NOTE

An Evaluation of Explicit Pseudo-Steady-State Approximation Schemes for Stiff ODE Systems from Chemical Kinetics

1. INTRODUCTION

Systems of ordinary differential equations (ODEs) describing chemical kinetics problems can be cast in the special form

$$y' = f(y) := P(y) - L(y)\,y, \qquad y(t) = [y_1(t), ..., y_m(t)]^T, \tag{1}$$

where

$$P(y) = [P_1(y), ..., P_m(y)]^T, \qquad L(y) = \operatorname{diag}[L_1(y), ..., L_m(y)], \tag{2}$$

and $\operatorname{diag}[L_1(y), ..., L_m(y)]$ stands for the diagonal matrix with entries $L_k(y)$ for $k = 1, ..., m$. The components $P_k(y)$ and $L_k(y)\,y_k$ are nonnegative and represent, respectively, the production and loss rate for the kth "concentration" y_k. The reciprocal of $L_k(y)$ is the physical time constant or characteristic reaction time for y_k. In most applications the range of time constants is large, which causes the ODE system to be stiff. For simplicity of presentation we here confine ourselves to the autonomous case. A chemical kinetics problem will become non-autonomous when the reaction constants are made time-dependent. This occurs, for example, in air pollution models involving temperature dependent reaction rates. All schemes discussed can be made non-autonomous in the usual way, either directly or by treating time as a dependent variable.

Many applications, such as those from air pollution modelling, give rise to a system of partial differential equations of the advection-diffusion-reaction type in which the ODE system (1) occurs as the reaction part. A popular approach for solving this kind of problems is operator splitting. Splitting involves the numerical integration of (1) at thousands of gridpoints, for each step of the operator splitting method. Because the costs of integrating (1) then readily dominate the total costs, one often applies so-called pseudo-steady-state approximation (PSSA) or asymptotic approximation schemes. Such schemes exploit the specific form (1) and, in contrast to modern, general purpose stiff solvers, they are explicit. Hence a PSSA scheme is extremely cheap, per integration step. PSSA schemes also show remarkably good stability, but can attain only low accuracy. An example of an algorithm based on this approach is CHEMEQ, which has been proposed by Young and Boris [15, 16] and recommended in [11, 13]. Early comparisons with the implicit backward differentiation (Gear type) codes DIFSUB [15, 6] and EPISODE [11, 2] are in favour of the PSSA approach and also in more recent work this approach is still applied and advocated [8, 9, 17, 18].

The present paper reports on results which are less favourable for PSSA, however. For a number of stiff ODEs from atmospheric chemistry we have compared a simple, easy-to-use PSSA solver with two state-of-the-art solvers from the stiff ODE field, viz., the implicit Runge–Kutta code RADAU5 developed by Hairer and Wanner [7] and the implicit backward differentiation code DASSL developed by Petzold [1]. In spite of their considerable overhead costs, our findings indicate that in most cases the implicit codes can be made more efficient, even for the low accuracy range that is of interest for reactive flow problems. In addition, the implicit codes prove also more reliable than the PSSA solver we have applied.

2. PSSA SCHEMES

If P and L are constant, then (1) can be solved exactly, i.e.,

$$y(t+\tau) = e^{-\tau L}y(t) + (I - e^{-\tau L})\,L^{-1}P. \tag{3}$$

This suggests considering the associated integration scheme,

$$y^{n+1} = R(-\tau L^n)\,y^n + \tau(R(-\tau L^n) - I)(-\tau L^n)^{-1}\,P^n, \qquad \tau = t_{n+1} - t_n, \tag{4}$$

where y^n is the approximation to $y(t_n)$, $L^n = L(y^n)$, $P^n = P(y^n)$, and $R(z) = e^z$ or a suitable consistent approximation, e.g., a Padé approximation. Note that (4) is explicit because L^n and $R(-\tau L^n)$ are diagonal matrices. A second attractive property is that nonnegative solutions are generated for any $\tau > 0$ if $R(z)$ and $(R(z)-1)/z$ are nonnegative for all $z \leqslant 0$.

If $P_k(y)$ and $L_k(y)$ vary very slowly, it is obvious that scheme (4) makes sense as a replacement of (3). This is the case when we are sufficiently close to the steady state value P_k/L_k. Due to the fact that for any given value y_k^n, the new approximation $y_k^{n+1} \to P_k/L_k$ for $\tau L_k \to \infty$, if $R(-\infty)=0$, scheme (4) is called a PSSA scheme or asymptotic approximation scheme. Scheme (4) is consistent of order one, because if we substitute the exact solution value $y(t_n)$ for y^n, we obtain

$$y^{n+1} = y(t_n) + \tau y'(t_n) + Q(-\tau L)\,\tau y'(t_n), \qquad L = L(y(t_n)), \tag{5}$$

and we see that first-order consistency holds for components for which $\tau \ll 1/L_k$, due to the fact that $Q(z) = (R(z)-z-1)/z = O(z)$, $z \to 0$. This concerns the nonstiff $(\tau \ll 1/L_k)$ components y_k. For the other, stiff components, the first-order consistency result is not true, since for these the order relation for $Q(z)$ cannot be imposed. In fact, we have $Q(z) \sim 1$, uniformly in $z \leqslant 0$, so that for the stiff components a zero consistency order holds. This provides an example of local order reduction (see [4, Chap. 7]). Of course, close to equilibrium this reduction will not be felt. However, if a component is not close to equilibrium and yet $\tau \gg 1/L_k$, the PSSA scheme may introduce the steady state too quickly. This indicates that the accuracy can be low and unpredictable, to some extent, for large complicated chemical kinetics problems containing widely different time scales.

The above accuracy observations are local and what counts is the global accuracy. The global accuracy depends on the consistency properties, on the stability, and hence on the precise way local errors accumulate to global ones. For well-established implicit Runge–Kutta schemes we know that in the transition from local to global, part of the local errors due to stiffness is often annihilated [4]. For the PSSA scheme a comprehensive error analysis seems not feasible, at least not for general nonlinear systems (1). Our practical experience is that schemes like (4) perform remarkably well with respect to stability, at least when taking into account that the stability is introduced solely by the diagonal matrix $L(y)$. However, albeit this may seem advantageous, it is also dangerous for schemes which are inaccurate, because it might give a false impression of reliability.

For nonstiff components second-order consistency can be obtained with the scheme

$$y^{n+1} = R(-\tau L^{n+1/2})\,y^n + \tau(R(-\tau L^{n+1/2}) - I) \times (-\tau L^{n+1/2})^{-1}\,P^{n+1/2}, \tag{6}$$

provided that $R(z)$ is second-order consistent and that the diagonal matrix $L^{n+1/2}$, as yet undefined, satisfies

$$L^{n+1/2} = L(y(t_n)) + \tau/2 L'(y(t_n))\,y'(t_n) + O(\tau^2), \tag{7}$$

upon substitution of the true solution (and similar for $P^{n+1/2}$). The second-order consistency follows from a straightforward Taylor expansion. Note the resemblance with (4). In particular, under the same conditions on $R(z)$, scheme (6) also guarantees nonnegative solutions.

Freedom exists in defining the intermediate values $L^{n+1/2}$ and $P^{n+1/2}$. If we put

$$\begin{aligned} L^{n+1/2} &= L((y^{n+1}+y^n)/2),\\ P^{n+1/2} &= P((y^{n+1}+y^n)/2) \end{aligned} \tag{8}$$

and select the second-order diagonal Padé approximation

$$R(z) = (2+z)/(2-z), \tag{9}$$

we in fact recover the classical, fully implicit midpoint rule. Likewise, with (9) and

$$\begin{aligned} L^{n+1/2} &= (L(y^n) + L(y^{n+1}))/2,\\ P^{n+1/2} &= (P(y^n) + P(y^{n+1}))/2, \end{aligned} \tag{10}$$

we obtain a third-order perturbation of the classical, fully implicit trapezoidal rule given by

$$y^{n+1} = y^n + \tau/2 f^n + \tau/2 f^{n+1} + \tau/4(L^{n+1} - L^n) \times (y^{n+1} - y^n). \tag{11}$$

From the equivalent implicit form

$$y^{n+1} = y^n + 2\tau(4I + \tau(L^{n+1} + L^n))^{-1} \times (P^{n+1} - L^{n+1}y^n + f^n), \tag{12}$$

the explicit, stiff corrector formula used in CHEMEQ [11, 16] is derived. Let ζ^{n+1} denote a predicted value and y^{n+1}, the associated corrected one. This explicit stiff corrector then reads

$$y^{n+1} = y^n + 2\tau(4I + \tau(L(\zeta^{n+1}) + L^n))^{-1} \times (P(\zeta^{n+1}) - L^n y^n + f^n). \tag{13}$$

Note that $L^{n+1}y^n$ is replaced by $L^n y^n$ and not by $L(\zeta^{n+1})\,y^n$. In [15; 13, p. 161] the implicit form

$$y^{n+1} = (T^{n+1} + T^n + \tau I)^{-1}\,[(T^{n+1} + T^n - \tau I)\,y^n + \tfrac{1}{2}\tau(T^{n+1} + T^n)(P^{n+1} + P^n)], \tag{14}$$

is used, where $T^n = (L^n)^{-1}$, i.e., the diagonal matrix of time constants. This formula is equivalent to (11) and (12), as it is also obtained from (6) and (9) by substitution of $T^{n+1/2} = (L^{n+1/2})^{-1}$ and by using definition (10) for $P^{n+1/2}$ and $T^{n+1/2}$.

We are not in favour of starting from (12) or (14), simply because (9) does not decay like e^z for $z<0$. Hence, small perturbations from the equilibrium state are no longer rapidly damped for $\tau L_k \to \infty$, which is the clue of the PSSA approach. Also note that (9) does not guarantee a nonnegative solution for all values of the stepsize τ. A more obvious choice is the second-order subdiagonal Padé approximation

$$R(z) = 1/(1 - z + \tfrac{1}{2}z^2), \tag{15}$$

since this one nicely mimics the damping of e^z for $z<0$ and also guarantees nonnegativity for (4) and (6). Trivially, one could use the exponential itself, like in [9], but this may lead to zero divisions in the expression $(R(z)-1)/z$ and hence requires an additional check or modification. The use of the approximation (15) avoids this.

The scheme for the PSSA solver that will be compared to RADAU5 and DASSL in Section 5 uses (15) and can now be defined. It has two stages. In stage one, the predictor stage, we apply (4). In stage two, the corrector stage, we apply (6), (10) while the first stage result ζ^{n+1} is substituted for y^{n+1},

$$\begin{aligned} &(I + Z + \tfrac{1}{2}Z^2)\,\zeta^{n+1} \\ &\quad = y^n + \tau(I + \tfrac{1}{2}Z)\,P^n, \qquad Z = \tau L^n, \\ &(I + Z + \tfrac{1}{2}Z^2)\,y^{n+1} \\ &\quad = y^n + \tau(I + \tfrac{1}{2}Z)\,P^{n+1/2}, \qquad Z = \tau L^{n+1/2}, \end{aligned} \tag{16}$$

where $L^{n+1/2} = \frac{1}{2}(L^n + L(\zeta^{n+1}))$ and $P^{n+1/2} = \frac{1}{2}(P^n + P(\zeta^{n+1}))$. One may also use (8) instead of (10). Numerical tests have revealed that this leads to only minor differences. Note that for components for which $L_k(y) = 0$, stage one yields the explicit Euler formula and stage two the explicit trapezoidal rule.

All experiments discussed in Section 5 have been repeated with (16) replaced by the two-stage scheme

$$\begin{aligned} &(I + Z)\,\zeta^{n+1} = y^n + \tau P^n, \qquad Z = \tau L^n, \\ &\quad y^{n+1} = \max\{0,\, y^n + 2\tau(4I + \tau(L(\zeta^{n+1}) + L^n))^{-1} \\ &\qquad \times (P(\zeta^{n+1}) - L^n y^n + f^n)\}, \end{aligned} \tag{17}$$

of which stage two comes from (13) and stage one results from (4) by defining $R(z) = 1/(1-z)$. The stage one formula is the predictor used in CHEMEQ. In all our experiments, (16) did perform notably better than this CHEMEQ pair. The difference in performance is due to the difference in asymptotic behaviour between the stability functions (9) and (15). In the remainder of this article results will only be given for (16).

Finally we note that in PSSA algorithms one usually assigns a steady state value P_k/L_k to components y_k for values of τL_k larger than a certain threshold. Similarly, for values of τL_k smaller than a second threshold, simple explicit integration formulas are introduced to replace the PSSA formulas. For large chemical systems a sensible choice of these thresholds is, in general, not an easy task. By using (15) there is less need to either, simply because for $\tau L_k \to \infty$ the exponential damping to steady state is assured, while for $\tau L_k \to 0$ the usual order concept applies and then the PSSA formulas behave as accurate as standard explicit formulas of the same order. In fact, these two limit cases are dealt with sufficiently accurately in the PSSA approach. However, for components not yet in steady state intermediate values of τL_k, associated with intermediate time constants, may give rise to less accurate results, due to the fact that the size of τL_k influences the local truncation error in a manner similar as outlined for formula (4).

3. CONNECTIONS WITH RUNGE–KUTTA METHODS

For (most) implicit RK methods, having $R(z)$ as stability function, (4) results after one modified Newton iteration if y^n is taken as the start vector and L^n as the (very crude) approximate Jacobian. Scheme (4) is also known as the most simple example of a RK–Rosenbrock or linearly implicit RK method [14, Chap. 4], when L^n is again interpreted as an approximate Jacobian. Also the most simple W-method [7, Section IV.7] can be brought in the form (4). Along the RK line it thus is possible to consider the application of PSSA type schemes of classical order higher than two. To see whether this could be advantageous, we have implemented the third-order, L-stable W-method W3L from [12], using L^n in all its four stages as the approximate Jacobian. Unfortunately, the performance of this explicit W3L turned out to be rather disappointing compared to the second-order PSSA scheme (16) (therefore no test results will be given for W3L). The main reasons are that W3L is roughly a factor three more expensive, per step, while it also generates negative solutions. Because the number of consistency conditions to be fulfilled increases very rapidly with the order, no further attempts have been made towards a scheme of order four or higher.

4. FOUR SOLVERS

We have experimented with four solvers, here called PSSA, RADAU, DASSL, and TRAP. PSSA is based on (16). The difference $E^{n+1} = y^{n+1} - \zeta^{n+1}$ is used as the local error estimator. Recall that the classical order of the two stages is one and two, respectively. Because the classical order concept is of limited value here (cf. Section 2), E^{n+1} will act as a crude estimate. However, for our comparison purpose E^{n+1} is good enough. Define

$$\|E^{n+1}\|_w = \max_k (|E_k^{n+1}|/W_k), \qquad W_k = \text{ATOL} + \text{RTOL}\,|y_k^n|, \tag{18}$$

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

$$\tau_{\text{new}} = \max(0.2, \min(8.0, 0.8/\sqrt{\|E^{n+1}\|_w}))\,\tau_{\text{old}}. \tag{19}$$

Hence the stepsize ratio is constrained by 0.2 and 8.0, which means that we allow a very rapid increase or decrease. It should be stressed that this is a highly desirable feature in the setting of operator splitting, due to the necessary restart within every operator splitting step. We copied these growth factors from the stepsize control from RADAU5.

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

$$\tau = \min_k (W_k/|f_k(y^0)|), \qquad W_k = \text{ATOL} + \text{RTOL}\,|y_k^0|. \tag{20}$$

Hence the initial step is chosen such that the first Taylor term $\tau f(y^0)$ satisfies the absolute/relative tolerance requirement. If this still results in a step rejection, we divide the initial guess by 10.0 until acception follows and then proceed as usual (cf. the RADAU5 strategy). Normally, however, (20) will lead to a rather small initial guess, which will be accepted and subsequently rapidly increased according to (19).

RADAU5 is the Nov. 14, 1989 version of the solver discussed in [7] and DASSL [1] is the double precision version DDASSL taken from netlib [5]. Both codes possess their own strategies and we have applied them as black boxes using only default options, except that the initial stepsize is determined by (20). Hence any possibility to let them run faster has been omitted. For example, the Jacobian matrix is computed numerically and always treated as a full matrix. Note that both can produce negative solution values. However, in the experiments reported here this has never been a problem.

TRAP is based on the explicit trapezoidal rule

$$\zeta^{n+1} = y^n + \tau f^n, \qquad y^{n+1} = y^n + \tfrac{1}{2}\tau(f^n + f(\zeta^{n+1})) \tag{21}$$

and chooses its stepsizes in precisely the same way as PSSA. We have used TRAP to assess in a simple way the degree of stiffness of the example problems. As a solver for stiff problems TRAP makes no sense, of course, due to the severe stability restriction. Note, however, that per integration step the costs of PSSA and TRAP are highly comparable, so that PSSA can be compared to TRAP to illustrate the large gain in stability of the explicit pseudo-steady-state formulas over standard explicit ones like (21).

5. THE NUMERICAL COMPARISON

We have selected three example problems from atmospheric chemistry, here called ATMOS7, ATMOS12, and ATMOS20 (the integer denotes the dimension). ATMOS7 is an atmospheric chemical relaxation test problem involving cesium and cesium ions. We borrowed this problem from [15, 16], where it is used to illustrate CHEMEQ. The two problems ATMOS12 and ATMOS20 emanate from an air pollution study and were borrowed from [10, 9], respectively. A complete specification of the problems is given in the appendix of the preprint to this paper which can be obtained from the first author.

In the tables of results we give the values (sd, cpu, steps), where sd is the number of significant digits for the maximum relative error,

$$\text{sd} = -{}^{10}\log\left(\max_k \frac{|y_k^n - y_k(T)|}{|y_k(T)|}\right), \tag{22}$$

cpu is the CPU time in seconds, and steps is the number of accepted plus rejected integration steps. Steps and cpu serve to measure the efficiency. Although cpu is an approximate value and implementation- and machine-dependent, the given values are good enough for comparison purposes (they have been checked by repeating all experiments several times). We confine ourselves to giving CPU times, in addition to steps, as it is not feasible to discuss and compare the full statistics of all integrations, due to a too widely differing workload of the three solvers. While the costs per step of PSSA can be expressed, approximately, as two $f(y)$-evaluations, similar for the two-stage explicit trapezoidal rule (21), the costs of the implicit solvers DASSL and RADAU are dominated by the numerical algebra overhead arising from the iterative modified Newton solution of the encountered systems of nonlinear algebraic equations. This overhead cannot be expressed in a simple cost unit per step, as it consists of Jacobian updates, of LU-decompositions, and of backsolves, which, in addition, for RADAU are also more expensive than for DASSL (see [7, Section IV.8]). Also note that the numbers of Jacobian updates and LU-decompositions always differ from the number of integration steps.

We also tabulate the initial stepsize τ_1 and the length T of the integration interval. For simplicity we have used one set of tolerances in all experiments with PSSA, RADAU, and DASSL, viz.

$$\begin{aligned} \text{ATOL} &= 10^{-6}\,\text{TOL}, \quad \text{RTOL} = \text{TOL}, \\ \text{TOL} &= 10^{-i} \quad \text{for} \quad i = 1, 2, 3, 4. \end{aligned} \tag{23}$$

It should be emphasized that for reactive flow problems a low level of accuracy suffices, say 1% (sd = 2). A higher level is redundant due to errors made in other (operator splitting) processes and uncertainties in the reaction constants of the chemistry model.

To assess the degree of stiffness of the three test problems, we have first applied TRAP using the tolerances RTOL = ATOL = 0.1. For these tolerances the stepsizes for ATMOS12 and ATMOS20 are completely restricted by stability, for all times. For ATMOS12 the variable stepsize mechanism selects as the maximum stable stepsize $2.0_{10^{-6}}$, approximately, which indicates a Lipschitz constant for the right-hand side function of approximately 10^6. For ATMOS20 these approximate values are, respectively, $1.3_{10^{-7}}$ and $1.5_{10^{+7}}$. Hence, both are excessively stiff. For ATMOS12 and ATMOS20 the explicit solver TRAP would require a total of about 60 million and 460 million accepted integration steps, respectively, over the selected time intervals (see Tables II, III). ATMOS7 differs from the other two problems, in the sense that for a number of components production strongly dominates over part of the integration steps. During these production steps the stepsize smoothly increases from about $1.2_{10^{-11}}$ to a maximum of about 3.25, which is then maintained until the final time $T = 1000$. When the maximum stepsize is reached, production has stopped and the stepsize is determined solely by stability. ATMOS7 thus turns out to be only moderately stiff. The integration for RTOL = ATOL = 0.1 with TRAP requires 6305 accepted steps plus three rejected ones and delivers sd = 1.73.

The results for PSSA, RADAU, and DASSL have been collected in Tables I–III. The following conclusions can be made: The explicit solver PSSA really beats the explicit solver TRAP, illustrating its much improved stability. In fact, with respect to stability, PSSA compares well with the implicit solvers. PSSA performs well for ATMOS7 for which it outperforms the two implicit solvers for TOL = 10^{-1}, 10^{-2}. For the smaller values of TOL the implicit codes become more efficient than PSSA due to their high order. This is of minor relevance, though, as high accuracy is redundant for the present application. The difference in the number of time steps between the two implicit solvers is due to the fact that RADAU5 admits a larger stepsize growth. As a result, for this problem RADAU5 manages to spent approximately the same CPU time as DASSL, despite its larger numerical algebra overhead. For a larger dimension this may no longer be true.

For ATMOS12 and ATMOS20 the situation is different and less promising for PSSA, despite the larger problem dimension which is a disadvantage for the implicit solvers in view of the numerical algebra overhead. PSSA now returns less accurate solutions and, hence, needs too many time steps near the 1% error level to beat the implicit solvers in speed. For ATMOS12 the explicit PSSA code still performs rather well, but for ATMOS20 the two implicit codes are significantly faster near the 1% error level. We conjecture that this is due to the presence of intermediate time constants rendering the PSSA approach inaccurate (cf. the remark at the end of Section 3). Clearly, the more complicated the chemical scheme, the more intermediate time constants can play this negative role. A closer inspection of Table II reveals that for ATMOS12 and a lower accuracy level, say about sd = 1.0 or 10% error, PSSA still compares favourably with both the implicit solvers. Unfortunately, for ATMOS20 the advantage of the low costs of PSSA is not borne out convincingly, not even at this rather low accuracy level. Finally, the experiments of this section have been

TABLE I

The Values (sd, cpu, steps) for ATMOS7 with $T = 1000$

(TOL, τ_1)	PSSA	RADAU5	DASSL
$(10^{-1}, 1.6_{10^{-18}})$	(1.53, 0.012, 116)	(1.82, 0.08, 49)	(1.43, 0.09, 134)
$(10^{-2}, 1.6_{10^{-19}})$	(2.44, 0.051, 456)	(2.82, 0.09, 49)	(1.97, 0.10, 175)
$(10^{-3}, 1.6_{10^{-20}})$	(3.43, 0.182, 1639)	(3.89, 0.11, 61)	(3.66, 0.15, 265)
$(10^{-4}, 1.6_{10^{-21}})$	(4.41, 0.610, 5479)	(5.63, 0.14, 92)	(3.96, 0.21, 356)

TABLE II

The Values (sd, cpu, steps) for ATMOS12 with $T = 120$

(TOL, τ_1)	PSSA	RADAU5	DASSL
$(10^{-1}, 2.5_{10^{-5}})$	(0.77, 0.003, 18)	(0.21, 0.08, 25)	(1.37, 0.04, 30)
$(10^{-2}, 2.5_{10^{-6}})$	(0.94, 0.006, 38)	(1.89, 0.05, 19)	(1.59, 0.05, 49)
$(10^{-3}, 2.5_{10^{-7}})$	(1.22, 0.020, 130)	(3.12, 0.06, 23)	(2.67, 0.07, 77)
$(10^{-4}, 2.5_{10^{-8}})$	(2.14, 0.090, 595)	(4.16, 0.09, 32)	(3.48, 0.09, 112)

TABLE III

The Values (sd, cpu, steps) for ATMOS20 with $T = 60$

(TOL, τ_1)	PSSA	RADAU5	DASSL
$(10^{-1}, 4.7_{10^{-07}})$	(0.09, 0.006, 29)	(2.08, 0.13, 20)	(1.20, 0.10, 42)
$(10^{-2}, 4.7_{10^{-08}})$	(0.41, 0.024, 123)	(4.17, 0.14, 23)	(2.09, 0.12, 69)
$(10^{-3}, 4.7_{10^{-09}})$	(1.13, 0.133, 676)	(4.86, 0.20, 32)	(3.56, 0.17, 108)
$(10^{-4}, 4.7_{10^{-10}})$	(2.27, 0.930, 4700)	(5.19, 0.31, 48)	(4.06, 0.24, 173)

repeated with the time interval halved. No notable differences with the results obtained on the original intervals were observed.

6. FINAL REMARKS

Two more chemical reaction kinetics problems have been experimented with, viz., the eight-species problem HIRES used as a test example in [7] and a 12-species chemical pyrolysis problem from [3]. These experiments again favour the implicit approach. We should also recall that common implicit integration formulas are conservative. The conservation error generated by implicit solvers only depends on the accuracy at which the implicit equations are solved. On the other hand, PSSA schemes are not conservative and therefore may easily generate larger conservation errors.

Despite their better performance and robustness, it is clear that standard implicit solvers will still require an enormous amount of CPU time in a reactive flow calculation where the chemical equations must be integrated at thousands of grid points at many operator splitting steps. Therefore, more research into fast implicit chemical integrators for reactive flow problems is needed. This research should be directed towards reducing the numerical algebra overhead costs spent in solving implicit relations for reactive flow calculations. Several possibilities to achieve this goal are conceivable and we plan to report on the subject in the near future.

ACKNOWLEDGMENTS

The research reported belongs to the project EUSMOG which is carried out in cooperation with the Air Laboratory of the RIVM—The Dutch National Institute of Public Health and Environmental Protection, Bilthoven. The RIVM is acknowledged for financial support.

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. G. D. Byrne and A. C. Hindmarsh, *ACM Trans. Math. Software* **1**, 71 (1975).
3. A. K. Datta, Technical Report MSH/67/84, Imperial Chemical Industries, Cheshire, 1967 (unpublished).
4. K. Dekker and J. G. Verwer, *Stability of Runge–Kutta Methods for Stiff Nonlinear Differential Equations* (North-Holland, Amsterdam, 1984).
5. J. J. Dongarra and E. Grosse, *Commun. ACM* **30**, 403 (1987).
6. C. W. Gear, *Commun. ACM* **14**, 185 (1971).
7. E. Hairer and G. Wanner, *Solving Ordinary Differential Equations. II. Stiff and Differential-Algebraic Problems* (Springer-Verlag, New York/Berlin, 1991).
8. Ø. Hov, Z. Zlatev, R. Berkowicz, A. Eliassen and L. P. Prahms, *Atmos. Environ.* **23**, 967 (1989).
9. F. A. A. M. de Leeuw, Report 228603005, National Institute of Public Health and Environmental Protection (RIVM), Bilthoven, The Netherlands, 1988 (unpublished).
10. F. A. A. M. de Leeuw, private communication, National Institute of Public Health and Environmental Protection (RIVM), Bilthoven, The Netherlands, 1993.
11. G. J. McRae, W. R. Goodin, and J. H. Seinfeld, *J. Comput. Phys.* **45**, 1 (1982).
12. A. Ostermann, Dissertation, Universität Innsbruck, 1988 (unpublished).
13. E. S. Oran and J. P. Boris, *Numerical Simulation of Reactive Flow* (Elsevier, Amsterdam/New York, 1987).
14. K. Strehmel and R. Weiner, *Linear-implizite Runge–Kutta Methoden und ihre Anwendung*, Teubner Texte zur Mathematik, Band 127 (Teubner, Stuttgart/Leipzig, 1992).
15. T. R. Young and J. P. Boris, *J. Phys. Chem.* **81**, 2424 (1977).
16. T. R. Young, NRL Memorandum Report 4091, Naval Research Laboratory, Washington, DC, 1979 (unpublished).
17. Z. Zlatev and J. Wasniewski, Report UNIC-92-05, Scientific Computing Group, Danmarks EDB-Center for Forskning og Uddannelse, 1992 (unpublished).
18. Z. Zlatev, J. Christensen, and Ø. Hov, *J. Atmos. Chem.* **15**, 1 (1992).

Received August 3, 1993; revised January 13, 1994

J. G. Verwer
M. van Loon

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