CONF-8509121--6

AUTOMATING SENSITIVITY ANALYSIS OF COMPUTER MODELS USING COMPUTER CALCULUS*

E. M. Oblow and F. G. Pin

Oak Ridge National Laboratory
Oak Ridge, TN 37831

CONF-8509121--6

DE85 018095

Invited Presentation in Proceedings of the Seventh Annual DOE/LLW Participant's Information Meeting, September 11-13, 1985, Las Vegas, Nevada.

*Research sponsored by the Office of Defense Waste and By-Products Management of the U.S. Department of Energy, under contract No. DE-AC05-840R21400 with Martin Marietta Energy Systems, Inc.

AUTOMATING SENSITIVITY ANALYSIS OF COMPUTER MODELS USING COMPUTER CALCULUS

E. M. Oblow and F. G. Pin
Oak Ridge National Laboratory
Oak Ridge, TN 37831

ABSTRACT

An automated procedure for performing sensitivity analyses has been developed. The procedure uses a new FORTRAN compiler with computer calculus capabilities to generate the derivatives needed to set up sensitivity equations. The new compiler is called GRESS - Gradient Enhanced Software System. Application of the automated procedure with "direct" and "adjoint" sensitivity theory for the analysis of non-linear, iterative systems of equations is discussed. Calculational efficiency consideration and techniques for adjoint sensitivity analysis are emphasized. The new approach is found to preserve the traditional advantages of adjoint theory while removing the tedious human effort previously needed to apply this theoretical methodology. Conclusions are drawn about the applicability of the automated procedure in numerical analysis and large-scale modelling sensitivity studies.

I. INTRODUCTION

Sensitivity theory has been used in many fields over the past two decades to assess the importance of variations in modeling data and parameters on calculated model results. This theory and its applications are summarized in Refs. 1-6 and the numerous references cited therein. The importance of this theory lies in its methodological diversity and wide ranging applicability in determining how sensitive analytic and calculated results are to model data.

For the purposes of this paper, several approaches to the sensitivity problem will be discussed but emphasis will be placed on the implementation of one particular approach - the adjoint sensitivity methodology. In problems where the number of model results of interest is limited but the data field needed to define the model is large, adjoint sensitivity theory is usually the method of choice.[7-10] The adjoint approach produces all model sensitivities for one model result using only a single additional

(i.e., adjoint) model calculation, no matter how large the model data field or how complex the model (including non-linear problems). The development of an adjoint model, however, requires considerable knowledge about the model equations and a long-term, high-level development effort. Due to the substantial man-power costs involved in such an effort, adjoint sensitivity theory has not enjoyed the widespread applicability that other techniques have had in large-scale modeling programs. In the past, adjoint methods have been applied more appropriately to small-scale analytical studies (see for instance Refs. 8, 11-14) or large-scale modeling programs with longer-term, higher-risk research goals (see for instance Refs. 15-17).

To try to remedy this situation, this paper will explore the possibility of automating the procedures needed to implement adjoint theory. Particular emphasis will be placed on the use of computer calculus to assist in the tedious process of constructing adjoint models from basic model equations as they exist in a computer code. The general problem addressed in this paper will be that of performing a sensitivity analysis on a large-scale computer model written in the FORTRAN programming language. The equations describing such a model and the sensitivity theories appropriate to its analysis will be specified in Section II. The use of computer calculus to assist in the development of a sensitivity analysis capability will be described in Section III and a full-scale calculus language, GRESS, used to automate this procedure will be outlined in Section IV. An analytic application of this procedure will be presented in Section V. The future applicability of this method, as well as its advantages and disadvantages will be discussed in Section VI.

II. THE GENERAL SENSITIVITY PROBLEM

A brief description of general sensitivity theory is given here as an aid to understanding the problem of applying this theory to computer models. The example to be discussed will be that of a general set of non-linear equations given as

$$\bar{y} = \bar{F}(\bar{y},\bar{c}) \tag{1}$$

where $\bar{y}$ represents the dependent variable being solved for, $\bar{c}$ represents the user specified model data or parameter set, and $\bar{F}$ defines the model equations. The particular form chosen in Eq. (1) is one that can be used

generally to represent equations coded in the FORTRAN programming language. The left-hand side of the equation represents the stored value of the variable calculated from the functional formula on the right-hand side. This form will be needed later for discussions about computer calculus and adjoint sensitivity code development.

Since the number of components of the vector $\bar{y}$ calculated in any typical large-scale modelling problem is large, it is useful to define a generic result for such a calculation that is of particular interest to the model user. Typically several results will be needed for analysis but in all cases they form a much smaller set than the actual set of $\bar{y}$ component values. A typical result will be defined as:

$$R = h(\bar{y}) \tag{2}$$

where R is a single number which is a function of the solution to Eq. (1) (i.e., $\bar{y}$). For notational ease, the generic parameter α_i will be used to denote any individual parameter, be it a component of the vector $\bar{c}$ or any other parameter vector defined later in the paper. The total number of parameters in the problem will be assumed to be M so that the index on α_i will run from 1 to M.

The basic problem in any sensitivity study is to find the rate of change in the result R arising from changes in any model parameters. For the generic parameter α_i, then, the quantity of interest is the numerical value of $dR/d\alpha_i$ given analytically by,

$$\frac{dR}{d\alpha_i} = \frac{\partial h}{\partial \bar{y}} \frac{d\bar{y}}{d\alpha_i} \tag{3}$$

Since the functional dependence of R on $\bar{y}$ through $h(\bar{y})$ is defined analytically by the model user, only $d\bar{y}/d\alpha_i$ needs to be generated in order to evaluate Eq. (3). The procedure needed to get $d\bar{y}/d\alpha_i$ is to differentiate Eq. (1) as follows,

$$\frac{d\bar{y}}{d\alpha_i} = \frac{\partial \bar{F}}{\partial \bar{y}} \frac{d\bar{y}}{d\alpha_i} + \frac{\partial \bar{F}}{\partial \bar{c}} \frac{d\bar{c}}{d\alpha_i} \tag{4}$$

Rearranging Eq. (4) yields the following set of coupled equations to solve for $d\bar{y}/d\alpha_i$,

$$(I - \frac{\partial \bar{F}}{\partial \bar{y}}) \frac{d\bar{y}}{d\alpha_i} = \frac{\partial \bar{F}}{\partial \bar{c}} \frac{d\bar{c}}{d\alpha_i} \tag{5}$$

or in more compact form,

$$A\bar{y}_i' = \bar{s}_i \qquad i=1,\ldots,M \tag{6}$$

where I is the identity matrix and A, $\bar{y}_i'$ and $\bar{s}_i$ are given by,

$$A \equiv I - \frac{\partial \bar{F}}{\partial \bar{y}} \tag{7}$$

$$\bar{y}_i' \equiv \frac{d\bar{y}}{d\alpha_i} \tag{8}$$

$$\bar{s}_i \equiv \frac{\partial \bar{F}}{\partial \bar{c}} \frac{d\bar{c}}{d\alpha_i} \tag{9}$$

If Eq. (6) were solved directly for $\bar{y}_i'$ the result could be used in Eq. (3) to evaluate $dR/d\alpha_i$. This method of sensitivity analysis is called the "direct" approach and is a classical methodology which has received a great deal of attention in the literature[3,4]. Its main drawback arises in large-scale applications where the size of the vector $\bar{c}$ (and therefore the number of α_i's whose sensitivities need to be evaluated) becomes prohibitively large. Since Eq. (6) must be solved each time a new α_i is defined, the computational expense puts this method out of reach as a practical sensitivity tool. Its practical value is therefore restricted to smaller-scale analytical problems or other cases where A can easily be inverted.

Since the ultimate objective of a large study, however, is still the evaluation of $dR/d\alpha_i$, the intermediary step of solving for $d\bar{y}/d\alpha_i$ and its inherent computational inefficiency can be avoided. For such problems the "adjoint" approach is far more applicable. In this methodology, use is made of the fact that Eq. (6) is linear in $\bar{y}_i'$, and an appropriate adjoint equation can therefore be developed specifically to evaluate Eq. (3).

Defining the matrix adjoint of A as A^* and using the usual definition of this adjoint gives the identity,

$$\bar{u}^{tr} A\bar{v} = \bar{v}^{tr} A^* \bar{u} \tag{10}$$

where $\bar{u}$ and $\bar{v}$ are arbitrary vectors and A^* is defined as,

$$A^* = A^{tr} \tag{11}$$

Here the "tr" superscript represents the transpose of the vector or matrix.

If specific vectors for the problem at hand are chosen for $\bar{u}$ and $\bar{v}$, the problem-specific adjoint equation can be set up as follows,

$$A^*\bar{y}^* = \bar{s}^* \tag{12}$$

where

$$A^* \equiv A^{tr} = (I - \frac{\partial \bar{F}}{\partial \bar{y}})^{tr} \tag{13}$$

Choosing $\bar{s}^*$ appropriately as,

$$\bar{s}^* \equiv (dh/d\bar{y})^{tr} \tag{14}$$

Eq. (3) can now be evaluated as follows,

$$\frac{dR}{d\alpha_i} = \bar{y}^{*tr} \frac{\partial \bar{F}}{\partial \bar{c}} \frac{d\bar{c}}{d\alpha_i} \quad i=1,\ldots,M \tag{15}$$

where $\bar{y}^*$ is now the solution to,

$$(I - \frac{\partial \bar{F}}{\partial \bar{y}})^{tr}\bar{y}^* = (\frac{dh}{d\bar{y}})^{tr} \tag{16}$$

The simplicity of the adjoint approach lies in the fact that Eq. (16) needs to be solved only once to get any and all sensitivities in the problem. This is a result of Eq. (16) being independent of the definition of α_i. The particular choice of α_i is only reflected in the evaluation of Eq. (15) which involves only simple vector products. In essence, the adjoint approach reduces the computational effort needed to evaluate $dR/d\alpha_i$ from solving many coupled linear equations to the evaluation of several vector products. For large scale systems with many thousands or even millions of parameters, this represents orders of magnitude in computational efficiency. Problems in sensitivity analysis that were practically unapproachable with the "direct" approach, can now be done in routine fashion.

It should be noted here that both the direct and adjoint equations [i.e., (6) and (16)] are in any case far easier to solve than the original model equations [i.e., Eq. (1)]. Both Eqs. (6) and (16) are linear

while Eq. (1) is non-linear. The direct and adjoint approaches, however, require the results of the original model equations to be available in order to set up Eqs. (6) and (16), since the A matrix and the vectors $\bar{s}$, and $\bar{s}^*$ depend on $\bar{y}$.

In order to perform any sensitivity analysis, then, the model user must first generate the matrices $\partial\bar{F}/\partial\bar{y}$ and $\partial\bar{F}/\partial\bar{c}$ from the original non-linear computer model. For large-scale problems this generally requires a great deal of painstaking human effort. The model equations must be derived from the computer coding, they must then be differentiated with respect to all parameters of interest, and a direct or adjoint set of equations must then be set up for computational solution. Successful automation of this procedure could greatly reduce the human effort involved, potentially by orders of magnitude. The incentive for automation of sensitivity model development [i.e., generation of Eqs. (6) and (16)] is therefore great indeed.

III. COMPUTER CALCULUS AND THE GRESS LANGUAGE

The basic problem in automating sensitivity analysis of computer codes is the generation of the derivative matrices and source terms for Eqs. (6) and (16). For simple models, the equations which need to be differentiated can readily be identified and isolated in function or arithmetic subroutines. In this form a variety of computer calculus languages[18-21] are available for efficiently evaluating the derivatives needed for sensitivity studies. In the case of very large modeling problems, this procedure in practice (and sometimes even in principle) can not be readily used. The model equations may be very complex and tied closely to and embedded in complex model logic and data-handling routines. In addition, for non-linear problems, the numerical solution procedure often precludes an easy separation of the modeling equations from other parts of the model coding structure. Under these circumstances a different computer calculus language and approach is required to attack large-scale problems.

In order to address large models, then, a computer language is needed which treats the entire model source code as a data stream. This language must act as a compiler, with an ability to search for mathematical modeling equations and generate the derivatives necessary for sensitivity analysis.

Such a language exists in a rudimentary form as part of a software package called PROSE,[22] which was developed in part to accelerate iterative numerical procedures using derivatives and Newton's method. The concept of searching for equations and generating appropriate derivatives lies at the core of the PROSE system and was the result of an idea originally developed for NASA applications.[23] To apply the PROSE concept to the sensitivity analysis problem, however, a new language had to be developed. The resulting FORTRAN language compiler developed for this purpose is called GRESS - Gradient-Enhanced Software System.[24] A more detailed description of this language can be found in Ref. 24.

The basic underlying principle of GRESS is to read the model source program and search for model equations. These are identified uniquely by the appearance in the FORTRAN source program of the "=" symbol. Since all FORTRAN "equations" so identified occur in the form of Eq. (1) (i.e., with a single dependent variable on the left-hand-side of such an expression), GRESS can search for and analyze each equation in terms of its functional dependence on $\bar{y}$ and $\bar{c}$ [see Eq. (1)]. The basic computer calculus operations of GRESS are then used in the computation of $\partial\bar{F}/\partial\bar{c}$ and $\partial\bar{F}/\partial\bar{y}$ for each expression encountered (i.e., each component of $\bar{F}$). The resulting values of these derivatives are stored in vector form for each component of $\bar{y}$. The differentiation is carried out analytically using calculus software for all permissible FORTRAN functions and operators and the results are computed and stored numerically using the local values of the independent and dependent variables.

A simple example of this procedure is illustrated in Figs. 1 and 2. The analysis goal of this example is to calculate $dR/d\alpha_i$, where x and y are the main dependent variables of interest and α_i is the independent variable (i.e., parameter) in the problem. In the GRESS procedure, the three given equations are processed to produce derivatives with respect to the single parameter of interest. An additional analytical equation is needed to produce the derivative for each dependent variable (i.e., there are three such new equations, one for x, y, and R). These equations are produced by the GRESS software analytically with numerical values being computed and stored at execution time.

SOURCE PROGRAM	ANALYTICAL EQUATIONS
READ(5) AI	α_i = value set externally
X = 2.0 * AI	$x = 2\alpha_i$
Y = X*EXP(2.0*X)	$y = xe^{2x}$
R = X*Y**2.0	$R = xy_2$

Fig. 1. Model source program and associated equations.

GRADIENT ENHANCED SOURCE PROGRAM	ANALYTICAL EQUATIONS
READ(5) AI	α_i = value set externally
DADAI = 1.0	$d\alpha_i/d\alpha_i = 1$
X = 2.0*AI	$x = 2\alpha_i$
DXDAI = 2.0*DADAI	$dx/d\alpha_i = 2(d\alpha_i/d\alpha_i)$
Y = X*EXP(2.0*X)	$y = xe^{2x}$
DYDAI = (2.0*X+1.0)* EXP(2.0*X)*DXDAI	$dy/d\alpha_i = (2x+1)e^{2x}(dx/d\alpha_i)$
R = X*Y**2.0	$R = xy_2$
DRDAI = 2.0*X*Y*DYDAI+ DXDAI*Y**2.0	$dR/d\alpha_i = 2xy(dy/d\alpha_i)+ (dx/d\alpha_i)y_2$

Fig. 2. Gradient enhanced model and associated equations.

The use of this chain-rule propagation technique is illustrated in all the equations in the example, since intermediate derivative values (i.e., $dx/d\alpha_i$ and $dy/d\alpha_i$) are required to evaluate the derivative of R with respect to α_i. In the GRESS language, chain-rule propagation is the key to the success of the method, since current values of all the dependent-variable derivative vectors are in general needed to propagate all derivative values explicitly through the complete system of model equations. GRESS has an elaborate and yet efficient scheme for accomplishing this propagation of derivatives through loops, user defined functions and subprograms.

IV. GRESS LANGUAGE

The major advantage of the GRESS language over other calculus-based software sytems is its ability to process the model source program as data for analysis. No separate effort is needed to specify the model other than to supply GRESS with the model source program. The key procedures which allow GRESS to produce a gradient-enhanced model are its abilities to search for the model equations and to propagate derivative values by the chain-rule of differential calculus. The search for equations is accomplished, as was stated before, by identifying all FORTRAN store operations designated by the "=" symbol. In this regard GRESS only recognizes real-variable store operations as valid equations (i.e., the left-hand side variable in a FORTRAN equation must be real), since continuous derivatives are to be calculated. The sum total of these "equations" embedded in the model language constitute the mathematical model. GRESS processes only these equations and enhances this part of the code with analytic derivative expressions and numerical derivative values.

A brief description of the GRESS procedures is as follows. To start, GRESS passes over the source program identifying all the model equations and their associated dependent real variables. Using a user-defined list of parameter names, a work space is set aside for computing derivative vectors for all the dependent variables in terms of these parameters. After storage and variable-identification procedures are completed, a second pass over the program replaces all the original equations by new ones which include derivative-vector-generation capabilities. This

operation is accomplished most efficiently by generating the new equations in a set of pseudo-machine-language interpretive instructions which are written out onto a scratch file. The old equations are replaced by a call to an interpretive software subroutine for run-time evaluation of derivatives. This approach was chosen to dynamically allocate derivative storage at run-time and reduce the size of the enhanced version of the model source program. In this procedure all calls to functions and subprograms which are user-defined are treated as part of the mathematical description of the model if the called routines contain appropriately defined equations. These program branches are also replaced by calls to interpretive subroutines.

The final GRESS version of the code consists of a source program with calls to interpretive software, a set of software subroutines which support these interpretive operations and a set of pseudo-machine-language instructions representing all the dependent-variable calculations and their respective derivatives. This complete package of source programs and instructions can then be compiled and run as a normal FORTRAN program to produce both conventional model results and gradient information. The gradient information can be used internally in the model (i.e., for accelerating iterative numerical methods) or it can be output for use in sensitivity analysis. In the latter mode, GRESS is able to produce the derivative matrices $\partial\bar{F}/\partial\bar{c}$ and $\partial\bar{F}/\partial\bar{y}$ for use in setting up sensitivity problems. [See Equations (5), (15) and (16)].

V. AUTOMATED SENSITIVITY ANALYSIS

The simplest example which illustrates the computational advantage of an automated sensitivity analysis capability is a computer-coded version of Eq. (1). This non-linear equation can be solved iteratively using the following general procedure,

$$\bar{y}^{(o)} = \bar{a} \tag{17a}$$

$$\bar{y}_o^{(n+1)} = \bar{F}(\bar{y}^{(n)}, \bar{c}) \tag{17b}$$

N iterations

$$\bar{y}^{(n+1)} = \bar{G}(\bar{y}_o^{(n+1)}, \bar{y}^{(n)}, \bar{b}) \tag{17c}$$

$$R = H(\bar{y}^{(N)}) \tag{17d}$$

Here $\bar{y}^{(o)}$ represents the initial guess needed to start the iteration procedure for $\bar{y}$ and is user-specified to be the vector $\bar{a}$; Eq. (17b) is the non-linear equation being solved; Eq. (17c) represents the general acceleration technique used to achieve convergence; R is the final result of interest and $\bar{y}^{(N)}$ is the final converged solution. Note here that the number of parameters in the problem is assumed to be M, and N is the number of iterations needed to achieve convergence of the solution. In analyzing this example, three approaches will be taken, each repesenting one of the alternative sensitivity methodologies.

Perturbation Approach

Eq. (17) can be analyzed for sensitivities in the most straightforward manner by using what will be denoted as the "perturbation approach." In this procedure the derivatives (i.e., sensitivities) of interest, $dR/d\alpha_i$ i=1,...M, can be approximated by making small perturbations in the individual problem parameters (i.e., each α_i) and recalculating R for each case. The sensitivities thus calculated are approximated by,

$$\frac{dR}{d\alpha_j} \cong \frac{\Delta R}{\Delta \alpha_j} = \frac{R_j - \hat{R}}{\alpha_j - \hat{\alpha}_j} \qquad j=1,...M \tag{18}$$

where the following runs were made to achieve these results:

$$\hat{R}: \text{ base case with } \alpha_i = \hat{\alpha}_i, \quad i=1,...,M \tag{19a}$$

$$R_j: \alpha_j \text{ perturbation with } \alpha_j \neq \hat{\alpha}_j \text{ and } \alpha_i = \hat{\alpha}_i,\ i \neq j, \quad i=1,...,M \quad j=1,...M \tag{19b}$$

If we now define a single computational unit as the time required to process a single iteration in solving Eq. (17), then it is clear that the computational expense for this approach is of order (M+1)N. That is, there are M runs (one for each parameter) and one base case calculation, each requiring the solution of Eq. (17) in N iterations. For large M this procedure becomes prohibitively expensive due to the repetitive use of the iterative techniques needed to solve Eq. (17).

Direct Sensitivity Approach

The classical sensitivity analysis technique for analyzing Eq. (17) is the "direct approach" derived in Section II. In this approach, derivative equations are solved for the sensitivities of interest and derivatives are calculated along with the regular calculations using an automated derivative-enhancement compiler (i.e., GRESS).

The computational expense of this approach is easily seen to be of order (M+1)N, since in each iteration both the values of $\bar{y}^{(n+1)}$ and the $\bar{y}_i^{(n+1)}$ (i=1,...,M) must be calculated. This is the same efficiency as the perturbation approach but with the added complication of having not only to achieve convergence for $\bar{y}$ but also for each $\bar{y}_i'$. Since $\bar{y}_i'$ is accelerated using the same functional scheme used for $\bar{y}$ additional problems can arise from using the same scaling parameters for both the $\bar{y}$ and $\bar{y}_i'$ calculations or from the fact that the starting values $\bar{y}_i^{(o)'}$ will always have components equal to zero or unity depending on the definition of c_i.

To avoid this latter problem $\bar{y}_i'$ can be solved for outside the original code by doing only a single iterative pass on the system with the GRESS language <u>after</u> convergence has been achieved. In this case, the necessary derivatives for the direct method can be generated at convergence with one iteration using $\bar{y}^{(o)} = \bar{y}^{(N)}$ and $\bar{y}_i^{(o)'} = d\bar{a}/d\alpha_i$ as starting values. The direct sensitivity problem can then be solved in a new code external to the original model with a linear equation-solving package. In this simple example the direct equations to be solved are analagous to the direct sensitivity equations derived in general for a non-linear system in Section II [i.e., Eqs. (3) and (5)].

For the sake of computational analysis, Eq. (5) will also be assumed to be solved by an iterative technique requiring N' iterations. The computational cost of this alternate direct approach is then found to be of order (N+M+1+MN'). The terms here represent, in order, the cost units of the N iterations to achieve convergence of Eq. (17), the M+1 computations to get the derivatives in the N+1st iteration and the MN' computations to achieve convergence of the linear system for $\bar{y}_i'$. This approach, therefore,

has a somewhat lower computational cost than either the conventional direct approach or the perturbation method.

Despite the fact that only a single iteration is required to set up the alternate direct-approach equations, an inordinate amount of computational expense is still involved in solving Eq. (5) when M is large. The advantage of using the alternate direct-approach, however, is that the solution of Eq. (5) can be controlled independently of Eq. (17). In many cases $M' \ll N$ and significant computational advantages can be derived from separating the original and derivative equations. Nevertheless, for large M, both direct approaches and the perturbation technique have similar computational orders and remain outside the bounds of practicality for large-model analysis.

Adjoint Approach

In automating the adjoint approach to sensitivity analysis, the GRESS language is used to enhance the original model equations to calculate $\partial\bar{F}/\partial\bar{y}$, $\partial\bar{F}/\partial\bar{c}$, and $\partial h/\partial\bar{y}$. Since an adjoint procedure for this problem requires the results of the final step to start the adjoint-equation solution,[7,11] no automation procedure can readily be devised to directly set up and solve the adjoint equations simultaneously with the original iteration loop. The single iteration procedure designed for the alternate direct approach is therefore used exclusively with the adjoint approach.

In this scheme, a single pass on the gradient-enhanced model equations is needed to produce $\partial\bar{F}/\partial\bar{y}$, $\partial\bar{F}/\partial\bar{c}$, and $\partial h/\partial\bar{y}$. Using the derivation presented in Section II [see Eq. (16)], an appropriate adjoint equation for the general iterative problem can then be written as:

$$\left(I - \frac{\partial\bar{F}}{\partial\bar{y}(N)}\right)^{tr}\bar{y}^{*} = \left(\frac{\partial h}{\partial\bar{y}(N)}\right)^{tr} \qquad (20)$$

This is a single adjoint equation for $\bar{y}^{*}$, independent of the definition of α_i, which can be solved outside the original model code. The desired sensitivity results for R_i' can be computed with little additional expense using the solution to Eq. (20) as follows [i.e., see Eq. (15)].

$$R_i' = \bar{y}^{*tr} \frac{\partial \bar{F}}{\partial \bar{c}} \frac{d\bar{c}}{d\alpha_i} \qquad i=1,\ldots,M \tag{21}$$

To analyze the computational efficiency of this approach, we assume that an iterative scheme is used to solve the adjoint problem given in Eq. (20) and that the scheme requires N' iterations to achieve convergence. The total computational cost of the adjoint scheme is then of order $(N+M+1+N')$. The difference between this result and the one for the direct approach is the elimination of the need for MN' computations outside the model code. Only a single adjoint equation needs to be solved to evaluate all the R_i''s, irrespective of the number of parameters in the problem. When M becomes very large, even the computational cost of the $N+N'$ iterations become negligible and the only significant cost in this approach is involved in generating the derivatives in the single pass on the gradient-enhanced equations.

The traditional efficiency of the adjoint method is thus maintained in its automated form. The only real computational expense in this approach is the cost of computing the derivatives required to set up the adjoint equations. The gradient-enhanced version of the code generates these required derivatives at a computational cost of order M. This cost replaces the human effort which is normally required to set up these equations. There can hardly be a problem in which this computational cost will not be preferable to the many man-months of human effort otherwise required to derive the adjoint equations for a major model.

VI. CONCLUSIONS

The major conclusion of this paper is that the automation of sensitivity analysis is both feasible and practical. Automation can be most efficiently accomplished using the adjoint sensitivity approach since optimum use can be made of the computational advantages of this methodology. The GRESS compiler for automating the differentiation process is a significant step in making these advances possible. It appears to be the first full-scale use of a compiler which can process FORTRAN source codes to determine model equations and derivatives for sensitivity analysis. Since GRESS can perform search and differentiation operations on an

existing computer model, no additional arrangement, grouping or definition is needed for the model equations.

Automated sensitivity capabilities are also a major step toward making adjoint sensitivity theory a competitive analysis technique for large-scale applications. The previous drawback of this methodology (i.e., the costly development effort needed to derive the adjoint equations for a model) no longer exists with the successful development of GRESS. All the computational advantages of the adjoint approach can therefore be used to efficiently perform exhaustive sensitivity studies which otherwise might not be undertaken. Such studies find direct applications in areas directly related to current low level waste management activities such as model evaluation (e.g. determining the most important (sensitive) model parameters thus avoiding "over modeling"), site characterization (e.g. determining the characterization data to concentrate on, thus avoiding "over collection" of data), predictive simulation (e.g. providing the uncertainty of predictions given the data uncertainty) and performance assessment (e.g. providing quantitative support for better interpretation of the results and for evaluating the reliability of predictions).

A final advantage of the GRESS language is its capability of providing derivatives for use both internal and external to large-scale model codes. Existence, uniqueness, convergence and numerical studies which depend on derivative information can now be undertaken on production versions of such large models. In the past these studies could be performed only analytically or on small tractable computer models. Large-scale sensitivity analysis should put engineering design work on a much firmer foundation.

REFERENCES

1. C. R. Weisbin, et al., Advances in Nuclear Science and Technology, 14 (1982).

2. W. M. Stacey, Jr., "Variational Methods in Nuclear Reactor Physics," Academic Press, NY (1974).

3. R. Tomovic´ and M. Vukobratovic´, "General Sensitivity Theory," American Elsevier, NY (1972).

4. P. M. Frank, "Introduction to Sensitivity Theory," Academic Press, NY (1978).

5. E. Greenspan, Advances in Nuclear Science and Technology, 9, 181 (1977).

6. R. H. Myers, Response Surface Methodology, Allyn and Bacon, Inc., Boston (1971).

7. D. C. Cacuci, et al., Nucl. Sci. Eng., 75, 88 (1980).

8. E. M. Oblow, Nucl. Sci. Eng., 68, 332 (1978).

9. D. G. Cacuci, J. Math. Phys., 22, 2794 and 22, 2803 (1981).

10. E. M. Oblow, Nucl. Sci. Eng., 59, 187 (1976) and 65, 428 (1978).

11. A. Gandini, Nucl. Sci. Eng., 77, 316 (1981).

12. D. C. Cacuci, et al., Nucl. Sci. Eng., 83, 112 (1983).

13. M. Demiralp and H. Rabitz, J. Chem. Phys., 74, 3362 (1981).

14. M. Becker, Nucl. Sci. Eng., 62, 296 (1977).

15. "FORSS - A Sensitivity and Uncertainty Analysis Code System," Radiation Shielding Information Center, ORNL, Report CCC-334.

16. R. G. Alsmiller, Jr., et al., "Adjoint Sensitivity Theory and Its Application to LEAP Model 22C," ORNL/TM-7789, Oak Ridge National Laboratory (1981) see also ORNL/TM-7245 (1980).

17. M. C. G. Hall and D. G. Cacuci, J. Atmos. Sci., 39, 2038 (1982).

18. L. B. Rall, Automatic Differentiation: Techniques and Applications, Springer-Verlag, NY (1981) and extensive references therein.

19. R. Kalaba et al., Appl. Math. and Comp., 9, 227 (1981).

20. R. E. Wengert, Comm. of ACM, 7, 463 (1964).

21. K. E. Hillstrom, "JAKEF - A Portable Symbolic Differentiator of Function Given by Algorithms," ANL-82-48, Argonne National Laboratory (1982).

22. "PROSE - A Language for Modern Computers," product of PROSE, Inc., Malaga Cove Plaza, Palos Verdes Estates Calif.

23. J. Thames, "Computing in Calculus," Research and Development, 26, 24 (1975).

24. E. M. Oblow, "GRESS - Gradient Enhanced Software System Version B User's Guide," ORNL/TM-8339. Oak Ridge National Laboratory (1983).