SIAM J. SCI. COMPUT.
Vol. 15, No. 5, pp. 1243–1250, September 1994

TIMELY COMMUNICATION

Under the "timely communications" policy for the SIAM Journal on Scientific Computing, papers that have significant timely content and do not exceed five pages automatically will be considered for a separate section of the journal with an accelerated reviewing process. It will be possible for the note to appear approximately six months after the date of acceptance.

GAUSS–SEIDEL ITERATION FOR STIFF ODES FROM CHEMICAL KINETICS*

J. G. VERWER†

Abstract. A simple Gauss–Seidel technique is proposed that exploits the special form of the chemical kinetics equations. Classical Aitken extrapolation is applied to accelerate convergence. The technique is meant for implementation in stiff solvers that are used in long range transport air pollution codes using operator splitting. Splitting necessarily gives rise to a great deal of integration restarts. Because the Gauss–Seidel iteration works matrix free, it has much less overhead than the modified Newton method. Start-up costs therefore can be kept low with this technique. Preliminary promising numerical results are presented for a prototype of a second order backward differentiation formula (BDF) solver applied to a stiff ordinary differential equation (ODE) from atmospheric chemistry. A favourable comparison with the general purpose BDF code DASSL is included. The matrix free technique may also be of interest for other chemically reacting fluid flow problems.

Key words. numerical stiff ODEs, chemical kinetics, air pollution modelling

AMS subject classifications. Primary: 65L05; Secondary: 80A30, 80A32

1. Introduction. Large scale, long range atmospheric air pollution models are computationally very expensive [10]. Usually the computational work is heavily dominated by the numerical treatment of the stiff ODE systems describing the chemical kinetics model in use. These ODE systems are of the nonlinear form

$$\frac{d}{dt}y = f(t,y) := P(t,y) - L(t,y)y, \qquad y(t) = (y_1(t), \ldots, y_m(t))^T, \tag{1.1}$$

where $P(t, y)$ is a vector and $L(t, y)$ is a diagonal matrix. The components $P_k(t, y)$ and $L_k(t, y)y_k$ are nonnegative and represent, respectively, production and loss terms. The reciprocal of L_k is the physical time constant or characteristic reaction time for compound y_k. Generally the range of time constants is large, which implies that in most applications the ODE system is stiff. For example, if we assume that the popular operator splitting approach is used, then a common situation is that (1.1) must be solved repeatedly, over several hundreds of split-step time intervals, at any of thousands of grid points. As a rule the length of this time interval is the stepsize used in the advection step, which for part of the chemical species occurring in atmospheric applications is much larger than the time constant. This introduces the stiffness. When the chemistry is nonlinear and many species are involved, say, 20 to 50, it is clear that a highly efficient stiff solver, tailored to the application under consideration, is indispensable. Due to the great number of restarts, one for each split-step time interval, it is

* Received by the editors October 20, 1993; accepted for publication (in revised form) March 4, 1994. This paper is one of a series on the development of algorithms for long range transport air pollution models (projects EUSMOG and CIRK). Financial support was provided by The Dutch National Institute of Public Health and Environmental Protection (RIVM).

† CWI, Research Institute of the Stichting Mathematisch Centrum, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands (janv@cwi.nl).

particularly important to use integrators that keep the inevitable start-up costs low, in comparison with a common stiff ODE integration. In addition, the integrator must be able to change stepsize rapidly with little costs because in practice rate coefficients in atmospheric chemistry models are temperature or time dependent. This dependency can cause sudden changes in the concentrations, which can be followed accurately and efficiently only if stepsizes can be adjusted efficiently.

2. The Gauss–Seidel iteration. The purpose of this note is to present some preliminary, but promising, results of a simple Gauss–Seidel iterative technique for solving implicit relations. Because this technique works matrix free, a change of stepsize involves no numerical algebra overhead at all. This results in low start-up cost compared to Newton-type iteration. The Gauss–Seidel technique is very cheap in the start-up phase where usually a few iterations are sufficient. When the stepsize increases, the convergence will normally slow down. In the experiments reported in this note we have successfully applied the classical Aitken extrapolation to accelerate convergence over a wide range of stepsizes. So far only a single extrapolation step has been considered. In the near future more sophisticated acceleration techniques will be the subject of further investigation. Moreover, switching between Gauss–Seidel and the modified Newton iteration will be examined. We also experimented with the Jacobi iteration, but less successfully. In the experiments reported here, the Gauss–Seidel technique significantly improved convergence, especially for larger stepsize values. Yet the Jacobi iteration may be of interest in the initial transient phase, because one Jacobi iteration is always cheaper than one Gauss–Seidel iteration. For simplicity of presentation, in this note we focus on the Gauss–Seidel iteration.

We have implemented the Gauss–Seidel iteration process in a prototype of a new solver. This prototype uses as the main integrator the variable step, second order BDF formula

$$y^{n+1} = Y^n + \gamma\tau f(t_{n+1}, y^{n+1}), \qquad \tau = t_{n+1} - t_n, \tag{2.1}$$

where $\gamma = (c+1)/(c+2)$, $c = (t_n - t_{n-1})/(t_{n+1} - t_n)$ and

$$Y^n = \left((c+1)^2 y^n - y^{n-1}\right)/(c^2 + 2c). \tag{2.2}$$

We emphasize that second order is sufficient because for reactive flow problems a low level of accuracy of, say, 1% is good enough. A higher level is thought to be superfluous, due to errors made in other (operator splitting) processes and uncertainties in the reaction constants of the chemistry models.

The Gauss–Seidel technique exploits the chemical kinetics form (1.1), by which (2.1) can be written as

$$y^{n+1} = F(y^{n+1}) := \left(I + \gamma\tau L(t_{n+1}, y^{n+1})\right)^{-1} \left(Y^n + \gamma\tau P(t_{n+1}, y^{n+1})\right). \tag{2.3}$$

The Gauss–Seidel iteration is now straightforwardly applied to the nonlinear system of equations $y = F(y)$ in the standard way. The diagonal form of L makes this process essentially an explicit one. Note that the technique can be implemented in a completely similar way into a diagonally implicit Runge–Kutta code, say, into one of order two with two stages. Due to the one-step nature, much larger differences between successive stepsizes can then be realized. A disadvantage is that more iterations are required per integration step. This is attributable to the computation of the second stage. In our future work, we will explore the efficiency of a two-stage Runge–Kutta method versus that of our BDF method.

Note that for components for which both P_k and L_k are constant in y, the solution of (2.3) is obtained in one iteration. This means that when individual components rapidly approach their steady state values P_k/L_k, they are handled very efficiently. In this connection the current iterative approach bears a resemblance to the explicit pseudo-steady approximation approach evaluated in [9]. The schemes evaluated in [9] show a very good stability behaviour, but proved to be so inaccurate that generally they cannot compete with state-of-the-art, general purpose codes like DASSL [1] and RADAU5 [6], even in the low accuracy range requested. A comparison with DASSL, presented in §4, will show that the iterative Gauss–Seidel technique offers better prospects for the development of fast, special purpose solvers for stiff ODEs in chemically reacting flows. Because the comparison is based on a single test problem, it is obvious that more experimental work is needed to justify completely the use of the Gauss–Seidel technique for chemical sub-models.

3. The prototype solver. We now present a brief outline of our prototype solver. This outline contains all the information needed to appreciate the experimental results presented in the next section. We begin with the variable stepsize strategy for the BDF method. Let E^{n+1} be a local error indicator and consider the weighted error norm

$$(3.1) \qquad ||E^{n+1}||_w = \max(|E_k^{n+1}|/W_k^n), \qquad W_k^n = ATOL + RTOL\,|y_k^n|,$$

where $ATOL$ and $RTOL$ are the absolute and relative error tolerance. If $||E^{n+1}||_w \leq 1.0$, then the integration step is accepted and otherwise rejected. The new stepsize τ_{new} is estimated by

$$(3.2) \qquad \tau_{\text{new}} = \max(0.5, \min(2.0, 0.8/\sqrt{||E^{n+1}||_w}))\tau_{\text{old}}.$$

The square root appears here since our local error indicator is

$$(3.3) \qquad E^{n+1} = \frac{2}{c+1}\left(cy^{n+1} - (1+c)y^n + y^{n-1}\right),$$

which yields $\tau^2 y''(t^n) + O(\tau^3)$ upon substitution of the exact solution. Hence we do not estimate the true local error of the BDF method, which is $O(\tau^3)$. The main reason is that we wish to avoid the explicit use of derivative values in E^{n+1}. As is well known, this would amplify small insignificant errors in the solution because of the stiffness, which hinders the prediction of the new stepsize. In codes using modified Newton iteration, this amplification is suppressed by an additional forward-backward substitution. Because we use Gauss–Seidel iteration, we cannot do this and therefore prefer to use the conservative estimate (3.3), which, in our experience, works well in combination with (3.2) and the iteration strategy described below. It is obvious, though, that more extensive tests are needed to justify (3.3) completely.

The missing starting value is computed with the implicit Euler method. To obtain a safe guess for the initial stepsize, we replace E^{n+1} in (3.1) by $\tau f(t_0, y^0)$ and define τ such that the weighted error norm is equal to one, i.e.,

$$(3.4) \qquad \tau = \min(W_k^0/|f_k(t_0, y^0)|).$$

Hence the initial step is chosen so that the first Taylor series term $\tau f(t_0, y^0)$ satisfies the absolute/relative tolerance requirement. The two-step scheme is then applied with the same stepsize and after that the variable stepsize mechanism is activated. Normally, (3.4) will lead to a rather small initial guess, which will be accepted and subsequently rapidly increased according to (3.2). This is also the case in the experiments reported here.

Let $y^{(i)}$ denote the approximation to y^{n+1} after i iterations with the Gauss–Seidel method for (2.3) or its counterpart for the implicit Euler method. Let $ITOL$ be a tolerance value. As an initial guess we use $y^{(0)} = y^n$. The first iterate $y^{(i)}$ satisfying

$$(3.5) \qquad ||y^{(i)} - y^{(i-1)}||_w \leq ITOL, \qquad i \geq 2,$$

is accepted as the new approximation y^{n+1}. Hence a minimum of two iterations is used at each timestep. The iteration is interrupted if $||y^{(i)} - y^{(i-1)}||_w > ||y^{(i-1)} - y^{(i-2)}||_w$. In the experiments reported here, such failure has not occurred.

Aitken extrapolation is applied for $i \geq 3$. Let $z^{(i)}$ be the Aitken extrapolated value at the ith iteration. Then, if (3.5) does not hold, $z^{(i)}$ is accepted as the new approximation y^{n+1} if

$$||z^{(i)} - z^{(i-1)}||_w \leq ITOL, \qquad i \geq 4. \tag{3.6}$$

Aitken extrapolation is applied to all components and consequently involves overhead, such as componentwise checks on zero division. For simple models the overhead may therefore annihilate the resulting speed in convergence. For the model example used below it has proven to be attractive, especially for larger stepsize values. For small stepsize values in a transient phase where only a few Gauss–Seidel iterations are used, the extrapolation has little effect, of course.

4. Numerical results. Following [8], [9], we present results for an air pollution model from atmospheric chemistry with 20 components and 25 reactions (see the Appendix). The initial data is such that an initial transient is present, while the Lipschitz constant is about $1.5\ 10^7$. This was estimated in [9] by means of an explicit Runge–Kutta integration. Hence the ODE system is very stiff, provided the integration interval is sufficiently large, which is true here. We will give results for $t = 1$ minute, which is slightly after the initial transient, and $t = 60$ minutes, at which time the solution gets close to its steady state.

We also include a comparison with DASSL (see [1]; we have used the double precision version DDASSL available from Netlib [5]). This general purpose BDF solver has been applied as a black box using only default options, except that the initial stepsize was also determined by (3.4). Both DASSL and the new BDF solver can produce negative solution values. However, in the experiments reported here we have not noticed this. It should be noted that the reaction constants in the model example are constant, so that outside the initial transient no sudden large changes in the concentrations occur. This slightly favours DASSL in our comparison. It should also be noted that DASSL is a differential algebraic equation (DAE) solver and hence carries some overhead needed to solve nonlinearly implicit DAEs, even when it is applied to ODEs in normal form, such as (1.1). When this overhead becomes truly noticeable, then it is likely that BDF codes developed for ODEs, like VODE [2] or LSODE [7], will be faster than DASSL. The interested reader is referred to [6, §V.5], where several codes are briefly compared. More specific results obtained with VODE for chemical kinetics problems can be found in [3], [4].

For the prototype solver, Table 1 yields the following information at the specified time $t = T$. SD = the number of significant digits for the maximum relative error, defined by

$$SD = -\log_{10}\left(\max_k \frac{|y_k^n - y_k(T)|}{|y_k(T)|}\right), \tag{4.1}$$

$STEPS$ = the number of integration steps, $ITER$ = the total number of Gauss–Seidel iterations, and CPU = cpu time in seconds. Although CPU is an approximate value and implementation and machine dependent, the given times are indicative for comparison purposes (with an accuracy of at most 0.01 sec. on a Silicon Graphics Indigo workstation, using the Fortran77 Compiler Options -c -r8 -O). In Table 1 results are given for the following two coupled values for $ATOL$ and $RTOL$,

$$ATOL = 10^{-6}TOL, \qquad RTOL = TOL, \qquad TOL = 10^{-1}, 10^{-2}. \tag{4.2}$$

Recall that for reactive flow problems a low level of accuracy suffices, say, 1% ($SD = 2$). For these two values of TOL, the initial stepsize τ_1 determined by (3.4) is equal to $4.7\ 10^{-7}$

and 4.7 10^{-8}, approximately. These small initial stepsizes reveal the initial transient and arise because we take *ATOL* rather small, which is desirable since a number of the concentrations are zero at the initial time and remain small for evolving time. To illustrate the convergence of the Gauss–Seidel iteration, two values for *ITOL* were used, viz., 10^{-2} and 10^{-3}. We emphasize that for the air pollution model considered here, $ITOL = 10^{-2}$ is sufficiently small to come close near the accuracy of the second order BDF formula. Note that inequalities (3.5) and (3.6) are based on the weighted error norm that contains *ATOL* and *RTOL*.

TABLE 1
Values (SD, CPU, STEPS, ITER) using Aitken extrapolation.

		$ITOL = 10^{-2}$	$ITOL = 10^{-3}$
$TOL = 10^{-1}$	$t = 1$	(1.87, 0.03, 42, 153)	(1.87, 0.03, 42, 183)
	$t = 60$	(2.11, 0.04, 56, 273)	(2.40, 0.05, 57, 351)
$TOL = 10^{-2}$	$t = 1$	(2.68, 0.06, 94, 369)	(2.68, 0.07, 94, 438)
	$t = 60$	(3.10, 0.09, 132, 663)	(3.08, 0.11, 132, 773)

We see from Table 1 that for the present example problem Gauss–Seidel iteration accelerated with Aitken extrapolation works very well. The rather high accuracy the prototype code yields for the tolerance values used is due to the conservative error indicator (3.3). Note that for $TOL = 10^{-1}$, $ITOL = 10^{-2}$, we are near the 1% error level. The average number of Gauss–Seidel iterations over the entire interval [0, 60] for this tolerance combination is approximately five. To illustrate the stepsize variation and convergence behaviour of the accelerated Gauss–Seidel iteration, Fig. 1 shows for this tolerance combination a plot of the stepsize sequence τ_n and of the associated number of iterations. We see that over a large range of stepsizes the number of Gauss–Seidel iterations remains limited. Only near the end of the interval, where we get close to the steady state and τ_n becomes quite large, the number of iterations starts to grow.

To show the effect of the Aitken extrapolation, we refer to Table 2, which gives the same information as Table 1, but without application of Aitken extrapolation. As to be expected, Aitken extrapolation becomes truly effective for the smaller tolerances $TOL = 10^{-2}$ and $ITOL = 10^{-3}$, while the convergence acceleration is largest for the larger stepsize values used when we approach steady state.

TABLE 2
Values (SD, CPU, STEPS, ITER) without using Aitken extrapolation.

		$ITOL = 10^{-2}$	$ITOL = 10^{-3}$
$TOL = 10^{-1}$	$t = 1$	(1.87, 0.03, 42, 171)	(1.87, 0.04, 42, 288)
	$t = 60$	(2.10, 0.05, 57, 450)	(2.39, 0.08, 57, 669)
$TOL = 10^{-2}$	$t = 1$	(2.68, 0.06, 94, 484)	(2.68, 0.09, 94, 754)
	$t = 60$	(3.07, 0.12, 132, 1016)	(3.08, 0.18, 132, 1537)

DASSL was also applied for the two coupled tolerances (4.2) and solved the problem without error test and convergence failures. The results of DASSL, in terms of *SD*, *CPU*, *STEPS*, *ITER*, and *JEVS*, are contained in Table 3. Now, *ITER* = the number of modified Newton iterations (or backsolves) and *JEVS* = the number of Jacobian evaluations (or LU decompositions). Note that DASSL computes the Jacobians by numerical differencing. DASSL yields results near the 1% error level for $TOL = 10^{-2}$. Comparing the results for $TOL = 10^{-1}$,

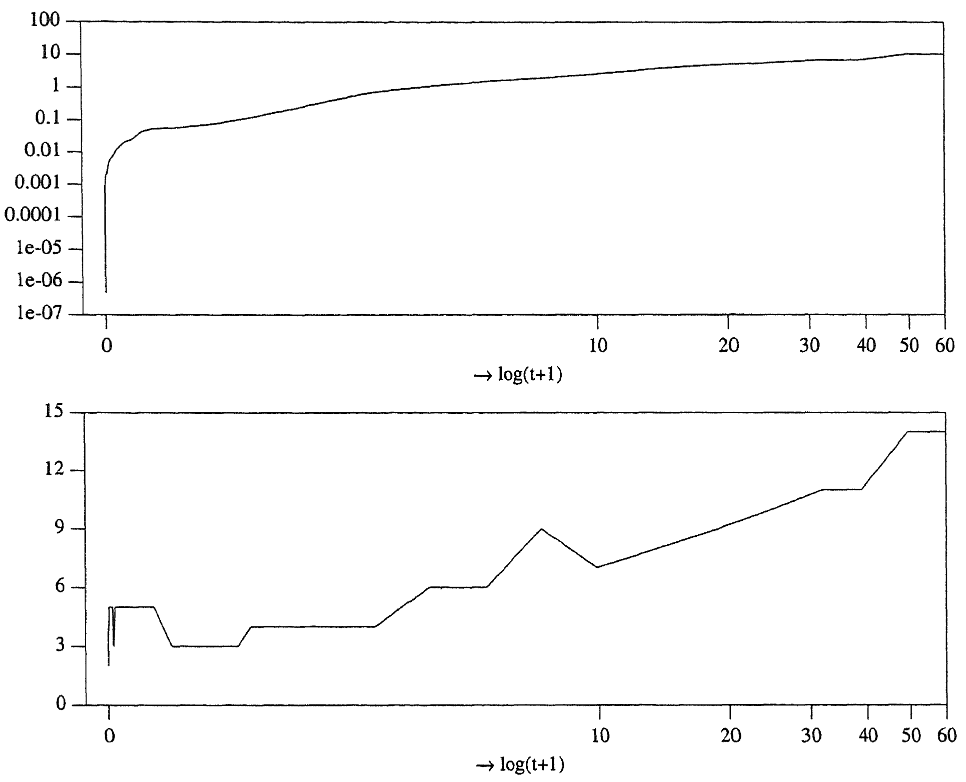

FIG. 1. *Stepsizes (upper plot) and number of Gauss–Seidel iterations (lower plot) for the tolerances $TOL = 10^{-1}$, $ITOL = 10^{-2}$ for $0 \le t \le 60$. Aitken extrapolation was applied.*

ITOL $= 10^{-2}$ from Table 1 with those in Table 3 for *TOL* $= 10^{-2}$, we see that near the 1% error level the prototype code runs approximately three times as fast as DASSL.

TABLE 3
Values (SD, CPU, STEPS, ITER, JEVS) for DASSL.

$TOL = 10^{-1}$	$t = 1$	(0.84, 0.07, 30, 42, 19)
	$t = 60$	(1.20, 0.09, 42, 60, 25)
$TOL = 10^{-2}$	$t = 1$	(2.17, 0.09, 49, 69, 19)
	$t = 60$	(2.09, 0.12, 69, 99, 25)

5. Conclusion. When solving atmospheric flow problems with operator splitting, stiff ODE integrations like the one discussed here must be carried out at thousands of grid points many times in succession. It is therefore of great practical interest to develop special purpose solvers which for this application run considerably faster than very efficient, general purpose codes like DASSL, of course without sacrificing accuracy and reliability. The preliminary results presented here show that our prototype solver, which as yet merely differs from a general purpose code in that a Gauss–Seidel iteration is applied instead of a Newton-type iteration, offers very good prospects for this purpose. An additional advantage of the Gauss–Seidel technique is that it reduces the storage requirements considerably. In large air pollution models the chemistry has to be carried out at thousands of grid points and, therefore, the storage requirement can be a restrictive factor. We will therefore continue our efforts towards the development of a fast stiff ODE solver for chemically reacting atmospheric flow problems along the lines proposed in this paper.

Appendix: Description of the example problem. We give the chemical model with the reaction constants r_k and define the ODE system through the reaction rates v_k, the production terms P_k, and the loss terms L_k. Also a highly accurate reference solution at $t = 1$ and $t = 60$ minutes with the corresponding initial values is given. This reference solution shows the order of the chemical species as used in the ODE system. The units for the rate constants are min^{-1} for first order reactions and $ppm^{-1}min^{-1}$ for the second order ones.

The chemical reactions for the model example.			
01. NO2	→	NO + O3P	.350E+00
02. NO + O3	→	NO2	.266E+02
03. HO2 + NO	→	NO2 + OH	.120E+05
04. HCHO	→	2HO2 + CO	.860E–03
05. HCHO	→	CO	.820E–03
06. HCHO + OH	→	HO2 + CO	.150E+05
07. ALD	→	MEO2 + HO2 + CO	.130E–03
08. ALD + OH	→	C2O3	.240E+05
09. C2O3 + NO	→	NO2 + MEO2 + CO2	.165E+05
10. C2O3 + NO2	→	PAN	.900E+04
11. PAN	→	C2O3 + NO2	.220E–01
12. MEO2 + NO	→	CH3O + NO2	.120E+05
13. CH3O	→	HCHO + HO2	.188E+01
14. NO2 + OH	→	HNO3	.163E+05
15. O3P	→	O3	.480E+07
16. O3	→	O1D	.350E–03
17. O3	→	O3P	.175E–01
18. O1D	→	2OH	.100E+09
19. O1D	→	O3P	.444E+12
20. SO2 + OH	→	SO4 + HO2	.124E+04
21. NO3	→	NO	.210E+01
22. NO3	→	NO2 + O3P	.578E+01
23. NO2 + O3	→	NO3	.474E–01
24. NO3 + NO2	→	N2O5	.178E+04
25. N2O5	→	NO3 + NO2	.312E+01

$$v_1 = r_1 y_1,\ v_2 = r_2 y_2 y_4,\ v_3 = r_3 y_5 y_2,\ v_4 = r_4 y_7,\ v_5 = r_5 y_7,\ v_6 = r_6 y_7 y_6,\ v_7 = r_7 y_9,$$

$$v_8 = r_8 y_9 y_6,\ v_9 = r_9 y_{11} y_2,\ v_{10} = r_{10} y_{11} y_1,\ v_{11} = r_{11} y_{13},\ v_{12} = r_{12} y_{10} y_2,\ v_{13} = r_{13} y_{14},$$

$$v_{14} = r_{14} y_1 y_6,\ v_{15} = r_{15} y_3,\ v_{16} = r_{16} y_4,\ v_{17} = r_{17} y_4,\ v_{18} = r_{18} y_{16},\ v_{19} = r_{19} y_{16},$$

$$v_{20} = r_{20} y_{17} y_6,\ v_{21} = r_{21} y_{19},\ v_{22} = r_{22} y_{19},\ v_{23} = r_{23} y_1 y_4,\ v_{24} = r_{24} y_{19} y_1,\ v_{25} = r_{25} y_{20};$$

$$P_1 = v_2 + v_3 + v_9 + v_{11} + v_{12} + v_{22} + v_{25},\ P_2 = v_1 + v_{21},\ P_3 = v_1 + v_{17} + v_{19} + v_{22},$$

$$P_4 = v_{15},\ P_5 = v_4 + v_4 + v_6 + v_7 + v_{13} + v_{20},\ P_6 = v_3 + v_{18} + v_{18},\ P_7 = v_{13},$$

$$P_8 = v_4 + v_5 + v_6 + v_7,\ P_9 = 0.0,\ P_{10} = v_7 + v_9,\ P_{11} = v_8 + v_{11},\ P_{12} = v_9,\ P_{13} = v_{10},$$

$$P_{14} = v_{12},\ P_{15} = v_{14},\ P_{16} = v_{16},\ P_{17} = 0.0,\ P_{18} = v_{20},\ P_{19} = v_{23} + v_{25},\ P_{20} = v_{24};$$

$$L_1 = r_1 + r_{10} y_{11} + r_{14} y_6 + r_{23} y_4 + r_{24} y_{19},\ L_2 = r_2 y_4 + r_3 y_5 + r_9 y_{11} + r_{12} y_{10},$$

$$L_3 = r_{15},\ L_4 = r_2 y_2 + r_{16} + r_{17} + r_{23} y_1,\ L_5 = r_3 y_2,\ L_6 = r_6 y_7 + r_8 y_9 + r_{14} y_1 + r_{20} y_{17},$$

$$L_7 = r_4 + r_5 + r_6 y_6,\ L_8 = 0.0,\ L_9 = r_7 + r_8 y_6,\ L_{10} = r_{12} y_2,\ L_{11} = r_9 y_2 + r_{10} y_1,$$

$$L_{12} = 0.0,\ L_{13} = r_{11},\ L_{14} = r_{13},\ L_{15} = 0.0,\ L_{16} = r_{18} + r_{19},\ L_{17} = r_{20} y_6,$$

$$L_{18} = 0.0,\ L_{19} = r_{21} + r_{22} + r_{24} y_1,\ L_{20} = r_{25}.$$

Concentration values in ppm at, respectively, $t = 0$ min., $t = 1$ min., and $t = 60$ min.				
1.	[NO2]	0	0.37326304298606E–01	0.56462554800124E–01
2.	[NO]	0.2	0.16251325412704E+00	0.13424841304232E+00
3.	[O3P]	0	0.27344389305923E–08	0.41397343310918E–08
4.	[O3]	0.04	0.32994065756620E–02	0.55231402074779E–02
5.	[HO2]	0	0.31151619385738E–06	0.20189772623092E–06
6.	[OH]	0	0.26534918500427E–06	0.14645418635004E–06
7.	[HCHO]	0.1	0.99423103666384E–01	0.77842491190039E–01
8.	[CO]	0.3	0.30061731277555E+00	0.32450753533953E+00
9.	[ALD]	0.01	0.99269949383466E–02	0.74940133838905E–02
10.	[MEO2]	0	0.29529601825322E–07	0.16222931573089E–07
11.	[C2O3]	0	0.20994901153721E–07	0.11358638332624E–07
12.	[CO2]	0	0.65714929538122E–04	0.22305059757112E–02
13.	[PAN]	0	0.59742964642105E–05	0.20871628827982E–03
14.	[CH3O]	0	0.27858639499927E–04	0.13969210168610E–04
15.	[HNO3]	0	0.13959464032840E–03	0.89648848569121E–02
16.	[O1D]	0	0.26002979092156E–17	0.43528463693250E–17
17.	[SO2]	0.007	0.69973974657447E–02	0.68992196962640E–02
18.	[SO4]	0	0.26025342552954E–05	0.10078030373603E–03
19.	[NO3]	0	0.38171954506700E–06	0.17721465139664E–05
20.	[N2O5]	0	0.72454590089901E–05	0.56829432922952E–04

REFERENCES

[1] K. E. Brenan, S. L. Campbell, and L. R. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, North-Holland, Amsterdam, 1989.

[2] P. N. Brown, G. D. Byrne, and A. C. Hindmarsh, *VODE: A variable step ODE solver*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 1038–1051.

[3] G. D. Byrne, *The taming of a co-polymerization problem with VODE*, IMPACT of Computing in Science and Engineering, 5 (1993), pp. 318–344.

[4] G. D. Byrne and A. M. Dean, *The numerical solution of some chemical kinetics models with VODE and CHEMKIN II*, Computers Chem., 17 (1993), pp. 297–302.

[5] J. J. Dongarra and E. Grosse, *Distribution of software via electronic mail*, Comm. ACM, 30 (1987), pp. 403–407. (Available through netlib@research.att.com.)

[6] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations* II. *Stiff and Differential-Algebraic Problems*, Springer-Verlag, Berlin, 1991.

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

[8] F.A.A.M. de Leeuw, *Numerical Solution of Ordinary Differential Equations Arising from Chemical Kinetics*, Report 228603005, National Institute of Public Health and Environmental Protection (RIVM), Bilthoven, The Netherlands, 1988.

[9] J. G. Verwer and M. van Loon, *An Evaluation of Explicit Pseudo-Steady State Approximation Schemes for Stiff ODE Systems from Chemical Kinetics*, CWI Report NM-R9312, Centre for Mathematics and Computer Science, Amsterdam, 1993; J. Comput. Phys., to appear.

[10] Z. Zlatev and J. Wasniewsky, *Large Scale Computations in Air Pollution Modelling*, Report UNIC-92-05, Scientific Computing Group, Danmarks EDB-Center for Forskning og Uddannelse, 1992.