The Simultaneous Solution and Sensitivity Analysis of Systems Described by Ordinary Differential Equations

JORGE R. LEIS and MARK A. KRAMER
Massachusetts Institute of Technology

The methodology for the simultaneous solution of ordinary differential equations and the associated first-order parametric sensitivity equations is presented, and a detailed description of its implementation as a modification of a widely disseminated implicit ODE solver is given. The error control strategy ensures that local error criteria are independently satisfied by both the model and sensitivity solutions. The internal logic effectuated by this implementation is detailed. Numerical testing of the algorithm is reported; results indicate that greater reliability and improved efficiency is offered over other sensitivity analysis methods.

Categories and Subject Descriptors: G.1.7 [**Numerical Analysis**]: Ordinary Differential Equations—*LSODE, ODESSA*; *initial value problems*; *error analysis*; *stiff equations*; I.6.4 [**Simulation and Modeling**]: Model Validation and Analysis; G.4 [**Mathematics of Computing**]: Mathematical Software—*efficiency*

General Terms: Algorithms, Measurement, Performance

Additional Key Words and Phrases: Model error, model prediction uncertainty, parameter variation, sensitivity analysis

1. INTRODUCTION

First-order sensitivity analysis involves examination of the effects of differential variations in the fixed coefficients or boundary conditions (parameters) of a mathematical model. The basic measures of sensitivity are the partial derivatives. ∂(model responses)/∂(model parameters), called the sensitivity coefficients of the model. These coefficients may be useful in uncertainty analysis and reduction of complex nonlinear models, examples being provided in the fields of chemical kinetics [1, 8] and econometrics [5]. Sensitivity calculations may also be required for gradient evaluation in optimizations [3], in experimental design and analysis [10, 20], and in many phases of chemical process design [22]. In this paper we address the calculation of sensitivity coefficients associated with systems of first-order ordinary differential equations. The algorithm falls into a class of methods

Authors' addresses: Jorge R. Leis, Shell Development Company, Westhollow Research Center, P.O. Box 1380, Houston, TX 77251-1380; Mark A. Kramer, Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139.

called explicit simultaneous methods, which have been shown to be very promising in terms of accuracy and efficiency [9]. The program ODESSA is a modification of the initial value solver LSODE [13], and shares very similar operational and portability features.

In the following section, a formal problem statement is given. This is followed in Section 3 by a summary of previous methods providing a basis for the numerical comparisons in Section 5. The current algorithm is described in Section 4.

In the accompanying paper [20], the ODESSA package is described in terms of its standard usage and optional capabilities. A description of the essential differences between ODESSA and LSODE are also provided to expedite familiarization to ODESSA for experienced LSODE users.

2. PROBLEM STATEMENT

The ODESSA modification of the LSODE package provides the capability for computing first-order sensitivity coefficients for systems of stiff or nonstiff ordinary differential equations (ODEs) of the general form:

$$\mathring{\underline{y}}(t; \underline{p}) = \frac{d\underline{y}(t; \underline{p})}{dt} = \underline{f}(\underline{y}, t; \underline{p}), \ \underline{y}(O) = \underline{y}^O, \tag{2.1}$$

where $\underline{y}$ is an N-dimensional dependent-variable vector, or model solution vector, $\underline{y}^O$ supplies the initial conditions, t is the independent variable, and $\underline{p}$ is an M-dimensional constant vector of (specified) model parameters. The governing equations for the first-order sensitivity coefficients are derived by differentiation of (2.1) with respect to the model parameters $\underline{p}$, yielding

$$\begin{gathered}\mathring{\underline{S}}(t) = \underline{J}(t)\underline{S}(t) + \frac{\partial \underline{f}(t)}{\partial \underline{p}} \\ S_{ij}^O = 1, \quad \text{if} \quad p_j = y_i^O, \quad \text{or} \quad 0, \text{ otherwise.}\end{gathered} \tag{2.2}$$

In (2.2), $\underline{S}(t)$ is the $N \times M$ sensitivity coefficient matrix $S_{ij} \equiv \partial y_i/\partial p_j$, $\underline{J}(t)$ is the $N \times N$ Jacobian matrix $J_{ij} \equiv \partial f_i/\partial y_j$, and $\partial \underline{f}(t)/\partial \underline{p}$ is an $N \times M$ matrix of partial derivatives $\partial f_i/\partial p_j$. The jth column in $\underline{S}(t)$, $\underline{s}_j(t)$, is the sensitivity solution vector for the jth model parameter p_j. Note that if p_j represents an initial condition on (2.1), then $\partial \underline{f}/\partial p_j = \underline{O}$.

Solution of (2.1) and (2.2) proceeds jointly in the ODESSA package, with information generated in the solution of (2.1) used in the solution of (2.2). The exact sequence of operations and further details are described in Section 4.

3. REVIEW OF PREVIOUS WORK

The simplest conceptual route to calculating first-order sensitivity coefficients is via a finite-difference approximation:

$$\frac{\partial \underline{y}(t)}{\partial p} \doteq \frac{\underline{y}(t; \underline{p} + \Delta p_j) - \underline{y}(t; \underline{p})}{\Delta p_j}, \tag{3.1}$$

where Δp_j is a finite perturbation in the jth element in $\underline{p}$. This method suffers from two drawbacks: the numerical values obtained may vary significantly with Δp_j [17], and repeated solution of the model (at least once for each parameter) is required. The former disadvantage may be overcome by integrating equations (2.1) and (2.2) simultaneously with an appropriate differential equation solver. For the full set of sensitivities, a combined $2N$-dimensional system must be solved M times. For some numerically stable systems, this dimensionality can be halved by decoupling (2.1) and (2.2), first solving and storing the solution to (2.1), and subsequently using this information to construct (i.e., interpolate) $\underline{J}(t)$ and $\partial \underline{f}(t)/\partial \underline{p}$ for (2.2). The decoupled system requires the solution of only $N(M + 1)$ ODEs. These two algorithms have been referred to as direct methods (DM) [6, 28].

For models in which $M > N$, it is often advantageous to use the Green's function method (GFM). The linear nature of (2.2) allows the construction of the homogeneous solution to this set of nonhomogeneous differential equations in the form of an $n \times n$ Green's function matrix. First-order sensitivities may then be expressed in integral rather than differential form. (The exact sequence of operations and other details are given in [7] and [15].) Because numerical quadrature is usually simpler than the solution of ODEs, the GFM is generally more efficient than the DM, if $M > N$.

The majority of computing time for the GFM is expended in the solution of the Green's function matrix. By exploiting the linearity of this equation, the analytically integrated Magnus (AIM) [16] modification of the GFM utilizes a specialized technique, the piecewise Magnus method (PMM), for this calculation, handling the problem as a matrix exponential rather than a succession of N, N-vector ODEs (see [16] and [17] for details). The total effort in the PMM solution of the Green's function matrix is proportional to N^3. By contrast, the effort expended in this solution by application of vector-ODE techniques is proportional to N^4. The additional factor of N significantly affects the relative efficiencies of the two approaches, favoring AIM for large problems.

The degree and complexity of the numerical manipulations and approximations required by the AIM implementation have raised concerns over its reliability. Recent research has focused on a class of algorithms designed to fully exploit the similarities between the mathematical structure of the model and the sensitivity equations in a more straightforward manner. Equations (2.1) and (2.2) have the same Jacobian, a feature that can be directly exploited in the sensitivity calculation. By applying a backward differentiation formula (BDF) [12] to the solution of (2.1) and (2.2), Lojek [24] showed that the model and sensitivity solution vectors can be jointly calculated using the same step size and order of integration. The method differs most fundamentally from the conventional DM in that the sensitivity solution vectors are computed noniteratively and concurrent with the solution of the model equations. Numerical results provided by Dunker [9] demonstrate the improved efficiency and accuracy of this new method. It was also discovered that Adams' formulas [12] can also be used to derive an explicit formula, enabling both stiff and nonstiff systems to be solved with a single implementation [22]. This method has also been extended to systems of coupled differential and algebraic equations [4, 23]. These earlier works provide the basis and motivation for this paper and the ODESSA package.

4. SOLUTION METHODOLOGY

The overall solution sequence for ODESSA given in Figure 1 is similar to a previous simultaneous sensitivity analysis method [9]. Within a successful integration step, $\underline{y}(t)$ is first advanced from t_{n-1} to t_n, followed by the propagation of $\underline{S}(t)$ using the identical step size and order of integration. This procedure is repeated across each successful integration step. The code used for the prediction, correction, and error control of the model solution vector $\underline{y}(t)$ is identical to that in LSODE (however, mandated by the sensitivity calculation, the Jacobian is never more than one step out of date). As detailed subsequently, the choice of integration step size and order are based on the behavior of both model and sensitivities solutions. As a background for the presentation of ODESSA, Section 4.1 is a brief summary of LSODE. A description of the implementation of ODESSA is contained in Section 4.2. Issues of solution stability and error control are addressed in Section 4.3. Section 4.4 contains a flow diagram depicting the sequence of calculations for the simultaneous integration of the model and sensitivity equations.

4.1 The ODE Solution

The LSODE package uses variable-order, predictor-corrector formulas for the integration of initial-value ODEs as described by (2.1). For stiff systems, Gear's backward differentiation formulas (BDFs) are used. Otherwise, Adams' method can be used. The particular method is selected by the user. The detailed mathematical treatment of these methods in the solution of initial-value problems can be found in several references [11, 12, 14]; no attempt is made here to reproduce these works. It is important to note, however, that a single representation for both Adams' formulas and BDFs exists within LSODE. This is accomplished through the use of Nordsieck arrays $\underline{U}$ [26], which store the solution history of $\underline{y}(t)$ as vectors of scaled derivatives at the current time step n:

$$\underline{U}_n = \left[\underline{y}_n \;\; h\dot{\underline{y}}_n \;\; \frac{h^2}{2!}\frac{d^2\underline{y}_n}{dt^2} \;\cdots\; \frac{h^k}{k!}\frac{d^k\underline{y}_n}{dt^k} \right]. \tag{4.1}$$

At issue are the relative convenience and efficiency of informationally equivalent implementations. The Nordsieck arrays facilitate local error estimation and, consequently, the simultaneous change of integration step size and order is easily effected through their use. They also allow a more efficient prediction, denoted by superscript (0), by use of the single equation:

$$U_n^{(0)} = \underline{D}\underline{U}_{n-1}, \tag{4.2}$$

where the matrix $\underline{D}$ is the Pascal triangle matrix. Considerable savings in computational effort may be achieved by carrying out the matrix multiplications implied in (4.2) by successive additions as suggested by Gear [11], [14].

Using the Nordsieck representation also allows a single corrector formula to be used:

$$\underline{U}_n = \underline{U}_n^{(0)} + \underline{l}\underline{a}_n^T, \tag{4.3}$$

where l is a normalized coefficient vector and superscript T denotes the transpose. The vector l is a function of the method and the integration order k. The

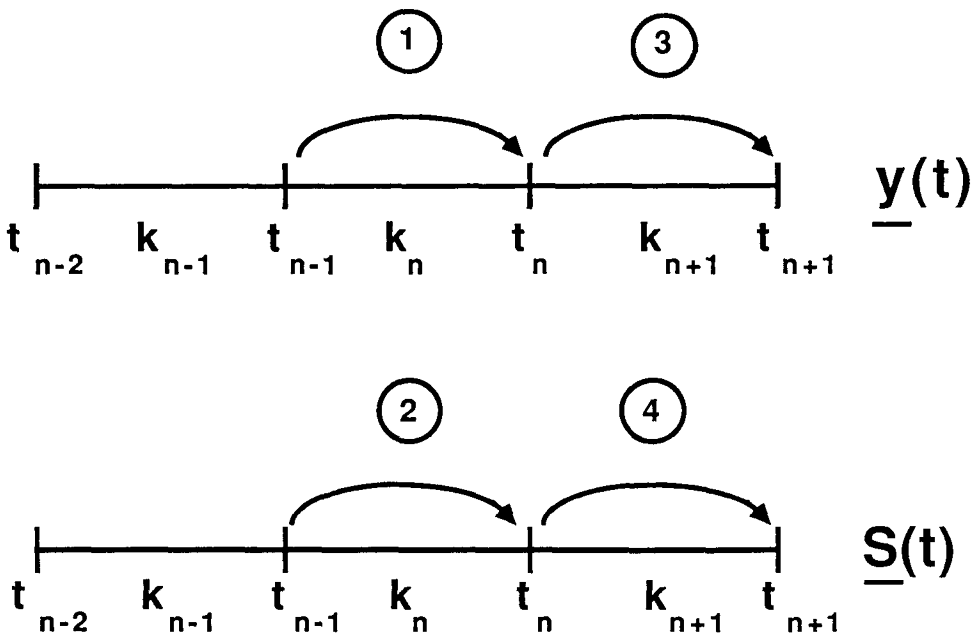

Fig. 1. Solution procedure for simultaneous sensitivity analysis. (Adapted from [9].) Circled numbers indicate sequence in which eqs. (2.1) and (2.2) are solved.

accumulated corrections vector, $\underline{a}_n$, is defined at t_n by

$$\underline{y}_n = \underline{y}_n^{(0)} + l_0\underline{a}_n, \tag{4.4}$$

$$h\overset{\circ}{\underline{y}}_n = h\overset{\circ}{\underline{y}}{}_n^{(0)} + \underline{a}_n, \tag{4.5}$$

$$h\overset{\circ}{\underline{y}}{}_n^{(0)} + \underline{a}_n = h\underline{f}(\underline{y}_n^{(0)} + l_0\underline{a}_n, t_n; \underline{p}). \tag{4.6}$$

The elements in $\underline{a}$ correspond to the scalar equations in $\underline{y}(t)$. Equations (4.4) and (4.5) are the first and second scalar equations of (4.3), respectively. Substitution of these equations into (2.1) yields (4.6), which represents the approximated solution to the ODEs at the current step t_n. In practice, the equalities in (4.6) hold only within finite tolerances that define the residuals $\underline{r}$ of the system (as functions of $\underline{a}_n$):

$$\underline{r}(\underline{a}_n) = h\underline{f}(\underline{y}_n^{(0)} + l_0\underline{a}_n, t_n) - (h\overset{\circ}{\underline{y}}{}_n^{(0)} + \underline{a}_n). \tag{4.7}$$

LSODE uses successive substitution or a modified Newton scheme to effectively satisfy the relations $\underline{r} \leq \underline{\varepsilon}_{\max}$, where $\underline{\varepsilon}_{\max}$ is a maximum error vector calculable from user-specified tolerances. (ODESSA uses the same Newton scheme but with a Jacobian that is at most one step old.) Clearly, at the predicted conditions $(\overset{\circ}{\underline{y}}{}_n^{(0)}, \underline{y}_n^{(0)}, t_n)$, the residuals are given by $\underline{r}(0)$.

Upon convergence of (4.7), the accumulated correction vector, $\underline{a}_n$, is used in the estimation of the local truncation errors, $\underline{e}_0$, in $\underline{y}(t)$ at the current integration step. The error control strategy in LSODE sets the integration step size and order to satisfy the following inequality at each integration step:

$$\frac{1}{N}\sum_{i=1}^{N}\left(\frac{\underset{\sim}{e}_{0,i}}{\underset{\sim}{\omega}_{0,i}}\right)^2 \le 1.0, \tag{4.8}$$

where the subscript i denotes the ith element of a vector. The above expression is the weighted root-mean-square norm of the error, subsequently denoted simply as $\| \underset{\sim}{e}_0/\underset{\sim}{\omega}_0 \|$. Here, N equals the number of ODEs in (2.1), and $\underset{\sim}{\omega}_0$ is the N-dimensional error weighting vector that is selected by the user to represent relative and/or absolute error control.

4.2 The Sensitivities Solution

The underlying idea of simultaneous methods can be illustrated by replacing the derivatives in (2.2) by a difference approximation. Rearrangement of the resulting expression yields an explicit formula for the solution of $\underline{S}(t)$ at t_n. For example, if the simple backward difference,

$$\overset{\circ}{\underline{S}}_n \doteq \frac{\underline{S}_n - \underline{S}_{n-1}}{h}, \tag{4.9}$$

is used, (2.2) can be rearranged to yield

$$\underline{S}_n = [\underline{I} - h\underline{J}_n]^{-1}\left[\underline{S}_{n-1} + h\left(\frac{\partial \underline{f}}{\partial \underline{p}}\right)_n\right], \tag{4.10}$$

where h $(=t_n - t_{n-1})$ is the integration step size, and all other terms are as previously defined in (2.2). Note that the required derivatives are evaluated at the current integration step, t_n, implying the solution of the model equations prior to the evaluation of equation (4.10). In practice, the sensitivities are calculated using the same integration step size (and integration order) as the model solution. This is accomplished without introducing additional storage by updating the sensitivities concurrently with the model solution. Also the coefficient matrix $(\underline{I} - h\underline{J}_n)$ is not inverted, but rather LU-decomposed. $\underline{S}_n$ is calculated by (repeated) back substitution (followed by forward substitution).

The algorithms in ODESSA are an extension of this approach. Instead of using the first-order formula (4.9), ODESSA approximates the derivative using the appropriate kth-order formula. (A detailed derivation using the original variable-order BDFs and Adams' formulas is presented in [23].) Integration step size and order are selected to satisfy local error criteria, with a secondary constraint added to secure the convergence of the corrector equations for $\underline{y}(t)$.

In ODESSA, as in LSODE, the integration formulas are represented in terms of Nordsieck arrays, defined for each parameter of interest. Prediction and correction are handled analogously to (4.2) and (4.3). The accumulated correction is defined in an equivalent manner to $\underset{\sim}{a}_n$ for the model solution. Rearrangement of the resulting residuals for the sensitivity equations results in an *explicit formula* for the solution of $\underset{\sim}{s}(t)$ at t_n:

$$\underset{\sim}{s}_n = [\underline{I} - hl_0\underline{J}_n]^{-1}(\underset{\sim}{s}_n^{(0)} - hl_0\overset{\circ}{\underset{\sim}{s}}{}_n^{(0)} + hl_0\partial\underline{f}_n/\partial p). \tag{4.11}$$

The Jacobian in (4.11), $\underline{J}_n$, is (usually) evaluated at current solution conditions $\underline{y}_n$, t_n. However, depending on the convergence of the model solution, $\underline{J}_n$ may be evaluated at t_n using the predicted values of $\underset{\sim}{y}_n$. Numerical tests indicate that no noticeable loss in accuracy results, and computational savings are realized with fewer Jacobian evaluations and decompositions necessary to solve a problem.

4.3 Error Control

Errors associated with the sensitivity calculations are computed in the equivalent manner to the errors associated with the model solution vector: Local errors are assumed to be proportional to the difference between corrected and predicted values. For a single parameter, p_1:

$$\underset{\sim}{e}_1 \propto \underset{\sim}{s}_n - \underset{\sim}{s}_n^{(0)}. \tag{4.12}$$

For the multiple parameter case, ODESSA controls the local errors according to inequalities of the form:

$$\| \underset{\sim}{e}_j / \underset{\sim}{\omega}_j \| \leq 1.0 \quad (j = 1, \ldots, M), \tag{4.13}$$

where the column vectors $\underset{\sim}{e}_j$ contain the estimated local errors for the corresponding sensitivity solution vectors $\underset{\sim}{s}_j$, and where $\underset{\sim}{\omega}_j$ are the vectors of weights selected by the user to represent relative and/or absolute error control for the sensitivities $\underset{\sim}{s}_j(t)$. We refer to the implementation where the inequalities in (4.13) and (4.8) are satisfied at every step in the integration as the mutually dependent error control strategy [22].

There are several possible alternative approaches. The model error vector $\underset{\sim}{e}_0$ can be augmented with the sensitivity error vectors $\underset{\sim}{e}_j$, $j = 1, \ldots, M$, to form a column vector $\underset{\sim}{E}$, to which the vector–norm inequality

$$\| \underset{\sim}{E} / \underset{\sim}{W} \| \leq 1.0 \tag{4.14}$$

is applied. This method is used in [4], which is a sequential sensitivity analysis modification of the DASSL [27] solver. Although computational effort is largely unaffected, numerical tests indicate that, at any given step, a majority of the inequalities in (4.13) are satisfied, thereby *compensating for larger errors* with respect to certain parameters in (4.14). This problem can be circumvented by using the maximum norm in place of the weighted RMS norm; however this yields unnecessarily stringent error control [12]. Another alternative, referred to as fully-dependent error control [22], is to rely on the error control provided in the solution of the model, as in [9]. This removes any direct control over errors in the sensitivity solution, and provides no guarantee against accumulation of errors in $\underset{\sim}{S}(t)$.

In the mutually dependent implementation, the dynamic behavior of both the model and the sensitivities are taken into account by applying an error test to each solution vector $\underset{\sim}{y}(t)$ and $\underset{\sim}{s}_j(t)$, $j = 1, \ldots, M$. This offers several advantages. Foremost among these are that the inequalities in (4.13) are satisfied on every step. This provides assurance that errors in the sensitivities are controlled. Secondly, since solution propagation, error estimation, and error checking can proceed independently for each solution vector, error test failures can be detected sooner on a given step, saving the computational expenses of correction, error estimation, and error checking associated with the solution vectors not yet considered. On average this will save the expense of considering half of the solution vectors on an integration step that is going to fail. If the error test fails for any vector $\underset{\sim}{e}_j$, $j = 1, \ldots M$, the step is immediately repeated beginning with the selection of a new step size, and possibly a new order of integration, depending on the severity of the error violation. Additionally, we have found that decreasing the tolerances on $\underline{y}(t)$ is less effective in improving the accuracy of $\underline{S}(t)$ than providing error control directly on $\underline{S}(t)$. Lastly, error tolerances can be specified independently on each solution vector, which may be useful in reducing the number of integration steps required to solve a problem. ODESSA stores the number of repeated integration steps attributable to any single solution vector to assist in setting individual tolerances. Mutually dependent implementations offer the greatest reliability and, of these, the ODESSA implementation is most reliable offering a higher degree of flexibility and (on average) greater efficiency.

4.4 Calculational Sequence

A detail of the sequence of calculations within ODESSA is presented in Figure 2. It should be immediately noticed that several nested loops exist with the initial task of selecting the integration step size and integration order. At the beginning of a problem, the order is set to one, and the initial step size is calculated as a function of the error tolerances supplied and the initial output time requested by the user. Optionally, the initial step size can be directly supplied by the user.

After the integration step size and order have been selected, the model and sensitivity solutions are predicted for at the next integration step using equation (4.2) (and the analogous form for the sensitivities). In ODESSA, the prediction step for both the model and sensitivities is carried out simultaneously. It is conceivable that greater efficiency can be achieved if the prediction of the sensitivities is performed after the model solution converges. This obviates the need to retract the predicted sensitivities to their values before prediction if the model solution fails to converge with the current step size. However, the savings are not seen to be significant. Prediction (or retraction) is carried out by successive additions (or subtractions), which is relatively inexpensive.

Only those components of the Nordsieck arrays contained in the residual (4.7), namely, $\underline{y}^{(0)}$ and $h\dot{\underline{y}}^{(0)}$, need be used in the model correction step. ODESSA provides two options to calculate $\underset{\sim}{a}_n$: successive substitution and Newton's method. If a sensitivity analysis is being performed, Newton's method must be used. Application of Newton's method to (4.7) yields the following iteration formula:

$$(\underset{\sim}{a}_n^{(m+1)} - \underset{\sim}{a}_n^{(m)}) = [\underline{I} - hl_0\underline{J}]^{-1}\underline{r}(\underset{\sim}{a}_n^{(m)}). \qquad (4.15)$$

The superscripts $(m + 1)$ and (m) refer to the iteration count. Comparing (4.15) with (4.11) reveals that the iteration matrix in (4.15) is of the exact same form as the coefficient matrix in (4.11). Since this matrix must be evaluated at least once per step as required by (4.11), the iteration matrix in (4.15) is never more than one step out of date, i.e., $\underline{J}_n$ in (4.15) is generally $\underline{J}_{n-1}$. It sometimes happens, however, that even at the previously corrected values $(\underline{y}_{n-1}, t_{n-1})$, the iteration matrix must be updated to converge the corrector at current conditions (y_n, t_n). If this occurs, the Jacobian is evaluated at the current predicted conditions $(\underline{y}_n^{(0)}, t_n)$. If the corrector still fails to converge, the step size is reduced, the

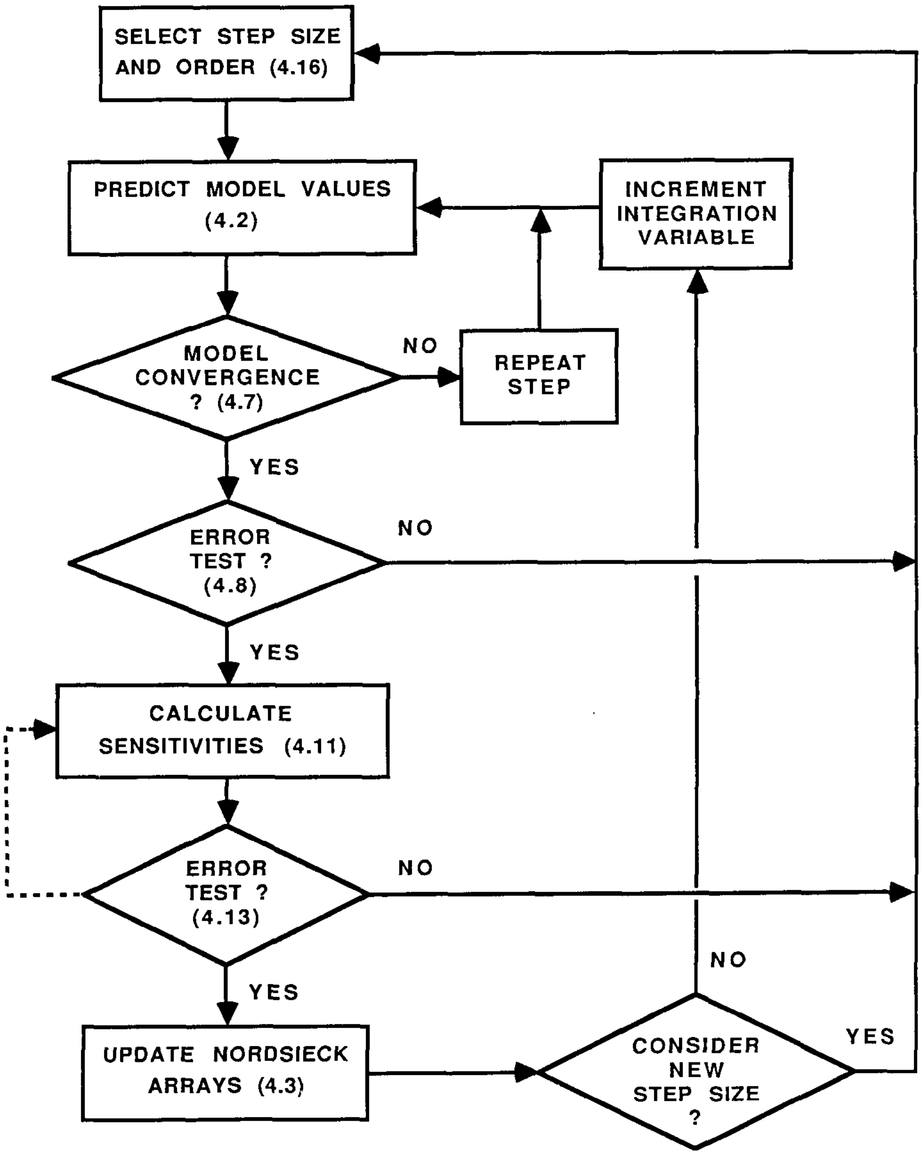
SELECT STEP SIZE AND ORDER (4.16)
PREDICT MODEL VALUES (4.2)
INCREMENT INTEGRATION VARIABLE
MODEL CONVERGENCE ? (4.7)
NO
REPEAT STEP
YES
ERROR TEST ? (4.8)
NO
YES
CALCULATE SENSITIVITIES (4.11)
ERROR TEST ? (4.13)
NO
NO
YES
UPDATE NORDSIECK ARRAYS (4.3)
CONSIDER NEW STEP SIZE ?
YES

Fig. 2. Calculational sequence of explicit simultaneous sensitivity analysis as implemented in ODESSA. Relevant equations appear in parentheses.

predicted values are retracted, and the step is tried again, beginning with the prediction of $\underline{y}(t)$ and $\underline{S}(t)$ using the now current step size. This involves a rescaling of the Nordsieck arrays $\underline{U}$ and $\underline{Z}$ at the previous step t_{n-1}. The maximum number of iterations allowed in ODESSA, if a sensitivity analysis is being performed, is four.

Following corrector convergence, the local error associated with $\underline{y}(t)$ is estimated and the error check is made. If the inequality in (4.8) is satisfied, the sensitivity calculations proceed. Otherwise the errors associated with the integration formula for one lower order are estimated, followed by the selection of a new step size, and possibly a lower order. The step is then retried until it succeeds, or until the smallest allowable step size with $k = 1$ fails. In the latter case, an error message is printed, and program execution is halted.

Correction (4.11), error estimation (4.12), and error checking (4.13) proceed serially for the sensitivity solution vectors $\underset{\sim}{s}_j(t), j = 1, \ldots, M$, as depicted by the dashed line in Figure 2. Failures of the error test initiate changes of integration order and step size, which are obtained from (analogous to Gear [14]):

$$\begin{aligned} \alpha &= \frac{1}{1.2}\left(\frac{1}{\max_j \| \underset{\sim}{e}_j/\underset{\sim}{\omega}_j \|}\right)^{1/(k+1)}, && j = 0, \ldots, M \text{ (order } k) \\ \alpha &= \frac{1}{1.3}\left(\frac{1}{\max_j \| \underset{\sim}{e}_j/\underset{\sim}{\omega}_j \|}\right)^{1/k}, && j = 0, \ldots, M \text{ (order } k-1) \qquad (4.16) \\ \alpha &= \frac{1}{1.4}\left(\frac{1}{\max_j \| \underset{\sim}{e}_j/\underset{\sim}{\omega}_j \|}\right)^{1/(k+2)}, && j = 0, \ldots, M \text{ (order } k+1) \end{aligned}$$

where α is the ratio of a new step size for the respective order to the current step size. The estimates of α are made after $k + 1$ steps have been taken since the last change in order or step size, or if the local error criterion is not satisfied (except in the latter case no attempt is made to increase the order). The desired value of α is the maximum value computed by use of (4.16). Like LSODE, ODESSA does not increase the step at the current order k unless the increase is greater than 10 percent, since a smaller increase is seen as not worth the computer time to perform it.

After the successful error check of all the solution vectors, the Nordsieck arrays are updated, and the integration continues. If $k + 1$ steps have been taken since the last change in order or step size, a new order and step size is selected by (4.16). Otherwise, the same step size is used to advance the solution.

5. NUMERICAL TESTING AND DISCUSSION

In this section, we apply ODESSA to two example problems. The first example is provided by a CO oxidation mechanism [29]. This mechanism is comprised of 52 reactions involving 11 species. In an earlier study [17], this problem was used as the basis of comparison between the AIM (analytically integrated Magnus), GFM (Green's function method), DM (direct method), and FD (finite difference) methods. The second example is provided by an ethane pyrolysis mechanism [18], comprised of 7 species and 5 reactions. Earlier studies [9, 16] have applied the AIM, GFM, DM, FD, and DDM (decoupled direct method) methods to calculate sensitivity coefficients for this problem.

5.1 CO Oxidation Example

In this example, we compare ODESSA and AIM, which was the most efficient algorithm in the previous study [17]. Performance comparisons between these methods is complicated by different error control strategies. To circumvent this problem, the error tolerances for each method were chosen to provide an averaged error of about 2 percent in the normalized sensitivities, $\partial \ln y_i/\partial \ln p_j$, whose values are greater than 10^{-3}. The results obtained are shown in Figure 3, which

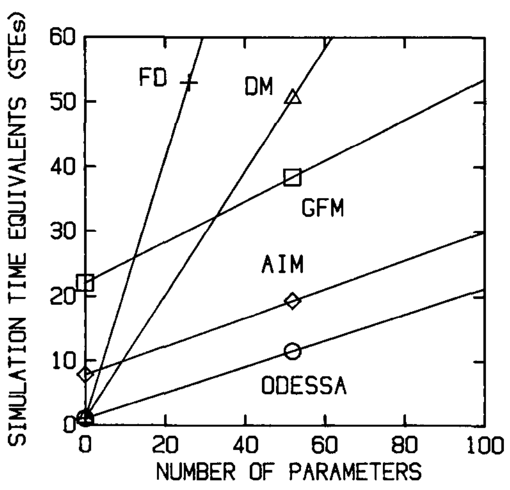

Fig. 3. Relative execution times for first-order sensitivities versus number of parameters analyzed. Methods shown are ODESSA (O), AIM (◇), GFM (□), DM (Δ), and FD (+).

is adapted from the original study. The computation time units are in simulation time equivalents (STEs). One STE equals the computational expense of solving the kinetics problem alone, which in this study equals 22.6 VAX-11/750 CPU seconds.

While AIM has a significant fixed cost associated with calculation of Green's functions, ODESSA has no such fixed cost. In addition, ODESSA had generally similar or lower marginal costs per sensitivity parameter. (Since each parameter requires a separate vector solution of (2.2), the effort required is an approximately linear function of the number of parameters, which was confirmed experimentally). Further efficiencies in ODESSA can be achieved by suppying the analytical derivatives $\partial \underline{f}/\partial \underline{p}$. Since the frequency of derivative calculations in ODESSA is generally greater than in AIM, analytical derivatives decrease the marginal cost per parameter in ODESSA more than in AIM. For this example, an improvement of one STE was realized by supplying the analytical derivatives. In all cases the analytical Jacobian was supplied.

In ODESSA, the overall expense for providing error control of $\underline{S}(t)$ through mutually dependent error control added only about 10 percent to the cost of the sensitivity calculation.

5.2 Ethane Pyrolysis Example

This relatively small-scale system is very stiff, and represents a significant challenge to sensitivity algorithms. In fact, both the DM and the GFM were unable to determine the full set of sensitivities with reasonable accuracy and efficiency [16]. The difficulties encountered by the GFM were overcome by the AIM modification [16], but still greater efficiency was reportedly achieved by the DDM [9]. Improvements in accuracy were also reported. The following compares ODESSA, DDM, and AIM with major emphasis on error control strategies.

We begin by considering the solution of the kinetic model by LSODE (ODESSA without sensitivities). In Table I, the error in $\underline{y}(t)$, the number of steps (NST), and the computation time (CPU seconds) are given as a function of the relative

Table I. Error in $\underline{y}(t)$ from Independent Kinetic Solution as a Function of the Specified Error Tolerances

RTOL[a]	$E(\underline{y})$[b]	NST[c,d]	CPU (sec)[d,e]
10^{-3}	405.6	5	0.26
10^{-4}	1112.8	25	1.47
10^{-5}	155.4	16	0.72
10^{-6}	0.5	15	0.65
10^{-7}	223.4	19	0.87
10^{-8}	0.1	51	2.02

[a] ATOL = RTOL * 10^{-2}; RTOL = Relative Tolerance, ATOL = Absolute Tolerance.
[b] See equation (5.1).
[c] Number of integration steps.
[d] Values reported at $t = 20.0$ seconds.
[e] Timing studies conducted on a VAX 11/750.

error tolerance RTOL. (The absolute error tolerance ATOL set at RTOL/100 throughout this study. ATOL and RTOL are used to calculate the error weights used in equation (4.8).) The error $E(\underline{y})$ is an average over the two output times, $t = 1.0$ and 20.0 seconds:

$$E(\underline{y}) = \frac{100}{14} \sum_{i=1}^{7} \left\{ \frac{|y_i^*(1.0) - y_i(1.0)|}{|y_i^*(1.0)|} + \frac{|y_i^*(20.0) - y_i(20.0)|}{|y_i^*(20.0)|} \right\}. \qquad (5.1)$$

The exact solution, denoted by superscript * , is obtained from an independent run with RTOL = 10^{-10}. The reasons for the nonmontic decrease in $E(\underline{y})$ as the tolerances are decreased are unclear, but the abnormally high accuracy in $\underline{y}(t)$ achieved with RTOL set to 10^{-6} is quite important because in comparing the AIM method and the DDM, Dunker [9] reports his findings at these tolerances. Not until RTOL is reduced below 10^{-8} do the errors in $\underline{y}(t)$ actually stabilize, that is, decrease monotonically, under various conditions. The results reported in [9] were therefore obtained under peculiar conditions conducive to fully dependent error control on $\underline{S}(t)$, and must be considered somewhat fortuitous.

In a separate series of ODESSA runs reported in Table II, the error in $\underline{y}(t)$ was again calculated as a function of the specified error tolerances. In these experiments, however, a sensitivity analysis was also performed. The sensitivities were calculated using the same tolerances applied to the solution of the model. The number of repeated steps (NRS) due to the sensitivity analysis are also listed; in each case, the rate constant k_1, for example, $\underline{s}_1(t) = -\partial \underline{y}(t)/\partial k_1$, is responsible. The simultaneous solution of kinetic and sensitivity equations requires a greater number of integration steps indicating the sensitivities are controlling the joint solution. In other words, *indirect error control provided by the solution $y(t)$ is inadequate for the solution $S(t)$*. Additionally, the mutually dependent error control strategy had a substantially positive influence on the stability of the solution, as indicated by the monotonically decreasing errors in

Table II. Error in $\underline{y}$ from Joint Solution as a Function of the Specified Error Tolerances

RTOL[a]	E($\underline{y}$)[b]	NST[c,d]	NRS[d,e]	CPU (sec)[d,f]
10^{-3}	154.5	20	4	3.33
10^{-4}	122.6	102	22	17.51
10^{-5}	9.5	57	10	10.10
10^{-6}	0.1	44	1	7.66
10^{-7}	0.0	63	2	11.14
10^{-8}	0.0	90	2	16.47

[a] ATOL = RTOL * 10^{-2}; RTOL = Relative Tolerance, ATOL = Absolute Tolerance.
[b] See equation (5.1).
[c] Number of integration steps.
[d] Values reported at $t = 20.0$ seconds.
[e] Number of repeated steps due to sensitivity analysis.
[f] Timing studies conducted on a VAX 11/750.

Table III. Error in $\underline{y}$ and $\underline{s}_1(t)$ as a Function of the Specified Error Tolerances

RTOL($\underline{y}$)[a]	RTOL($\underline{s}_1$)[b]	E($\underline{y}$)[c]	E($\underline{s}_1$)[c]	NST[d,e]	NRS[e,f]	CPU (sec)[e,g]
10^{-8}	10^{-8}	0.0	0.0	90	2	16.47
10^{-8}	10^{-6}	0.1	0.1	51	0	8.83
10^{-6}	10^{-6}	0.1	0.1	44	1	7.66
10^{-6}	10^{-5}	0.2	0.3	30	2	5.37

[a] ATOL = RTOL * 10^{-2}; RTOL = Relative Tolerance, ATOL = Absolute Tolerance.
[b] Tolerances for $\underline{s}_j$, $j = 2, \ldots, 5$, equal tolerances for $\underline{y}$.
[c] See equation (5.1).
[d] Number of integration steps.
[e] Values reported at $t = 20.0$ seconds.
[f] Number of repeated steps due to sensitivity analysis.
[g] Timing studies conducted on a VAX 11/750.

Table II. The computational effort associated with mutually dependent error control on $\underline{S}(t)$ proved to be less than 5 percent of the total computational effort.

Mutually dependent error control in ODESSA allows considerable flexibility in specifying the error tolerances. Table III contains the errors in $\underline{y}(t)$ and $\underline{s}_1(t)$ as a function of the error tolerances applied to each solution vector. Equivalent tolerances were applied to $\underline{s}_j(t)$, $j = 2, \ldots, 5$, as to $\underline{y}(t)$. The results show four sets of tolerances, each yielding solutions that are accurate to within $\frac{1}{2}$ percent, but varying considerably in the computational effort required. The first and third entries correspond to conditions from Table II where equivalent tolerances were applied to $\underline{y}(t)$ and $\underline{S}(t)$. The second entry demonstrates that indirect error control provided by $\underline{y}(t)$ may be sufficient for the joint solution. However, greater efficiency is achieved if the direct error control on $\underline{S}(t)$ is active. (Compare the second and the final entries.) The final entry further demonstrates that the joint solution is governed by the requirements of $\underline{s}_1(t)$. Increasing the tolerances on this solution vector alone results in still less computational effort. These results

Table IV. Computational Effort for Initial Condition Sensitivities*

Set of I.C. Sensitivities	NST[a]	CPU (sec)[a,b]
Complete set[c]	88	14.81
Complete set[d]	75	12.70
$\partial \underline{y}/\partial[C_2H_6]_0$[c,e]	51	9.47
$\partial \underline{y}/\partial[C_2H_6]_0$[c,f]	28	4.68
Complete set[d,g]	15	2.60

* Reported values are for averaged errors no greater than 1% except where noted.

[a] Values reported at $t = 20.0$ seconds.

[b] Timing studies conducted on a VAX 11/750.

[c] For absolute values greater than 10^{-6}.

[d] For absolute values greater than 10^{-3}.

[e] Fully dependent error control: RTOL($\underline{y}$) = 10^{-8}, ATOL($\underline{y}$) = 10^{-10}.

[f] Mutually dependent error control: RTOL($\underline{y}$) = 10^{-6}, ATOL($\underline{y}$) = 10^{-8}; RTOL($\partial \underline{y}/\partial[C_2H_6]_0$) = ATOL($\partial \underline{y}/\partial[C_2H_6]_0$) $=10^{-1}$, all other tolerances set to 10^4.

[g] Fully dependent error control: RTOL($\underline{y}$) = 10^{-6}; ATOL($\underline{y}$) = 10^{-8}; averaged errors as low as 1.8% for $\partial \ln \underline{y}/\partial \ln[C_2H_6]_0$.

indicate that mutually dependent error control, as implemented in ODESSA, offers the greatest flexibility to minimize the computational effort required for the solution of any given problem.

The relative performance of ODESSA on the coupled set of rate constant and initial condition sensitivities was also examined. Initial results indicated that a savings in computational effort of about 15 percent was realized by ODESSA over the AIM method. However since only the normalized initial condition sensitivities with respect to $[C_2H_6]_0$ are nonzero, the initial error criterion, which was based on the normalized sensitivities, proved to be unreasonable. In subsequent experiments, ODESSA was used to calculate the initial condition sensitivities under various conditions subject to global error criteria based on the unnormalized quantities. Selected results, presented in Table IV, show that ODESSA requires considerable effort to accurately calculate the full set of initial condition sensitivities. In fact, considerably greater effort is required than that which is predicted by the marginal cost of successive rate constant sensitivities. Although these observations were unique to this example, they suggest that the AIM method may offer computational savings for certain problems where the full set of initial condition sensitivities is desired.

6. SUMMARY

The successful implementation of ODESSA has provided improved reliability at greater efficiency than other sensitivity analysis methods, and has added a degree of flexibility previously unattained. The algorithm utilizes proven numerical methods and retains the operational and portability features of the widely accepted software package LSODE upon which it is based.

REFERENCES

1. ATHERTON, R. W., SCHAINKER, R. B., AND DUCOT, E. R. On the statistical sensitivity analysis of models for chemical kinetics. *AIChE J. 21*, 3 (May 1975), 441–448.
2. BARD, Y. *Nonlinear Parameter Estimation.* Academic Press, New York, 1974.
3. BEVERIDGE, G. S. G., AND SCHECHTER, R. S. *Optimization: Theory and Practice.* McGraw-Hill, New York, 1970.
4. CARACOTSIOS, M., AND STEWART, W. E. Sensitivity and analysis of initial value problems with mixed ODE's and algebraic equations. *Comput. Chem. Eng. 9*, 4 (1985), 359–365.
5. DE JONGH, D. C. J. Structural parameter sensitivity of the 'limits of growth' world model. *Appl. Math. Model. 2* (June 1978), 77–80.
6. DICKINSON, R. P., AND GELINAS, R. J. Sensitivity analysis of ordinary differential equation systems—a direct method. *J. Comput. Phys. 21* (1976), 123–143.
7. DOUGHERTY, E. P., AND RABITZ, H. A computational algorithm for the Green's function method of sensitivity analysis in chemical kinetics. *Int. J. Chem. Kinet. XI* (1979), 1237–1248.
8. DOUGHERTY, E. P., AND RABITZ, H. Computational kinetics and sensitivity analysis of hydrogen-oxygen combustion. *J. Chem. Phys. 72*, 12 (June 1980), 6571–6586.
9. DUNKER, A. M. The decoupled direct method for calculating sensitivity coefficients in chemical kinetics. *J. Chem. Phys. 81* (1984), 2385–2393.
10. ENO, L., BEUMEE, J. G. B., AND RABITZ, H. Sensitivity analysis of experimental data. *Appl. Math. Comput. 16* (1985), 153–163.
11. GEAR, C. W. The automatic integration of ordinary differential equations. *Commun. ACM 14* (March 1971), 176–179.
12. GEAR, C. W. *Numerical Initial Value Problems in Ordinary Differential Equations.* Prentice-Hall, Englewood Cliffs, N.J., 1971.
13. HINDMARSH, A. C. LSODE and LSODI, two new initial value ordinary differential equation solvers. *ACM-SIGNUM Newsl. 15*, 4 (1980), 10–11.
14. HOLLAND, C. D., AND LIAPIS, A. I. *Computer Methods for Solving Dynamic Separation Problems.* McGraw-Hill, New York, 1983.
15. HWANG, J. T., DOUGHERTY, E. P., RABITZ, S., AND RABITZ, H. The Green's function method of sensitivity analysis in chemical kinetics. *J. Chem. Phys. 69*, 11 (Dec. 1978), 5180–5191.
16. KRAMER, M. A., CALO, J. M., AND RABITZ, H. An improved computational method for sensitivity analysis: Green's function method with 'AIM'. *Appl. Math. Model. 5* (Dec. 1981), 432–441.
17. KRAMER, M. A., RABITZ, H., CALO, J. M., AND KEE, R. J. Sensitivity analysis in chemical kinetics: Recent developments and computational comparisons. *Int. J. Chem. Kinet. 16* (1984), 559–578.
18. LAYOKUN, S. K., AND SLATER, D. H. Mechanism and kinetics of propane pyrolysis. *Ind. Eng. Chem. Process Des. Dev. 18*, 2 (1979), 232–236.
19. LEHMAN, S. L., AND STARK, L. W. Three algorithms for interpreting models consisting of ordinary differential equations: Sensitivity coefficients, sensitivity functions, global optimization. *Math. Biosciences 62* (1982), 107–122.
20. LEIS, J. R., AND KRAMER, M. A. Algorithm 658: ODESSA—An ordinary differential equation solver with explicit simultaneous sensitivity analysis. *ACM Trans. Math. Software.* This issue, pp. 61–67.
21. LEIS, J. R., AND KRAMER, M. A. Sensitivity analysis in process design. ILP 4-15, Massachusetts Institute of Technology, Dept. of Chemical Engineering, Feb. 1985.
22. LEIS, J. R., AND KRAMER, M. A. Sensitivity analysis of general differential systems. AIChE National Meeting, San Francisco, Cal., paper 27c (Nov. 1984).
23. LEIS, J. R., AND KRAMER, M. A. Sensitivity analysis of systems of differential and algebraic equations. *Comput. Chem. Eng. 9*, 1 (1985), 93–96.
24. LOJEK, B. Sensitivity analysis of non-linear circuits. *IEE Proceedings 129*, Pt. G. (1982), 85–88.
25. MAGNUS, W. On the exponential solution of differential equations for a linear operator. *Commun. Pure Appl. Math.*, 7 (1954) 649.
26. NORDSIECK, A. On numerical integration of ordinary differential equations. *Math. Comp. 16*, 1 (Jan. 1962), 22–49.

27. PETZOLD, L. R. A description of DASSL: A differential/algebraic system solver. Sandia Report SAND82-8637, Sandia, Sept. 1982.
28. TOMOVIC, R., AND VOKOBRATOVIC, M. *General Sensitivity Theory*. Elsevier, New York, 1972.
29. YETTER, R., DRYER, F., AND RABITZ, H. Some interpretive aspects of elementary sensitivity gradients in combustion kinetics modelling. *Combustion and Flame 59*, 2 (Feb. 1985), 107–133.

Received March 1986; revised February 1987 and November 1987; accepted January 1988