ELSEVIER

Journal of Computational and Applied Mathematics 100 (1998) 11–21

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

The use of approximate factorization in stiff ODE solvers

P.J. van der Houwen *, B.P. Sommeijer

CWI, P.O. Box 94079, 1090GB Amsterdam, The Netherlands

Received 19 November 1997; received in revised form 25 June 1998

Abstract

We consider implicit integration methods for the numerical solution of stiff initial-value problems. In applying such methods, the implicit relations are usually solved by Newton iteration. However, it often happens that in subintervals of the integration interval the problem is nonstiff or mildly stiff with respect to the stepsize. In these nonstiff subintervals, we do not need the (expensive) Newton iteration process. This motivated us to look for an iteration process that converges in mildly stiff situations and is less costly than Newton iteration. The process we have in mind uses modified Newton iteration as the outer iteration process and a linear solver for solving the linear Newton systems as an inner iteration process. This linear solver is based on an approximate factorization of the Newton system matrix by splitting this matrix into its lower and upper triangular part. The purpose of this paper is to combine fixed point iteration, approximate factorization iteration and Newton iteration into one iteration process for use in initial-value problems where the degree of stiffness is changing during the integration.

Keywords: Numerical analysis; Initial-value problems; Ordinary differential equations; Approximate factorization

1. Introduction

We consider implicit integration methods for the numerical solution of stiff initial-value problems (IVPs) for the ordinary differential equation (ODE)

$$\frac{dy}{dt} = f(y), \quad y, f \in \mathbb{R}^d, \quad t \geqslant t_0. \tag{1.1}$$

The conventional way of solving the implicit relations is the modified Newton (MN) iteration process. MN iteration possesses a relatively large convergence region, that is, if h is the stepsize used, then the quantity $\|h\partial f/\partial y\|$ is allowed to be of large magnitude. However, it often happens that in subintervals of the integration interval the problem is nonstiff or mildly stiff with respect to h, implying that $\|h\partial f/\partial y\|$ is of moderate size. In these nonstiff subintervals, we do not need the (expensive) MN iteration process with its large convergence region and we would like to use a less costly method in

* Corresponding author. E-mail: senna@cwi.nl.

0377-0427/98/$ – see front matter
PII: S0377-0427(98)00125-3

steps where $\|h\partial \boldsymbol{f}/\partial \boldsymbol{y}\|$ is of modest size. For example, one may use an explicit method or one may apply an iteration method which is less costly than MN iteration. In this paper, we focus on using less costly iteration processes. One such iteration procedure is fixed point (FP) iteration. Unfortunately, the convergence of FP iteration is not satisfactory unless $\|h\partial \boldsymbol{f}/\partial \boldsymbol{y}\|$ is quite small. This motivated us to look for an iteration process that converges much faster than FP iteration in nonstiff situations, and is cheaper than MN iteration. The process we have in mind uses MN iteration as the outer iteration process and a linear solver for solving the linear Newton systems as an inner iteration process. This linear solver is based on an approximate factorization of the Newton system matrix by splitting it into two (or more) "convenient" matrices. The inner–outer iteration process will be referred to as approximate factorization (AF) iteration. The technique of approximate factorization is well known in the design of suitable discretizations of partial differential equations (PDEs) [4], but this can also be used for the design of suitable iteration processes. In the PDE case, the particular splitting of the Newton matrix is crucial for the rate of convergence in AF iteration. In [1, 5] we analysed and applied AF iteration to time-dependent PDEs and used splittings that correspond to the spatial dimensions of the PDE. In this paper, we do not restrict (1.1) to PDEs and use a generally applicable splitting which splits the Newton matrix in its lower and upper triangular part. Then, in one-inner-iteration mode, each AF iteration requires one right-hand side evaluation $\boldsymbol{f}$ and one d-dimensional forward/backward substitution. Hence, when compared with an MN iteration process using a direct linear solver for the linear Newton systems, we see that the AF iterations are equally expensive as the MN iterations, but AF iteration does not require the $O(d^3)$ factorization costs associated with MN iteration. Surprisingly, this extremely simple lower–upper triangular splitting is quite effective in actual applications.

The purpose of this paper is to combine FP, AF and MN iterations into one iteration process for use in IVPs where the degree of stiffness is changing in the integration interval.

2. Approximate factorization iteration

We consider implicit IVP solvers in which the implicit relations to be solved are of the form

$$\boldsymbol{R}(\boldsymbol{y}) = 0, \quad \boldsymbol{R}(\boldsymbol{y}) := \boldsymbol{y} - \delta h \boldsymbol{f}(\boldsymbol{y}) - \boldsymbol{v}, \tag{2.1}$$

where h is the stepsize of the numerical method, δ is a positive parameter, $\boldsymbol{y}$ represents a numerical approximation to the solution of (1.1), and $\boldsymbol{v}$ is a given vector $\in \mathbb{R}^d$. In nonstiff parts of the integration interval, a possible way of solving (2.1) is FP iteration

$$\boldsymbol{y}^{(j)} = \boldsymbol{y}^{(j-1)} - \boldsymbol{R}(\boldsymbol{y}^{(j-1)}), \quad j = 1, 2, \ldots, m, \tag{2.2}$$

where $\boldsymbol{y}^{(0)}$ is to be provided by a predictor formula. However, this process only converges rapidly if $\delta h\|\partial \boldsymbol{f}/\partial \boldsymbol{y}\|$ is sufficiently small. We want to accelerate the rate of convergence. Our starting point is the MN iteration process

$$(I - \delta hJ)(\boldsymbol{y}^{(j)} - \boldsymbol{y}^{(j-1)}) = -\boldsymbol{R}(\boldsymbol{y}^{(j-1)}), \quad j = 1, 2, \ldots, m, \tag{2.3}$$

where J is an approximation to the Jacobian matrix $\partial \boldsymbol{f}/\partial \boldsymbol{y}$. Each iteration in (2.3) requires the solution of a d-dimensional linear system for the Newton correction $\boldsymbol{y}^{(j)} - \boldsymbol{y}^{(j-1)}$. We shall employ

special iterative linear solvers which converge much faster than FP iteration. Let J be split into a lower triangular matrix L and an upper triangular matrix U; and consider the iteration method

$$\begin{aligned} &\Pi(\boldsymbol{y}^{(j,v)} - \boldsymbol{y}^{(j,v-1)}) = (I - \delta hJ)(\boldsymbol{y}^{(j-1)} - \boldsymbol{y}^{(j,v-1)}) - \boldsymbol{R}(\boldsymbol{y}^{(j-1)}), \\ &\boldsymbol{y}^{(j,0)} = \boldsymbol{y}^{(j-1)}, \quad \boldsymbol{y}^{(j)} = \boldsymbol{y}^{(j,r)}, \quad \Pi := (I - \delta hL)(I - \delta hU), \quad J = L + U, \end{aligned} \tag{2.4}$$

where $v = 1, 2, \ldots, r$ and $j = 1, \ldots, m$ represent the inner and outer iteration indices, respectively. The matrix Π is called an approximate factorization of the MN iteration matrix $I - \delta hJ$ and process (2.4) will be referred to as AF iteration. Each inner iteration in (2.4) requires the solution of two linear systems with triangular system matrices (that is, a forward/backward substitution), and, except for the very first inner iteration, a matrix–vector multiplication.

2.1. Rate of convergence

In order to get insight into the convergence of the process (2.4), we first look at the convergence of the MN method (2.3). From (2.1) and (2.3) it follows that

$$\boldsymbol{y}^{(j)} - \boldsymbol{y} = \delta h(I - \delta hJ)^{-1}\boldsymbol{g}(\boldsymbol{y}^{(j-1)} - \boldsymbol{y}), \quad \boldsymbol{g}(\epsilon) := \boldsymbol{f}(\boldsymbol{y} + \epsilon) - \boldsymbol{f}(\boldsymbol{y}) - J\epsilon. \tag{2.5}$$

If the function $\boldsymbol{g}$ possesses a Lipschitz constant $C(h)$ in the neighbourhood of the origin (with respect to the norm $\|\cdot\|$), then we have the estimate

$$\|\boldsymbol{y}^{(j)} - \boldsymbol{y}\| \leqslant \delta hC(h)\|(I - \delta hJ)^{-1}\|\|\boldsymbol{y}^{(j-1)} - \boldsymbol{y}\|. \tag{2.6}$$

If we assume that the logarithmic norm of J with respect to the Euclidean norm $\|\cdot\|_2$ is nonpositive, i.e. $\mu_2[J] \leqslant 0$, then the value of $\|(I - \delta hJ)^{-1}\|_2$ can be estimated by means of the Von Neumann theorem (cf. e.g. [3, p. 179]). This yields $\|(I - \delta hJ)^{-1}\|_2 \leqslant \max\{|1 - z|^{-1}: \mathrm{Re}(z) \leqslant 0\} = 1$, so that we have the estimate

$$\|\boldsymbol{y}^{(j)} - \boldsymbol{y}\|_2 \leqslant \delta hC(h)\|\boldsymbol{y}^{(j-1)} - \boldsymbol{y}\|_2. \tag{2.7}$$

The value of $C(h)$ is determined by $\|\tilde{J} - J\|_2$, where the entry $\tilde{J}_{ij}$ of the matrix $\tilde{J}$ equals $\partial f_i/\partial y_j$ at some point on the line segment $\theta\boldsymbol{y}^{(j-1)} + (1 - \theta)\boldsymbol{y}$, $0 \leqslant \theta \leqslant 1$. Hence, if J is a sufficiently close approximation to the Jacobian $\partial\boldsymbol{f}/\partial\boldsymbol{y}$ evaluated at $\boldsymbol{y}$, then $C(h)$ is quite small.

Next, we consider the rate of convergence of the AF process (2.4). The inner iteration error $\boldsymbol{y}^{(j,v)} - \boldsymbol{y}^{(j)}$ satisfies the recursion

$$\begin{aligned} &\boldsymbol{y}^{(j,v)} - \boldsymbol{y}^{(j)} = M(\boldsymbol{y}^{(j,v-1)} - \boldsymbol{y}^{(j)}), \\ &M := I - \Pi^{-1}(I - \delta hJ) = \delta^2h^2(I - \delta hU)^{-1}(I - \delta hL)^{-1}LU. \end{aligned} \tag{2.8}$$

From this recursion and again using the Von Neumann theorem, we immediately have the convergence result:

Theorem 2.1. *Let $\mu_2[.]$ denote the logarithmic norm with respect to the Euclidean norm $\|\cdot\|_2$ and let $\delta > 0$, $\mu_2[L] \leqslant 0$, $\mu_2[U] \leqslant 0$. Then, a sufficient condition for convergence of the inner iteration*

process (2.4) *is*

$$h<\frac{1}{\delta\sqrt{\|LU\|_2}}. \tag{2.9}$$

For the overall convergence of the inner–outer iteration processes (2.4) we also need the recursion (2.5). From (2.5) and (2.8) we obtain

$$\boldsymbol{y}^{(j,v)}-\boldsymbol{y}=M(\boldsymbol{y}^{(j,v-1)}-\boldsymbol{y}^{(j)})+\delta h(I-\delta hJ)^{-1}\boldsymbol{g}(\boldsymbol{y}^{(j-1)}-\boldsymbol{y}). \tag{2.10}$$

After r inner iterations, recursion (2.10) yields

$$\boldsymbol{y}^{(j,r)}-\boldsymbol{y}=M^r(\boldsymbol{y}^{(j-1,r)}-\boldsymbol{y})+\delta h(I-M^r)(I-\delta hJ)^{-1}\boldsymbol{g}(\boldsymbol{y}^{(j-1,r)}-\boldsymbol{y}), \tag{2.11}$$

where we have set $\boldsymbol{y}^{(j,0)}=\boldsymbol{y}^{(j-1,r)}$. As in (2.7), we assume that $\mu_2[J]\leqslant 0$ yielding the estimate

$$\|\boldsymbol{y}^{(j,r)}-\boldsymbol{y}\|_2\leqslant K(r,h)\|\boldsymbol{y}^{(j-1,r)}-\boldsymbol{y}\|_2,\quad K(r,h):=\|M^r\|_2+\delta hC(h)\|I-M^r\|_2. \tag{2.12}$$

Since $M=(\delta h)^2LU(1+\mathrm{O}(h))$, we see that the amplification factor is approximately given by

$$K_{\mathrm{AF}}(r,h)\approx(\delta h)^{2r}\|(LU)^r\|_2+\delta hC(h). \tag{2.13}$$

Here, $\delta^{2r}h^{2r}\|(LU)^r\|_2$ represents the (accumulated) inner amplification factor and $\delta hC(h)$ the MN amplification factor. Expression (2.13) indicates that there is no point in choosing r too large, because $K_{\mathrm{AF}}(r,h)$ is always bounded below by the Newton amplification factor.

2.2. Comparison with FP iteration

Let us compare the AF amplification factor $K(r,h)$ with the amplification factor associated with the FP iteration process. By observing that FP iteration (2.2) is obtained from MN iteration (2.3) by setting $J=0$, we see that (2.7) implies the estimate

$$\|\boldsymbol{y}^{(j)}-\boldsymbol{y}\|_2\leqslant K_{\mathrm{FP}}(h)\|\boldsymbol{y}^{(j-1)}-\boldsymbol{y}\|_2,\quad K_{\mathrm{FP}}(h):=\delta hC_{\mathrm{FP}}, \tag{2.14}$$

where C_{FP} is a Lipschitz constant for $\boldsymbol{f}$ in the neighbourhood of the numerical solution. This constant can be approximated by $\|J\|_2=\|L+U\|_2$, provided that J is a sufficiently close approximation to the Jacobian of $\boldsymbol{f}$ at the point $\boldsymbol{y}$. Then, it follows from (2.13) and (2.14) that AF iteration converges faster than FP iteration by a factor

$$\alpha(r,h):=\frac{K_{\mathrm{FP}}(h)}{K_{\mathrm{AF}}(r,h)}\approx\frac{\|L+U\|_2}{(\delta h)^{2r-1}\|(LU)^r\|_2+C(h)}. \tag{2.15}$$

Assuming that $\alpha(r,h)$ is at least as large as $\alpha(1,h)$, we find that

$$\alpha(r,h)\geqslant\alpha(1,h)\geqslant\frac{\|L+U\|_2}{2\max\{\delta h\|LU\|_2,C(h)\}}. \tag{2.16}$$

In cases where the AF contribution in the amplification factor $K_{\mathrm{AF}}(1,h)$ dominates the Newton contribution, that is, if $\delta h\|LU\|_2\geqslant C(h)$, we may expect that even in one-inner-iteration mode, AF

iteration is faster than FP iteration for stepsizes less than $\|L+U\|_2(2\delta\|LU\|_2)^{-1}$. Note that the one-inner-iteration mode of AF iteration reduces to the simple scheme

$$\Pi(y^{(j)}-y^{(j-1)})=-R(y^{(j-1)}),\quad \Pi:=(I-\delta hL)(I-\delta hU),\quad J=L+U,\ j=1,\ldots,m. \tag{2.4'}$$

Finally, we consider the interval of "fast" convergence of FP and AF iteration defined by the interval of h-values where the amplification factor is less than or equal to a small number q (say $q\leqslant 0.1$). From (2.14) and (2.12) it follows that these intervals are, respectively, determined by $[0,H(q)]$ with

$$H_{\mathrm{FP}}(q)=\frac{q}{\delta\|L+U\|_2},\quad H_{\mathrm{AF}}(q)=\left(\frac{q}{\delta^2\|LU\|_2}\right)^{1/2}, \tag{2.17}$$

where we assumed that $C_{\mathrm{FP}}\approx\|L+U\|_2$ and where we ignored the contribution of the MN iteration to the amplification factor $K_{\mathrm{AF}}(1,h)$ of AF iteration. Thus, if $\sqrt{q\|LU\|_2}<\|L+U\|_2$, then the corresponding interval of AF convergent h-values is larger than the corresponding interval of FP convergent h-values. For example, in problem (3.3a) of Kaps [6], we have $\sqrt{\|LU\|_2}\approx\frac{1}{4}\|L+U\|_2$, so that the convergence interval of AF iteration is a factor $4/\sqrt{q}$ larger than that of FP iteration.

3. Numerical illustration

Consider the 4th-order Runge–Kutta method of Butcher (cf. [2, p. 205]) based on Lobatto quadrature:

$$y_{n+1/2}=y_n+\tfrac{1}{4}h(f(t_n,y_n)+f(t_{n+1/2},y_{n+1/2})), \tag{3.1a}$$

$$y^*_{n+1}=y_n+hf(t_{n+1/2},y_{n+1/2}), \tag{3.1b}$$

$$y_{n+1}=y_n+\tfrac{1}{6}h(f(t_n,y_n)+4f(t_{n+1/2},y_{n+1/2})+f(t_{n+1},y^*_{n+1})). \tag{3.1c}$$

This method has one implicit stage $y_{n+1/2}$ which has to satisfy an equation of the form (2.1). Let us solve this equation by means of FP iteration (2.2) and by the one-inner-iteration mode of AF iteration (2.4'). In both cases, we used the last step value predictor $y^{(0)}_{n+1/2}=y_n$ and we updated the Jacobian in each step. Furthermore, we implemented the Butcher–Lobatto method (3.1) such that y^*_{n+1} and y_{n+1} are, respectively, approximated by (cf. [8])

$$y^*_{n+1}\approx u^*_{n+1}:=y_n+4(y^{(m)}_{n+1/2}-y_n)-hf(t_n,y_n), \tag{3.2b}$$

$$y_{n+1}\approx u_{n+1}:=\tfrac{1}{3}y_n+\tfrac{2}{3}u^*_{n+1}+\tfrac{1}{6}h(f(t_n,y_n)+f(t_{n+1},u^*_{n+1})). \tag{3.2c}$$

We observe that for autonomous problems, the last step value predictor implies that the first iteration does not require a new right-hand side evaluation, so that each integration step requires $m+1$ right-hand side evaluations in the case of FP iteration and $m+1$ right-hand side evaluations and m forward/backward substitutions in the case of AF iteration. In the case of nonautonomous problems, we have the same costs if in each step, the first residue is evaluated at the point (t_n,y_n).

The iteration error $y_{n+1/2}^{(m)} - y_{n+1/2} = O(h^{\theta m+1})$, where $\theta = 1$ and $\theta = 2$ in the FP and AF iteration case, respectively. Hence,

$$u_{n+1}^* - y_{n+1}^* = 4(y_{n+1/2}^{(m)} - y_n) - hf(t_n, y_n) - hf(t_{n+1/2}, y_{n+1/2})$$
$$= 4(y_{n+1/2}^{(m)} - y_{n+1/2}) = O(h^{\theta m+1}).$$

It is now easily verified that $u_{n+1} = y_{n+1} + O(h^{\theta m+1})$, so that the global error becomes $O(h^{\theta m} + h^4)$. Thus, two AF iterations already yield a 4th-order scheme.

3.1. Comparison of FP, AF and MN iteration

We used a test problem of Kaps [6]

$$\frac{dy_1}{dt} = -12y_1 + 10y_2^2, \quad \frac{dy_2}{dt} = y_1 - y_2(1 + y_2), \quad y_1(0) = y_2(0) = 1, \quad 0 \leqslant t \leqslant 5, \tag{3.3a}$$

and three test problems from the books of Hairer–Nørsett–Wanner:

JACB[2, p. 236], $0 \leqslant t \leqslant 20$, (3.3b)

TWOB[2, p. 236], $0 \leqslant t \leqslant 20$, (3.3c)

HIRES[3, p. 157], $0 \leqslant t \leqslant 1$. (3.3d)

In all examples, we used for the AF mode a lower–upper triangular splitting $J = (J_{ij}) = L + U$, where the diagonal entries of L and U equal $\frac{1}{2}J_{ii}$.

In order to clearly see the algorithmic effects, we used a fixed stepsize strategy. Table 1 lists in a logarithmic scale the relative end point errors, that is, the numbers of correct significant digits (csd) at the end point (negative values are indicated by $-$). These figures show that in most cases AF iteration has more or less converged within two iterations, whereas FP iteration requires at least four iterations. Furthermore, in these examples, AF iteration performs as robustly as MN iteration and converges only slightly less fast.

3.2. Combination of FP, AF and MN iteration

Suppose that we use FP, AF and MN iteration, respectively, in the nonstiff, mildly stiff and stiff parts of the integration interval. Then, we avoid unnecessary evaluations of the Jacobian, forward/backward substitutions and LU decompositions. Again, we used a fixed stepsize strategy, applied AF iteration in one-inner-iteration mode, and we chose the following simple switching strategy:

$$\begin{array}{lll} \text{If} & \delta h\|J\|_\infty < a & \text{then FP iteration,} \\ \text{If} & a \leqslant \delta h\|J\|_\infty < b & \text{then AF iteration,} \\ \text{If} & b \leqslant \delta h\|J\|_\infty & \text{then MN iteration.} \end{array} \tag{3.4a}$$

Here a and b are constants depending on the integration method used. Furthermore, in the very first step we used MN iteration, and if in FP mode more than 4 iterations were needed, then we switched

Table 1
Relative precision at the end point produced by the Butcher–Lobatto method {(3.1a), (3.2b), (3.2c)}

Prblm	m	$h=\frac{1}{4}$			$h=\frac{1}{8}$			$h=\frac{1}{16}$			$h=\frac{1}{32}$		
		FP	AF	MN	FP	AF	MN	FP	AF	MN	FP	AF	MN
(3.3a)	1	—	1.3	2.3	0.2	2.1	2.7	0.6	2.7	3.3	0.9	3.3	3.9
	2	1.1	2.5	2.7	1.7	3.6	4.0	2.3	4.8	5.2	2.9	6.0	6.5
	3	1.9	2.8		2.7	4.0		3.6	5.3		4.5	6.5	
	4	2.2			3.2			4.5			5.7		
	...	...	...	...	...	...	...	...	...	...	...	...	...
	∞		2.7			4.0			5.2			6.5	
(3.3b)	1	—	1.8	1.3	0.3	2.8	1.9	0.6	3.0	2.5	0.9	3.5	3.1
	2	0.8	3.3	3.5	1.5	4.6	4.7	2.1	5.9	5.9	2.7	7.1	7.1
	3	2.0	3.5		2.9			3.8			4.7		
	4	3.8			4.9			6.0			7.2		
	...	...	...	...	...	...	...	...	...	...	...	...	...
	∞		3.5			4.7			6.0			7.2	
(3.3c)	1	—	—	—	—	—	—	—	—	—	—	—	—
	2	—	—	—	—	0.9	0.5	—	2.1	1.9	—	3.4	3.7
	3	—	—	—	—	0.6	0.6	0.7	2.0	2.0	1.7	3.3	3.3
	4	—	—	—	0.5			1.9			3.2	3.3	3.3
	...	...	...	...	...	...	...	...	...	...	...	...	...
	∞		—			0.6			2.0			3.3	
(3.3d)	1	—	—	0.1	1.0	1.4	2.7	1.4	1.9	3.3	1.7	2.4	3.5
	2	—	—	—	—	1.8	1.8	2.9	4.1	3.9	3.3	4.6	5.3
	3	—	—	—	0.6			3.2	3.9		3.7	5.2	
	4	—	—		—			3.2			4.1		
	...	...	...	...	...	...	...	...	...	...	...	...	...
	∞		—			1.9			3.9			5.2	

to AF mode in the next step. In all iteration modes, the iteration process was stopped if

$$\|\boldsymbol{R}(\boldsymbol{y}^{(j)})[\boldsymbol{y}^{(j)}+10^{-6}\boldsymbol{e}]^{-1}\|_\infty < h^{p+1}, \tag{3.4b}$$

where p is the order of the integration method, $\boldsymbol{e}$ is a vector with unit entries, and where the vector operations are meant to be componentwise.

We applied the Butcher–Lobatto method {(3.1a), (3.2b), (3.2c)} and we set $a=\frac{1}{2}$, $b=3$ and $p=4$ in the strategy formulas (3.4). In Tables 2(a)–2(d), we listed for the four problems (3.3) the correct number of significant digits obtained at the end point (csd), the averaged number of right-hand sides per step (rhs), the averaged number of Jacobians per step (jac), the averaged number of forward/backward substitutions per step (fbs), and the averaged number of LU decompositions per step (lud), respectively. A comparison with Table 1 reveals that in most cases the implicit stage equation (3.1a) in the Butcher–Lobatto equation is more or less solved and that the averaged number of iterations per step given by $m=\text{rhs}-1$ is at most about 6 and usually about 4 or 5. Furthermore, lud is always small, so that the iteration process is mostly in FP or AF mode, as should be expected when integrating nonstiff or mildly stiff problems.

Table 2(a)
Results for problem (3.3a)

h	csd	rhs	jac	fbs	lud
1/4	2.7	4.95	1.00	3.95	0.05
1/8	4.0	6.27	0.67	2.67	0.02
1/16	5.2	6.45	0.51	2.04	0.01
1/32	6.5	6.04	0.50	1.99	0.01

Table 2(b)
Results for problem (3.3b)

h	csd	rhs	jac	fbs	lud
1/4	2.8	4.25	0.02	0.05	0.01
1/8	4.3	4.73	0.01	0.01	0.01
1/16	6.2	4.93	0.01	0.02	0.00
1/32	7.3	4.99	0.01	0.02	0.00

Table 2(c)
Results for problem (3.3c)

h	csd	rhs	jac	fbs	lud
1/4	—	—	—	—	—
1/8	0.5	5.01	0.22	0.74	0.01
1/16	2.0	5.09	0.21	0.66	0.00
1/32	3.3	5.09	0.22	0.69	0.00

Table 2(d)
Results for problem (3.3d)

h	csd	rhs	jac	fbs	lud
1/4	—	—	—	—	—
1/8	1.9	6.37	0.75	3.37	0.12
1/16	3.9	6.69	0.62	2.75	0.06
1/32	5.3	6.97	0.50	2.06	0.03

3.3. *Using the various modes in an automatic code*

Next, we show the effect of the various modes when implemented in an automatic ODE solver. For that purpose we selected the code PSODE as our starting point. The main features of this code are described below and for further details we refer to [9]. The code itself is available from the second author of this paper.

3.3.1. *The code PSODE*

PSODE is based on the L-stable, four-stage Radau IIA method of order seven and is aimed to solve initial value problems for stiff ODEs. The implicit relation to be solved in each step is of the form

$$\boldsymbol{Y}_n = \boldsymbol{e} \otimes \boldsymbol{y}_n + h_n(A \otimes I)\boldsymbol{F}(t_n\boldsymbol{e} + h_n\boldsymbol{c}, \boldsymbol{Y}_n), \tag{3.5}$$

where $\boldsymbol{Y}_n$ is the so-called stage vector containing the four approximations $\boldsymbol{Y}_{n,i}$, $i=1,\ldots,4$ to the solution vector $\boldsymbol{y}(t)$ in the points $t_n + c_i h_n$, defined by the abscissae vector $\boldsymbol{c}=(c_i)$. Furthermore, $\boldsymbol{F}(t_n\boldsymbol{e} + h_n\boldsymbol{c}, \boldsymbol{Y}_n) = (\boldsymbol{f}(t_n + c_1h_n, \boldsymbol{Y}_{n,1})^{\mathrm{T}}, \ldots, \boldsymbol{f}(t_n + c_4h_n, \boldsymbol{Y}_{n,4})^{\mathrm{T}})^{\mathrm{T}}$ contains the corresponding derivative vectors. A is the parameter matrix of the Radau method and h_n is the current stepsize $t_{n+1} - t_n$. The symbol $\otimes$ denotes the Kronecker product and the vector $\boldsymbol{e}=(1,1,1,1)^{\mathrm{T}}$. Since the Radau IIA method is stiffly accurate ($c_4 = 1$), the new approximation $\boldsymbol{y}_{n+1}$ equals $\boldsymbol{Y}_{n,4}$.

To solve (3.5) for the stage vector $\boldsymbol{Y}_n$ leads to a huge linear algebra problem, since this system is of dimension $4d$. To reduce the computational work involved to an acceptable level, PSODE uses the following iteration process [9]

$$\boldsymbol{Y}_n^{(j)} - h_n(D \otimes I)\boldsymbol{F}(t_n\boldsymbol{e} + h_n\boldsymbol{c}, \boldsymbol{Y}_n^{(j)}) = \boldsymbol{e} \otimes \boldsymbol{y}_n + h_n((A - D) \otimes I)\boldsymbol{F}(t_n\boldsymbol{e} + h_n\boldsymbol{c}, \boldsymbol{Y}_n^{(j-1)}), \tag{3.6}$$

where the iterates $\boldsymbol{Y}_n^{(j)}$ (hopefully) converge to $\boldsymbol{Y}_n$ for $j=1,2,\ldots$. Here, the matrix D has been chosen to be of diagonal form which has the advantage that each of the four components $\boldsymbol{Y}_{n,i}^{(j)}$ of $\boldsymbol{Y}_n^{(j)}$ can be solved from a d-dimensional system, which is of form (2.1). Moreover, if PSODE

runs on a parallel machine with (at least) four processors, then the components vectors $\boldsymbol{Y}_{n,i}^{(j)}$ can be solved concurrently. This was exactly the motivation in [9] to construct methods of this type. A convergence analysis showed that (for the Radau IIA method) the choice $D \approx$ diag(0.3205, 0.0891, 0.1817, 0.2334) leads to an optimal damping of the stiff components in the iteration error.

In PSODE, the iterate $\boldsymbol{Y}_n^{(j)}$ is solved from the nonlinear relation (3.6) by applying just one modified Newton iteration, using $\boldsymbol{Y}_n^{(j-1)}$ as its initial guess. It is straightforwardly verified that the resulting expression for each of the components of $\boldsymbol{Y}_n^{(j)}$ is of the form (2.3) with δ replaced by the corresponding element of the matrix D.

The code PSODE is equipped with a number of control mechanisms to perform an automatic integration of stiff ODEs. We shall briefly describe the most important ones.

First, the local truncation error is controlled by calculating a reference solution. This reference solution $\boldsymbol{y}_{\text{ref}}$ is a linear combination of $\boldsymbol{y}_n$, $h_n\boldsymbol{f}(t_n,\boldsymbol{y}_n)$, and the four final (say, the mth) iterates $\boldsymbol{Y}_{n,i}^{(m)}$, where the weights are such that we obtain fourth-order accuracy (see [9] for more details). Following an idea of Shampine, an estimate for the local truncation error is now defined by $(I - d_4h_n\partial\boldsymbol{f}/\partial\boldsymbol{y})^{-1}(\boldsymbol{y}_{\text{ref}} - \boldsymbol{y}_{n+1})$, where d_4 is the fourth entry in the diagonal matrix D. The premultiplication by the matrix $(I - d_4h_n\partial\boldsymbol{f}/\partial\boldsymbol{y})^{-1}$ is meant to obtain a bounded estimate for the linear test problem $y' = \lambda y$ in case $h_n\lambda \to \infty$ (see also [3, p. 134]).

Then the usual formula $h_{n+1} = \text{fac}_{\text{safe}}(\text{TOL}/\|\text{local error}\|)^{1/5}h_n$ is applied to predict a new stepsize. The safety factor fac_{safe} has been set to 0.9 in PSODE and the stepsize ratio h_{n+1}/h_n is restricted to the interval [0.1, 4]. Finally, some additional strategies have been implemented to further fine-tune the stepsize selection.

With respect to the control of the convergence behaviour of the iteration process (3.6) a sophisticated strategy is needed. The convergence control used in PSODE is quite similar to the one used in the code RADAU5 (cf. [3, p. 130]). Skipping the details we mention that the iteration is interrupted in case of a too slow convergence (which includes divergence). With a "fresh" Jacobian and a halved stepsize the iteration is retried. The process is considered to be converged as soon as the (scaled, Eucledean) norm of the update of the last iterate is less than $0.1 * \text{TOL}$. We conclude this brief description of PSODE by mentioning that the prediction $\boldsymbol{Y}_n^{(0)}$ to start the iteration is obtained by extrapolating the collocation polynomial calculated in the preceding step.

To get an impression of the usefulness of the AF and FP modes in the context of a stiff ODE solver, we extended PSODE with a strategy to automatically switch between the various modes. We emphasize that in the "original" PSODE only the MN mode has been used. As a switching criterion we use the simple test (3.4a) as described in the previous section (with $\delta = \max d_i = 0.3205, a = \frac{1}{2}$ and $b = 3$). This test is activated in all cases where either the stepsize or the Jacobian has been changed.

The FP–AF–MN version of PSODE was applied to the nonstiff or mildly stiff test problems (3.3). We nicely observed that the MN mode has never been selected (in fact, almost all steps were performed in FP mode).

3.3.2. *The HIRES problem*

Next, we perform a more severe test with the code by applying it to the HIRES problem (3.3d) including the non-transient interval, viz. we choose $0 \leqslant t \leqslant 321.8122$, as in [3]. Results for the original PSODE as well as for the extended version are given in Table 3. Analogously to the tables

Table 3
Results for the HIRES problem on [0, 321.8122]

	PSODE in MN mode						PSODE in FP–AF–MN mode					
TOL	N	csd	rhs	jac	fbs	lud	N	csd	rhs	jac	fbs	lud
10^{-2}	36	3.2	6.94	0.42	5.94	1.00	36	2.7	6.92	0.42	5.11	0.61
10^{-3}	51	4.3	6.86	0.41	5.86	1.00	56	4.0	7.00	0.41	5.16	0.66
10^{-4}	63	4.8	6.83	0.35	5.83	0.98	70	4.7	7.04	0.36	4.86	0.61

in Section 3.2, the listed number of operations (like rhs, jac, fbs, and lud) have been scaled by the number of (successful) steps, which is denoted by N in Table 3. It should be remarked that the values of rhs, fbs and lud listed in Table 3 take into account the fact that the four implicit subsystems in (3.6) can be solved in parallel, that is, each four right-hand sides, each four forward/backward substitutions and each four LU decompositions can be counted for one.

As we see from this table, approximately 40% of the steps could be iterated with the AF or FP mode, saving a lot of the expensive LU-factorizations. Moreover, the numbers of right-hand side and Jacobian evaluations remain more or less constant and the number of forward/backward substitutions is reduced.

3.3.3. A combustion problem

Our second example is a problem from combustion theory and is a two-dimensional version of a similar test problem used in [7]. It is described by the PDEs

$$\frac{\partial c}{\partial t} = \frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2} - Dce^{-\delta/T}, \quad L\frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \alpha Dce^{-\delta/T}, \quad t>0, \tag{3.7}$$

defined on the unit square $0<x,\ y<1$. The initial conditions are given by $c=T=1$, on the whole unit square. At $x=y=0$, we impose homogeneous Neumann conditions, and at $x=y=1$ Dirichlet conditions are prescribed ($c=T=1$).

The variables c and T denote the concentration and temperature of a chemical during a reaction. At the origin, a so-called "hot-spot" is developed for the temperature. Initially, the temperature slowly increases but suddenly, at the ignition point, it explodes to about $1+\alpha$, and initiates a reaction front which propagates towards the boundaries $x=y=1$, where a boundary layer is formed. Finally, the temperature distribution reaches a steady state. The problem parameters are given by $L=0.9$, $\alpha=1$, $\delta=20$, and $D=Re^{\delta}/(\alpha\delta)$, with $R=5$. The integration interval is $0<t<0.3$, which is sufficiently long to reach the steady state.

The equations in (3.7) are discretized on a uniform spatial grid with mesh size Δ using second-order, symmetric differences. Defining $\Delta=1/(N+0.5)$, with N as the number of grid points in both directions, and introducing artificial points outside the region at a distance $\Delta/2$, the homogeneous Neumann boundary conditions are easily discretized by symmetric differences. In this test example we set $N=20$, resulting in a system of $2*20^2=800$ ODEs. Again we applied both versions of PSODE to problem (3.7) and the results are given in Table 4. To measure the errors, we calculated a reference solution of the ODE system, using a stringent tolerance. Hence, the errors displayed in the table are only due to the time integration and do not interfere with spatial discretization errors.

Table 4
Results for the combustion problem (3.8) on [0, 0.3]

	PSODE in MN mode						PSODE in FP–AF–MN mode					
TOL	N	csd	rhs	jac	fbs	lud	N	csd	rhs	jac	fbs	lud
10^{-2}	187	2.4	7.66	0.51	6.66	1.14	241	2.4	6.72	0.59	5.71	0.53
10^{-3}	259	3.5	8.08	0.49	7.08	1.19	300	3.9	7.55	0.58	5.80	0.41

During ignition, the problem becomes locally unstable, forcing any integration method to take small timesteps in order to accurately follow the solution. Also the travelling reaction front limits the time step for accuracy reasons. In this part of the integration interval, the extended version of PSODE frequently uses steps in AF or FP mode. Only for small t-values and near the steady state, large steps are possible and the code switches to MN mode. The nature of this flame propagation problem is nicely illustrated by the numerical experiments. From Table 4 we see that the number of steps and the corresponding accuracies of both versions are more or less comparable. However, the number of right-hand side evaluations and forward/backward substitutions per step are smaller for the mixed-mode version, but especially the number of LU-decompositions has been drastically reduced, which saves a lot of CPU time for a problem of this size. Finally, we remark that the lud-numbers for the original PSODE are >1, which indicates that the MN mode encountered convergence problems. The remedy to halve the stepsize, forces PSODE to calculate a new LU-decomposition in the same step.

References

[1] C. Eichler, P.J. van der Houwen, B.P. Sommeijer, Analysis of approximate factorization in iteration methods, Appl. Numer. Math. 28 (1998) 245–258.
[2] E. Hairer, S.P. Nørsett, G. Wanner, Solving ordinary differential equations, Nonstiff Problems, vol. I, Springer, Berlin, 1987.
[3] E. Hairer, G. Wanner, Solving ordinary differential equations, Stiff and Differential-Algebraic Problems, vol. II, Springer, Berlin, 1991.
[4] C. Hirsch, Numerical computation of internal and external flows, Fundamentals of Numerical Discretization, vol. 1, Wiley, New York, 1988.
[5] P.J. van der Houwen, B.P. Sommeijer, J. Kok, The iterative solution of fully implicit discretizations of three-dimensional transport models, Appl. Numer. Math. 25 (1997) 243–256.
[6] P. Kaps, Rosenbrock-type methods, in: G. Dahlquist, R. Jeltsch (Eds.), Numerical Methods for Stiff initial value Problems Bericht Nr.9, Inst. für Geometrie und Praktische Mathematik der RWTH Aachen, 1981.
[7] P.K. Moore, R.H. Dillon, A comparison of preconditioners in the solution of parabolic systems in three space dimensions using DASPK and a high order finite element method, Appl. Numer. Math. 20 (1996) 117–128.
[8] L.F. Shampine, Implementation of implicit formulas for the solution of ODEs, SIAM J. Sci. Statist. Comput. 1 (1980) 103–118.
[9] B.P. Sommeijer, Parallel-iterated Runge–Kutta methods for stiff ordinary differential equations, J. Comput. Appl. Math. 45 (1993) 151–168.