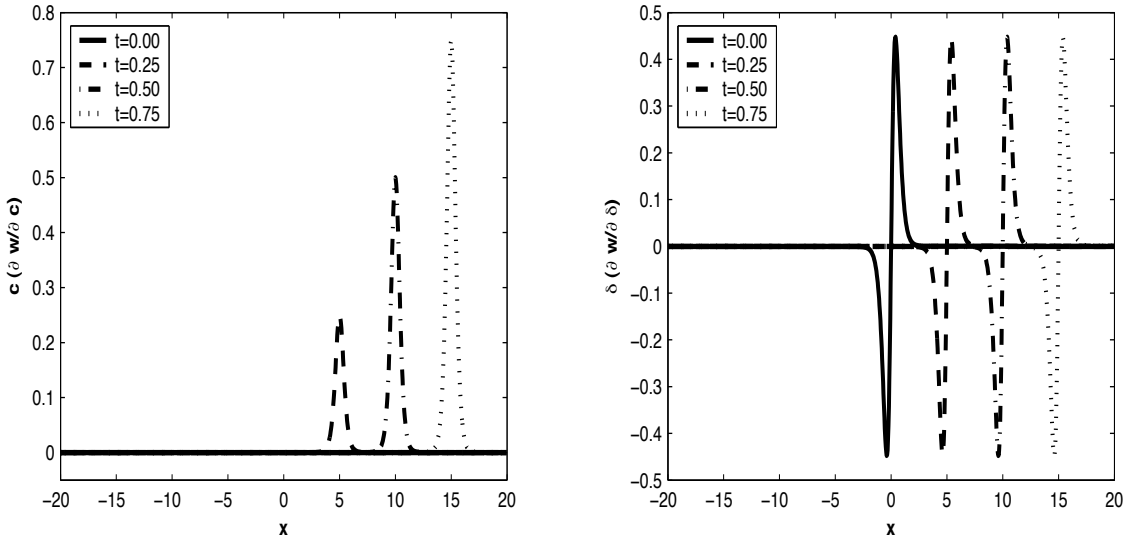

Fig. 2. Scaled sensitivities for thermal wave problem. (a) Scaled sensitivity with respect to speed. (b) Scaled sensitivity with respect to width.

Table 2
Thermal Wave Problem. Solver Statistics: Solution and Sensitivities.

	ATOL $= 1e-4$, RTOL $= 1e-4$				ATOL $= 1e-8$, RTOL $= 1e-8$			
	FD	CSD	derivify	ADIC	FD	CSD	derivify	ADIC
Time (sec)	0.9	9.8	1.41	1.06	2.82	38.34	5.42	4.03
RelErr w	1.03e-3	1.03e-3	1.03e-3	1.03e-3	6.43e-4	6.43e-4	6.43e-4	6.43e-4
RelErr cw_c	9.19e-3	9.19e-3	9.19e-3	9.19e-3	6.61e-3	6.61e-3	6.61e-3	6.61e-3
RelErr δw_δ	5.03e-2	5.03e-2	5.03e-2	5.03e-2	2.78e-2	2.78e-2	2.78e-2	2.78e-2
NST	452	452	452	452	1,028	1,373	1,373	1,373
NFE	907	907	907	907	2,062	2,750	2,750	2,750
NFES	1,904	1,904	1,904	1,904	4,230	2,824	2,824	2,824
NNI	452	452	452	452	1,031	1,374	1,374	1,374
NNIS	946	946	946	946	2,109	2,818	2,818	2,818
NLI	2,127	2,127	2,127	2,127	3,108	4,197	4,197	4,197

5.2 2D Diurnal Kinetics Problem

Figures 3, 4, and 5 compare the results obtained when computing solutions and sensitivities as an augmented system, using finite differences for computing sensitivity derivatives. The reference solution for the comparison, at the given spatial resolution, is obtained by using

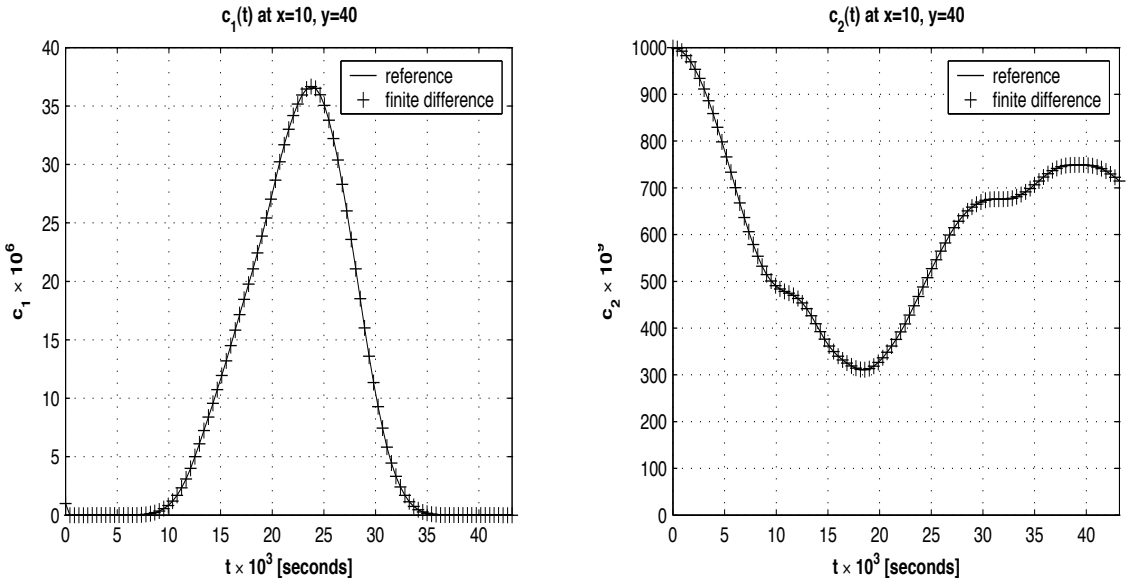

Fig. 3. Solution values computed at the center of the two-dimensional grid for species c_1 and species c_2.

very tight tolerances for RTOL and ATOL and an exact technique for computing sensitivity derivatives (i.e., automatic differentiation). Figures 3a and 3b compare the values for the two species c_1 and c_2 as computed at the center of the two-dimensional grid. The solution values are in good agreement in this case; however, Figures 4 and 5 show a large disparity in the results from the sensitivity computations. The discrepancy is apparent primarily in the c_1 component of the sensitivity vectors; see Figures 4a and 5a. From Figure 4, we especially note that the c_1 component remains negative despite the late rise above zero for the reference solution. From Figure 5, the c_1 components also fails to track the reference solution as the latter moves away from zero. The c_2 component of the sensitivities also suffers from accuracy problems, yet evidently maintains the correct qualitative behavior. The interesting features of this example are the improved accuracy in certain components of the sensitivity vectors and the fact that the solution y is in such close agreement with the reference solution. Table 3 presents the accuracy and run times of the various methods on this problem.

5.3 Summary

For the thermal wave problem, the solver statistics of CVODES roughly agree independent of which derivative technique is used. In this case, the main considerations are ease of implementation and run time. The complex-step method is relatively simple to implement but suffers from the slowness of the arithmetic with complex numbers. For the diurnal kinetics problem, the finite-difference method is easiest to implement and gives the fastest run times, but the sensitivities are found to be completely incorrect halfway through the simulation.

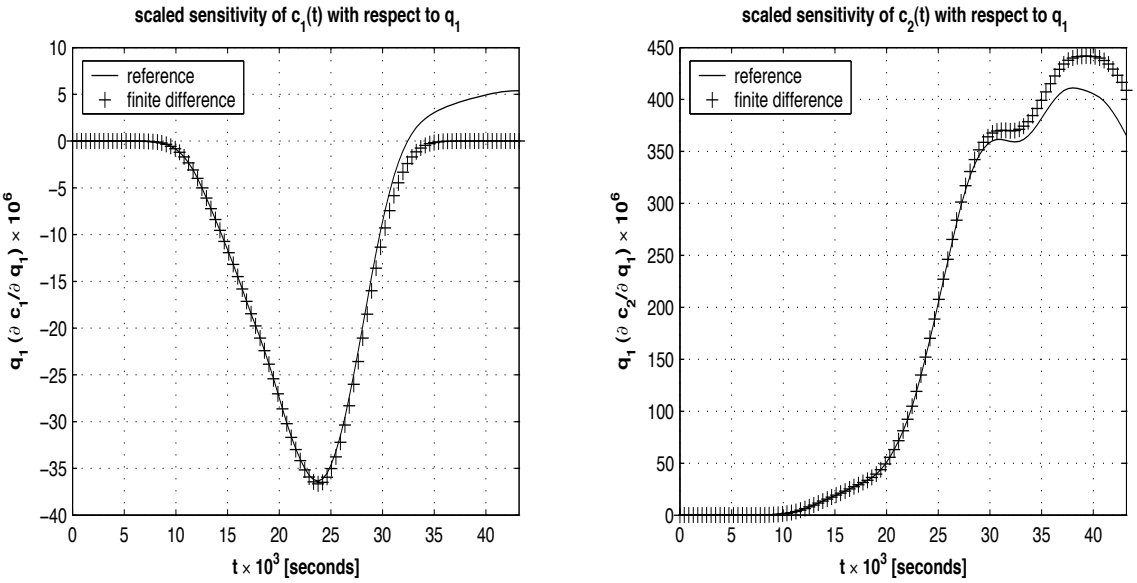

Fig. 4. Values as computed at the center of the two-dimensional grid for the scaled sensitivity of species with respect to q_1.

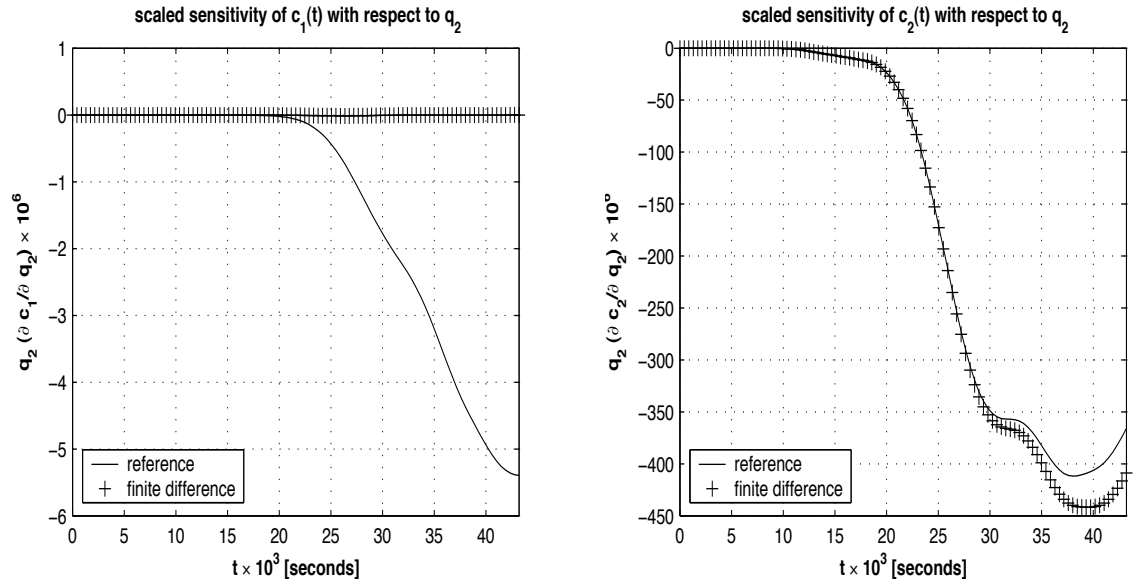

Fig. 5. Values as computed at the center of the two-dimensional grid for the scaled sensitivity of species with respect to q_2.

6 Conclusions

We compared three methods for computing (approximate) derivatives for use in the forward sensitivity analysis of time-dependent problems. These techniques all have tradeoffs in terms of their accuracy, ease of implementation, and performance cost. One conclusion is that the default finite-difference techniques can give misleading sensitivity results despite the accuracy to which their original solution variables are computed. This was demonstrated in the diurnal kinetics example. The failure of finite differences in this role can be difficult to

Table 3
Diurnal Kinetics Problem. Solver Statistics: Solution and Sensitivities.

	ATOL $= 1e+2$, RTOL $= 1e-5$				ATOL $= 1.0$, RTOL $= 1e-7$			
	FD	CSD	derivify	ADIC	FD	CSD	derivify	ADIC
Time (sec)	9.5	91.9	38.1	16.8	failed	387.9	124.5	68.9
NST	985	1,203	1,520	1,080	n/a	5,045	4,989	4,646
NFE	2,018	2,629	3,264	2,358	n/a	11,495	11,308	10,532
NFES	4,198	2,891	3,576	2,596	n/a	13,383	13,113	12,248
NNI	1,020	1,337	1,652	1,201	n/a	5,785	5,689	5,294
NNIS	2,093	2,885	3,570	2,590	n/a	13,381	13,111	12,246
NLI	3,306	2,998	3,792	2,852	n/a	11,176	10,999	10,262

detect, especially since our intuition about the behavior of solution sensitivities is not well developed. A second finding, and recommended safeguard, is to pursue convenient methods for computing (nearly) exact sensitivity derivatives. The derivify implementation of automatic differentiation is ideal as a simplified and inexpensive method for obtaining derivatives via an analytic method. Third, we note that the current implementations of complex-step techniques and automatic differentiations can be improved to reduce the overhead in using them. With advances in this area, and tools for automating their use, sensitivity analysis for scientific simulations will be made more reliable and perhaps more widely used.

Acknowledgments

We thank Radu Serban, Alan Hindmarsh, and other interested researchers and collaborators. Lee's work was performed under the auspices of the U.S. DOE by LLNL under contract no. W-7405-Eng-48. Hovland's work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, Office of Science, under Contract W-31-109-Eng-38.

The submitted manuscript has been created by the University of Chicago as Operator of Argonne National Laboratory ("Argonne") under Contract No. W-31-109-ENG-38 with the U.S. Department of Energy. The U.S. Government retains for itself, and others acting on its behalf, a paid-up, nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government.

References

[1] S. L. Lee, C. S. Woodward, F. Graziani, Analyzing radiation diffusion using time-dependent sensitivity-based techniques, J. Comp. Phys. 192 (2003) 211–230.

[2] H. Rabitz, M. Kramer, D. Dacol, Sensitivity analysis in chemical kinetics, Ann. Rev. Phys. Chem. 34 (1983) 419–461.

[3] R. Serban, A. C. Hindmarsh, CVODES: An ODE solver with sensitivity analysis capabilities, Tech. Rep. UCRL-JP-200039, Lawrence Livermore National Laboratory, Livermore, CA, submitted to ACM Transactions on Mathematical Software (Sep. 2003).

[4] C. Bischof, L. Roh, ADIC–An extensible automatic differentiation tool for ANSI-C, Software–Practice & Experience 27 (12) (1997) 1427–1456.

[5] C. Bischof, A. Carle, P. Khademi, A. Mauer, ADIFOR 2.0: Automatic differentiation of Fortran 77 programs, IEEE Computational Science & Engineering 3 (1996) 18–32.

[6] M. Caracotsios, W. E. Stewart, Sensitivity analysis of initial value problems with mixed ODEs and algebraic equations, Computers and Chemical Engineering 9 (1985) 359–365.

[7] M. Kramer, J. Leis, The simultaneous solution and sensitivity analysis of systems described by ordinary differential equations, ACM Transactions on Mathematical Software 14 (1988) 45–60.

[8] W. Feehery, J. Tolsma, P. Barton, Efficient sensitivity analysis of large-scale differential-algebraic systems, Applied Numerical Mathematics 25 (1) (1997) 41–54.

[9] T. Maly, L. Petzold, Numerical methods and software for sensitivity analysis of differential-algebraic systems, Applied Numerical Mathematics 20 (1997) 57–79.

[10] S. Li, L. R. Petzold, Design of new DASPK for sensitivity analysis, Tech. Rep. 1999-28, University of California at Santa Barbara (May 1999).

[11] S. L. Lee, A. C. Hindmarsh, P. N. Brown, User documentation for SensPVODE, a variant of PVODE for sensitivity analysis, Tech. Rep. UCRL-MA-140211, Lawrence Livermore National Laboratory (2000).

[12] J. R. Martins, P. Sturdza, J. J. Alonso, The complex-step derivative approximation, ACM Transactions on Mathematical Software 29 (3) (2003) 245–262.

[13] W. Anderson, J. Newman, D. Whitfield, E. Nielsen, Sensitivity analysis for the Navier-Stokes equations on unstructured meshes using complex variables, AIAA Paper 99-3294 .

[14] J. Newman, W. Anderson, L. Whitfield, Multidisciplinary sensitivity derivatives using complex variables, Tech. Rep. MSSU-COE-ERC-98-08, Computational Fluid Dynamics Laboratory (Jul. 1998).

[15] J. Martins, I. Kroo, J. Alonso, An automated method for sensitivity analysis using complex variables, AIAA Paper 2000-0689 .

[16] J. Martins, J. Alonso, J. Reuther, Complete configuration aero-structural optimization using a coupled sensitivity analysis method, AIAA Paper 2002-5402 .

[17] L. I. Cerviño, T. R. Bewley, On the extension of the complex-step derivative technique to pseudospectral algorithms, J. Comp. Phys. 187 (2003) 544–549.

[18] A. C. Hindmarsh, R. Serban, User documentation for CVODES, an ODE solver with sensitivity analysis capabilities, Tech. Rep. UCRL-MA-148813, Lawrence Livermore National Laboratory (Jul. 2002).

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

[20] C. H. Bischof, P. D. Hovland, B. Norris, Implementation of automatic differentiation tools, Higher-Order and Symbolic Computation To appear. Also available as Argonne National Laboratory Technical Report ANL/MCS-1153-0404.

[21] P. Sturdza, The `complexify.h` and `derivify.h` C++ class libraries, available at http://mdolab.utias.utoronto.ca/c++.html.

[22] J. Lyness, C. Moler, Numerical differentiation of analytic functions, SIAM J. Numer. Anal. 4 (2) (1967) 202–210.

[23] J. Lyness, Numerical algorithms based on the theory of complex variables, in: Proceedings of the ACM 22nd National Meeting (Washington D.C.), ACM, New York, 1967, pp. 125–133.

[24] W. Squire, G. Trapp, Using complex variables to estimate derivatives of real functions, SIAM Review 40 (1) (1998) 110–112.

[25] J. Martins, P. Sturdza, J. Alonso, The connection between the complex-step derivative approximation and algorithmic differentiation, AIAA Paper 2001-0921 .

[26] A. Griewank, personal communication. (1998).

[27] A. M. Lesk, Dynamic computation of derivatives, Communications of the ACM 10 (9) (1967) 571–572.

[28] A. I. Volpert, V. A. Volpert, V. A. Volpert, Traveling wave solutions of parabolic systems, American Mathematical Society, Providence, R.I., 1994.

[29] O. M. Knio, H. N. Najm, P. S. Wyckoff, A semi-implicit numerical scheme for reacting flow: II. Stiff, operator-split formulation, J. Comp. Phys. 154 (1999) 428–467.