IMA Journal of Numerical Analysis (2001) **21**, 367–385

Parallel iteration of the extended backward differentiation formulas

J. E. FRANK[†]
Delft University of Technology, The Netherlands
AND
P. J. VAN DER HOUWEN[‡]
CWI, Amsterdam, The Netherlands

[Received 26 March 1999 and in revised form 29 February 2000]

The extended backward differentiation formulas (EBDFs) and their modified form (MEBDF) were proposed by Cash in the 1980s for solving initial value problems (IVPs) for stiff systems of ordinary differential equations (ODEs). In a recent performance evaluation of various IVP solvers, including a variable-step-variable-order implementation of the MEBDF method by Cash, it turned out that the MEBDF code often performs more efficiently than codes like RADAU5, DASSL and VODE. This motivated us to look at possible parallel implementations of the MEBDF method. Each MEBDF step essentially consists of successively solving three non-linear systems by means of modified Newton iteration using the same Jacobian matrix. In a direct implementation of the MEBDF method on a parallel computer system, the only scope for (coarse grain) parallelism consists of a number of parallel vector updates. However, all forward–backward substitutions and all right-hand-side evaluations have to be done in sequence. In this paper, our starting point is the original (unmodified) EBDF method. As a consequence, two different Jacobian matrices are involved in the modified Newton method, but on a parallel computer system, the effective Jacobian-evaluation and the LU decomposition costs are not increased. Furthermore, we consider the simultaneous solution, rather than the successive solution, of the three non-linear systems, so that in each iteration the forward–backward substitutions and the right-hand-side evaluations can be done concurrently. A mutual comparison of the performance of the parallel EBDF approach and the MEBDF approach shows that we can expect a speed-up factor of about 2 on three processors.

Keywords: numerical analysis; iteration methods; initial value problems; extended BDFs; parallelism.

1. Introduction

The extended backward differentiation formulas (EBDFs) were proposed by Cash (1980) in 1980 for solving initial value problems (IVPs) for stiff systems of ordinary differential equations (ODEs):

$$\frac{dy}{dt} = f(y),\ y, f \in \mathcal{R}^d,\ t \geqslant t_0. \tag{1}$$

[†]Current address: CWI, Amsterdam.

[‡]The investigations reported in this paper were partly supported by the Dutch Technology Foundation STW.

To conserve notation we will consider the autonomous problem, but the results of this paper can be trivially extended to derivatives with explicit dependence on time, i.e. $f = f(t, y)$. Each EBDF step essentially consists of successively solving three non-linear systems by means of (modified) Newton iteration. Since two different Jacobian matrices are involved, the method needs two different LU decompositions after each Jacobian update, or change of step size. In order to reduce the LU costs, Cash (1983) modified the EBDF methods (MEBDF methods) such that only one LU decomposition is required.

In a recent performance evaluation (Lioen (1998)) of various IVP solvers, including a variable-step-variable-order implementation of the MEBDF method due to Cash, it turned out that the MEBDF code often performs more efficiently than codes like RADAU5 (Hairer & Wanner, 1998), DASSL (Petzold, 1991) and VODE (Brown *et al.*, 1992). This motivated us to look at possible parallel implementations of the MEBDF method.

In a direct implementation of MEBDF on a parallel computer system, the only scope for (coarse grain) parallelism consists of a number of parallel vector updates. However, all forward–backward substitutions and all right-hand-side evaluations have to be done in sequence. In this paper, our starting point is the original (unmodified) EBDF method. As a consequence, two different Jacobian matrices are involved in the modified Newton method, but on a parallel computer system, the effective costs of the Jacobian evaluations and the LU decompositions are not increased. Furthermore, we consider the simultaneous solution, rather than the successive solution, of the three non-linear systems, so that in each iteration the forward–backward substitutions and the right-hand-side evaluations can be done concurrently. A mutual comparison of the performance of the parallel EBDF and MEBDF approaches shows that we can expect a speed-up factor of about 2 on three processors.

2. The EBDF and MEBDF methods of Cash

The EBDF method of Cash (1980) is based on the formula

$$y_{n+1} = a_1 y_n + a_2 y_{n-1} + \cdots + a_k y_{n-k+1} + h b_0 f(y_{n+1}) + h b_1 f(y_{n+2}) \tag{2}$$

for computing an approximation y_{n+1} to the exact solution $y(t_{n+1})$ of (1). Here, y_{n+2} is an approximation to $y(t_{n+2})$ obtained by some predictor formula and the coefficients a_i and b_i are determined by imposing the conditions for order $k+1$ accuracy. Cash used the standard (implicit) BDF as a predictor,

$$\begin{aligned} u_{n+1} &= \bar{a}_1 y_n + \bar{a}_2 y_{n-1} + \cdots + \bar{a}_k y_{n-k+1} + h\bar{b}_0 f(u_{n+1}), \\ u_{n+2} &= \bar{a}_1 u_{n+1} + \bar{a}_2 y_n + \cdots + \bar{a}_k y_{n-k+2} + h\bar{b}_0 f(u_{n+2}), \end{aligned} \tag{3a}$$

to obtain an approximation u_{n+2} for the 'future' value y_{n+2}. Thus, y_{n+1} is computed from the equation

$$y_{n+1} = a_1 y_n + a_2 y_{n-1} + \cdots + a_k y_{n-k+1} + h b_0 f(y_{n+1}) + h b_1 f(u_{n+2}). \tag{3b}$$

The coefficients $\bar{a}_i$ and $\bar{b}_0$ are the BDF coefficients. The internal vectors u_{n+1} and u_{n+2} defined by (3a) have order of accuracy k and the external (or output) vector y_{n+1} defined by (3b) has order of accuracy $k+1$. Hence the stage order s equals k and the actual order

TABLE 1
Coefficients $\{\bar{a}_i, \bar{b}_0\}$ *in the BDF formulas (3a)*

k	$\bar{a}_1\delta$	$\bar{a}_2\delta$	$\bar{a}_3\delta$	$\bar{a}_4\delta$	$\bar{a}_5\delta$	$b_0\delta$	δ
2	4	−1				2	3
3	18	−9	2			6	11
4	48	−36	16	−3		12	25
5	300	−300	200	−75	12	60	137

TABLE 2
Coefficients $\{a_i, b_0, b_1\}$ *in the EBDF and MEBDF formulas (3b) and (3c)*

k	$a_1\delta$	$a_2\delta$	$a_3\delta$	$a_4\delta$	$a_5\delta$	$b_0\delta$	$b_1\delta$	δ
2	28	−5				22	−4	23
3	279	−99	17			150	−18	197
4	4 008	−2 124	728	−111		1644	−144	2501
5	26 550	−18 700	9600	−2925	394	8820	−600	14 919

p equals $k + 1$. Furthermore, (3a) and (3b) possess a considerably larger stability region than the classical BDF method of order $p = k + 1$. This can be explained by observing that the underlying corrector formula (2) is much more stable than the classical BDF (for $k > 1$). For future reference, the coefficients $\{\bar{a}_i, \bar{b}_0\}$ and $\{a_i, b_0, b_1\}$ are given in Tables 1 and 2 for $k = 2, \ldots, 5$. The MEBDF method arises from the EBDF method (3a) and (3b) by replacing (3b) with the formula (see (Cash, 1983; Hairer, 1996))

$$y_{n+1} = a_1 y_n + a_2 y_{n-1} + \cdots + a_k y_{n-k+1} + h\bar{b}_0 f(y_{n+1}) + h(b_0 - \bar{b}_0) f(u_{n+1}) + h b_1 f(u_{n+2}). \tag{3c}$$

The advantage is that the modified Newton iteration of the subsystems (3a) and (3b) can use the same LU decomposition. Furthermore, the order of accuracy is not affected and the stability regions are even slightly larger than for the EBDF methods.

Note that in (3b) and (3c) each of the required derivatives $f(u_{n+1})$ and $f(u_{n+2})$ can be computed either with an additional function evaluation or by solving the corresponding formula (3a) for the (converged) derivative as a linear combination of back values, whichever is cheaper (Cash (1983)).

2.1 *The implicit relations*

The EBDF and MEBDF methods are implicit in u_{n+1}, u_{n+2} and y_{n+1}, and use the back values $y_{n-k+1}, \ldots, y_n$ as input. Let us define the stage vector Y_{n+1} and the input vector V_n

according to

$$Y_{n+1} = \begin{pmatrix} u_{n+1} \\ u_{n+2} \\ y_{n+1} \end{pmatrix}, \quad V_n = \begin{pmatrix} y_{n-k+1} \\ \vdots \\ y_n \end{pmatrix}.$$

Then, using tensor notation, both the EBDF method (3a) and (3b) and the MEBDF method (3a) and (3c) can be represented in the compact form

$$(B \otimes I)Y_{n+1} - h(C \otimes I)F(Y_{n+1}) = (E \otimes I)V_n. \tag{4}$$

Here, $\otimes$ denotes the Kronecker product, h the step size $t_{n+1} - t_n$, and $F(Y_{n+1})$ contains the right- hand sides $f(u_{n+1})$, $f(u_{n+2})$, $f(y_{n+1})$. We denote the identity matrix by I, and its dimension will always be clear from the context. In the EBDF case, B, C and E are defined by

$$B := \begin{bmatrix} 1 & 0 & 0 \\ -\bar{a}_1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, C := \begin{bmatrix} \bar{b}_0 & 0 & 0 \\ 0 & \bar{b}_0 & 0 \\ 0 & b_1 & b_0 \end{bmatrix}, E := \begin{bmatrix} \bar{a}_k & \bar{a}_{k-1} & \cdots & \bar{a}_1 \\ 0 & \bar{a}_k & \cdots & \bar{a}_2 \\ a_k & a_{k-1} & \cdots & a_1 \end{bmatrix}. \tag{5}$$

In the MEBDF case, the last row of the matrix C changes to $(b_0 - \bar{b}_0, b_1, \bar{b}_0)$.

2.2 *Iteration processes*

Instead of solving for the unknown components u_{n+1}, u_{n+2} and y_{n+1} of Y_{n+1} *sequentially*, as was the original approach of Cash, we consider here an approach where these components are solved *simultaneously*, that is, the subsystems in the EBDF or MEBDF method are solved simultaneously. As we shall see below, one option in this approach is approximating the matrix $B^{-1}C$ by a diagonal matrix. The relative error of the diagonal approximation will be smaller if the diagonal elements of $B^{-1}C$ are large compared to the off-diagonal elements. For this reason, we choose the EBDF method, rather than the MEBDF method, as our starting point because the additional non-zero off-diagonal element of the MEBDF matrix is taken out of the diagonal such that the row sum remains the same.

Premultiplying (4) by $B^{-1} \otimes I$, we can rewrite the method in the form

$$R_n(Y_{n+1}) = 0, \tag{6}$$

$$R_n(Y) := Y - h(A \otimes I)F(Y) - (B^{-1}E \otimes I)V_n, \tag{7}$$

$$A := B^{-1}C = \begin{bmatrix} \bar{b}_0 & 0 & 0 \\ \bar{a}_1\bar{b}_0 & \bar{b}_0 & 0 \\ 0 & b_1 & b_0 \end{bmatrix}.$$

Let us iterate the system of implicit relations $R_n(Y_{n+1}) = 0$ by the Newton-type method

$$(I - A^* \otimes hJ_{n+1})(Y^{(j)} - Y^{(j-1)}) = -R_n(Y^{(j-1)}), \quad j = 1, \ldots, m, \tag{8}$$

where A^* is a suitably chosen matrix, $Y^{(0)}$ is an initial approximation to Y_{n+1} and J_{n+1} is an approximation to the Jacobian matrix of the right-hand-side function in (1) at t_{n+1}. Note that an approximation J_{n+1} given by a evaluation of $J(y_k)$ for some $k \leqslant n$ may be sufficient for this purpose (that is, it may be unnecessary to update the Jacobian at every step). The decision to update the Jacobian is a strategy issue which must be addressed in a full implementation. In many cases it is possible to take several time steps before updating the Jacobian. Suppose that the first and third components of $Y^{(0)}$ are defined by the second component of the stage vector Y_n computed in the preceding step, and that the second component of $Y^{(0)}$ is obtained by extrapolation of the most recent approximations available at the points $t_{n+1}, t_n, \ldots, t_{n-k+1}$. Then, $Y^{(0)}$ has order of accuracy $p = k$ and is expected to be an excellent initial approximation to Y_{n+1}. Moreover, the computational costs are negligible. For further discussion on the practical implementation of modified Newton iterations, see, for example, (Shampine, 1994, p. 405 ff).

It is tempting to set $A^* = A$, resulting in the familiar (modified) Newton method, and to try to diagonalize (8) by a Butcher similarity transformation $\tilde{Y}^{(j)} = (Q^{-1} \otimes I)Y^{(j)}$ such that the matrix $Q^{-1}AQ$ is diagonal. Unfortunately, the matrix A is defective, so that this does not work. However, if we approximate A by the matrix

$$A^* = \begin{bmatrix} \bar{b}_0 & 0 & 0 \\ 0 & \bar{b}_0 & 0 \\ c_1 & c_2 & b_0 \end{bmatrix}, \tag{9a}$$

then diagonalization is possible. It can be shown that

$$Q := \begin{bmatrix} q_1 & 0 & 0 \\ q_0 & q_2 & 0 \\ \frac{c_1 q_1 + c_2 q_2}{\bar{b}_0 - b_0} & \frac{c_2 q_2}{\bar{b}_0 - b_0} & q_3 \end{bmatrix} \Rightarrow Q^{-1}A^*Q = D, \quad D := \operatorname{diag}(\bar{b}_0, \bar{b}_0, b_0) \tag{9b}$$

for all non-zero diagonal entries of Q. This family of transformation matrices does not represent all possible transformation matrices Q with the property $Q^{-1}A^*Q = D$, but for our purposes we do not need more generality.

Using (9a) and (9b), we can define the transformed iteration method

$$\begin{aligned} (I - D \otimes hJ_{n+1})(\tilde{Y}^{(j)} - \tilde{Y}^{(j-1)}) &= -(Q^{-1} \otimes I)R_n(Y^{(j-1)}), \quad Y^{(j)} \\ &= (Q \otimes I)\tilde{Y}^{(j)}, \quad j = 1, \ldots, m. \end{aligned} \tag{10}$$

We shall refer to (9a), (9b) and (10) as the *transformed* EBDF method. In particular, we may set $c_1 = c_2 = 0$ in (9a), i.e. $A^* = D$, so that we can use $Q = I$, which avoids transformation costs. We shall call (8), $A^* = D$, simply the *diagonal* EBDF method. All the iteration processes (9a), (9b) and (10) and (8), $A^* = D$ have the advantage of possessing a lot of additional intrinsic parallelism when compared with the MEBDF method as implemented in Cash (1983), where the three equations in (3a) and (3c) are solved sequentially by Newton iteration (to be referred to as *sequential* MEBDF). First, the two LU decompositions can be obtained in parallel and, second, in each iteration, the forward–backward substitutions for the three subsystems and the three components of the residue function $R_n(Y^{(j)})$ can be computed in parallel. Furthermore, in the case of (9a), (9b) and (10), the similarity transformation can largely be computed concurrently, at the cost of some data relocation.

Let us compare the iteration cost of diagonal and transformed EBDF on three processors with that of sequential MEBDF. Suppose that, respectively, m_1, m_2 and m_3 iterations are required to solve the three MEBDF equations in (3a) and (3c) in sequence and m iterations are required for parallel EBDF. Then, sequential MEBDF needs one LU decomposition, $m_1+m_2+m_3$ sequential forward–backward substitutions and $m_1+m_2+m_3$ sequential evaluations of f. Thus, diagonal and transformed EBDF (if we ignore the transformation costs) are less costly than sequential MEBDF if $m < m_1 + m_2 + m_3$. Finally, we remark that in an actual implementation of diagonal and transformed EBDF, it is sometimes advantageous to use in the system matrix in (8) the Jacobian J_{n+1} in the blocks of the first and third row and an approximation J_{n+2} of the Jacobian at t_{n+2} in the blocks of the second row (see Section 4). Since these Jacobians can again be evaluated concurrently, the effective costs do not increase.

3. Convergence of diagonal and transformed EBDF

Let us consider the rate of convergence of the iteration process (8). Defining the iteration error $\varepsilon^{(j)} := Y^{(j)} - Y_{n+1}$, we subtract the exact solution relation:

$$(I - A^* \otimes hJ_{n+1})(Y_{n+1} - Y_{n+1}) = -R_n(Y_{n+1})$$

from (8) getting

$$\begin{aligned}(I - A^* \otimes hJ_{n+1})(\varepsilon^{(j)} - \varepsilon^{(j-1)}) &= R_n(Y_{n+1}) - R_n(Y_{n+1} + \varepsilon^{(j-1)}) \\ &= -\varepsilon^{(j-1)} + h(A \otimes I)[F(Y_{n+1} + \varepsilon^{(j-1)}) - F(Y_{n+1})] \\ &= -(I - A \otimes hJ_{n+1})\varepsilon^{(j-1)} \\ &\quad - h(A \otimes I)(I \otimes J_{n+1})\varepsilon^{(j-1)} \\ &\quad + h(A \otimes I)[F(Y_{n+1} + \varepsilon^{(j-1)}) - F(Y_{n+1})],\end{aligned}$$

or

$$\begin{aligned}(I - A^* \otimes hJ_{n+1})\varepsilon^{(j)} &= (A - A^*) \otimes hJ_{n+1}\varepsilon^{(j-1)} + h(A \otimes I) \\ &\quad \times [F(Y_{n+1} + \varepsilon^{(j-1)}) - F(Y_{n+1}) - (I \otimes J_{n+1})\varepsilon^{(j-1)}].\end{aligned}$$

Defining

$$\begin{aligned}M &:= (I - A^* \otimes hJ_{n+1})^{-1}((A - A^*) \otimes hJ_{n+1}), \\ L &:= (I - A^* \otimes hJ_{n+1})^{-1}(A \otimes I), \\ \Phi(\varepsilon) &:= F(Y_{n+1} + \varepsilon) - F(Y_{n+1}) - (I \otimes J_{n+1})\varepsilon,\end{aligned}$$

the error recursion becomes

$$\varepsilon^{(j)} = M\varepsilon^{(j-1)} + hL\Phi(\varepsilon^{(j-1)}),\ j \geqslant 1. \tag{11}$$

Hence,

$$\varepsilon^{(j)} = M^j\varepsilon^{(0)} + hM^{j-1}L\Phi(\varepsilon^{(0)}) + \cdots + hML\Phi(\varepsilon^{(j-2)}) + hL\Phi(\varepsilon^{(j-1)}). \tag{12}$$

3.1 *The rate of convergence*

Let A^* be defined by (9a), so that A^* has the same diagonal entries as A. Then the 3-by-3 lower block-triangular matrix M has zero diagonal blocks

$$M = (I - A^* \otimes hJ_{n+1})^{-1} \begin{bmatrix} 0 & & \\ \bar{a}_1\bar{b}_0 hJ_{n+1} & 0 & \\ -c_1 hJ_{n+1} & (b_1 - c_2)hJ_{n+1} & 0 \end{bmatrix}, \tag{13}$$

so that M^j vanishes for all $j \geqslant 3$. Thus,

$$\begin{aligned} \varepsilon^{(1)} &= M\varepsilon^{(0)} + hL\Phi(\varepsilon^{(0)}), \\ \varepsilon^{(2)} &= M^2\varepsilon^{(0)} + hML\Phi(\varepsilon^{(0)})hL\Phi(\varepsilon^{(1)}), \\ \varepsilon^{(j)} &= hM^2L\Phi(\varepsilon^{(j-3)}) + hML\Phi(\varepsilon^{(j-2)}) + hL\Phi(\varepsilon^{(j-1)}),\ j \geqslant 3. \end{aligned} \tag{12'}$$

In the case of *linear* problems, where the function Φ vanishes, we have convergence *within three iterations* (of course, if $A^* = A$, then (8) reduces to the modified Newton, which converges within one iteration for linear problems).

For *non-linear* problems, we consider the first-order approximation to (12'). Let us write $\Phi(\varepsilon) = K\varepsilon + O(\varepsilon^2)$, where K is the (3-by-3 block-diagonal) Jacobian matrix of $\Phi(\varepsilon)$ at $\varepsilon = 0$. Let $N := hLK$, and neglecting terms of degree higher than one in ε, the first-order approximation to (12') becomes

$$\begin{aligned} \varepsilon^{(1)} &= (M + N)\varepsilon^{(0)}, \\ \varepsilon^{(2)} &= (M + N)^2\varepsilon^{(0)}, \\ \varepsilon^{(j)} &= M^2N\varepsilon^{(j-3)} + MN\varepsilon^{(j-2)} + N\varepsilon^{(j-1)},\ j \geqslant 3. \end{aligned} \tag{14}$$

In the modified Newton case ($A^* = A$), we have $M = 0$, so that the first-order error recursion (14) reduces to $\varepsilon^{(j)} = N\varepsilon^{(j-1)}$, $j \geqslant 1$. However, if $M \neq 0$, then both N and M play a role in the rate of convergence. We consider the first few iteration errors taking the structure of the matrices M and N into account. From (9a) and (11) it can be seen that N has a 3-by-3 block lower triangular structure. It follows from the nilpotent structure of M (13), and the fact that the product of a lower triangular matrix and a lower triangular nilpotent matrix maintains the same nilpotent structure (independent of the order of multiplication), that all matrix products in (14) containing three or more factors M vanish, so that

$$\begin{aligned} \varepsilon^{(1)} &= (M + N)\varepsilon^{(0)}, \\ \varepsilon^{(2)} &= (M^2 + MN + NM + N^2)\varepsilon(0), \\ \varepsilon^{(3)} &= (M^2N + MNM + MN^2 + NM^2 + NMN + N^2M + N^3)\varepsilon^{(0)}, \\ \varepsilon^{(4)} &= (M^2N^2 + (MN)^2 + MN^2M + NM^2N + (NM)^2 + N^2M^2 + O(N^3))\varepsilon^{(0)}, \\ \varepsilon^{(j)} &= O(N^{j-2})\varepsilon^{(0)},\ j \geqslant 5. \end{aligned} \tag{14'}$$

where the notation $O(N^i)$ is used for terms containing at least i factors of N. We summarize the preceding derivations in the following theorem.

THEOREM 3.1 In the error recursion (11), let the function Φ satisfy $\Phi(\varepsilon) = K\varepsilon + O(\varepsilon^2)$ and let $N := hLK$. Then the first-order approximation to the error recursion (12) is given by

$\{\varepsilon^{(j)} = N\varepsilon^{(j-1)},\ j \geqslant 1\}$ if $A^* = A$ (modified Newton),

$\{\varepsilon^{(j)} = (M+N)^j\varepsilon^{(0)},\ j = 1,2;\ \varepsilon^{(j)} = O(N^{j-2})\varepsilon^{(0)},\ j \geqslant 3\}$ if A^* is defined by (9a).

Thus, if A^* is defined by (9a), then after at most two iterations the rate of convergence is comparable with that of the modified Newton, for all values of c_1 and c_2. However, in the transformed EBDF case with $c_2 = b_1$, this is already achieved after one iteration, because for this choice M assumes the form (see (13))

$$M = \begin{bmatrix} 0 & & \\ \times & 0 & \\ \times & 0 & 0 \end{bmatrix}.$$

Both NM and MN have this same structure; hence, all matrix products in (14′) containing two or more factors M vanish. It follows that

$$\varepsilon^{(1)} = (M+N)\varepsilon^{(0)}, \quad \varepsilon^{(j)} = O(N^{j-1})\varepsilon^{(0)},\ j \geqslant 2.$$

Thus, we have proved the theorem.

THEOREM 3.2 Let the conditions of Theorem 3.1 be satisfied and let $c_2 = b_1$ in (9a). Then, for all c_1 the first-order approximation to the error recursion (12) associated with transformed EBDF (9a), (9b) and (10) is given by

$$\{\varepsilon^{(1)} = (M+N)\varepsilon^{(0)};\quad \varepsilon^{(j)} = O(N^{j-1})\varepsilon^{(0)}\, j \geqslant 2\}.$$

3.2 *Amplification factors*

Theorems 3.1 and 3.2 show that transformed and diagonal EBDF may converge slower than the Newton in the first iteration and the first two iterations, respectively. The reason is that the magnitude of M is expected to be much greater than that of N. We shall consider the effect of the amplification matrix $(M+N)^j \approx M^j$ on the initial error $\varepsilon^{(0)}$. Although M has only zero eigenvalues, the magnitude of M is not necessarily small. Let us expand $\varepsilon^{(0)}$ with respect to the vectors $a \otimes v$, where v is an eigenvector of the Jacobian matrix J_{n+1}. Since

$$M^j(a \otimes v) = (Z^j(z) \otimes I)(a \otimes v), \quad Z(z) := z(I - zA^*)^{-1}(A - A^*), \quad z := h\lambda(J_{n+1}), \tag{15}$$

we are interested in the size of $\|Z^j(z)\|_\infty$. It follows from (9a) that

$$Z(z) = \frac{z}{(1-b_0z)(1-\bar{b}_0z)} \begin{bmatrix} 0 & 0 & 0 \\ \bar{a}_1\bar{b}_0(1-b_0z) & 0 & 0 \\ (c_1 + c_2\bar{a}_1)\bar{b}_0z - c_1 & (b_1 - c_2)(1-\bar{b}_0z) & 0 \end{bmatrix}.$$

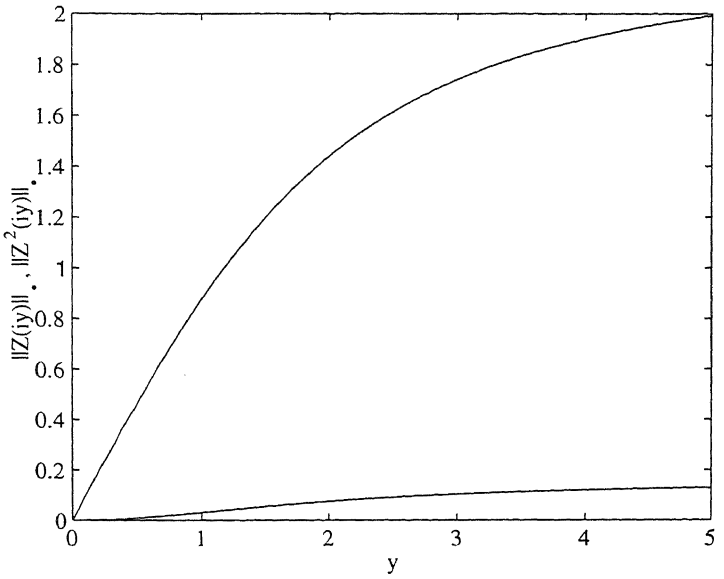

FIG. 1. The amplification factors (16) for the sixth-order iterated EBDF method ($k = 5$).

Assuming that the eigenvalues $\lambda(J_{n+1})$ are in the left half- plane, then $\|Z^j(z)\|_\infty$ is maximal along the imaginary axis, so that we set $z = iy$. Since c_1 and c_2 do not appear in the second row of Z, we can do no better than choosing these parameters such that the third row is inferior to the second in determining the infinity norm. We verified that this is the case, both for the diagonal Newton, where $c_1 = c_2 = 0$, and for the transformed Newton with $c_1 = 0$ and $c_2 = b_1$. For these cases we find

$$\|Z(iy)\|_\infty = \frac{\bar{a}_1\bar{b}_0|y|}{\sqrt{1+\bar{b}_0^2y^2}}, \quad \|Z^2(iy)\|_\infty = \frac{\bar{a}_1\bar{b}_0|b_1 - c_2||y^2|}{\sqrt{1+\bar{b}_0^2y^2}\sqrt{1+b_0^2y^2}}, \tag{16}$$

showing that $\|Z(iy)\|_\infty$ monotonically increases from 0 to $\bar{a}_1$ and $\|Z^2(iy)\|_\infty$ monotonically increases from 0 to $\bar{a}_1|b_1 - c_2|b_0^{-1}$. Since $\bar{a}_1 > 1$ (see Table 2), we should expect that the stiff components in the iteration error are amplified in the first iteration. However, in the second iteration, the transformed EBDF already has a zero amplification factor and the diagonal EBDF has quite a small amplification factor because $\bar{a}_1|b_1|b_0^{-1} \ll 1$. Figure 1 illustrates the behaviour of $\|Z(iy)\|_\infty$ and $\|Z^2(iy)\|_\infty$ as given by (16) for the sixth-order diagonal EBDF method ($k = 5$) used in the numerical experiments.

4. Numerical experiments

In this section, we compare the accuracy obtained with a sixth- order ($k = 5$) sequential MEBDF method and the transformed ($c_1 = 0$, $c_2 = b_1$) and diagonal ($c_1 = c_2 = 0$) EBDF methods. Our selection of the sixth-order method was motivated by the development of a *four*-stage, sixth-order *L-stable* method by Psihoyios and Psihoyios (1998), the

parallelization of which we discuss in a companion article (Frank (2000)) and compare with the three-stage methods of this one. In a full implementation one would of course desire a mechanism for varying the step size. However, since at this point we are primarily interested in the algorithmic properties, we will consider only *fixed* step sizes in order to separate the strategy effects. Some possibilities for variable step size implementations use backward difference arrays (Cash (1983); Hairer *et al.* (1993)) or Nordsieck vectors (Hairer *et al.* (1993)).

In all of the experiments, we computed the initial iterates by taking the most recent approximation available at each time level or, if not yet available (in the case of the 'future' value u_{n+2}), by $(k+1)$-point extrapolation of already computed approximations. The experiments include results obtained by the three methods, where the Jacobian matrix J_{n+1} is evaluated in each step using the future-point approximation to y_{n+1} from the preceding step. Moreover, we included results obtained by diagonal EBDF using two Jacobians J_{n+1} and J_{n+2}, where the y-argument in J_{n+2} is determined by extrapolation of already computed y-values (these two Jacobians can of course be evaluated in parallel). This version will be denoted by EBDF(2). In a realistic implementation, one would only update the Jacobian when necessary to improve convergence.

The starting values were computed either from the exact solution if available or by applying the fifth-order Radau IIA method with a five times smaller step size. We took two well-known test problems from the literature having no transient phase, which allows us to use fixed step sizes, viz. a problem posed by Kaps (1981):

$$\frac{dy_1}{dt} = -1002y_1 + 1000y_2^2, \quad \frac{dy_2}{dt} = y_1 - y_2(1+y_2), \quad y_1(0) = y_2(0) = 1, \quad 0 \leqslant t \leqslant 5, \tag{17}$$

with exact solution $y_1 = e^{-2t}$, $y_2 = e^{-t}$, and the problem

$$\text{HIRES on } [5,\ 321{\cdot}8122], \tag{18}$$

where the initial conditions at $t = 5$ were obtained by integrating the HIRES problem given in Hairer (1996, p. 157) on $[0, 5]$. It turns out that these problems are relatively easy in the sense that the three methods converge within one or two iterations. Therefore, we also used the more difficult problem

$$\begin{aligned} y_1' &= -1000(y_1^3 y_2^6 - \cos^3(t)\sin^6(t)) - \sin(t), && y_1(0) = 1, \\ y_2' &= -1000(y_2^5 y_3^4 - \sin^5(t)\sin^4(t)) + \cos(t), && y_2(0) = 0, \quad 0 \leqslant t \leqslant 1 \\ y_3' &= -1000(y_1^2 y_3^3 - \cos^2(t)\sin^3(t)) + \cos(t), && y_3(0) = 0, \end{aligned} \tag{19}$$

with exact solution $y_1 = \cos(t)$ and $y_2 = y_3 = \sin(t)$. Because of its strong non-linearity it is a more suitable test problem for showing the differences in rate of convergence of the three methods. Finally, we tested the problem

$$\begin{aligned} y_1' &= -0{\cdot}04y_1 + 10^4 y_2 y_3 - 0{\cdot}96e^{-t}, && y_1(0) = 1, \\ y_2' &= 0{\cdot}04y_1 - 10^4 y_2 y_3 - 10^7 (y_2)^2 - 0{\cdot}04e^{-t}, && y_2(0) = 0, \quad 0 \leqslant t \leqslant t_{\text{end}}, \\ y_3' &= 3\,10^7 (y_2)^2 + e^{-t}, && y_3(0) = 0, \end{aligned} \tag{20}$$

TABLE 3
Values of scd for problem (17)

N	Method	$m=1$	$m=2$	$m=3$	...	$m=\infty$
10	Sequential MEBDF	4·7			...	4·7
	Transformed EBDF	*	4·5		...	4·5
	Diagonal EBDF	*	4·7	4·5	...	4·5
	Diagonal EBDF(2)	*	4·7	4·5	...	4·5
20	Sequential MEBDF	6·5			...	6·5
	Transformed EBDF	*	6·3		...	6·3
	Diagonal EBDF	*	6·4	6·3	...	6·3
	Diagonal EBDF(2)	*	6·4	6·3	...	6·3
40	Sequential MEBDF	8·3			...	8·3
	Transformed EBDF	*	8·1		...	8·1
	Diagonal EBDF	*	8·2	8·1	...	8·1
	Diagonal EBDF(2)	*	8·2	8·1	...	8·1

with exact solution $y_1 = e^{-t}$, $y_2 = 0$, $y_3 = 1 - e^{-t}$. This problem has the same highly stiff Jacobian matrix as the famous Robertson problem (Robertson, 1966), but it is modified by adding non-homogeneous terms, so that it possesses for the given initial values a solution without transient phase. System (20) resembles the original Robertson problem more as t increases. Note that the numerical integration process will become unstable if negative approximations to $y_2(t)$ are generated.

In our numerical experiments, we denoted the number of steps by N, the number of iterations in each iteration process by m , and the total number of iterations by M (not including the iterations needed to compute the starting values). Note that for fixed values of m and N, sequential MEBDF requires three times more *sequential* right-hand-side evaluations and forward–backward substitutions than the transformed and diagonal EBDF-type methods, because sequential MEBDF solves three subsystems per step. Hence, for sequential MEBDF, the value of M is three times greater. The accuracy is given by the number of significant correct digits (*scd*); that is, we write the maximal *absolute* end-point error in the form 10^{-scd}. In the tables of results, we shall indicate negative *scd* values by *.

4.1 *Fixed numbers of iterations*

We start by applying the three methods with a prescribed number of iterations m. In the case of the HIRES problem (18) where no exact solution is available, the starting values were provided by the Radau IIA method using ten iterations. Tables 3 and 4 list for given values of m and N the resulting *scd* values for the problems (17) and (18). These results show that in almost all cases sequential MEBDF finds the solution in one

TABLE 4
Values of scd for problem (18)

N	Method	$m=1$	$m=2$	$m=3$	$m=4$	...	$m=\infty$
10	Sequential MEBDF	2·2	2·7	2·8	2·7	...	2·7
	Transformed EBDF	*	3·1	2·6	2·7	...	2·7
	Diagonal EBDF	*	2·8	2·5	2·7	...	2·7
	Diagonal EBDF(2)	*	*	2·4	0·6	...	*
20	Sequential MEBDF	3·4	3·3			...	3·3
	Transformed EBDF	*	3·3			...	3·3
	Diagonal EBDF	*	3·6	3·4	3·3	...	3·3
	Diagonal EBDF(2)	*	3·2	3·1	3·3	...	*
40	Sequential MEBDF	4·3	4·2			...	4·2
	Transformed EBDF	*	4·3			...	4·3
	Diagonal EBDF	*	4·4	4·3		...	4·3
	Diagonal EBDF(2)	*	4·3			...	4·3

iteration per subsystem, whereas transformed or diagonal EBDF needs two iterations for the whole system (note that transformed and diagonal EBDF show a comparable convergence behaviour). Diagonal EBDF(2) behaves poorly for the HIRES problem (18) due to the relatively large time steps which destroy the quality of the Jacobian J_{n+2} (recall that the argument in J_{n+2} is based on extrapolation of preceding y-values). Only for the smallest step size in Table 4 (i.e. $h \approx 7{\cdot}9$) does the diagonal EBDF(2) method converge. As to the order behaviour, for the Kaps problem the order $p = 6$ of the methods is reproduced, but for the HIRES problem the step size is too large to observe asymptotic convergence.

In order to see more clearly the differences in convergence rates, we now integrate the highly non-linear problem (19). Surprisingly, the numbers of iterations to reach the converged solution is more or less comparable for all methods and differs by at most one iteration. Furthermore, in this example, the additional Jacobian J_{n+2} used in diagonal EBDF(2) improves the initial rate of convergence considerably. The $N = 20$ and $N = 40$ results indicate that again only the stage order $s = 5$ is shown (since the experiments were run with 14 decimals precision, the $N = 80$ results did not reach the expected value $scd = 14{\cdot}3$). Finally, we integrate the highly stiff modified Robertson problem (20) with $t_{\text{end}} = 1$. Here, the performance is similar to that for the Kaps problem (17). Apparently, the methods are able to compute positive approximations to the second component $y_2(t)$.

4.2 *Variable number of iterations*

If the number of iterations is adjusted for each non-linear system (or subsystem in the case of sequential MEBDF) to be solved, then the efficiency is obviously improved because we avoid the situation where the (sub)system solutions have quite different accuracies. Moreover, in such a dynamic approach, sequential MEBDF can take advantage of the fact that it solves the subsystems *successively* instead of simultaneously, as done in

TABLE 5
Values of scd for problem (19)

N	Method	$m=1$	$m=2$	$m=3$	$m=4$	$m=5$	$m=6$	$m=7$	...	$m=\infty$
20	Seq. MEBDF	5·3	9·3	10·3	10·9	10·8	10·9		...	10·9
	Transf. EBDF	*	7·6	11·2	11·4	11·3			...	11·3
	Diag. EBDF	*	5·0	9·8	10·5	10·9	11·4	11·3	...	11·3
	Diag. EBDF(2)	*	11·4	11·3					...	11·3
40	Seq. MEBDF	11·7	12·3	12·4	12·5	12·4			...	12·4
	Transf. EBDF	*	12·9	12·8					...	12·8
	Diag. EBDF	*	11·3	12·3	12·9	12·8			...	12·8
	Diag. EBDF(2)	*	13·1	12·8					...	12·8
80	Seq. MEBDF	13·5	13·9	13·8					...	13·8
	Transf. EBDF	*	13·8						...	13·8
	Diag. EBDF	*	13·5	13·8					...	13·8
	Diag. EBDF(2)	*	13·8						...	13·8

TABLE 6
Values of scd for problem (20) with $t_{end} = 1$

N	Method	$m=1$	$m=2$	...	$m=\infty$
10	Sequential MEBDF	7·9		...	7·9
	Transformed EBDF	7·9		...	7·9
	Diagonal EBDF	7·8	7·9	...	7·9
	Diagonal EBDF(2)	7·9		...	7·9
20	Sequential MEBDF	9·6		...	9·6
	Transformed EBDF	6·4	9·6	...	9·6
	Diagonal EBDF	*	9·6	...	9·6
	Diagonal EBDF(2)	4·9	9·6	...	9·6
40	Sequential MEBDF	11·3		...	11·3
	Transformed EBDF	*	11·3	...	11·3
	Diagonal EBDF	*	11·3	...	11·3
	Diagonal EBDF(2)	*	11·3	...	11·3

the transformed and diagonal EBDF methods. Hence, we also obtain a more honest comparison.

In our dynamic iteration strategy, we used the stopping strategy described in Hairer (1996, p. 130). This stopping strategy depends on a given tolerance parameter *Tol*, because it presupposes the use of automatic step size selection based on keeping the local truncation error (*LTE*) close to *Tol*. Since we focus on convergence aspects we want to use fixed step sizes, so that we have to replace *Tol* by some estimate of *LTE*. In our case, the difference $u_n - y_n$ from the preceding step provides us with a free estimate of *LTE*. We define the

TABLE 7
Values of M for problem (17) with $\kappa = 0{\cdot}1$

Method	$scd = 5$	$scd = 6$	$scd = 7$	$scd = 8$	$scd = 9$	$scd = 10$
Sequential MEBDF	29	49	79	123	187	282
Diagonal EBDF	20	32	59	106	160	244
Diagonal EBDF(2)	20	32	62	97	153	235

TABLE 8
Values of M for problem (18) with $\kappa = 0{\cdot}1$

Method	$scd = 4$	$scd = 5$	$scd = 6$	$scd = 7$
Sequential MEBDF	126	210	305	406
Diagonal EBDF	83	133	189	241
Diagonal EBDF(2)	72	122	177	234

damping parameter θ_m and the accumulated damping parameter η_m as

$$\theta_m := \frac{\|Y^{(m)} - Y^{(m-1)}\|_\infty}{\|Y^{(m-1)} - Y^{(m-2)}\|_\infty}, \quad \eta_0 := (\eta_{\text{old}})^{0.8}, \quad \eta_m := \frac{\theta_m}{1 - \theta_m}, \quad m \geqslant 1, \tag{21a}$$

where η_{old} equals the η_m from the preceding step (bounded below by the machine precision). Then, the stopping criterion described in Hairer (1996) yields for the number of iterations m the condition

$$\eta_m \|Y^{(m)} - Y^{(m-1)}\|_\infty \leqslant \kappa \|u_n - y_n\|_\infty. \tag{22b}$$

Here, κ is a control parameter. The implicit relations are solved more accurately as κ is smaller. For the problems (17)–(20), we performed experiments where the number of steps was chosen such that a prescribed *scd* value was obtained. For these problems, the *maximal* number of iterations in the subsequent iteration processes was prescribed, viz. $m = 5, 10, 20$ and 10, respectively. In problem (18), where no exact solution is available, we used the Radau starting method with $m = 10$. Tables 7–11 list the total number of iterations M needed to obtain a given *scd* value. Since transformed and diagonal EBDF exhibit a similar convergence behaviour, we only listed *scd* values for the easier implementable diagonal EBDF methods. From these results we may conclude that the total number of iterations is always less for the diagonal EBDF methods. Furthermore, diagonal EBDF(2) is now performing quite well for the HIRES problem because the step size is adjusted to the required accuracy. On the basis of the above results, we can derive theoretical speed-up factors for the efficiency of the iteration part of the methods. Table 12 presents such efficiency speed-up factors by comparing M-values (averaged over the *scd* values) for sequential MEBDF and diagonal EBDF(2).

4.3 *Code timings*

Finally we will give an indication of how our formulation of the diagonal EBDF method compares with the sequential MEBDF method of Cash when implemented on a parallel

TABLE 9
Values of M for problem (19) with $\kappa = 0{\cdot}1$

Method	$scd = 10$	$scd = 11$	$scd = 12$	$scd = 13$
Sequential MEBDF	103	118	157	231
Diagonal EBDF	131	125	121	140
Diagonal EBDF(2)	32	38	67	105

TABLE 10
Values of M for problem (20) with $t_{\text{end}} = 1$ *and* $\kappa = 0{\cdot}1$

Method	$scd = 8$	$scd = 9$	$scd = 10$	$scd = 11$	$scd = 12$	$scd = 13$
Sequential MEBDF	21	39	66	107	168	260
Diagonal EBDF	9	17	29	49	74	114
Diagonal EBDF(2)	8	17	30	48	74	115

TABLE 11
Values of M for problem (20) with $t_{\text{end}} = 10$ *and* $\kappa = 0{\cdot}1$

Method	$scd = 3$	$scd = 4$	$scd = 5$	$scd = 6$	$scd = 7$	$scd = 8$
Sequential MEBDF	31	60	101	163	255	392
Diagonal EBDF	19	37	47	75	119	184
Diagonal EBDF(2)	18	26	47	76	119	184

shared memory machine, in this case a Cray C916. Parallel speed-ups in this section were obtained using the Autotasking Expert Analysis tool (Cray Research Inc., 1994) available on Cray computer systems, which estimates the speed-up that would be obtained by a program run on a dedicated multiprocessor system, based on the observed performance of an arbitrarily loaded system. Since the ATExpert tool measures speed-up with respect to the same code run on a single processor, it is important in order to obtain meaningful results that no redundant work be performed within parallel sections of the code. The tests in this section were run with a *fixed* number of Newton iterations per time step to clearly distinguish the parallel performance in the absence of iteration strategies. We have taken many more time steps in the experiments of this section to reduce the effects of initialization costs, such as memory allocation and start-up procedure.

There is, of course, a certain amount of parallelism available in sequential MEBDF. For each of the three relations in (3a) and (3c), a non-linear system must be solved with a (modified) Newton method, in which the following tasks have varying degrees of parallelism:

1. evaluation of the Jacobian J_{n+1};
2. evaluation of the right-hand side;
3. update of the solution vector;
4. computation of an LU decomposition of the system matrix $I - \bar{b}_0 h J_{n+1}$;
5. execution of a forward–backward substitution.

TABLE 12
Theoretical iteration speed-up of diagonal EBDF(2)

Problem	Speed-up
(17)	1·3
(18)	1·7
(19)	2·7
(20)	2·2

These tasks all contain a number of independent operations which are proportional to the problem dimension d (*parallelism across the space*, in the classification of Gear (1993)), and are present in sequential MEBDF as well as in the diagonal EBDF methods. However, we are interested in an additional, coarser grained parallelism, orthogonal to these parallelizations, such as the concurrent computation of LU decompositions and forward–backward substitutions for the three subsystems (*parallelism across the method*). This kind of parallelism is not available if the subsystems are solved successively as in sequential MEBDF. However, by solving the subsystems simultaneously, as in diagonal EBDF, all of items 1 through to 5 above can be computed in parallel for the three subsystems. In the following subsections we present timings concerning the effect of concurrent computation of the various tasks in the diagonal EBDF.

4.3.1 *LU decompositions.* Since the computation of LU decompositions is generally considered to be expensive, we first discuss the effect on the CPU time of computing the LU decomposition of the matrices $I - \bar{b}_0 h J_{n+1}$ and $I - b_0 h J_{n+2}$ needed in diagonal EBDF concurrently. Since the Jacobians are factored only once per time step, the effect of factoring them concurrently becomes less important as more iterations are needed. Table 13 shows for $N = 1280$ time steps the speed-up figures obtained from a two-processor implementation of diagonal EBDF in which only the two LU decompositions are computed in parallel. Apparently, for the problems (17)–(20), the parallel computation of the LU decompositions does not lead to a substantial speed-up, even for the eight-dimensional HIRES problem (18). Of course, for higher dimensional problems, the speed-up will *increase*. On the other hand, a more sophisticated implementation, where the Jacobian is only updated every few steps, will *decrease* the speed-up attained by concurrent decomposition of Jacobians. Therefore, a substantial speed-up of a parallel implementation of diagonal EBDF should not be expected from the parallel computation of the LU decompositions alone.

4.3.2 *Overhead costs.* The diagonal EBDF approach incurs a small increase in cost due to the fact that the most recently computed function evaluations $f(u_{n+1}^{(j-1)})$ and $f(u_{n+2}^{(j-1)})$ must be updated in the second and third components of the residue in (8), whereas these are constant components of the residue functions if the subsystems are solved in sequence.

TABLE 13
Speed-ups attained by concurrent decomposition of Jacobians in diagonal EBDF

Problem	$m = 2$	$m = 3$	$m = 4$	$m = 5$
(17)	0·97	0·97	1·00	1·00
(18)	1·18	1·14	1·11	1·10
(19)	1·03	1·02	1·02	1·02
(20)	1·04	1·03	1·03	1·02

TABLE 14
Ratio of serial CPU times for sequential and diagonal Newton

Problem	$m = 2$	$m = 3$	$m = 4$	$m = 5$
(17)	0·94	0·91	0·89	0·89
(18)	0·98	0·97	0·97	0·96
(19)	1·01	0·99	0·97	0·97
(20)	0·98	0·96	0·95	0·94

Hence, these additional costs have to be considered as overhead costs. In order to estimate these costs, we compared diagonal EBDF with sequential EBDF. The latter method is understood to be the method obtained if the EBDF subsystems in (3a) and (3b) are solved sequentially. An indication of the significance of this overhead is provided in Table 14, in which the ratio of serial CPU times for sequential EBDF and diagonal EBDF is compared for $N = 1280$ time steps. These figures show that the increase in sequential overhead is quite modest.

4.3.3 *Overall speed-up factors.* Table 15 shows the ATExpert observed speed-up of the diagonal EBDF approach on three processors over sequential MEBDF on one processor for $N = 1280$ time steps and $m = 5$ iterations. It is noteworthy that this speed-up is essentially independent of the number of Newton iterations. In the table we have also listed the dimension of each system (the non-autonomous terms of problems (19) and (20) have been implemented as an extra dimension). The attainable speed-up is highest for the HIRES problem, which has dimension 8, and lowest for the Kaps problem of dimension 2. As observed in Section 4.1, we suffer only a slight loss in convergence rate when changing from sequential MEBDF to diagonal EBDF. Hence, we may expect comparable accuracies for equal numbers of steps N and iterations m, so that the CPU speed-up factors in Table 15 are also an indication of the speed-up of *efficiency* (that is, CPU speed-up under the condition of equal accuracies).

TABLE 15
Speed-up of diagonal EBDF on three processors

Problem	d	$m = 5$
(17)	2	1·8
(18)	8	2·3
(19)	4	2·0
(20)	4	2·0

Acknowledgements

The authors are very grateful for the comments and suggestions of the referees.

REFERENCES

BROWN, P. N., HINDMARSH, A. C., & BYRNE, G. D. 1992 VODE: a variable coefficient ODE solver. Available at `http://www.netlib.org./ode/vode.f`.

CASH, J. R. 1980 On the integration of stiff ODEs using extended backward differentiation formulae. *Numer. Math.* **34**, 235–246.

CASH, J. R. 1983 The integration of stiff initial value problems in ODEs using modified extended backward differentiation formulae. *Comput. Math. Appl.* **5**, 645–657. Software available at `http://www.ma.ic.ac.uk/~jcash/IVP_software/finaldae/readme.html`.

1994 Cray Research Inc. *CF77 Commands and directives, SR-3771*, 6.0 edn.

FRANK, J. E. & VAN DER HOUWEN, P. J. 2000 Diagonalizable extended backward differentiation formulas. *BIT* **40**, 497–512.

GEAR, C. W. 1993 Massive parallelism across time in ODEs. *Appl. Numer. Math.* **11**, 27–44 *Proc. Int. Conf. on Parallel Methods for Ordinary Differential Equations, Grado (It), Sept10–13, 1991.*

HAIRER, E., NORSETT, S. P., & WANNER, G. 1993 *Solving Ordinary Differential Equations, I. Nonstiff Problems*, 2nd edn, Berlin: Springer.

HAIRER, E. & WANNER, G. 1996 *Solving Ordinary Differential Equations, II. Stiff and Differential-Algebraic Problems*, 2nd edn, Berlin: Springer.

HAIRER, E. & WANNER, G. 1998 RADAU. Available at `ftp://ftp.unige.ch/pub/doc/math/stiff/radau.f`.

KAPS, P. 1981 Rosenbrock-type methods. *Numerical Methods for Stiff Initial Value Problems, Bericht nr. 9.* (G. Dahlquist & R. Jeltsch eds). Inst. für Geometrie und Praktische Mathematik der RWTH Aachen, Germany.

LIOEN, W. M. & DE SWART, J. J. B. 1998 Test set for IVP solvers, Release 2.0. Available at `http://www.cwi.nl/cwi/projects/IVPtestset/`.

PETZOLD, L. R. 1991 DASSL: a differential/algebraic system solver. Available at `http://www.netlib.org/ode/ddassl.f`.

PSIHOYIOS, G.-Y. & CASH, J. R. 1998 A stability result for general linear methods with characteristic function having real poles only. *BIT* **38**, 612–617.

ROBERTSON, H. H. 1966 The solution of a set of reaction rate equations. *Numerical Analysis, an Introduction*. (J. Walsh ed.). Academic, pp 178–182.

SHAMPINE, L. F. 1994 *Numerical Solution of Ordinary Differential Equations*. New York: Chapman & Hall.