Pergamon

Atmospheric Environment Vol. 30, No. 1, pp. 49–58, 1996

1352-2310/96 $9.50 + 0.00

1352-2310(95)00283-9

A COMPARISON OF STIFF ODE SOLVERS FOR ATMOSPHERIC CHEMISTRY PROBLEMS*

J. G. VERWER,† J. G. BLOM, M. VAN LOON and E. J. SPEE

Center for Mathematics and Computer Science (CWI), P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

(*First received* 7 *March* 1995 *and in final form* 15 *June* 1995)

Abstract—In the operator splitting solution of atmospheric transport-chemistry problems modeling air pollution, a major task is the numerical integration of the stiff systems of ordinary differential equations describing the chemical transformations. In this paper a numerical comparison is presented between two special purpose solvers developed for this task.

Key word index: Atmospheric chemistry, air pollution modeling, numerical stiff ODEs.

1. INTRODUCTION

This paper deals with the numerical integration of the initial value problem for stiff systems of ordinary differential equations (ODEs) from atmospheric chemistry. Although the numerical stiff ODE field is well developed and an interesting variety of efficient and quite reliable stiff ODE solvers is available (Hairer and Wanner, 1991), the general experience is that for three space dimensional transport-chemistry problems, where stiff ODE integrations are carried out at thousands of grid points, still faster tailor-made solvers are needed. In this paper we compare two such solvers on a set of three atmospheric chemistry problems from practice.

The first solver is TWOSTEP, a simple code based on the implicit, second-order, two-step backward differentiation formula (BDF). The code has been designed as a special purpose solver for atmospheric chemistry problems. The solver is special purpose in the sense that Gauss–Seidel iteration is used for approximately solving the implicitly defined BDF solution (Verwer, 1994; Verwer and Simpson, 1994), rather than the more commonly used iterative modified Newton technique. The Gauss–Seidel iteration renders the integration explicit which implies low start-up costs and a low memory demand. This is of advantage in an operator splitting application of a stiff solver. The Gauss–Seidel iteration is related to the straightforward (Jacobi or Picard) iteration used in the quasi-steady-state-approximation (QSSA) approach, to which it compares very favorably (Verwer and Simpson, 1994).

The second solver is VODE, the variable-coefficient ordinary differential equation solver from Brown *et al.* (1989) which is comparable with LSODE, the "Livermore solver" from Hindmarsh (1980) which is often used in the field of atmospheric chemistry. Hence, VODE is a general, state of the art, variable order implicit BDF code. From the user point of view VODE and LSODE are very comparable. For example, they have a similar user interface. We pay special attention to sparsity of the Jacobian matrix in an attempt to reduce the time VODE spends in the iterative modified Newton process for solving the nonlinear implicit BDF relations. As is well-known, for large chemical models it is precisely this process which often renders an implicit ODE solver too expensive for application to real life, the three-dimensional transport-chemistry models and which in the past has led to the development of special purpose QSSA methods (Hesstvedt *et al.*, 1978; Høv *et al.*, 1978). We emphasize that exploiting sparsity has been shown before to be advantageous for atmospheric chemistry problems (Jacobson and Turco, 1994). Since VODE is known as an efficient solver for chemical kinetics problems, optimal use of sparsity should make it one of the best candidates from the numerical stiff ODE field for tailor-made solution of atmospheric transport-chemistry problems.

Our main purpose thus is to test TWOSTEP against VODE, provided with sparse matrix routines, and to check whether the claims made for this explicit code (Verwer, 1994; Verwer and Simpson, 1994) are confirmed if we largely economize on the modified

* This report is one of a series on the development of algorithms for long-range transport air pollution models. We gratefully acknowledge support from the RIVM for the projects EUSMOG and CIRK.

† Email: Jan.Verwer@cwi.nl.

Newton process by exploiting sparsity of the Jacobian matrix. Section 2 is devoted to the two solvers. In Section 2.1 we explain the notion of Gauss–Seidel iteration as implemented in TWOSTEP and briefly discuss how this solver works. In Section 2.2 we discuss how we have modified the public domain version of VODE with regard to exploiting sparsity. Section 3 contains results for our three different test problems. Concluding remarks are given in the final Section 4.

2. THE TWO SOLVERS

2.1. *TWOSTEP*

Atmospheric chemical kinetics systems can be cast in the nonlinear form

$$\frac{\mathrm{d}}{\mathrm{d}t}y = f(t, y) := P(t, y) - L(t, y)y,$$

$$y(t) = (y_1(t), \ldots, y_m(t))^{\mathrm{T}} \tag{1}$$

where y is a vector of concentrations, $P(t, y)$ is a vector function and $L(t, y)$ a diagonal matrix. The components $P_k(t, y)$, $L_k(t, y)y_k$ are nonnegative and represent, respectively, production and loss terms for compound y_k. For the numerical solution we consider the variable step size, two-step BDF formula

$$y^{n+1} = Y^n + \gamma\tau f(t_{n+1}, y^{n+1}),$$

$$\tau = t_{n+1} - t_n \tag{2}$$

where y^n is the approximation for $y(t_n)$, $\gamma = (c + 1)/(c + 2)$, $c = (t_n - t_{n-1})/(t_{n+1} - t_n)$ and

$$Y^n = ((c + 1)^2 y^n - y^{n-1})/(c^2 + 2c). \tag{3}$$

We have chosen the second-order BDF formula in view of the modest accuracy requirement. The approach we follow can also be examined for higher-order BDF formulas. Our use of the Gauss–Seidel technique exploits the production-loss form (1), by which equation (2) can be written as

$$y^{n+1} = F(y^{n+1}) := (I + \gamma\tau L(t_{n+1}, y^{n+1}))^{-1} \times (Y^n + \gamma\tau P(t_{n+1}, y^{n+1})). \tag{4}$$

The Gauss–Seidel technique is now applied to the nonlinear system of equations $y = F(y)$. That is, given the iterate $y^{(i)}$ as the ith approximation for the sought solution y^{n+1}, the Gauss–Seidel iteration implemented in TWOSTEP computes the next iterate $y^{(i+1)}$ by the componentwise formula

$$y_k^{(i+1)} = F_k(y_1^{(i+1)}, \ldots, y_{k-1}^{(i+1)}, y_k^{(i)}, \ldots, y_m^{(i)}),$$

$$k = 1, \ldots, m. \tag{5}$$

Inspection of the operator F reveals that application of equation (5) results in an explicit computation (no systems of equations need to be solved) due to the diagonal form of L. More precisely, for the computation of $y_k^{(i+1)}$ only division by the scalar variable

$$1 + \gamma\tau L_k(t_{n+1}, v)$$

is required, where v denotes the intermediate vector

$$v = [y_1^{(i+1)}, \ldots, y_{k-1}^{(i+1)}, y_k^{(i)}, \ldots, y_m^{(i)}]^{\mathrm{T}}.$$

The fact that in v the first $k - 1$ components are taken from the new $(i + 1)$st iterate, makes equation (5) a Gauss–Seidel type iteration process. It is also possible to take the first k components in v from the $(i + 1)$st iterate. Then the method becomes scalarly implicit, since $L_k(t, y)$ may depend on y_k. We prefer to avoid this scalar implicitness. Further, would we have taken all m components in v from the old iterate $y^{(i)}$, then an iteration method of Jacobi or Picard type would result. In this case equation (5) would be replaced by

$$y_k^{(i+1)} = F_k(y_1^{(i)}, \ldots, y_{k-1}^{(i)}, y_k^{(i)}, \ldots, y_m^{(i)}),$$

$$k = 1, \ldots, m. \tag{6}$$

Computationally there is not much difference between equations (5) and (6). However, equation (5) is in a sense half-implicit, as it uses solution components as soon as these have been updated which is characteristic for the Gauss–Seidel approach. The half-implicitness of course serves to improve the convergence of the iteration. Due to this half-implicitness, the order of the species obviously affects the accuracy of the final iterate, especially when only a few iterations are spent, which we advocate. However, our experience so far is that the influence of the order of the species on the accuracy is of minor importance. For our test Problems I and II the order is given explicitly.

Noteworthy is that QSSA methods also use the iteration formula (6), but with a different operator F. In QSSA methods, F is based on the exponential solution formula which is exact for species for which both P_k and L_k are constant in y (cf. Hesstvedt *et al.*, 1978; Høv *et al.*, 1978). We start from the BDF integration formula (2) which in general is more accurate than QSSA formulas. Furthermore, for components for which both P_k and L_k are constant in y, the solution with either equation (5) or equation (6) is obtained in one iteration. Consequently, when individual components rapidly approach their steady-state value P_k/L_k, they are handled efficiently and accurately by equations (5) and (6). In this respect, the current iterative approach bears a resemblance with the explicit QSSA approach. However, our experience is that the TWOSTEP code based on the Gauss–Seidel technique (5) is more efficient than well developed QSSA solvers (cf. Verwer and Simpson, 1994).

For a further discussion on TWOSTEP‡ with regard to implementation aspects for formula (2), such

‡ A copy of the Fortran 77 listing can be obtained from the first author by Email: Jan.Verwer@cwi.nl.

as the variable step size and start-up strategy, we refer to Verwer and Simpson (1994). In the comparisons carried out in this paper, the code has been applied in two different ways, in the remainder indicated by TWOSTEP1 and TWOSTEP2.

TWOSTEP1. By TWOSTEP1 we mean the standard black box use which here basically means that at any time step two Gauss–Seidel iterations are used. It should be noted that two iterations are by far not enough to let the Gauss–Seidel iteration fully converge. Our experience is that the overall accuracy of the code improves with the number of iterations, but the efficiency generally not. We therefore prefer a small number of iterations, which is attractive in any case after a restart with a small step size. In all tests the step size is variable and governed by user defined tolerances (specified later), but constrained to a minimum and maximum. That is, in all tests we prescribe a starting, a minimal and maximal step size. These are, in seconds,

$$\tau_{start} = 1, \quad \tau_{min} = 1, \quad \tau_{max} = 900. \tag{7}$$

Step sizes below 1 s are redundant. The minimal time constants of importance for photochemical chemistry models are about 1 min in size and species with a time constant smaller than 1 s almost instantaneously get in their (solution-dependent) steady state when they are perturbed. Hence the choice of 1 s, is reasonable and safe compared to 1 min. We emphasize that without the 1 s, lower bound extremely small steps could be selected by the variable step size selection scheme, since atmospheric chemistry problems containing photochemical reactions can possess time constants as small as 10^{-8} to 10^{-9} s and step size selection mechanisms do signal these extremely small time constants. While the 1 s lower bound is imposed for reasons of efficiency, the 900 s upper bound serves to protect the code for taking too large step sizes. This upperbound is reasonable on chemical grounds. With step sizes much larger than 15 min the reliability of numerical computations may degrade.

TWOSTEP2. TWOSTEP2 refers to the same way of application, but in addition certain *ad hoc* rules are used to exploit special problem properties. This means that special techniques like lumping or group iteration are combined with Guass–Seidel iteration process. These *ad hoc* rules will be discussed with the test problems and of course serve to obtain a more efficient numerical solution process.

2.2. *VODE*

VODE is a variable-coefficient ordinary differential equation solver based on the implicit BDF formulas (Brown *et al.*, 1989; Hairer and Wanner, 1991). VODE might be called a successor of the "Livermore solver" LSODE from Hindmarsh (1980) which is often used in the field of atmospheric chemistry. For a discussion of the mathematical techniques implemented in VODE we refer to Brown *et al.* (1989) and Hairer and Wanner (1991). As mentioned in the introduction, here we focus on the sparsity modifications we implemented in the public domain version which we obtained from Netlib (Dongarra and Grosse, 1987). To illustrate the wide efficiency range for this general solver, it has been used in three different ways. These will be indicated by VODE1, VODE2 and VODE3.

VODE1. VODE1 concerns the standard black box use, i.e. no optional input is used and the method parameters ITASK (not essential for our comparison) and MF are set to 4 and 22, respectively. The choice MF = 22 implies that the code generates the Jacobian automatically by numerical differencing. This choice also invokes the use of the standard, full matrix linear algebra routines DGEFA (factoring) and DGESL (backsolves) from the linear algebra software package LINPACK (Dongarra *et al.*, 1979) for solving the linear systems arising in the iterative modified Newton solution of the implicit BDF relations. MF = 22 also implies extra storage because both the Jacobian and its factored form (LU-decomposition) are stored. This saves Jacobian updates, on the other hand additional storage may be a disadvantage for higher space dimensional problems. Because no optional input is used, there is no constraint on the step size selection. For example, the code selects its own starting step size. To sum up, VODE1 is the easiest, most user friendly way of calling as it requires no extra effort on the part of the user whatsoever. This, of course, does have a price in terms of CPU time for ODE systems with a large number of components, as we will see later.

VODE2. In the second manner of calling VODE, special problem properties are exploited so as to obtain a more efficient numerical solution process. VODE2 means use of ITASK = 4 and MF = 21. The choice MF = 21 is important since this implies that Jacobians have to be provided by the user in analytic form. We emphasize that this already can save CPU time because of sparsity. However, VODE2 still uses the same (full matrix) linear algebra routines DGEFA and DGESL as VODE1 does. Hence the sparsity is here not yet exploited in the solution of the linear systems. Like MF = 22, the choice MF = 21 implies that extra storage is needed.

A second important difference with VODE1 is that for VODE2 we also prescribe the stepsize constraints (7) used by TWOSTEP, for the same reasons. However, it is necessary to overrule the rejection strategy to enable equation (7). Without this, overruling the code returns with an error message due to the constraint $\tau_{min} = 1$ and interrupts the integration, simply because the automatic local error control of VODE does signal time constants smaller than 1 s, and is not allowed to reduce the stepsize due to equation (7). In general, this is perfectly all-right, of course, and VODE should not be blamed for it. We repeat that for stiff photochemical chemistry problems step sizes below 1 s, are redundant for global accuracy. This lower bound will not be recovered in the results,

unless extremely high accuracies are sought for. In theory, the lower bound might cause problems for the convergence of the iterative modified Newton process, resulting in an interruption as a result of a convergence failure. We have not experienced this in our tests and do not expect it to happen for other problems either, because on a scale of 1 s, the Jacobian matrix is not expected to change substantially.

VODE3. VODE3 is identical to VODE2, except that now the sparsity of the Jacobian is exploited to reduce the costs of solving the linear algebraic systems arising in the modified Newton iteration. The idea is easily explained. Consider a linear system of algebraic equations of dimension m,

$$Mv = b. \tag{8}$$

Suppose that equation (8) is directly solved by first factoring M and then doing a backsolve. Factoring means that M is written as $M = LU$, where L is a lower and U is an upper triangular matrix (LU-decomposition). Then equation (8) reads $LUv = b$ and the vector v is found by the backsolve, which involves first solving the lower triangular system $Lw = b$ followed by solving the upper triangular system $Uv = w$. Obviously, the solution of triangular systems is trivial. This way of solving equation (8) is a standard procedure in numerical algebra and is the usual approach followed in stiff ODE solvers. Now suppose that M is very sparse, i.e. M has very many zero entries. If to a large extent the sparsity pattern of M can be maintained in the LU-decomposition, then the costs of the factoring and the backsolve can be reduced substantially, simple by omitting all redundant calculations in which a zero occurs. For large systems this is very attractive, as the number of operations for the factoring and backsolve amount to, approximately, $m^3/3$ and m^2 for a full (no zero entries) matrix. As a result, for a large dimension the costs are high, especially those for the factoring. Standard LU-decomposition routines like those used in VODE1/2, treat the matrix as full and hence do not exploit sparsity at all.

To exploit the sparsity in the LU-decomposition, we have reordered the species in equation (1) such that the most dense rows in the Jacobian reside in the bottom of the matrix and the most sparse rows at the top. If this is the case, then the fill-in in L and U is greatly reduced. For our test examples we have carried out the reorderings using facilities offered by the computer algebra package MAPLE (Char *et al.*, 1991). We note in passing that one and the same ordering is used for the whole time interval. At night, when photochemical reactions are switched off, the sparsity is somewhat larger, but for simplicity we have not used this small advantage. Then, after having determined the fill-in that remains from the factoring, for which purpose again MAPLE is used, the linear systems can be solved quite efficiently with routines that omit all redundant calculations in which a zero occurs. For this purpose we have used slightly modified versions of the ILU (Incomplete LU) routines DSILUS (factoring) and DSLUI2 (backsolves) from the sparse linear algebra package (SLAP). SLAP is a public domain code written by Greenbaum and Seager (with contributions of several other authors) that is available from Netlib (Dongarra and Grosse, 1987). Hence these two sparse matrix routines replace the full matrix LINPACK routines DGEFA and DGESL, respectively. Like the LINPACK routines, they factorize and backsolve, but omit all redundant calculations in which a zero occurs. It should be remarked, though, that now no longer pivoting occurs in the matrix factorization. This could give rise to errors in the linear system solution which otherwise would have been avoided. We have not experienced problems of this sort. Of course, if the factorization fails, then the step size control of VODE will detect this and a change in the step size will improve matters. It seems that for solving stiff ODEs pivoting is not often required (cf. Jacobson and Turco, 1994; Sherman and Hindmarsh, 1980).

Finally, Table 1 illustrates the sparsity for the three test examples treated in Section 3. The table obviously predicts a large reduction in CPU time for Problem III. For Problems I and II the gain will be less since for these two the dimension is modest. Note that for other atmospheric chemical kinetics problems with a large dimension, a similar level of sparsity exists as for our Problem III. For example, Jacobson and Turco (1994) discuss a smog chemistry problem of dimension 92 with only 695 nonzero entries in the Jacobian. The fill-in only increases this to 839.

3 RESULTS FOR THREE TEST PROBLEMS

3.1. *Set-up of experiments*

The solvers are tested as if they were used in an operator splitting environment. This means that we split up the total integration interval in a lot of subintervals, representing the length of advection steps taken in the operator splitting. For each subinterval we then restart the integration of the stiff solver. All three test examples are based on chemical transformations of which part are photochemical. This means that part of the reaction constants are determined by time of the day dependent photolysis rates which undergo a near discontinuity at sunrise and sunset. In all three examples, we take the same integration interval covering 112 h. This interval starts at 04.00 h at day one and ends at 20.00 h at day five. Thus the time interval is sufficiently long to include a number of diurnal cycles for the important photochemical transformations and to include a large set of different initial conditions due to the restarts.

The total integration interval of 112 h is split up in 56 2-h subintervals which involves 56 restarts. Our measure of accuracy is based on the relative root mean square error RRMS_k for each species k, taken over the endpoints of all 2-h intervals over the 112 h.

Table 1. Sparsity data

	Dimension	Nonzero entries	Fill-in	% Nonzeros LU
Problem I	15	57	0	25
Problem II	18	111	12	38
Problem III	66	500	107	14

Hence,

$$\mathrm{RRMS}_k = \sqrt{\frac{\sum_{n=1}^{N} (y_k^n - y_k(t_n))^2}{\sum_{n=1}^{N} (y_k(t_n))^2}} \tag{9}$$

where $N = 56$, $t_n = 14{,}400 + 7200n$ s and $y_k(t_n)$ represents a sufficiently accurate reference solution. We then calculate the number of significant digits for the average of RRMS_k, defined by

$$\mathrm{SDA} = -\log_{10}\left(\frac{1}{m}\sum_{k=1}^{m} \mathrm{RRMS}_k\right). \tag{10}$$

Our comparison focuses on modest accuracy, i.e. relative accuracies near 1%, since higher accuracy levels are redundant for the actual practice of three-dimensional air pollution modeling. For all three test problems we will use the same set of relative error tolerances rtol and absolute error tolerances atol for the variable step size control, viz.

$$\mathrm{rtol} = 10^{-l}, \quad \mathrm{atol} = 1.0, \quad l = 1, 2, 3, 4, 5 \tag{11}$$

for all species. In all three test problems, the unit of concentration is number of molecules per cm^3. We therefore effectively invoke a relative error control. For some species (radicals) the concentration can be smaller than unity, but these values are insignificant for the overall solution and require no local error control. Since the two solvers use quite different solution techniques, and are therefore difficult to compare, efficiency is measured by CPU time. In the figures showing the results, we thus plot the SDA values against the measured CPU times (in unit seconds).

3.2. *Example Problem I: the EUSMOG chemistry*

The chemical model is identical to the one described in Van Loon (1994). This model is a highly parameterized version of the EMEP MSC-W model (Simpson *et al.*, 1993; Simpson, 1994) that will be used in Section 5. It consists of 15 reactions between 15 species:

1. $\mathrm{NO} + \mathrm{O_3} \rightarrow \mathrm{NO_2}$ $\quad k_1 = 2.0\,10^{-12}\exp(-1400/T)$
2. $\mathrm{NO_2} + h\nu \rightarrow \mathrm{NO} + \mathrm{O_3}$ $\quad k_2 = 1.45\,10^{-2}\exp(-0.4/\cos Z)$
3. $\mathrm{NO_2} + \mathrm{OH} \rightarrow \mathrm{NO_{3a}}$ $\quad k_3 = 1.68\,10^{-12}\exp(560/T)$
4. $2\mathrm{NO_2} + \mathrm{O_3} \rightarrow 2\mathrm{NO_{3a}}$ $\quad k_4 =$ see Van Loon (1994)
5. $\mathrm{O_3} + h\nu \rightarrow b_1\mathrm{OH} + b_2\mathrm{O_3}$ $\quad k_5 = 2.0\,10^{-4}\exp(-1.4/\cos Z)$
6. $\mathrm{C_2H_6} + \mathrm{OH} \rightarrow a_1\mathrm{O_3}$ $\quad k_6 = 8.7\,10^{-12}\exp(-1070/T)$
7. $\mathrm{C_4H_{10}} + \mathrm{OH} \rightarrow a_2\mathrm{O_3}$ $\quad k_7 = 1.4\,10^{-11}\exp(-559/T)$
8. $\mathrm{C_2H_4} + \mathrm{OH} \rightarrow a_3\mathrm{O_3}$ $\quad k_8 = 1.66\,10^{-12}\exp(474/T)$
9. $\mathrm{C_3H_6} + \mathrm{OH} \rightarrow a_4\mathrm{O_3}$ $\quad k_9 = 4.1\,10^{-12}\exp(545/T)$
10. $\mathrm{XYL} + \mathrm{OH} \rightarrow a_5\mathrm{O_3}$ $\quad k_{10} = 1.4\,10^{-11}$
11. $\mathrm{ISO} + \mathrm{OH} \rightarrow a_6\mathrm{O_3}$ $\quad k_{11} = 2.55\,10^{-11}\exp(410/T)$
12. $\mathrm{CO} + \mathrm{OH} \rightarrow a_7\mathrm{O_3}$ $\quad k_{12} = 2.4\,10^{-13}$
13. $\mathrm{CH_4} + \mathrm{OH} \rightarrow a_8\mathrm{O_3}$ $\quad k_{13} = 2.9\,10^{-12}\exp(-1820/T)$
14. $\mathrm{SO_2} + \mathrm{OH} \rightarrow \mathrm{SO_4}$ $\quad k_{14} = 2.32\,10^{-10}\exp(-457/T)$
15. $\mathrm{SO_2} \rightarrow \mathrm{SO_4}$ $\quad k_{15} = 1.39\,10^{-6}$.

The parameter Z denotes the solar zenith angle and T is the temperature in Kelvin. In Van Loon (1994) the above set of reactions is part of a smog prediction model, consisting, apart from chemistry, of advection, horizontal and vertical diffusion, emission and deposition. Each of these processes is treated in an operator splitting context. This means that per split step for each grid cell one ODE describing the chemical reactions has to be solved. Here, however, we only carry out box calculations with the chemical model. In order to get more realistic concentration profiles, emission terms Q_i and deposition terms vg_i have been added. For $\mathrm{NO_2}$, $\mathrm{O_3}$ and $\mathrm{SO_2}$ deposition is specified and for NO, the VOCs and $\mathrm{SO_2}$, emission. All time-dependent coefficients are updated at the beginning of the 2-h intervals and then frozen for the rest of the time. A specification of all parameters and input data defining the complete ODE system is given in Appendix A of the preprint Verwer *et al.* (1995). Evaluation of the eigenvalues of the Jacobian matrix for various conditions has shown that the eigenvalues range from -10^{1} to $-10^{-10}\,\mathrm{s}^{-1}$, approximately, indicating that the system is (moderately) stiff. Recall that time steps of a few minutes size should be achievable for an efficient code.

First we mention the order in which the components are treated in the Gauss–Seidel process (equal for TWOSTEP1/2): $\mathrm{NO_2}$, NO, $\mathrm{O_3}$, OH, $\mathrm{NO_{3a}}$, the VOCs, $\mathrm{SO_2}$ and $\mathrm{SO_4}$. As TWOSTEP1 is the standard way to use TWOSTEP, we now only describe how we exploited special problem characteristics in TWOSTEP2, in order to improve the convergence of the Gauss–Seidel iteration. In the chemistry literature the approach we follow is called "lumping" (Hesstvedt *et al.*, 1978; Høv *et al.*, 1978). In our case, the lumping

involves the introduction of the two "new" species $NO_x = NO_2 + NO$ and $O_x = NO_2 + O_3$. For both a differential equation can be specified with positive production and loss terms. The differential equation for NO_x, for example, takes the form

$$\begin{aligned}\frac{d}{dt}NO_x &= -k_3 \cdot NO_2 \cdot OH - 2k_4 \cdot (NO_2)^2 \cdot O_3 \\ &\quad - vg_1 \cdot NO_2 + Q_{NO} \\ &= -k_3 \cdot (NO_x - NO) \cdot OH \\ &\quad - 2k_4 \cdot NO_2 \cdot O_3 \cdot (NO_x - NO) \\ &\quad - vg_1 \cdot (NO_x - NO) + Q_{NO} \\ &= P_{NO_x} - L_{NO_x} NO_x, \end{aligned} \tag{12}$$

where

$$\begin{aligned} P_{NO_x} &= k_3 \cdot NO \cdot OH + 2k_4 NO_2 \cdot O_3 \cdot NO \\ &\quad + vg_1 \cdot NO + Q_{NO} \end{aligned} \tag{13}$$

and

$$L_{NO_x} = k_3 \cdot OH + 2k_4 \cdot NO_2 \cdot O_3 + vg_1. \tag{14}$$

When we would compute the implicit solution for the original ODE system augmented with the lump species exactly, the lumping relations also hold for this exact implicit solution. That is, at any nth time step we have

$$NO_{x^n} = NO_{2^n} + NO^n, \quad O_{x^n} = NO_{2^n} + O_{3^n}. \tag{15}$$

This, however, is not true for the approximate solution obtained with Gauss–Seidel iteration. The idea behind the lumping technique is to impose the lumping relations (15) on the solution obtained after each Gauss–Seidel iteration, thus hoping that this will improve the convergence to the exact implicit solution. In our case, the lumping of NO_2 and NO into NO_x, etc. underlies the assumption that the first two reactions from above are in some sense dominant in the whole set of chemical transformations. Because, if we consider only reactions 1 and 2, then we have

$$\frac{d}{dt}NO_x = 0, \quad \frac{d}{dt}O_x = 0 \tag{16}$$

showing that for these two reactions NO_x and O_x are conserved. Consequently, if the first two reactions are truly dominant in the whole system, then NO_x and O_x are expected to vary slowly. This, in turn, implies that the integration of the differential equation for NO_x and O_x can be done accurately, so that imposing relation (15) by correcting one of the grouped species will make sense.

We perform the integration of the new species as follows. At the end of a Gauss–Seidel iteration, we first compute NO_x from the BDF formula (2), using the production-loss form. We thus get

$$NO_x = (Y + \gamma\tau P)/(1.0 + \gamma\tau L), \tag{17}$$

where P and L are evaluated at the solution generated by the last Gauss–Seidel iteration and Y denotes the history term (3) of the BDF formula for NO_x. Next, if $NO_2 > NO$, then NO_2 is recomputed from NO_x, otherwise NO is recomputed. In the same way O_x is computed and O_3 or NO_2 is recalculated from O_x. Consequently, relation (15) now holds after any Gauss–Seidel iteration. In Appendix C of the preprint Verwer *et al.* (1995), it is shown that lumping can be interpreted as a simple form of preconditioning.

For VODE1 and VODE2 we use the same ordering of components as TWOSTEP in the Gauss–Seidel

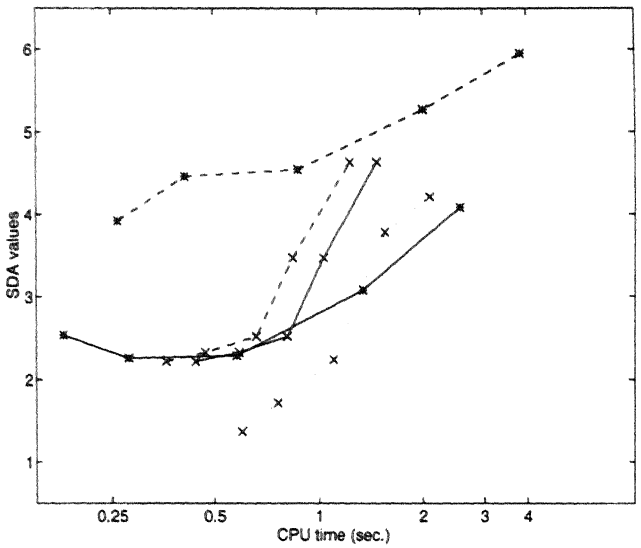

Fig. 1. Results for Problem I. TWOSTEP1 (*, solid), TWOSTEP2 (*, dashed), VODE1 (x, dotted), VODE2 (x, solid), VODE3 (x, dashed).

process. This ordering results in a large amount of fill-in elements in the LU-decomposition of the matrix $P = I - \gamma\tau J$, J denoting the Jacobian. Whereas the original matrix P has 57 nonzero elements, the sum of the nonzero elements in L and U is 148. For the present chemical system it is possible to reorder the components in such a way that no fill-in elements arise at all, i.e. the total number of nonzero elements in L and U is equal to 57, which number should be compared with 225 for the standard LU-decomposition. For completeness we give the new order of the components: first the VOCs, followed by SO_2, SO_4, NO, NO_{3a}, NO_2, O_3 and OH. This new ordering is used by VODE3.

Figure 1 shows all accuracy–efficiency plots for Problem I. The marks on the plots correspond with the five tolerances (11). Interestingly, lumping improves the TWOSTEP solution more than expected and in fact brings it very close to the true implicit BDF solution. This is shown in Fig. 2 where again the plots for TWOSTEP1 and TWOSTEP2 are given, together with the plot for the implicit solution. For the three different cases the same step sizes were used. We see that the plots for TWOSTEP2 and the implicit solution practically coincide, showing that the lumping indeed has improved the convergence of the Gauss–Seidel iteration as used in TWOSTEP1. Recall that only two iterations have been carried out. Hence for this chemistry the lumping works out very well and because the additional costs are negligible, it is attractive to use. We should note, however, that lumping is problem dependent and in general improves accuracy certainly not as much as here.

The fact that there is hardly any difference in the TWOSTEP1/2 accuracies for the larger tolerances, which also occurs for VODE2 and VODE3, is due to the upper step size bound of 900 s. Yet, for TWOSTEP the CPU time becomes significantly larger when the tolerance is decreased. This is caused by coefficient updates at the beginning of every 2-h interval, which introduces small, but fast initial transients. These initial transients have disappeared near the end of the 2-h intervals and have no influence on the accuracies there, but their presence obviously reduces the step size in the initial phase of the integrations. This behavior is characteristic for operator splitting applications, indicating that for TWOSTEP the choice of a minimal step size is of practical importance.

As expected, VODE2 outperforms VODE1. We found that this is due to the step size restriction (11) and not a result of using the exact sparse Jacobian instead of a full numerical approximation. For the present model, with only 15 components, the overhead of this numerical approximation is too small to become visible in the results. Restriction (7) prevents VODE2 from taking very large step sizes which will reduce the accuracy at the end of the 2-h intervals, but also prevents VODE2 from taking very small step sizes lower than 1 s in the initial phase. As noted before, these small step sizes are of no relevance for the accuracies we measure. VODE2 spends only about 30% in routines that handle the factorization and the backsolve, which of course is too small to get much gain in CPU time by replacing VODE2 by VODE3. The latter needs approximately 20% less

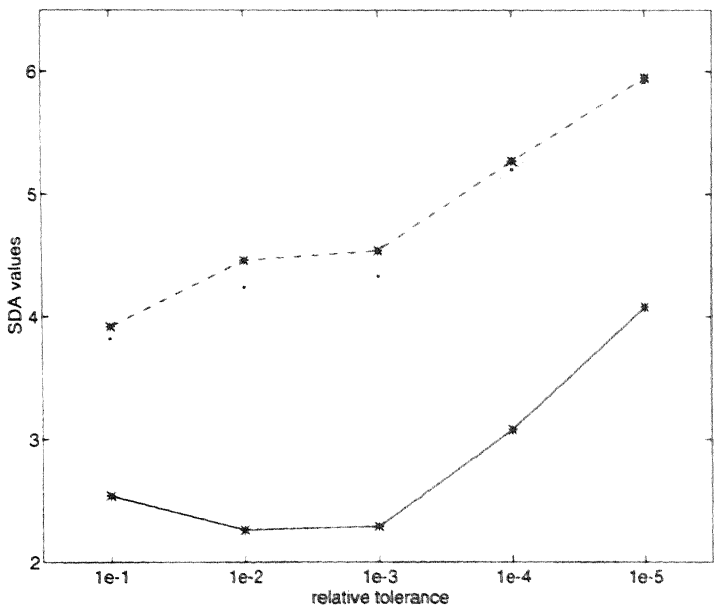

Fig. 2. Results for Problem I. TWOSTEP1 (*, solid), TWOSTEP2 (*, dashed), the implicit BDF solution (dotted).

CPU time than VODE2. These numbers reveal that by using the sparse matrix routines, the linear algebra costs have been reduced by a factor three. Finally, when we compare with the most efficient VODE version, which is VODE3, we can conclude that TWOSTEP2 outperforms VODE3 convincingly. Also TWOSTEP1 is faster in the 1% error range, although the step size selection needs some more attention for this test example.

3.3. *Example Problem II: the methane CIRK chemistry*

We obtained our second chemical model from The (1994). This model is used in long term, global studies and describes a methane oxidation cycle. It consists of 46 reactions between 19 species. Thirteen reactions depend on the solar zenith angle which, different from Problem I, is taken continuous and hence calculated in each time step. The problem is very stiff. Eigenvalues of the Jacobian lie between -10^9 and $0\ s^{-1}$, approximately. There are two extremely large eigenvalues which originate from the free radicals O^1D and O^3P. These are absent in Problem I, which explains the modest stiffness of that problem. A complete description of the model defining the ODE system used in the experiments can be found in Appendix B of the preprint by Verwer *et al.* (1995).

The order of the components used in the Gauss–Seidel process is (equal for TWOSTEP1/2): O^1D, O^3P, OH, NO_3, HO_2, N_2O_5, NO, NO_2, O_3, HNO_3, HO_2NO_2, HNO_2, H_2O_2, HCHO, CH_3OOH, CH_3O_2, CH_4, NO_x, O_x. TWOSTEP2 uses the same NO_x and O_x lumping as for Problem I. VODE1 and VODE2 use a slightly different order with O_x omitted. For the modified Newton process as used by VODE1/2 the order is to a great extent irrelevant, while also the lumped species play no role here. The sequence used by VODE1/2 results in 31 fill-in elements in the LU-decomposition. Reordering leads to a fill-in of 12 elements. Thus the total number of nonzeroes after reordering is $111 + 12 = 123$. The new sequence used by VODE3 reads CH_4, O^1D, HNO_2, H_2O_2, N_2O_5, HNO_3, HO_2NO_2, CH_3OOH, O_3, HCHO, CH_3O_2, NO_3, O^3P, NO, NO_2, HO_2, NO_x, OH.

Figure 3 shows all results obtained for Problem II. First we notice that, similar as for Problem I, the simple lumping trick improves the TWOSTEP accuracy considerably and for minor costs. The VODE results compare well with those for Problem I. Supplying VODE with an analytical Jacobian and a minimal and maximal step size improves the performance significantly (VODE2). However, here also the gain in CPU from using the sparsity of the Jacobian in the Jacobian evaluation is low, only 10%. Similar as for Problem I, this also holds for the change to VODE3 where the sparsity is exploited in the solution of the linear systems. In the accuracy region of greatest practical interest, both solvers perform well although TWOSTEP is again the most efficient one.

3.4. *Example Problem III: the EMEP chemistry*

The third example problem is identical to the urban test case reported in Verwer and Simpson (1994) for the EMEP MSC-W ozone chemistry model. This chemistry model consists of about 140 reactions

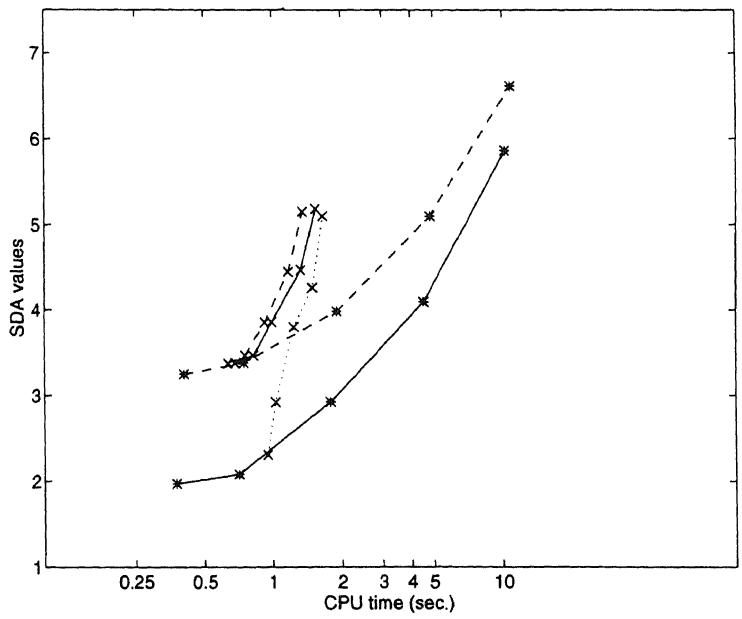

Fig. 3. Results for Problem II. TWOSTEP1 (*, solid), TWOSTEP2 (*, dashed), VODE1 (x, dotted), VODE2 (x, solid), VODE3 (x, dashed).

between 66 species. The model is state of the art in the field of regional air pollution modeling. Rate coefficients are often variable, depending on temperature and, for some, humidity. The model takes into account emission inputs and dry and wet removal processes. Photolysis rates obviously depend on solar elevation and cloudiness. These rates vary continuously in time, but undergo a discontinuity at sunset and sunrise. As regards to stiffness, Problem III is comparable with Problem II. Because the model is too large to describe here, we refer to Simpson *et al.* (1993) and Simpson (1994) for more details.

Figure 4 shows all accuracy–efficiency plots we obtained for Problem III. TWOSTEP2 now differs from the version used before. For TWOSTEP2 also two GS-iterations were used, but within each such iteration five group iterations on the $NO_y + O_3$ group are added (cf. Verwer and Simpson, 1994). The species in this group are strongly coupled, so it makes sense to perform this group iteration. We emphasize that this group iteration involves a minor change in the code and hence is very simply applicable. Because the group consists of only seven species, the additional work is minor and it obviously improves the Gauss–Seidel iteration. The TWOSTEP2 result should be compared with the best result obtained for VODE, which clearly is the VODE3 case. We see that for the accuracy range of greatest practical interest, TWOSTEP2 and VODE3 are comparable. For higher accuracies the variable order VODE3 is more efficient because it then uses the higher-order BDF formulas. The figure also nicely illustrates that by an intelligent use, standard stiff ODE codes like VODE can be improved dramatically. In the low accuracy range, VODE3 is about six times more efficient than VODE1. We emphasize that the difference between VODE2 and VODE3 is only due to the use of the sparse matrix techniques, which works out very well for this test problem due to its large number of components. The difference between VODE1 and VODE2 is due to using the analytical sparse Jacobian and the step size constraints (7). Both reduce part of the CPU time needed by the black box version VODE1.

4. CONCLUDING REMARKS

The MAPLE tools for automatically computing the analytical Jacobian and for deriving the datastructure for the ILU routines are easy to use. The sparse matrix technique based on the ILU routines from the SLAP library handles the solution of the linear systems well. We have encountered no difficulties in using VODE3, which solves the linear systems without pivoting. Similar experiences were reported by Jacobson and Turco (1994) and Sherman and Hindmarsh (1980).

For large problems from atmospheric chemistry, like the EMEP MSC-W model, the sparse matrix technique can lead to significant savings in CPU time for codes like VODE. This experience corresponds with the results reported by Jacobson and Turco (1994). For atmospheric chemistry models of a more moderate size, like the EUSMOG and CIRK model, the gain by exploiting sparsity does hardly pay. For such models, with about 20 species say, the solution costs of the linear systems in VODE are simply too low compared to the costs of all other calculations.

There is room for both TWOSTEP and VODE. When used in an intelligent way, both solve our test examples efficiently. In the low accuracy region

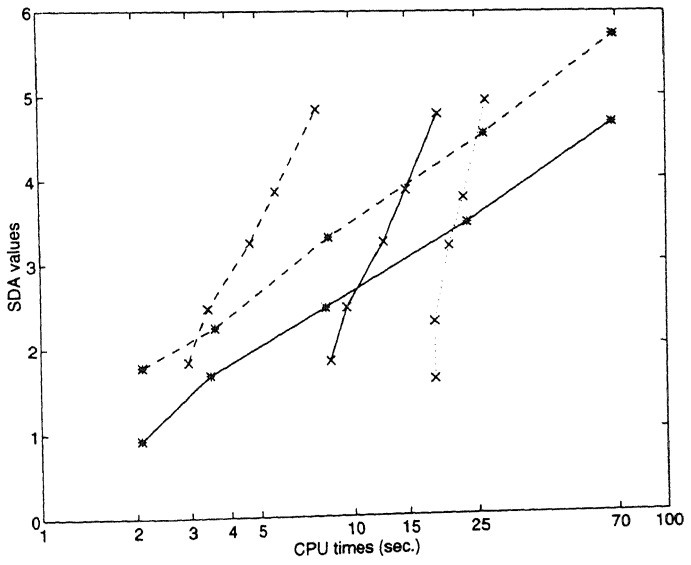

Fig. 4. Results for Problem III. TWOSTEP1 (*, solid), TWOSTEP2 (*, dashed), VODE1 (x, dotted), VODE2 (x, solid), VODE3 (x, dashed).

TWOSTEP always seems to be somewhat faster. Obviously, the lumping technique and/or the group iteration are recommendable for TWOSTEP when only a few Gauss–Seidel iterations are used. Lumping, however, is problem dependent which means that each time a few reactions are added, it might turn out necessary to reconsider the components treated in the lumping process to retain its efficiency. This, of course, is a disadvantage.

An advantage of Gauss–Seidel iteration is that it works matrix free and hence the memory demand is low, which is of interest when grid vectorization is employed. As shown in Verwer *et al.* (1995), Gauss–Seidel iteration can be nearly optimally vectorized over the grid, in a similar way as modified Newton combined with sparse solution techniques in the code SMVGEAR (Jacobson and Turco, 1994).

A further attractive feature of Gauss–Seidel iteration is that it can be efficiently extended to solve chemistry and vertical turbulent diffusion in a coupled way (Verwer *et al.*, 1995). This is not true for the modified Newton process as regards the exploitation of sparsity. If diffusion is coupled with chemistry, then the sparsity of the chemistry Jacobian is almost completely lost in the factorization of the banded linear system.

REFERENCES

Brown P. N., Byrne G. D. and Hindmarsh A. C. (1989) VODE: a variable coefficient ODE solver. *SIAM J. Sci. Stat. Comput.* **10,** 1038–1051.

Char B. W., Geddes K. O., Gonnet G. H., Monagan M. B. and Watt S. M. (1991) *Maple V Language Reference Manual.* Springer, New York.

Dongarra J. J. and Grosse E. (1987) Distribution of software via electronic mail. *Commun. ACM* **30,** 403–407 (netlib@research.att.com).

Dongarra J. J., Moler C. B., Bunch J. R. and Stewart G. W. (1979) *LINPACK Users' Guide.* SIAM, Philadelphia.

Hairer E. and Wanner G. (1991) *Solving Ordinary Differential Equations II. Stiff and Differential Algebraic Problems.* Springer, Berlin.

Hesstvedt E., Hov Ø. and Isaksen I. S. A. (1978) Quasi-steady state approximation in air pollution modeling: comparison of two numerical schemes for oxidant prediction. *Int. J. Chem. Kinet.* **10,** 971–994.

Hindmarsh A. C. (1980) LSODE and LSODI, two new initial value ordinary differential equation solvers. *ACM-SIGNUM Newsletter* **15,** 10–11.

Hov Ø., Isaksen I. S. A. and Hesstvedt E. (1978) A numerical method to predict secondary air pollutants with an application on oxidant generation in an urban atmosphere. *WMO Publication* **510,** 219–226.

Jacobson M. Z. and Turco R. P. (1994) SMVGEAR: A sparse-matrix, vectorized gear code for atmospheric models. *Atmospheric Environment* **28,** 273–284.

Van Loon M. (1994) Numerical smog prediction I: the physical and chemical model. CWI Report NM-R9411, Centre for Mathematics and Computer Science, Amsterdam.

Sherman A. H. and Hindmarsh A. C. (1980) GEARS: a package for the solution of sparse, stiff ordinary differential equations. Lawrence Livermore Laboratory Report UCRL-84102, Livermore, California.

Simpson D. (1994) Biogenic VOC in Europe. Part II: implications for ozone control strategies (submitted).

Simpson D., Andersson-Sköld Y. and Jenkin M. E. (1993) Updating the chemical scheme for the EMEP MSC-W, The Norwegian Meteorological Institute, Oslo.

The H. (1994) Private Communication. National Institute for Environmental Protection and Hygiene (RIVM), Bilthoven, The Netherlands.

Verwer J. G. (1994) Gauss–Seidel iteration for stiff ODEs from chemical kinetics. *SIAM J. Sci. Comput.* **15,** 1243–1250.

Verwer J. G. and Simpson D. (1994) Explicit methods for stiff ODEs from atmospheric chemistry. CWI Report NM-R9409, Center for Mathematics and Computer Science, Amsterdam. *Appl. Numer. Math.* (to appear).

Verwer J. G., Blom J. G. and Hundsdorfer W. H. (1995) An implicit–explicit approach for atmospheric transport-chemistry problems, CWI Report NM-R9501, Center for Mathematics and Computer Science, Amsterdam.

Verwer J. G., Blom J. G., van Loon M. and Spee E. J. (1995) A comparison of stiff ODE solvers for atmospheric chemistry problems, CWI Report NM-R9505, Center for Mathematics and Computer Science, Amsterdam (preprint to this article, obtainable from http://www.cwi.nl/cwi/projects/nw1.4.html).

Pergamon

Atmospheric Environment Vol. 30, No. 9, p. 1531, 1996
Elsevier Science Ltd
Printed in Great Britain

ERRATUM

"A comparison of stiff ODE solvers for atmospheric chemistry problems" by J. G. Verwer, J. G. Blom, M. Van Loon and E. J. Spee (*Atmospheric Environment* **30**, 49–58).

On page 54, equation (15) should read as follows:

$$NO_x^n = NO_2^n + NO^n, \qquad O_x^n = NO_2^n + O_3^n$$

On page 53, in reactions 3 and 4, and also on page 55, line 13, NO_{3a} should read:

$$NO_3^a$$

There is an error in Figure 3 on page 56. The authors regret that the SDA was not computed according to formula (10) on page 53. A recalculation gives the figure shown below. The SDA differ between 0.6 and 1.0 from the previous values, but the conclusions remain unchanged.

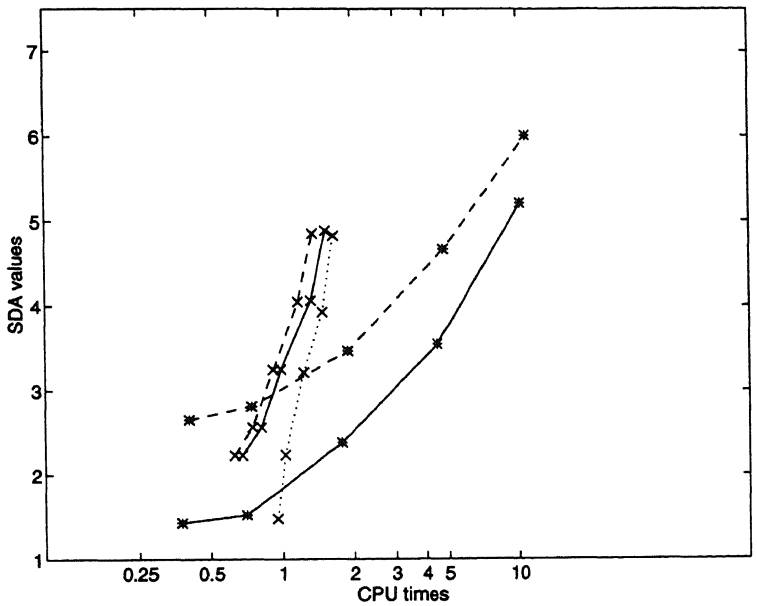

Fig. 3. Results for Problem II. TWOSTEP1 (*, solid), TWOSTEP2 (*, dashed), VODE1 (×, dotted), VODE2 (×, solid), VODE3 (×, dashed).

Pergamon

Atmospheric Environment Vol. 30, No. 9, p. 1531, 1996
Elsevier Science Ltd
Printed in Great Britain

ERRATUM

"A comparison of stiff ODE solvers for atmospheric chemistry problems" by J. G. Verwer, J. G. Blom, M. Van Loon and E. J. Spee (*Atmospheric Environment* **30,** 49–58).

On page 54, equation (15) should read as follows:

$$NO_x^n = NO_2^n + NO^n, \qquad O_x^n = NO_2^n + O_3^n$$

On page 53, in reactions 3 and 4, and also on page 55, line 13, NO_{3a} should read:

$$NO_3^a$$

There is an error in Figure 3 on page 56. The authors regret that the SDA was not computed according to formula (10) on page 53. A recalculation gives the figure shown below. The SDA differ between 0.6 and 1.0 from the previous values, but the conclusions remain unchanged.

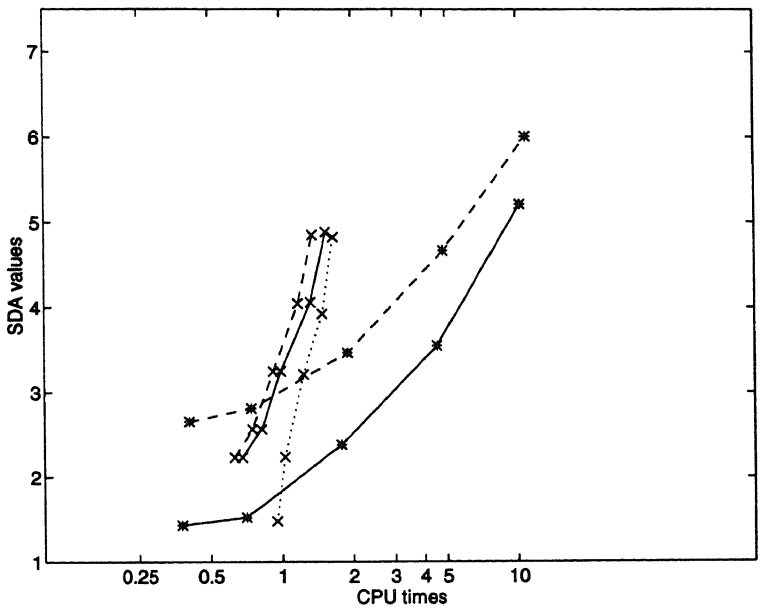

Fig. 3. Results for Problem II. TWOSTEP1 (*, solid), TWOSTEP2 (*, dashed), VODE1 (×, dotted), VODE2 (×, solid), VODE3 (×, dashed).