LA-UR- 00-5453

UNCLASSIFIED

NECDC

October 2000

LA-UR-00 -????
Approved for public release
Distribution is unlimited

Title: AUTOMATIC DIFFERENTIATION OF AN EULERIAN HYDROCODE (U)

Author(s): Rudolph J. Henninger (CCS-2)
Alan Carle (Rice University)
Paul J. Maudlin (T-3)

Submitted to: Nuclear Explosives Code Developers Collaborations (NECDC) 2000
Oakland, CA
October 23 – 27, 2000

Los Alamos

NATIONAL LABORATORY

DISCLAIMER

Automatic Differentiation of an Eulerian Hydrocode (U)

R. J. Henninger, P. J. Maudlin
Los Alamos National Laboratory
Alan Carle
Rice University

Automatic differentiation (AD) is applied to a two-dimensional Eulerian hyrodynamics computer code (hydrocode) to provide gradients that will be used for design optimization and uncertainty analysis. We examine AD in both the forward and adjoint (reverse) mode using Automatic DIfferentiation of FORtran (ADIFOR, version 3.0). Setup time, accuracy, and run times are described for three problems. The test set consists of a one-dimensional shock-propagation problem, a two-dimensional metal-jet-formation problem and a two-dimensional shell-collapse problem. Setup time for ADIFOR was approximately one month starting from a simplified, fixed-dimension version of the original code. ADIFOR produced accurate (as compared to finite difference) gradients in both modes for all of the problems. These test problems had 17 independent variables. We find that the forward mode is up to39% slower and the adjoint mode is at least 11% faster than finding the gradient by means of finite differences. Problems of real interest will certainly have more independent variables. The adjoint mode is thus favored since the computational time increases only slightly for additional independent variables. (U)

Keywords: hydrodynamics, forward and adjoint sensitivities

Introduction

The purpose of this project has been to provide sensitivities of results from an Eulerian hydrodynamics computer code (hydrocode) (1) for use in design-optimization and uncertainty analyses. We began (2) by applying an equation-based sensitivity technique used successfully in the early eighties that was applied to reactor-safety thermal-hydraulics problems (3,4), which is called Differential Sensitivity Theory (DST) (5,6). In these reactor-safety applications, DST was found to provide accurate sensitivities (3,4). The methodology is as follows. The system of partial differential equations (the forward or physical PDEs) is assembled, differentiated with respect to the independent variables of interest, and adjointed (5). The resulting adjoint PDEs are then solved using straightforward numerical operators. The forward-variable solutions when needed for the adjoint solutions are provided by the original computer code that solves the physical (or forward) problem. In the present hydrocode application, acceptable results were obtained for one-material, one-dimensional problems. The DST results were then improved by means of "compatible" finite difference operators (7,8). We have seen, however, that DST techniques do not produce accurate values for sensitivities to all of the independent variables of interest and for problems with discontinuities such as a multi-material problem (9). To obtain accurate sensitivities for arbitrary numerical resolution a more code-based approach was then tried. We attempted to apply automatic differentiation (AD) in the forward mode using Automatic DIfferentiation of FORtran (ADIFOR, version 2.0)(10) and the Tangent-linear and Adjoint Model Compiler (TAMC)(11) in the forward and adjoint mode. We were successful for one-dimensional problems in both modes but failed to obtain accurate sensitivities in the adjoint mode for two-dimensional problems (12,13).

Here we present the successful results for two-dimensional problems in both the forward and adjoint modes using ADIFOR, version 3.0 (14). In what follows, we describe AD methods in the context of their use for a hydrocode. We then examine setup time, results, accuracy, and computer run times for three test problems obtained by ADIFOR. Finally, we outline our plans for future work.

AD Methods for a Hydrocode

Forward vs. Adjoint Modes of Differentiation. Both code- and equation-based methods can be implemented in either the forward or adjoint mode. By forward and adjoint, we mean the direction through the code and in time in which the derivative values are obtained. The forward mode of differentiation involves determining the necessary derivatives by following the code logic in the forward direction (and forward in time); while for the adjoint mode, the derivatives are determined by following the code logic in the reverse direction (and backward in time). Which of these is more useful and efficient depends on the relative numbers of independent variables of interest and responses of interest. The forward mode is more efficient for determining the sensitivity of many responses to one or a few independent variables, while the adjoint mode is better suited for sensitivities of one or a few responses with respect to many independent variables.

For an optimization process the response of interest is a so-called cost function that, for example, computes the sum of the squares of the difference between the calculated and a desired data set. There may be many independent variables of interest (these could be the material model parameters and the initial and boundary conditions). For this problem one would choose the adjoint mode, which is most efficient for one response and many independent variables. The most recent version of ADIFOR (14) that is used here is applied in both the forward and adjoint mode.

Mechanics of Adjoint Differentiation. Consider that a program is represented in terms of a flow diagram such as that shown in Figure 1. This is a simplification, but the principle will stand for more complex situations. The quantity α is the input, R is the output (presumably a scalar cost function in the case of an optimization process), the modules **A**, **B**, and **C** are the processes or transforms to which the input is subjected, and the output of process **A** is **y**, and that of **B** is **z**, and R is the output of **C**. This sequence of processes is considered the forward or physical calculation.

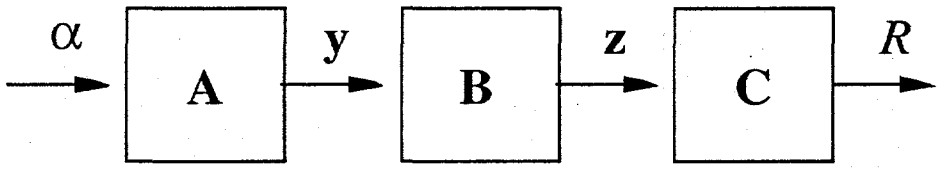

Figure 1. Data flow diagram of the forward calculation.

The quantities α, **y**, and **z** are general data structures; they can consist of mixed types of data structures. Some of the data may even be parameters that affect the transformations or processes themselves. There is no loss in generality if they are thought of as being carried along in the sequence of data structures up to the module in which they are used. [In fact, it is necessary that the input to a process must be all that is required in order to determine the output of that process. The intermediate data structures could represent all of the data, and the only changes in variables from input to output of the process are in variables affected by that particular process.] With all of that in mind, these structures can have high dimensionality. We also do not place any restrictions on the processes, other than that they be differentiable (functions not obviously differentiable can often be handled). By requiring that the input to a process be all that is necessary to produce the output, we have thus required that each transformation or process is self-contained.

Considering the possibly high dimensionality of the data structures, storing the sensitivity matrices of the transformations, such as $\frac{\partial y_i}{\partial \alpha_j}$ for all i and j, is likely to be extremely costly, because one is multiplying the dimensionalities of α and **y**, which may already be large. The chain rule, however, allows the calculation of the response R with respect to the i^{Th} component of α

$$\frac{\partial R}{\partial \alpha_i} = \sum_k^K \sum_j^J \frac{\partial R}{\partial z_k} \frac{\partial z_k}{\partial y_j} \frac{\partial y_j}{\partial \alpha_i} \tag{1}$$

Even if a process is nonlinear, the expression above amounts to a $K \times J$ matrix multiplication, each element of which specifies the differential response of an output variable with respect to a differential change of an input variable. The order of summations can be done two ways, either over j first or over k first. If the summation is done over j first, one is proceeding in the same direction as the forward calculation and is therefore in the forward mode discussed above. The data-flow diagram in Figure 2 illustrates the forward mode process (where subscript derivative notation is used *i. e.*, y_α is the derivative of y with respect to α). By studying the data flow diagram, one can see how the forward mode is not optimum for a situation with many independent variables and one output of interest. If the dimension (I) of α is large, then the results of the first summation $\frac{\partial y}{\partial \alpha}$ can be very large (dimensionality = $I \times J$), and so on through the process until the last step, which reduces to $\frac{\partial R}{\partial \alpha}$. The response R is a scalar, so that our final result has just the dimensionality of α, while the intermediate results had the dimensionality of either $I \times J$ or $I \times K$. As these data structures can be very large, this can result in extremely large intermediate results that need to be stored.

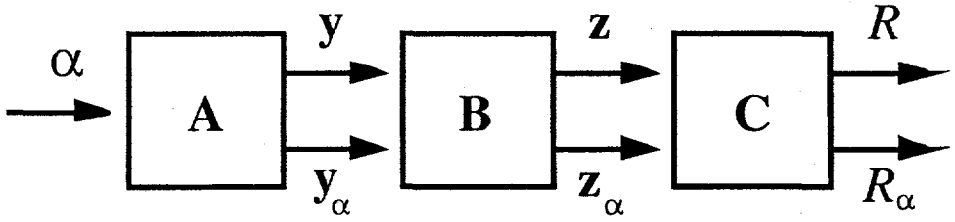

Figure 2. Data flow diagram of the forward derivative calculation.

Summing over k first, on the other hand, yields the adjoint mode, and the sequence of events goes backwards from R. Figure 3 illustrates this sequence in a data-flow diagram. The notation for the derivatives is given in the same way that it was for the forward mode. The adjoint mode of differentiation can be seen to be useful; since R (a scalar) is what is always being differentiated, the dimensionality of the possibly large data structures are never multiplied together as they are in the forward mode. Instead of storing the matrix of the adjoint of each process ($\frac{\partial R}{\partial z}$, $\frac{\partial z}{\partial y}$, and $\frac{\partial y}{\partial \alpha}$), only the intermediate data structures are formed and stored. Thus the requirement for storing these data structures is only about double that required to store the structures for the forward calculation (the forward calculation structures are also required for the sensitivity calculation if the processes are non-linear).

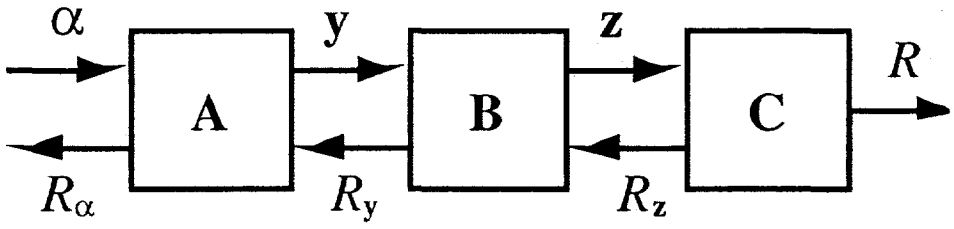

Figure 3. Data flow diagram of the adjoint derivative calculation.

Figure 3 presents the adjoint differentiation technique, and it is the method that is followed for AD implementation. The automatic differentiation tool ADIFOR (version 3) is discussed next.

ADIFOR. AD tools have several stages involved in getting from the original code to an executable code with derivative coding included as indicated in Figures 2 and 3. The first step is to submit the original code to a precompiler. This precompiler analyzes the code and modifies it to include code that calculates the derivatives of interest. The output of this step is enhanced code, with some calls to external subroutines. For a non-linear hydrocode, information from the forward calculation is needed in the adjoint calculation. Independent storage or recalculation can satisfy the need for this information. The second step in the process is to determine and setup the required storage. For a large problem, storage of the entire forward solution is impossible. A technique called checkpointing is required. This technique consists of dumping the solution at checkpoints as the forward solution is generated. The forward solution is stored from the final checkpoint to the final time of the forward calculation. One then calculates the adjoint solution backward from the final state. One then recalculates the forward solution from the second to the last checkpoint. This process is repeated until the starting time of the forward calculation is reached. The last step is to compile the enhanced code, auxiliary storage code and the adjoint code, including run-time libraries that satisfy the external subroutine calls. The current version of ADIFOR works only on Fortran77 code. There is a version for C, known as ADIC.

Problem Descriptions and Results

Setup time, accuracy, and run times are described for three problems. The problem test set consists of a one-dimensional shock-propagation problem, a two-dimensional metal-jet-formation problem, and a two-dimensional shell-collapse problem. Setup time for ADIFOR was approximately one month starting from a simplified fixed-dimension version of the original code. Creation of the simplified code and getting it running on an SGI platform represents several months of work.

One-Dimensional Shock Problem. The first test problem is the one-dimensional impact of a copper plate with a rigid boundary where the plate has an initial velocity of 0.05 cm/μs (500 m/s) as indicated in Fig. 4.

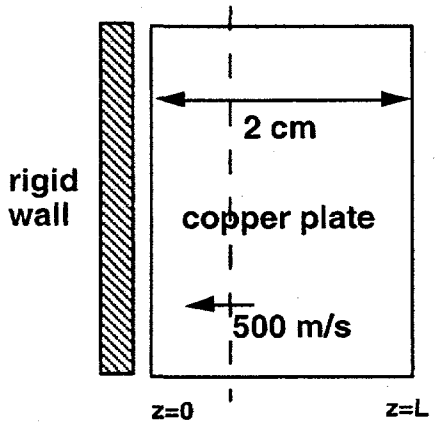

Figure 4. One-dimensional shock problem. The response R is given by Eq. 2; z_0 is at the dashed line.

The cooper is represented by a Mie-Grueneisen equation of state (EOS) and an elastic-perfectly-plastic strength model. Upon impact the plate experiences a right-going shock that compresses the material to the Hugoniot pressure of 20.89 kbar (density of 9.961 g/cm^3). The plate is 2 cm in thickness and is divided into forty 0.5-mm cells for the numerical computations. Running with a two-dimensional code required 3 cells in the transverse direction. The impact

problem was simulated out to a final time of 2.0 μs. The response for problem was arbitrarily chosen to be the time-averaged density at the spatial location $z_0 = 0.6$ cm, i.e.,

$$R = \int_t \int_z \rho\, \delta(z - z_o) \frac{1}{t_f} dz\, dt = \bar{\rho}(z_o). \tag{2}$$

Sensitivity results for this impact problem are given in Table 1 for the time-average density located at 0.6 cm (R_O = 9.236 g/cm^3). This table lists the 17 problem parameters (independent variables) in columns one through three, the sensitivities $\partial R/\partial\alpha$ as determined by both forward and adjoint ADIFOR-produced code in column four, and the finite difference sensitivities $\Delta R/\Delta\alpha$ in column five (obtained via finite perturbations of each parameter appearing in the physical equation set) for validation purposes. Constructing the 17 finite difference sensitivities in column five required 18 forward calculations. Runtimes for the various methods are compared and contrasted below.

Table 1. Comparison of one-dimensional shock problem parameters and sensitivities.

α	**Description**	**Value**	$\partial R/\partial\alpha$ *	$\Delta R/\Delta\alpha$
$\rho(t=0)$	Initial Density (g/cm^3)	8.93	1.0830336	1.0830335
$u_z(t=0)$	Initial Velocity (m/s)	-0.050	-6.2245379	-6.2245225
$e(t=0)$	Initial Internal Energy (Mbar-cm^3/g)	0.0	2.5850554	2.5850481
$s_{zz}(t=0)$	Initial Axial Deviatoric Stress (Mbar)	0.0	0.68905919	0.68905837
s	Shock Velocity Constant	1.489	0.034749960	0.03474934
c_0	Sound Speed (cm/μs)	0.394	0.68660301	0.68660035
ρ_0	Nominal EOS Density (g/cm^3)	8.93	-0.052257965	-0.052258786
Γ	Second Grueneisen Ratio	2.002	0.0077603254	0.00775988
c_L	Linear Artificial Viscosity Constant	0.8	-0.000746556	-0.0074685
c_Q	Quadratic Artificial Viscosity Constant	16.0	0.0001929072	0.00019286
$u_z(z=0)$	Outflow Velocity (cm/μs)	0.00	7.5631986	7.5633874
$\rho(z=L)$	Inflow Density (g/cm^3)	8.93	0.0014047381	0.00140464
$e(z=L)$	Inflow Internal Energy (Mbar-cm^3/g)	0.0	0.16173630	0.16173605
$u_z(z=L)$	Inflow Velocity (cm/μs)	-0.05	-0.074715761	-0.0746979
$s_{zz}(t=0)$	Inflow Axial Deviatoric Stress (Mbar)	0.0	-0.009010984	-0.00901169
Y_0	Yield strength (Mbar)	0.050	0.51783506	0.51782111
G	Shear Modulus (Mbar)	0.50	-0.014570571	-0.01457216

* R = Time-averaged density at 0.6 cm over 2 μs, R_O = 9.236 g/cm^3

Jet-Formation Problem. The second test problem is a two-dimensional Cartesian jet-formation problem in which a copper bar impacts a rigid boundary as is shown in Figure 2. The bar has an initial axial velocity ($u_z(t=0)$) of 0.7 km/s and was run with three transverse velocity ($u_x(t=0)$) cases: 0.0, -0.1, and –0.7 km/s respectively. For the non-zero transverse velocities a jet is formed that flows along the axis. The response of interest is the jet tip speed that is obtained by

following a marker particle that is placed on axis at the right side of the copper bar (shown as a black dot in Figure 5).

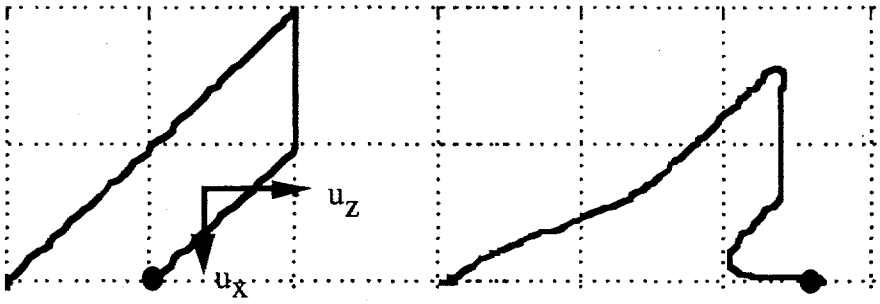

Figure 5. Copper bar impacting a rigid boundary subsequently forming a jet when the transverse velocity ($u_x(t=0)$) is non-zero; the response is the marker particle (shown as the black dot) velocity at the end of the calculation.

For this example the problem parameters are the same as those listed in Table 1. The problem was run with $c_L = 0.2$, $c_Q = 2.0$, $\rho(z=L)=0$, $u_z(z=L)=0$, $Y_0 = 0$, and $G = 0$ and to a final time of 1 μs. The initial velocity is the axial velocity ($u_z(t=0) = 0., -0.01, or -0.07$ cm/μs). The axial velocity was chosen because the sensitivity of the final velocity to the initial axial velocity is 1.000 for the zero transverse velocity case. This seemingly trivial result provided an excellent test of the advection-scheme adjoint code. When 1.000 was obtained for the initial velocity sensitivity, the other sensitivities agreed well with the forward results obtained by ADIFOR (which is viewed as the "correct" solution) and by finite differences. The sensitivities produced by the adjoint code for –0.01 cm/μs also agreed well with those of the forward code and finite differences. It was not possible to produce reasonable sensitivities for times greater than 0.8 μs by any method for the –0.07 cm/μs transverse velocity case. Examination of the computed results for this case showed that the sensitivity was proportional to the marker particle acceleration, which became unstable after 0.8 μs. A different response choice (other than following a marker particle) will be necessary to obtain a numerically smoother characterization of the jet tip speed and stable sensitivities. We intend to explore this issue in future work. The run times for the forward and adjoint codes are examined below.

Shell-Collapse Problem. The third test problem is a free-running shell collapse. In this problem, a spherical shell of elasto-plastic material is given an initial velocity toward its center. During the collapse, the shell thickens, and the kinetic energy is irreversibly converted to internal energy. Under the appropriate initial conditions, the shell will stop at a finite inner radius when all of its kinetic energy has been dissipated. Ignoring elastic effects, the equations of motion for the shell are the one-dimensional, spherical geometry, Lagrangian equations subject to the constraint of the Von Mises yield criterion. The equations are simplified further by invoking incompressibility in the plastic flow. The initial velocity distribution in the shell is given by:

$$u(r) = u_0\left(r_0/r\right)^2 \tag{3}$$

where u_0 is the initial velocity at the inner radius r_0.

Verney (16) has provided the analytical solution for the plastic work done during the collapse of the shell from r_0 to r_0'. By equating the plastic work to the initial kinetic energy, an initial velocity distribution may be determined that is consistent with a specified final inner radius. A density of 1.845 g/cm^3 and a yield stress of 3.3 kbar were assumed. The initial shell radius is 8 cm and its thickness is 2 cm. For a final inner radius r_0' of 3 cm, the initial inner-radius velocity is 0.067504 cm/μs.

Using this velocity, the fully collapsed (>99 % kinetic energy converted) inner radius was calculated to be approximately 3 cm, which is in good agreement with the analytical result. For timing comparison purposes we used all 17 parameters that are listed in Table 1. Of these, the eleven that affected the result are listed in Table 2. For the analytical solution the only parameter that matters is the yield strength. The yield strength is also the parameter with the largest sensitivity. To the number of digits listed in Table 2 the forward- and adjoint-calculated sensitivities were identical.

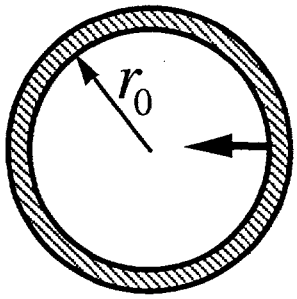

Figure 6. Shell- collapse problem initial geometry. The inner radius r_0 is 8 cm; the shell thickness is 2 cm.

Table 2. Shell-collapse problem active parameters and their sensitivities.

α	**Description**	**Value**	$\partial R/\partial\alpha$ *
$\rho(t=0)$	Initial Density (g/cm^3)	1.845	-3.0224558
$e(t=0)$	Initial Internal Energy (Mbar-cm^3/g)	0.0	1.03613223
$s_{zz}(t=0)$	Initial Axial Deviatoric Stress (Mbar)	0.0	11.1253570
s	Shock Velocity Constant	1.489	-0.002254506
c_0	Sound Speed (cm/μs)	0.394	-0.027052578
ρ_0	Nominal EOS Density (g/cm^3)	1.845	-2.2454823
Γ	Second Grueneisen Ratio	2.002	-0.026381213
c_L	Linear Artificial Viscosity Constant	0.0	0.0006064444
c_Q	Quadratic Artificial Viscosity Constant	2.0	0.0000215145
Y_0	Yield strength (Mbar)	0.0033	2033.7093367
G	Shear Modulus (Mbar)	1.51	0.0033900528

* R = Radius at end of collapse (100 μs) $r_0' = 3$ cm

Accuracy and Timing Comparisons. ADIFOR-processed code provided accurate (as compared to finite difference) parameter gradients in both the forward and adjoint modes for all of the problems. The number of independent variables versus the number of responses plays a key role in this decision. The run times for the various methods used to obtain the test problem sensitivities are listed in Table 3. These test problems had 17 independent variables and one response. We find that the ADIFOR forward mode is up to 39% slower and the ADIFOR adjoint

mode is at least 11% faster than finding the gradient by means of finite differences. Problems of real interest will certainly have more independent variables thus favoring the adjoint mode.

Table 3. Comparison of computational times on an SGI Origin 2000 for test problems with 17 independent variables and one response.

Problem	1D Shock	Jet	Shell
Problem Information			
Cells in 2D	3 X 40	60 X 100	42 X 42
Time steps	400	100	1000
Run Times CPU seconds			
Finite Difference	36	126	347
ADIFOR-Forward	15	146	484
ADIFOR-Adjoint	12	63	309

Summary and Future Work

Since last reporting in this forum we have applied the automatic differentiation tool ADIFOR (version 3.0) to MESA2D (a Fortran77 code) and have obtained accurate sensitivities for three test problems in both the forward and adjoint modes. With this capability in hand we can now apply these sensitivities to uncertainty analyses and optimization problems. The next step will be to apply the Frotran90 version of ADIFOR that is under development.

References

1. D. J. Cagliostro, D. A. Mandell, L. A. Schwalbe, T. F. Adams, and E. J. Chapyak, "MESA 3-D Calculations of Armor Penetration by Projectiles with Combined Obliquity and Yaw," *Int. J. Impact Engineering*, **10**, (1990).

2. P. J. Maudlin, R. J. Henninger, and E. N. Harstad, "Application of Differential Sensitivity Theory to Continuum Mechanics," Proc. ASME Winter Annual Meeting, 93, New Orleans, Louisiana (November 28-December 3, 1993).

3. P. J. Maudlin, C. V. Parks and C. F. Weber, "Thermal-Hydraulic Differential Sensitivity Theory," ASME paper No. 80-WA/HT-56, Proc. ASME Annual Winter Conference (1980).

4. C. V. Parks and P. J. Maudlin, "Application of Differential Sensitivity Theory to a Neutronic/Thermal Hydraulic Reactor Safety Code," *Nucl. Technol.*, **54**, 38 (1981).

5. D. G. Cacuci, C. F. Weber, E. M. Oblow and J. H. Marable, "Sensitivity Theory for General Systems of Nonlinear Equations," *Nucl. Sci. Eng.*, **75**, 88 (1980).

6. D. G. Cacuci, P. J. Maudlin and C. V. Parks, "Adjoint Sensitivity Analysis of Extremum-Type Responses in Reactor Safety," *Nucl. Sci. Eng.*, **83**, 112 (1983).

7. R. Henninger, P. Maudlin, and M. L. Rightley, "Accuracy of Differential Sensitivity for One-Dimensional Shock Problems," LA-UR-97-596, 1997 APS Shock Compression of Condensed Matter Conference, Amherst, MA (27 July -1 August, 1997).

8. M. Shashkov, *Conservative Finite Difference Methods on General Grids*, CRC Press, Boca Raton (1996).

9. R. J. Henninger, P. J. Maudlin, and E. N. Harstad, "Differential Sensitivity Theory Applied to the MESA2D Code for Multi-Material Problems, " Proceedings of the APS Meeting on Shock Compression of Condensed Matter, 283, Seattle, WA, (August 1995).

10. C. Bischof, A. Carle, P. Khademi, and A. Mauer, "The ADIFOR 2.0 System for the Automatic Differentiation of Fortran 77 Programs," Argonne National Laboratory Report ANL-MCS-P481-1194 (1995).
11. R. Giering, "Tangent Linear and Adjoint Model Compiler Users Manual," Manual Version 1.1, TAMC Version 4.76, 1997.
12. R. J. Henninger, M. L. Rightley, and P. J. Maudlin, "Code Differentiation Applied to MESA," 1998 Nuclear Explosives Code Development Conference, LA-UR-98-5103, Las Vegas, NV (25-30 October 1998).
13. R. J. Henninger, M. L. Rightley, and P. J. Maudlin, "Code Differentiation for Hydrodynamic Model Optimization," Shock Compression of Condensed Matter – 1999, Proceedings of the Conference of the American Physical Society, Topical Group on Shock Compression of Condensed Matter, LA-UR-99-3075, pages 359-362, Snowbird, UT (27 June – 2 July 1999).
14. – "ADIFOR 3.0 Overview", Rice University Technical Report CAAM-TR-00-02 (February 2000).
15. F. H. Harlow and A. A. Amsden, "Fluid Dynamics," LA-4700, Los Alamos Scientific Laboratory, 1971.
16. D. Verney, "Evaluation de la Limite Elastique et Cuivre et de l'Uranium par des Experiences d'Implosion 'Lente'," *Behavior of Dense Media under Dynamic Pressures*, Symposium H. D. P., International Union of Theoretical and Applied Mechanics, page 293, Gordon and Breach, New York (1968).