ELSEVIER

Applied Numerical Mathematics 32 (2000) 161–174

APPLIED NUMERICAL MATHEMATICS

www.elsevier.nl/locate/apnum

Sensitivity analysis of differential–algebraic equations: A comparison of methods on a special problem ☆

Shengtai Li [a], Linda Petzold [a,*], Wenjie Zhu [b]

[a] *Department of Computer Science, University of California, Santa Barbara, CA 93106-5070, USA*
[b] *Department of Computer Science, University of Minnesota, Minneapolis, MN 55455, USA*

Abstract

We compare several methods for sensitivity analysis of differential–algebraic equations (DAEs). Computational complexity, efficiency and numerical conditioning issues are discussed. Numerical results for a chemical kinetics problem arising in model reduction are presented. © 2000 IMACS. Published by Elsevier Science B.V. All rights reserved.

Keywords: Differential–algebraic equations; Sensitivity analysis; Automatic differentiation

1. Introduction

Sensitivity analysis of large-scale differential–algebraic systems is important in many engineering and scientific applications, such as chemical, mechanical and electrical engineering and economics. Sensitivity analysis generates essential information for parameter estimation, optimization, control, model simplification and experimental design. Consequently, algorithms which perform such an analysis in an efficient and rapid manner are invaluable to researchers in many fields.

Usually, the DAE and sensitivity systems are solved in sequence, taking into account the linearity of the sensitivity equations. These methods are so-called *staggered direct methods* [3,5,7,9,11,17]. On each time step, first the state variables of the DAE system are computed by the nonlinear corrector step and then the linear sensitivity system is solved directly. This method is sometimes considered to be inefficient because it needs to evaluate and factor the Jacobian matrix at each step, which is often the most costly part of the computation. However, as we will point out later, for some special problems, this may not be the case. We will also see that for problems with an ill-conditioned iteration matrix, the original implementation [3] of the method can be unreliable.

☆ This work was partially supported by DOE contract number DE-F603-98ER25354 and Exxon Research and Engineering Co., with computing resources from the DOE National Energy Research Scientific Computing Center and the Minnesota Supercomputer Institute.

* Corresponding author. E-mail: petzold@engineering.ucsb.edu

PII: S0168-9274(99)00020-3

Another class of methods is the so called *simultaneous corrector* method of Maly and Petzold [13]. In this method the state variables and the sensitivity variables are solved simultaneously in the nonlinear corrector step. In many cases, this method can be as efficient as the staggered corrector method which we will talk about later, because it evaluates and factors the Jacobian matrix only when necessary.

Most recently, a new method has been proposed by Feehery and Barton [6]. This method is called the *staggered corrector* method. It is similar to the staggered direct method. The important difference is that instead of using the factorization of the Jacobian matrix computed at the current step directly to solve the linear sensitivity equations, it uses the factorization of the Jacobian matrix at some past step and Newton iteration to solve the linear sensitivity system. Thus, it saves on the number of factorizations of the Jacobian matrix, which can be the most expensive part of the computation for large problems.

There has often been a tacit assumption in the sensitivity analysis literature that the size of the DAE system is much larger than the number of sensitivity parameters. However, in some optimization and optimal control problems the number of optimization parameters, i.e., sensitivity parameters, can be much larger than the number of equations in the DAE system [15,16].

In this paper, we focus on a class of application problems where the size of the original DAE system is moderate but there are many sensitivity parameters. Our newly-designed software DASPK3.0 [12] for solving DAEs and their sensitivities is used in all of our examples. Although one integration of the system is not costly, in a design optimization the original and sensitivity systems will be integrated many times. Thus, efficiency is a very important issue. We will compare the existing sensitivity analysis methods. Besides efficiency, numerical stability is also an important consideration.

In [6,12], the simultaneous and/or staggered corrector methods were compared with the staggered direct method. It was concluded that both of the corrector methods were more efficient than the staggered direct method, and the staggered corrector method is slightly more efficient than the simultaneous corrector method for most problems. However, if the cost of a function evaluation is more than the cost of factorization of the Jacobian matrix (e.g., for the chemical kinetic problem considered in Section 3), the staggered direct method [3] can be more efficient, although it may fail for an ill-conditioned problem. In this paper we describe how to modify the implementation of the staggered direct method of [3] so that it is as reliable for ill-conditioned problems as the staggered corrector method.

In either of the previous generation of DASSL-based solvers (DASSL, DASPK or their sensitivity analysis solvers DASSLSO [13] and DASPKSO), finite differencing (forward or central) is chosen as the default method to evaluate the Jacobian matrix for the direct method, the matrix–vector product in the Krylov iteration, and the residuals for the sensitivity equations in case exact input is not available. For most well-scaled and smooth problems, the results using finite differencing are nearly as good as for exact input. However, for some badly-scaled problems, finite-differencing cannot get accurate results for the sensitivities. For some strongly nonlinear problems, exact input of the Jacobian in the direct method can greatly improve the accuracy and efficiency. The automatic differentiation tool ADIFOR [1] can generate a subroutine to compute the Jacobian matrix and/or the sensitivity equations with accuracy up to round-off error. In our experience, ADIFOR-generated derivative code usually outperforms divided-difference approximations. In DASPK3.0 [12], we provide an option to use ADIFOR to compute the derivatives. We embed the ADIFOR-generated routine in such a way that the previous user interface is not altered very much.

1.1. Sensitivity analysis problem

To illustrate the basic approach for sensitivity analysis, consider the general DAE system with parameters,

$$F(t, y, y', p) = 0, \qquad y(0) = y_0, \tag{1}$$

where $y \in \mathbb{R}^{n_y}$, $p \in \mathbb{R}^{n_p}$. Here, n_y and n_p are the dimension and the number of parameters in the original DAE system, respectively. Sensitivity analysis entails finding the derivative of the above system with respect to each parameter. This produces an additional $n_s = n_p \cdot n_y$ sensitivity equations which, together with the original system, yields

$$F(t, y, y', p) = 0, \tag{2}$$

$$\frac{\partial F}{\partial y} s_i + \frac{\partial F}{\partial y'} s_i' + \frac{\partial F}{\partial p_i} = 0, \quad i = 1, \ldots, n_p, \tag{3}$$

where $s_i = \partial y / \partial p_i$. There are three well-established direct methods to solve the system (3) as we mentioned before. Other methods such as the Green's function method will not be discussed here.

In the following discussion, we assume that the basic algorithm for the DAE problem (1) is DASSL [2]. For each time step, a predictor polynomial that interpolates the solution at previous time steps is used to obtain an initial guess $y_{n+1}^{(0)}$ and $y_{n+1}^{\prime(0)}$. Then a modified Newton iteration is used to solve the corrector formula for y_{n+1}. DASSL uses the fixed leading coefficient form of the kth order BDF formula for the corrector [2]. For all three sensitivity methods, the predictor step for the sensitivity equations is the same but the corrector formula is solved in different ways which we will describe in detail.

There are two ways of using ADIFOR to generate the sensitivity residuals (3); we will investigate both of them. The first way is to generate the matrix–vector products in (3) directly (without computing the Jacobian on each time step) via the seed-matrix option of ADIFOR. The second possibility is to form the Jacobian matrices via ADIFOR and then to multiply by the vector in (3). We will call this the *matrix times vector method*.

1.2. Simultaneous corrector method

Defining $Y = [y, s_1, \ldots, s_{n_p}]^{\mathrm{T}}$ and

$$\boldsymbol{F} = \left[F(t, y, y', p), \frac{\partial F}{\partial y} s_1 + \frac{\partial F}{\partial y'} s_1' + \frac{\partial F}{\partial p_1}, \ldots, \frac{\partial F}{\partial y} s_{n_p} + \frac{\partial F}{\partial y'} s_{n_p}' + \frac{\partial F}{\partial p_{n_p}} \right]^{\mathrm{T}},$$

the combined system can be rewritten

$$\boldsymbol{F}(t, Y, Y', p) = 0, \qquad Y(0) = \begin{pmatrix} y_0 \\ \dfrac{\partial y_0}{\partial p_1} \\ \vdots \\ \dfrac{\partial y_0}{\partial p_{n_p}} \end{pmatrix}. \tag{4}$$

The simultaneous corrector method solves (4) as one nonlinear dynamic system without making use of the linearity of the sensitivity equations and the fact that the linear sensiti vity equations can be obtained exactly after solving for the state variables.

Approximating the solution to the combined system by a numerical method, for example, the kth order BDF formula with step size h_{n+1}, yields the nonlinear system for the corrector iteration

$$G(Y_{n+1}) = \boldsymbol{F}\left(t_{n+1}, Y_{n+1}, Y'^{(0)}_{n+1} - \frac{\alpha_s}{h_{n+1}}\left(Y_{n+1} - Y^{(0)}_{n+1}\right), p\right) = 0 \tag{5}$$

for Y_{n+1}, where $Y^{(0)}_{n+1}$ and $Y'^{(0)}_{n+1}$ are predictor values of Y_{n+1} and Y'_{n+1}, which are obtained via interpolation of previous solution values [2]. Also, α_s is the fixed leading coefficient which is defined in [2] and is not important to our discussion here.

Newton's method for the nonlinear system produces the iteration

$$Y^{(k+1)}_{n+1} = Y^{(k)}_{n+1} - \boldsymbol{J}^{-1} G\left(Y^{(k)}_{n+1}\right), \tag{6}$$

where

$$\boldsymbol{J} = \begin{bmatrix} J & & & & \\ J_1 & J & & & \\ J_2 & 0 & J & & \\ \vdots & \vdots & \vdots & \ddots & \\ J_{n_p} & 0 & \dots & 0 & J \end{bmatrix}, \tag{7}$$

$$J = \alpha\frac{\partial F}{\partial y'} + \frac{\partial F}{\partial y}, \quad J_i = \frac{\partial J}{\partial y}s_i + \frac{\partial J}{\partial p_i} \quad \text{and} \quad \alpha = -\alpha_s/h_{n+1}.$$

In the direct option of DASPK3.0, the full Jacobian matrix $\boldsymbol{J}$ is not actually computed. Instead, it is approximated by its block diagonal in the Newton iteration. It has been shown in [13] that the resulting iteration is two-step quadratically convergent for full Newton, and convergent for modified Newton iteration.

We assume here that all first order partial derivatives can be obtained analytically via the automatic differentiation software ADIFOR. Thus the Jacobian matrix J and sensitivity equations are obtained analytically without using the finite difference method, which can be unreliable in some circumstances.

In the simultaneous corrector method, the Jacobian matrix J is evaluated and factored only when the current version does not converge or is expected to not converge the Newton iteration. However, this is at the cost of having a Jacobian matrix $\boldsymbol{J}$ which is complex and which is approximated by its block diagonal part. This approximation may increase the number of Newton iterations needed to converge.

By using the seed-matrix option of ADIFOR, the simultaneous corrector method can avoid computing the full Jacobian matrix and hence save computation when there are just a few parameters. Even though the Jacobian matrix is not evaluated on each step, the sensitivity equations are obtained implicitly using the recently evaluated Jacobian matrix at the latest Newton iteration. This ensures the accuracy of the sensitivity equations.

1.3. Staggered direct method

Noting that the differential equations for the state variables are independent of the sensitivity variables, the staggered direct method first solves the differential equations for the state variables at each time step.

After the Newton iteration for the state variables has converged, the coefficient matrices of the sensitivity equations are obtained at the current step. This requires updating the Jacobian matrix every time step. In this way the sensitivity equations are obtained exactly at each step. The linear sensitivity equations are discretized with the same numerical scheme as the state variables. For example, using the kth order BDF method with step-size h_{n+1} as before, we solve, for the ith parameter,

$$\frac{\partial F}{\partial y'_{n+1}}\left(s'^{(0)}_{i_{(n+1)}} - \frac{\alpha_s}{h_{n+1}}\left(s_{i_{(n+1)}} - s^{(0)}_{i_{(n+1)}}\right)\right) + \frac{\partial F}{\partial y_{n+1}} s_{i_{(n+1)}} + \frac{\partial F}{\partial p_i} = 0, \tag{8}$$

for $s_{i_{(n+1)}}$.

Because (8) is linear, it can be solved for $s_{i_{(n+1)}}$ directly without using Newton iteration

$$J s_{i_{(n+1)}} = \left(-\frac{\partial F}{\partial y'_{n+1}}\beta_i - \frac{\partial F}{\partial p_i}\right), \tag{9}$$

where $\beta_i = s'^{(0)}_{i_{(n+1)}} - \alpha s^{(0)}_{i_{(n+1)}}$. However, to solve the linear system in this way requires computation and factorization of the Jacobian matrix at each step and also extra storage for the matrix $\partial F/\partial y'_{n+1}$.[1] Moreover, this implementation may fail when the matrix J is ill-conditioned, which happened in our examples in Section 3. This is because the right-hand side of Eq. (9) can be very large and can introduce large round-off errors when J is ill-conditioned, and the large round-off errors can cause the error tests in ODE/DAE solvers to fail.

To handle ill-conditioned problems, we propose the following modification. Rather than solving (9) directly, instead solve

$$J\delta = J s^{(0)}_{i_{(n+1)}} + \frac{\partial F}{\partial y'_{n+1}}\beta + \frac{\partial F}{\partial p_i}, \tag{10}$$

where $\delta = s^{(0)}_{i_{(n+1)}} - s_{i_{(n+1)}}$. The right-hand side of (10) is easy to obtain by the function evaluations of the sensitivity equations. It does not require any extra storage and special handling. What is important is that the right-hand side now becomes much smaller than that of Eq. (9), and the round-off error will be much reduced.

We note that in practice, the Jacobian matrix is evaluated using the predicted values of the state variables y and y' and then factored. This new Jacobian is then used in the Newton iterations for the state variables and in forming the sensitivity equations. Since the order of the predictor is the same as the corrector, the Jacobian matrix evaluated at the predicted values is an accurate approximation to the Jacobian evaluated after the corrector step. In this way the sensitivity equations are obtained accurately and also the Jacobian is always current for the Newton iterations for the state variables. Thus there are fewer Newton iterations for the state variables for problems with a rapidly changing Jacobian matrix.

[1] Alternatively, the seed-matrix option of ADIFOR2.0 can also be used to obtain the matrix–vector product $(\partial F/\partial y'_{n+1})\beta_i$ to improve the efficiency and avoid the storing of $\partial F/\partial y'_{n+1}$.

1.4. Staggered corrector method

The staggered corrector method is similar to the staggered direct method. However, instead of solving the linear system (8) directly, a Newton iteration is used:

$$s_{i_{(n+1)}}^{(k+1)} = s_{i_{(n+1)}}^{(k)} - \widehat{J}^{-1}\left(J s_{i_{(n+1)}}^{(k)} + \frac{\partial F}{\partial y'_{n+1}}\beta_i + \frac{\partial F}{\partial p_i}\right), \tag{11}$$

where $\widehat{J}$ may be a factored Jacobian matrix which is saved from a past step and used in the Newton iteration for the state variables, while J is the current unfactored Jacobian. The updated Jacobian will be factored if the current version can not converge the Newton iteration. Using this method, an extra copy of the unfactored updated Jacobian J should be stored for computing the residuals in later iterations as in [6]. However, as in the simultaneous corrector method, we can use the seed-matrix option of ADIFOR to avoid evaluating the Jacobian and matrix–vector product, thus the Jacobian matrix is evaluated and factored only when necessary. This can substantially improve the efficiency of the staggered corrector method when there are just a few parameters and/or the function evaluation is not costly. However, in the case of a large number of parameters and costly function evaluation, evaluating the Jacobian first and then computing the sensitivity equations via the matrix times vector method may be more efficient. We will give an analysis in the next section.

2. Comparison of the methods

A comprehensive comparison of these methods is difficult if not impossible. Since we are interested in the differences between the methods, we will focus on their computational complexity and numerical performance for a particular class of problems from chemical kinetics. We will assume that all of the derivatives are computed analytically, to simplify the comparison. In all three methods, only first order derivatives are actually used.

2.1. Computational complexity

We consider the computational complexity of these three methods at each time step. The total number of time steps these methods take differs from problem to problem; we will discuss this later.

The only differences between these three methods are how the Jacobian matrices are updated and factored and how the sensitivity equations are obtained. We mainly consider using the seed-matrix option of ADIFOR2.0. However, the order of the total cost for both the seed-matrix and matrix times vector methods turns out to be similar as we will see later.

As in [6], we first introduce in Table 1 some notation for costs to facilitate our discussion. We use C_{lin} to denote the cost for solving the linear system of size n_y.

For the simultaneous corrector method, suppose that the Jacobian is evaluated and factored every N_{l} steps and the number of Newton iterations for the state variables and sensitivity variables are the same: N_{it_0}. The natural choice for evaluating the sensitivity equations is the seed-matrix option of ADIFOR. At each step the cost for this computation is

$$C_{\text{SimCor}} = \frac{C_{\text{eval}} + C_{\text{fac}}}{N_{\text{l}}} + N_{\text{it}_0}(C_{\text{lin}}) + N_{\text{it}_0}(n_p C_{\text{lin}} + C_{\text{seed}} + C_{\text{par}}). \tag{12}$$

Table 1
Notation for costs

C_{fun}	residual evaluation for F
C_{par}	evaluation of $[\partial F/\partial p_1, \ldots, \partial F/\partial p_{n_p}]$
C_{fac}	factorization of Jacobian matrix J
C_{eval}	evaluation of Jacobian matrix J
C_{mvp}	matrix times vector of J and $[s_1, \ldots, s_{n_p}]$
C_{seed}	seed-matrix option for $J \times [s_1, \ldots, s_{n_p}]$
C_{lin}	solving the linear system using back-substitution

The matrix times vector method is not a good choice for the simultaneous corrector method because it requires the matrix evaluation at each Newton iteration.

Since the Jacobian is updated and factored every step and no iteration is used when solving the linear system, the cost for the staggered direct method is

$$C_{\text{StaDir}} = C_{\text{eval}} + C_{\text{fac}} + N_{\text{it}_1}(C_{\text{lin}} + C_{\text{fun}}) + n_p C_{\text{lin}} + C_{\text{seed}} + C_{\text{par}}, \tag{13}$$

where N_{it_1} is the number of Newton iteration only for state variables with an updated and factored Jacobian at every time step. For this method, the matrix–vector product method for evaluating sensitivity equations makes sense and the C_{seed} term in the above cost formula can be replaced by C_{mvp} to obtain the cost using the matrix times vector method, which yields

$$C_{\text{StaDir2}} = C_{\text{eval}} + C_{\text{fac}} + N_{\text{it}_1}(C_{\text{lin}} + C_{\text{fun}}) + n_p C_{\text{lin}} + C_{\text{mvp}} + C_{\text{par}}. \tag{14}$$

Suppose that the frequency of factoring the Jacobian for the staggered corrector method is the same as for the simultaneous corrector method, and the number of Newton iterations for the state variables and sensitivity variables are N_{it_2} and N_{it_3}, respectively. Since $\partial F/\partial p_i$ is computed only once and will be saved for later iterations, the cost of the staggered corrector method is

$$C_{\text{StaCor1}} = \frac{C_{\text{eval}} + C_{\text{fac}}}{N_{\text{l}}} + N_{\text{it}_2}(C_{\text{lin}} + C_{\text{fun}}) + N_{\text{it}_3}(n_p C_{\text{lin}} + C_{\text{seed}}) + C_{\text{par}}. \tag{15}$$

For this method, if the matrix times vector method is used to evaluate the sensitivity equations, the Jacobian will be computed at every step and the cost will be

$$C_{\text{StaCor2}} = C_{\text{eval}} + \frac{C_{\text{fac}}}{N_{\text{l}}} + N_{\text{it}_2}(C_{\text{lin}} + C_{\text{fun}}) + N_{\text{it}_3}(n_p C_{\text{lin}} + C_{\text{mvp}}) + C_{\text{par}}. \tag{16}$$

Note that we distinguish the number of Newton iterations for each method with different subscripts. For a general problem, their relationship usually satisfies $N_{\text{it}_1} \leqslant \max(N_{\text{it}_2}, N_{\text{it}_3}) \leqslant N_{\text{it}_0}$.

2.2. *Efficiency*

From the computational complexity of the three methods we can see that the simultaneous corrector method will be nearly as efficient as the staggered corrector method for problems where C_{eval} and C_{fac} dominate the costs, i.e., when $n_y \gg n_p$, if the seed matrix option is used. The main extra cost for the

simultaneous method is that it needs to compute $\partial F/\partial p_i$ at every nonlinear iteration. This extra cost can be eliminated by using the values obtained at the predictor, which will not affect the accuracy of the solution. Another consideration is that, as pointed out in [6], when the old Jacobian matrix can not converge the Newton iteration or the error test is not satisfied with the current step size, only the work on the state variables is wasted for the staggered corrector method, whereas the simultaneous corrector method will waste more work on the sensitivity variables.

To compare the costs of (15) and (16), we consider the problem of evaluating the Jacobian J of a vector function F with respect to an n vector of variables y. Suppose that $\partial F/\partial p$ is also evaluated by ADIFOR. The cost of evaluating J is related to C_{fun} by

$$C_{\text{eval}} \simeq a \cdot n \cdot C_{\text{fun}},$$

where $a = 3$ for the basic forward mode of automatic differentiation. If only a product of the Jacobian with some vector p is required, the cost of $J \cdot p$ is

$$\text{COST}(J \cdot p) \simeq a \cdot C_{\text{fun}}.$$

If the sparse forward mode (with SparsLinC option) is used in automatic differentiation, the cost satisfies

$$C_{\text{eval}} \simeq a \cdot n' \cdot C_{\text{fun}},$$

where n' is the maximum number of nonzero entries in any row of the Jacobian. Suppose the average number of nonlinear iterations is n_i, and the number of sensitivity parameters is n_p. Then the cost of ADIFOR with the seed matrix option includes n_p matrix–vector products for each nonlinear iteration. The total cost can be approximated by

$$n_i n_p \cdot a_1 \cdot C_{\text{fun}}.$$

The cost of direct evaluation by the matrix times vector method includes the cost of evaluation of the Jacobian and the two matrix times vector operations, which is

$$a_2(m' + n'_p) \cdot C_{\text{fun}} + n_i n_p m' n,$$

where m' and n'_p are the maximum number of nonzero entries in any row of the Jacobian ($\partial F/\partial y$, $\partial F/\partial y'$), and ($\partial F/\partial p$), respectively. We use a_1 and a_2 here to distinguish the coefficients for two different approaches because a_1 is usually much larger than a_2. The matrix times vector method is better when

$$(m' + n'_p)a_2 \cdot C_{\text{fun}} + n_i n_p m' n < n_i n_p a_1 \cdot C_{\text{fun}},$$

which results in

$$n_p > \frac{(m' + n'_p)a_2}{n_i a_1}, \tag{17}$$

and

$$C_{\text{fun}} > \frac{n_i n_p m' n}{n_i n_p a_1 - (m' + n'_p)a_2}. \tag{18}$$

Eqs. (17) and (18) imply two conditions for the matrix times vector method to be advantageous over the seed matrix option. First n_p must be large enough. For example, if $a_1 = a_2$, $n'_p = n_p$, and $n_i = 2$, then $n_p > m'$, which means the number of sensitivity parameters must be larger than the maximum number of nonzero entries in any row of the Jacobian ($\partial F/\partial y$, $\partial F/\partial y'$). The second condition is that the evaluation of the function F must be costly enough, which is defined by (18).

Since the Jacobian is updated at each time step for the staggered direct method, the matrix times vector method should be better than the seed-matrix option in ADIFOR if the function evaluation is expensive. There is the possibility that the staggered direct method can perform better than the other two methods. Comparing Eqs. (14) and (16), we find that if

$$C_{\text{fac}}\left(1-\frac{1}{N_{\text{l}}}\right) < (N_{\text{it}_3}-1)(n_p C_{\text{lin}} + C_{\text{mvp}}), \tag{19}$$

then the staggered direct method is better than the staggered corrector method with the matrix times vector option. Suppose the half bandwidth of the Jacobian is n_b,

$$C_{\text{fac}} \simeq 2n_y n_b^2, \qquad C_{\text{lin}} \simeq 2n_y n_b, \qquad C_{\text{mvp}} \simeq 2n_p n_y n_b. \tag{20}$$

After substituting (20) into (19), we obtain

$$2(N_{\text{it}_3}-1)n_p > (1-1/N_{\text{l}})n_b. \tag{21}$$

Eq. (21) implies that the number of Newton iterations for the sensitivity variables (N_{it_3}) must be larger than 1, and the number of parameters n_p must be large enough. This situation does occur in a class of model reduction problems [16] as we will see later.

Our numerical experiments with a class of chemical model reduction problems indicate that the staggered corrector method has fewer nonlinear iteration failures and more error test failures than the simultaneous corrector method. The reason for fewer nonlinear iteration failures is that with the staggered corrector method, when the state variables can be solved without trouble, so can the sensitivity variables: a Newton iteration failure for the sensitivity variables usually occurs when the Jacobian is out of date and once an updated factorization of the Jacobian matrix is obtained, there will usually be no Newton iteration failure for the sensitivity variables. The reason for more error test failures is not fully understood.

3. Numerical experiments

In this section we compare the numerical performance of the three methods using a chemical model reduction problem [16] as an example. In this problem, an optimization method is used to reduce the number of reactions in the original chemical mechanism to obtain a reduced mechanism which can approximate the important features of the original mechanism accurately. This aids in understanding of the original mechanism; also the reduced mechanism can be used in place of the original one in later computations to save computational work. A norm, which is chosen according to the future use of the reduced mechanism, is to be minimized by the optimizer. The gradient information about the norm, needed in the optimization, is obtained via sensitivity analysis of the differential equations defining the norm.

The species concentrations and temperature can be described by a system of ordinary differential equations

$$y' = \sum_{r=1}^{N} S_r F_r(y), \qquad y(0) = y_0, \tag{22}$$

where y is the vector of species concentrations and temperature, and $S_r \in \mathbb{R}^n$ is the stoichiometric vector for reaction r. Here n is the dimension of y, N is the number of reactions in the original mechanism and $F_r(y)$ is the reaction rate of reaction r.

Thus the optimization problem is to find a set of parameters $d_i = 0$ or 1, $i = 1, \ldots, N$ which defines the approximate system

$$z' = \sum_{r=1}^{N} S_r d_r F_r(z), \qquad z(0) = y_0, \tag{23}$$

and makes the error norm $||y - z||$ as small as possible. A reaction is kept in the reduced mechanism if its corresponding parameter is 1 and it is deleted from the original mechanism if the value of the parameter is 0. To achieve a reduction of the number of reactions, we also require $\sum_{r=1}^{N} d_r = k \ll N$.

We can see from the definition of the optimization problem that the number of optimization parameters is much larger than the number of ODEs. This is due to the fact that the number of reactions in a chemical mechanism is usually much larger than the number of species.

3.1. Example 1

The first numerical experiments are done on an already reduced chemical mechanism which is obtained from the Exxon model [14] which contains 116 species and 447 reversible reactions. The reduced model contains 45 species and 79 reactions. We try to reduce this model further. This is a relatively small problem. The sensitivity analysis results are listed below. Details about the dynamic system and parameter values are omitted. The number of state variables in this problem is 46, and the number of parameters is 54. The residual function is highly nonlinear and very expensive to evaluate.

We first introduce some notation in Table 2 for different methods. In the following tables, the nonlinear iterations for the simultaneous corrector methods are counted for state variables and sensitivity variables together, since in this method the Newton iteration is performed on these variables simultaneously. For the staggered corrector methods, since separate Newton iterations are used for the state and sensitivity variables, the nonlinear iterations are counted separately ("Non. iter." is used for state variables and "Non. iter. SA" is used for sensitivity variables). All of the computations are performed on a Linux86 PC with 450 MHz CPU.

In Table 3, the staggered corrector methods and the simultaneous methods have a few more nonlinear iteration failures or error test failures than the staggered direct methods. This results in more time steps for these two methods. However, since the number of parameters is very small, the gain from fewer Jacobian evaluations and factorizations for these two methods with the seed matrix option is more than the loss from more time steps. But we will see that this is only the case when the number of parameters is small compared with the size of the original system.

Table 2
Notation for different methods

SDS	Staggered direct method with seed-matrix option
SDM	Staggered direct method with matrix–vector product
STCS	Staggered corrector method with seed matrix option
STCM	Staggered corrector method with matrix–vector product
SICS	Simultaneous corrector method with seed-matrix option

In this example, the staggered corrector and simultaneous corrector methods with the seed-matrix option are better than the matrix times vector method. This is due to the fact that the number of sensitivity parameters is just one. Note that the simultaneous corrector method has more convergence test failures than other methods. These convergence test failures generate more nonlinear iterations. However, it is comparable in efficiency with the staggered corrector method.

In Table 4, we show the results for a sensitivity analysis with 54 parameters. We can see that the seed-matrix option is no longer better than the matrix times vector methods. This is because the number of parameters (54) is larger than the number of equations (46). The staggered direct method is more efficient than the staggered corrector methods for this example. This is because the cost of function evaluations dominates that of Jacobian evaluations and factorizations for this example.

Table 3
Example 1. Sensitivity analysis results for one parameter

Methods	SDS	SDM	STCS	STCM	SICS
Time steps	583	648	982	1006	916
Jac. eva.	592	657	86	85	135
Jac. fac.	592	657	86	1091	135
Err. fail.	9	9	44	39	40
Non. iter.	1181	1310	1397	1392	1572
Non. iter. SA	588	655	1358	1333	1572
Non. it. fail.	0	0	0	0	12
CPU sec.	10.26	9.5	4.4	15.60	4.51

Table 4
Example 1. Sensitivity analysis results for 54 parameters

Methods	SDS	SDM	SDM2*	STCS	STCM	SICS
Time steps	984	1020	6549	1641	1665	1713
Jac. eva.	992	1025	9792	105	1770	106
Jac. fac.	992	1025	9792	105	105	106
Err. fail.	8	5	3243	48	46	30
Non. iter.	1611	1653	13900	2011	2032	2532
Non. iter. SA	992	1025	9792	2198	2214	2532
Non. it. fail.	0	0	0	1	1	2
CPU sec.	46.04	29.59	257.41	66.91	52.67	74.33

*The SDM2 represents the original staggered direct method of [3] with matrix times vector method. The output time is 0.0093 for SDM2 and 1.0 for the others.

We should point out that this problem is ill-conditioned. The original implementation (SDM2 in Table 4) of the staggered direct method [3] took excessively small steps (about 10^{-6}) and we stopped the integration at time 0.0093. We noticed that SDM2 had a larger number of error test failures and Jacobian factorizations. We also compared the solutions at this time point and found that the solution given by SDM2 was similar to those of the other methods but with largest relative error.

3.2. Example 2

A similar experiment on the original Exxon model has also been done. In this example, there are about 117 equations in the original DAE system and about 447 sensitivity parameters.

From Table 5, we can see that the performance of the staggered direct qmethod with two different options is almost the same. However, for the staggered corrector method, the seed matrix option is much more efficient than the matrix times vector option, and is even more efficient than the staggered direct method. This is because we have only one sensitivity parameter. It may not be true after we increase the number of parameters, as we will see in the next table. We also note that the simultaneous corrector method is as efficient as the staggered corrector method when the seed matrix option is chosen.

Comparing the results in Table 6, we find that the matrix times vector method is much more efficient than the seed matrix option, in contrast to the results of Table 5. We note that the test problems here have the property that the number of differential equations for the state variables is not very large and the function evaluation is very expensive. Thus the cost of the Jacobian evaluation and matrix factorization does not dominate. The condition (18) is satisfied. The other reason why the matrix times vector method is better for this example is that the number of sensitivity parameters ($= 447$) is much larger than the number of equations ($= 117$) and the condition (17) is satisfied. Since both conditions are satisfied, the matrix times vector method should be more efficient than the seed matrix option according to our analysis in Section 2.2.

As in example 1, this problem cannot be solved effectively by the original staggered direct method [3].

Table 5
Example 2. Sensitivity analysis results for one parameter

Methods	SDS	SDM	STCS	STCM	SICS
Time steps	840	879	1306	1611	1079
Jac. eva.	848	889	98	1713	91
Jac. fac.	848	889	98	102	91
Err. fail.	6	8	25	35	24
Non. iter.	1380	1436	1691	2019	1699
Non. iter. SA	846	887	1843	2214	1699
Non. it. fail.	2	2	2	2	2
CPU sec.	176.94	184.81	59.09	333.65	42.46

Table 6
Example 2. Sensitivity analysis results for 447 parameters

Methods	SDS	SDM	STCS	STCM	SICS
Time steps	998	997	1692	1594	1581
Jac. eva.	1006	1004	101	1712	77
Jac. fac.	1006	1004	101	118	77
Err. fail.	6	5	33	35	12
Non. iter.	1142	1492	1988	1919	2412
Non. iter. SA	1004	1002	2357	2261	2412
Non. it. fail.	2	2	2	2	2
CPU sec.	3494.69	782.47	7506.32	1502.56	7282.20

4. Conclusion

In this paper, we compared the complexity and efficiency of three sensitivity analysis methods on a special class of problems. We found that the simultaneous corrector method can be made nearly as efficient as the staggered corrector method with the seed-matrix option of ADIFOR, provided that the diagonal approximation of the full Jacobian converges the Newton iteration well, which happens for most problems.

We found that for problems with an ill-conditioned iteration matrix, the original implementation [3] of the staggered direct method can be unreliable. However, the modified staggered direct method is efficient and reliable for problems which have a large number of sensitivity parameters and a very costly function evaluation that dominates the cost of the Jacobian factorization, or for problems with a rapidly changing iteration matrix.

For evaluation of the sensitivity equations, the matrix times vector method is more efficient than the seed matrix option of ADIFOR when the two requirements (17) and (18) we proposed are satisfied. Roughly speaking, this will be the case for problems with costly function evaluations and a number of sensitivity parameters which is large relative to the maximum number of nonzero entries in any row of the Jacobian. Because the relative efficiencies of the sensitivity analysis methods are so problem-dependent, we have included all of the methods as options in DASPK3.0. Based on our overall experience, here are our recommendations. For problems with few parameters, use the staggered corrector or simultaneous corrector method with the seed-matrix option. For problems with many parameters, the staggered direct or staggered corrector method with the matrix times vector option will often be the best choice.

References

[1] C. Bischof, A. Carle, G. Corliss, A. Griewank, P. Hovland, ADIFOR—Generating derivative codes from Fortran programs, Scientific Programming 1 (1992) 11–29.

[2] K.E. Brenan, S.L. Campbell, L.R. Petzold, Numerical Solution of Initial-Value Problems in Differential–Algebraic Equations, 2nd Edition, SIAM, Philadelphia, PA, 1995.

[3] M. Caracotsios, W.E. Stewart, Sensitivity analysis of initial value problems with mixed ODEs and algebraic equations, Comput. Chem. Engrg. 9 (4) (1985) 359–365.
[4] D. Clancey, Automatic differentiation in the numerical solution and sensitivity analysis of differential algebraic equations, in preparation, Department of Computer Science and Engineering, University of Minnesota.
[5] J.E. Dennis, R.B. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Prentice-Hall, Englewood Cliffs, NJ, 1983.
[6] W.F. Feehery, J.E. Tolsma, P.I. Barton, Efficient sensitivity analysis of large-scale differential–algebraic systems, Appl. Numer. Math. 25 (1997) 41–54.
[7] P.E. Gill, W. Murray, M.H. Wright, Practical Optimization, Academic Press, New York, 1981.
[8] G.H. Golub, C.F. Van Loan, Matrix Computations, 2nd Edition, Johns Hopkins, Baltimore, MD, 1989.
[9] E.J. Haug, R. Wehage, N.C. Barman, Design sensitivity analysis of planar mechanism and machine dynamics, Trans. ASME 103 (1981).
[10] C.T. Kelley, C.T. Miller, M.D. Tocci, Termination of Newton/Chord iterations and the method of lines, SIAM J. Sci. Comput. 19 (1998) 280–290.
[11] M.A. Kramer, J.R. Leis, The simultaneous solution and sensitivity analysis of systems described by ordinary differential equations, ACM Trans. Math. Software 14 (1988) 45–60.
[12] S. Li, L.R. Petzold, Design of new DASPK3.0 for sensitivity analysis, Technical Report, Department of Computer Science, University of California, Santa Barbara, 1999.
[13] T. Maly, L.R. Petzold, Numerical methods and software for sensitivity analysis of differential–algebraic systems, Appl. Numer. Math. 20 (1997) 57–79.
[14] C.A. Mims, R. Mauti, A.M. Dean, K.D. Rose, Radical chemistry in methane oxidative coupling: tracing of ethylene secondary. Reactions with computer models and isotopes, J. Phys. Chem. 98 (1994) 13357–13372.
[15] L.R. Petzold, J.B. Rosen, P.E. Gill, L.O. Jay, K. Park, Numerical optimal control of parabolic PDEs using DASOPT, in: L. Bregler, T. Coleman, A. Conn, F. Santosa (Eds), Large Scale Optimization with Applications, Part II: Optimal Design and Control, IMA Volumes in Mathematics and its Applications, Vol. 93, 1997, 271–300.
[16] L.R. Petzold, W. Zhu, Model reduction for chemical kinetics: An optimization approach, AICHE J. April (1999) 869–886.
[17] H. Rabitz, M. Kramer, D. Dacol, Sensitivity analysis in chemical kinetics, Ann. Rev. Phys. Chem. 34 (1983) 419–461.