Centrum voor Wiskunde en Informatica

REPORT*RAPPORT*

Parallel Iteration of the Extended Backward Differentiation Formulas

J.E. Frank, P.J. van der Houwen

Modelling, Analysis and Simulation (MAS)

MAS-R9913 May 31, 1999

Report MAS-R9913
ISSN 1386-3703

Parallel Iteration of the Extended Backward Differentiation Formulas

J.E. Frank
TU Delft, Fac. ITS
P.O. Box 356, 2600 AJ Delft, The Netherlands

P.J. van der Houwen
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

ABSTRACT
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 nonlinear 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 righthand 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 nonlinear systems, so that in each iteration the forward-backward substitutions and the righthand 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 speedup factor of about 2 on 3 processors.

1991 Mathematics Subject Classification: 65L06
Keywords and Phrases: numerical analysis, iteration methods, initial-value problems, extended BDFs, parallelism
Note: Work carried out under project MAS 2.3 - 'Numerical Algorithms for Initial Value Problems.' The investigations reported in this paper were partly supported by STW and TU Delft.

1. Introduction

The extended backward differentiation formulas (EBDFs) were proposed by Cash[2] in 1980 for solving initial-value problems (IVPs) for stiff systems of ordinary differential equations (ODEs)

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

Each EBDF step essentially consists of successively solving three nonlinear 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 stepsize. In order to reduce the LU costs, Cash[3] modified the EBDF methods (MEBDF methods) such that only one LU decomposition is required.

In a recent performance evaluation [9] 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 [7], DASSL[10] and VODE[1]. This motivated us to look at possible parallel implementations of the MEBDF method.

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 righthand 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 nonlinear systems, so that in each iteration the forward-backward substitutions and the righthand 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 speedup factor of about 2 on 3 processors.

2. The EBDF and MEBDF methods of Cash

The EBDF method of Cash[2] is based on the formula

$$y_{n+1} = a_1 y_n + a_2 y_{n-1} + \cdots + a_k y_{n-k+1} + hb_0 f(y_{n+1}) + hb_1 f(y_{n+2}) \tag{2.1}$$

for computing an approximation y_{n+1} to the exact solution $y(t_{n+1})$ of (1.1). 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 BDF predictor formula

$$\begin{aligned} u_{n+1} &= \bar{a}_1 y_n + \bar{a}_2 y_{n-1} + \cdots + \bar{a}_k y_{n-k+1} + h\bar{b}_0 f(u_{n+1}), \\ u_{n+2} &= \bar{a}_1 u_{n+1} + \bar{a}_2 y_n + \cdots + \bar{a}_k y_{n-k+2} + h\bar{b}_0 f(u_{n+2}), \end{aligned} \tag{2.2a}$$

to obtain a prediction u_{n+2} for the ‘future’ value y_{n+2}. Thus, y_{n+1} is computed from the equation

$$y_{n+1} = a_1 y_n + a_2 y_{n-1} + \cdots + a_k y_{n-k+1} + hb_0 f(y_{n+1}) + hb_1 f(u_{n+2}). \tag{2.2b}$$

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 (2.2a) have order of accuracy k and the external (or output) vector y_{n+1} defined by (2.2b) has order of accuracy $k+1$. Hence the stage order s equals k and the actual order p equals $k+1$. Furthermore, {(2.2a),(2.2b)} possesses 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 (2.1) is much more stable than the classical BDF. For future reference, the coefficients $\{\bar{a}_i, \bar{b}_0\}$ and $\{a_i, b_0, b_1\}$ are given in the Tables 2.1 and 2.2 for $k = 2, \ldots, 5$. The MEBDF methods arise from the EBDF method {(2.2a),(2.2b)} by replacing (2.2b) with the formula (see Cash [3])

$$y_{n+1} = a_1 y_n + a_2 y_{n-1} + \cdots + a_k y_{n-k+1} + h\bar{b}_0 f(y_{n+1}) + h(b_0 - \bar{b}_0) f(u_{n+1}) + hb_1 f(u_{n+2}). \tag{2.2c}$$

The advantage is that the modified Newton iteration of the subsystems (2.2a) and (2.2b) 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.

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

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

Table 2.1: Coefficients $\{\bar{a}_i, \bar{b}_0\}$ in the BDF formulas (2.2a).

k	$\bar{a}_1 d$	$\bar{a}_2 d$	$\bar{a}_3 d$	$\bar{a}_4 d$	$\bar{a}_5 d$	$\bar{b}_0 d$	d
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 2.2: Coefficients $\{a_i, b_0, b_1\}$ in the EBDF and MEBDF formulas (2.2b) and (2.2c).

k	$a_1 d$	$a_2 d$	$a_3 d$	$a_4 d$	$a_5 d$	$b_0 d$	$b_1 d$	d
2	28	-5				22	-4	23
3	279	-99	17			150	-18	197
4	4008	-2124	728	-111		1644	-144	2501
5	26550	-18700	9600	-2925	394	8820	-600	14919

Then, using tensor notation, both the EBDF method $\{(2.2a), (2.2b)\}$ and the MEBDF method $\{(2.2a), (2.2c)\}$ can be represented in the compact form

$$(B \otimes I)Y_{n+1} - h(C \otimes I)F(Y_{n+1}) = (E \otimes I)V_n, \tag{2.3}$$

Here, I is the d-by-d identity matrix, $\otimes$ the Kronecker product, h the stepsize $t_{n+1} - t_n$, and $F(Y_{n+1})$ contains the righthand sides $f(u_{n+1})$, $f(u_{n+2})$, $f(y_{n+1})$. In the EBDF case, B, C and E are defined by

$$B := \begin{pmatrix} 1 & 0 & 0 \\ -\bar{a}_1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, C := \begin{pmatrix} \bar{b}_0 & 0 & 0 \\ 0 & \bar{b}_0 & 0 \\ 0 & b_1 & b_0 \end{pmatrix}, E := \begin{pmatrix} \bar{a}_k & \bar{a}_{k-1} & \cdots & \bar{a}_1 \\ 0 & \bar{a}_k & \cdots & \bar{a}_2 \\ a_k & a_{k-1} & \cdots & a_1 \end{pmatrix}. \tag{2.4}$$

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

2.2 Iteration processes
Instead of solving 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 here consider 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. Since the error of the diagonal approximation will be smaller as $B^{-1}C$ is itself closer to a diagonal matrix, we choose the EBDF method, rather than the MEBDF method, as our starting point, because in the EBDF case the first entry in the last row of $B^{-1}C$ vanishes and in the MEBDF case it does not.

Premultiplying (2.3) by $B^{-1} \otimes I$, we can rewrite it in the form

$$\begin{aligned} &R_n(Y_{n+1}) = 0, R_n(Y) := Y - h(A \otimes I)F(Y) - (B^{-1}E \otimes I)V_n, \\ &A := B^{-1}C = \begin{pmatrix} \bar{b}_0 & 0 & 0 \\ \bar{a}_1\bar{b}_0 & \bar{b}_0 & 0 \\ 0 & b_1 & b_0 \end{pmatrix}. \end{aligned} \tag{2.5}$$

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

$$(I - A^* \otimes hJ_{n+1})(Y^{(j)} - Y^{(j-1)}) = -R_n(Y^{(j-1)}),\ j = 1, \ldots, m, \tag{2.6}$$

where I is again the identity matrix, 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 righthand side function in (1.1) at t_{n+1} (note that in the preceding timestep, an approximation to y_{n+1} has been computed). Suppose that the first and third component 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.

It is tempting to set $A^* = A$ resulting in the familiar (modified) Newton method and to try to diagonalize (2.6) 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

$$A^* = \begin{pmatrix} \bar{b}_0 & 0 & 0 \\ 0 & \bar{b}_0 & 0 \\ c_1 & c_2 & b_0 \end{pmatrix}, \tag{2.7a}$$

then diagonalization is possible. It can be shown that

$$Q := \begin{pmatrix} q_1 & 0 & 0 \\ q_0 & q_2 & 0 \\ \frac{c_1 q_1 + c_2 q_2}{\bar{b}_0 - b_0} & \frac{c_2 q_2}{\bar{b}_0 - b_0} & q_3 \end{pmatrix} \Rightarrow Q^{-1}A^*Q = D, \ D := \mathrm{diag}(\bar{b}_0, \bar{b}_0, b_0) \tag{2.7b}$$

for all nonzero 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 {(2.7a),(2.7b)}, we can define the transformed iteration method

$$(I - D \otimes hJ_{n+1})(\tilde{Y}^{(j)} - \tilde{Y}^{(j-1)}) = -(Q^{-1} \otimes I)R_n(Y^{(j-1)}), \ Y^{(j)} = (Q \otimes I)\tilde{Y}^{(j)}, \ j = 1, \ldots, m. \tag{2.8}$$

We shall refer to {(2.7a),(2.7b),(2.8)} as the *transformed* EBDF method. In particular, we may set $c_1 = c_2 = 0$ in (2.7a), i.e. $A^* = D$, so that we can use $Q = I$ which avoids transformation costs. We shall call {(2.6), $A^* = D$} simply the *diagonal* EBDF method. Both iteration processes {(2.7a),(2.7b),(2.8)} and {(2.6), $A^* = D$} have the advantage of possessing a lot of additional intrinsic parallelism when compared with the MEBDF method as implemented in Cash[3] where the three equations in {(2.2a),(2.2c)} are solved sequentially by Newton iteration (to be referred to as *sequential* MEBDF). Firstly, the two LU decompositions can be obtained in parallel; and secondly, in each iteration the forward-backward substitutions for the 3 subsystems, the three components of the residue function $R_n(Y^{(j)})$, and in the case of {(2.7a),(2.7b),(2.8)} the similarity transformation can be computed in parallel.

Let us compare the iteration cost of diagonal and transformed EBDF with that of sequential MEBDF. Suppose that respectively m_1, m_2 and m_3 iterations are required. Then, sequential MEBDF requires 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 (2.6) 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 (2.6). Defining the iteration error $\varepsilon^{(j)} := Y^{(j)} - Y_{n+1}$, we derive for (2.6) the error recursion

$$\begin{array}{ll} \varepsilon^{(j)} = M\varepsilon^{(j-1)} + hL\Phi\left(\varepsilon^{(j-1)}\right), \ j \geq 1, & \\ M := \left(I - A^* \otimes hJ_{n+1}\right)^{-1}\left((A - A^*) \otimes hJ_{n+1}\right), \quad L := (I - A^* \otimes hJ_{n+1})^{-1}(A \otimes I), & (3.1) \\ \Phi(\varepsilon) := F(Y_{n+1} + \varepsilon) - F(Y_{n+1}) - (I \otimes J_{n+1})\varepsilon. & \end{array}$$

Hence,

$$\varepsilon^{(j)} = M^j\varepsilon^{(0)} + hM^{j-1}L\Phi(\varepsilon^{(0)}) + \cdots + hML\Phi(\varepsilon^{(j-2)}) + hL\Phi(\varepsilon^{(j-1)}). \qquad (3.2)$$

3.1 The rate of convergence

Let A^* be defined by (2.7a), so that A^* has the same diagonal entries as A. Then the 3-by-3 lower block-triangular matrix M has zero diagonal blocks, so that M^j vanishes for all $j \geq 3$. Thus,

$$\begin{array}{ll} \varepsilon^{(1)} = M\varepsilon^{(0)} + hL\Phi(\varepsilon^{(0)}), & \\ \varepsilon^{(2)} = M^2\varepsilon^{(0)} + hML\Phi(\varepsilon^{(0)})hL\Phi(\varepsilon^{(1)}), & (3.2') \\ \varepsilon^{(j)} = hM^2L\Phi(\varepsilon^{(j-3)}) + hML\Phi(\varepsilon^{(j-2)}) + hL\Phi(\varepsilon^{(j-1)}), \ j \geq 3. & \end{array}$$

Thus, in the case of *linear* problems, where the function Φ vanishes, we have convergence *within three iterations* for all matrices A^* with the same diagonal entries as A (of course, if $A^* = A$, then (2.6) reduces to modified Newton which converges within one iteration for linear problems). For *nonlinear* problems, we consider the first-order approximation to (3.2′). Let us write $\Phi(\varepsilon) = K\varepsilon + O(\varepsilon^2)$ and $hLK = N$, where K is the (3-by-3 block-diagonal) Jacobian matrix of $\Phi(\varepsilon)$ at $\varepsilon = 0$. By ignoring second and higher powers of ε, the first-order approximation to (3.2′) becomes

$$\begin{array}{ll} \varepsilon^{(1)} = (M + N)\varepsilon^{(0)}, \ N := hLK, & \\ \varepsilon^{(2)} = (M + N)^2\varepsilon^{(0)}, & (3.3) \\ \varepsilon^{(j)} = M^2N\varepsilon^{(j-3)} + MN\varepsilon^{(j-2)} + N\varepsilon^{(j-1)}, j \geq 3. & \end{array}$$

In the Newton case $A^* = A$, we have $M = O$, so that the first-order error recursion (3.3) reduces to $\varepsilon^{(j)} = N\varepsilon^{(j-1)}, \ j \geq 1$. However, if $M \neq O$, 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 (2.7a) and (3.1) it follows that M and N are 3-by-3 block matrices with the structure

$$M = (I - D \otimes hJ_n)^{-1}\begin{pmatrix} O & & \\ \bar{a}_1\bar{b}_0 \otimes hJ_n & O & \\ \times & (b1 - c2) \otimes hJ_n & O \end{pmatrix}, \quad N = \begin{pmatrix} \times & & \\ \times & \times & \\ \times & \times & \times \end{pmatrix}. \qquad (3.4)$$

Hence, all matrix products in (3.3) containing three or more factors M vanish, so that

$$\begin{array}{ll} \varepsilon^{(1)} = (M + N)\varepsilon^{(0)}, & \\ \varepsilon^{(2)} = (M^2 + MN + NM + N^2)\varepsilon(0), & \\ \varepsilon^{(3)} = (M^2N + MNM + NM^2 + MN^2 + NMN + N^2M + N^3)\varepsilon^{(0)}, & (3.3') \\ \varepsilon^{(4)} = (M^2N^2 + (MN)^2 + MN^2M + NM^2N + (NM)^2 + N^2M^2 + O(N^3))\varepsilon^{(0)}, & \\ \varepsilon^{(j)} = O(N^{j-2})\varepsilon^{(0)}, j \geq 5. & \end{array}$$

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

Theorem 3.1 *Let in the error recursion (3.1) the function Φ satisfy $\Phi(\varepsilon) = K\varepsilon + O(\varepsilon^2)$, and let $N := hLK$. Then the first-order approximation to the error recursion (3.2) is given by*

$$\begin{array}{l} \left\{\varepsilon^{(j)} = N\varepsilon^{(j-1)}, \ j \geq 1\right\} \ \textit{if } A^* = A \textit{ (modified Newton)}, \\ \left\{\varepsilon^{(j)} = (M + N)^j\varepsilon^{(0)}, \ j = 1, 2; \ \varepsilon^{(j)} = O(N^{j-2})\varepsilon^{(0)}, \ j \geq 3\right\} \ \textit{if } A^* \textit{ is defined by (2.7a).} \end{array}$$

Thus, if A^* is defined by (2.7a), then after at most two iterations the rate of convergence is comparable with that of 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 (3.4))

$$M = \begin{pmatrix} O & & \\ \times & O & \\ \times & O & O \end{pmatrix}.$$

Hence, all matrix products in (3.3′) containing factors M^p with $p \geq 2$ vanish. Furthermore, $MNM = (MN)^2 = (NM)^2 = MN^2M = O$. Using these relations, it can be shown that

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

Thus, we have proved:

Theorem 3.2 *Let the conditions of Theorem 3.1 be satisfied, and let in (2.7a) $c_2 = b_1$. Then, for all c_1 the first-order approximation to the error recursion (3.2) associated with transformed EBDF $\{$(2.7a),(2.7b),(2.8)$\}$ is given by*

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

3.2 Amplification factors

Theorems 3.1 and 3.2 show that transformed and diagonal EBDF may converge slower than 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

$$M^j(a \otimes v) = (Z^j(z) \otimes I)(a \otimes v), Z(z) := z(I - zA^*)^{-1}(A - A^*),\ z := h\lambda(J_{n+1}), \tag{3.5}$$

we are interested in the size of $\|Z^j(z)\|$. It follows from (2.7a) that

$$Z(z) = \frac{z}{(1-b_0 z)(1-\bar{b}_0 z)} \begin{pmatrix} 0 & 0 & 0 \\ \bar{a}_1\bar{b}_0(1-b_0 z) & 0 & 0 \\ (c_1 + c_2\bar{a}_1)\bar{b}_0 z - c_1 & (b_1 - c_2)(1-\bar{b}_0 z) & 0 \end{pmatrix}.$$

Assuming that the eigenvalues $\lambda(J_{n+1})$ are in the left halfplane, $\|Z^j(z)\|$ is maximal along the imaginary axis, so that we set $z = iy$. Let us assume that the parameters c_1 and c_2 are chosen such that $\|Z^j(iy)\|_\infty$ is determined by the second row of Z. We verified that this happens in the case of diagonal Newton where $c_1 = c_2 = 0$ and in the case of transformed Newton with $c_1 = 0$ or $c_1 = \bar{a}_1\bar{b}_0 b_1(\bar{b}_0 - b_0)^{-1}$ and with $c_2 = b_1$. For these cases we find

$$\|Z(iy)\|_\infty = \frac{\bar{a}_1\bar{b}_0|y|}{\sqrt{1+\bar{b}_0^2 y^2}},\ \|Z^2(iy)\|_\infty = \frac{\bar{a}_1\bar{b}_0|b_1 - c_2||y^2|}{\sqrt{1+b_0^2 y^2}\sqrt{1+b_0^2 y^2}}. \tag{3.6}$$

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 2.2), we should expect that the stiff components in the iteration error are amplified in the first iteration. However, in the second iteration, transformed EBDF already has zero amplification factors and diagonal EBDF has quite small amplification factors because $\bar{a}_1|b_1|b_0^{-1} \ll 1$. Figure 3.1 illustrates the behaviour of $\|Z(iy)\|_\infty$ and $\|Z^2(iy)\|_\infty$ as given by (3.6) for the 6th-order diagonal EBDF method ($k = 5$).

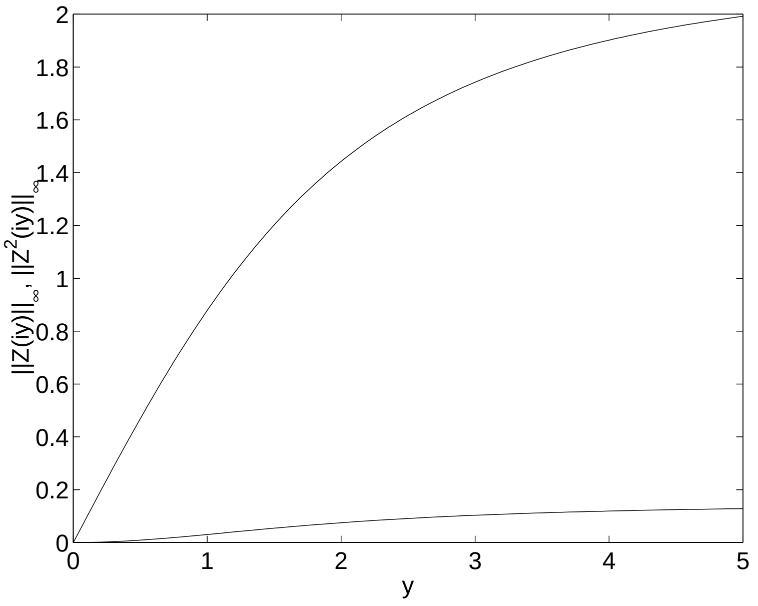

Figure 3.1: The amplification factors (3.6) for the 6th-order, iterated EBDF method ($k = 5$).

4. Numerical experiments

In this section, we compare the accuracy obtained by the sequential MEBDF method and the transformed and diagonal EBDF methods for *fixed* stepsizes and *fixed* number of iterations. The fixed-stepsize and fixed-number-of-iterations strategy is chosen in order to see the algorithmic properties separately from strategy effects. In the three approaches, we computed the initial iterates by taking the most recent approximation available 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 obtained 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). The methods were tested for $k = 5$. The starting values were obtained either from the exact solution if available or by applying the 5th-order Radau IIA method with a 5 times smaller stepsize. We took two well-known test problems from the literature such that there is no transient phase, allowing us to use fixed stepsizes; viz. a problem posed by Kaps[8].

$$\frac{dy_1}{dt} = -1002y_1 + 1000y_2^2,\ \frac{dy_2}{dt} = y_1 - y_2(1+y_2),\ y_1(0) = y_2(0) = 1,\ 0 \le t \le 5, \tag{4.1}$$

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

$$\text{HIRES on } [5, 321.8122], \tag{4.2}$$

where the initial conditions at $t = 5$ were obtained by integrating the HIRES problem given in ([6], 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

$$\begin{aligned} y_1' &= -1000(y_1^3 y_2^6 - \cos^3(t)\sin^6(t)) - \sin(t), & y_1(0) &= 1, \\ y_2' &= -1000(y_2^5 y_3^4 - \sin^5(t)\sin^4(t)) + \cos(t), & y_2(0) &= 0, \qquad 0 \le t \le 1 \\ y_3' &= -1000(y_1^2 y_3^3 - \cos^2(t)\sin^3(t)) + \cos(t), & y_3(0) &= 0, \end{aligned} \tag{4.3}$$

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

$$\begin{aligned} y_1' &= -0.04y_1 + 10^4 y_2 y_3 - 0.96e^{-t}, & y_1(0) &= 1, \\ y_2' &= 0.04y_1 - 10^4 y_2 y_3 - 10^7 (y_2)^2 - 0.04e^{-t}, & y_2(0) &= 0, \qquad 0 \le t \le t_{\text{end}}, \\ y_3' &= 3\,10^7 (y_2)^2 + e^{-t}, & y_3(0) &= 0, \end{aligned} \tag{4.4}$$

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[11], but it is modified by adding nonhomogeneous terms, so that it possesses for the given initial values a solution without transient phase. System (4.4) 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* righthand 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 (4.2) where no exact solution is available, the starting values were provided by the Radau IIA method using 10 iterations. Tables 4.1 and 4.2 list for given values of m and N the resulting *scd*-values for the problems (4.1) and (4.2). These results show that in almost all cases sequential MEBDF finds the solution in one 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 (4.2) due to the relatively large timesteps 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 stepsize in Table 4.2 (i.e. $h \approx 7.9$) does the diagonal EBDF(2) method converge. As to the order behaviour, in the Kaps problem the order $p = 6$ of the methods is shown, but the HIRES problem only shows their stage order $s = 5$.

In order to see more clearly the differences in convergence rates, we now integrate the highly nonlinear problem (4.3). Surprisingly, the numbers of iterations to reach the converged solution is more or less comparable for all methods and differ 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 $scd = 14.3$). Finally, we integrate the highly stiff modified Robertson problem (4.4). Here, the performance is similar to that in the Kaps problem (4.1). Apparently, the methods are able to compute positive approximations to the second component $y_2(t)$.

Table 4.1: Values of *scd* for problem (4.1).

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

Table 4.2: Values of *scd* for problem (4.2).

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

4.2 Variable number of iterations

If the number of iterations is adjusted to each nonlinear 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 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 ([6], p. 130). This stopping strategy depends on a given tolerance parameter *Tol*, because it presupposes the use of automatic stepsize selection based on keeping the local truncation error *LTE* close to *Tol*. Since we focus on convergence aspects we want to use fixed stepsizes, so that we have to replace *Tol* by some

Table 4.3: Values of scd for problem (4.3).

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 4.4: Values of scd for problem (4.4) on $[0, 1]$.

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

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 damping parameter θ_m and the accumulated damping parameter η_m:

$$\theta_m := \frac{\|Y^{(m)} - Y^{(m-1)}\|_\infty}{\|Y^{(m-1)} - Y^{(m-2)}\|_\infty},\ \eta_0 := (\eta_{\text{old}})^{0.8}, \eta_m := \frac{\theta_m}{1 - \theta_m}, m \geq 1, \tag{4.5a}$$

where η_{old} equals the η_m from the preceding step (bounded below by the machine precision). Then, the stopping criterion described in [6] yields for the number of iterations m the condition

$$\eta_m \|Y^{(m)} - Y^{(m-1)}\|_\infty \leq \kappa \|u_n - y_n\|_\infty. \tag{4.6b}$$

Here, κ is a control parameter. The implicit relations are solved more accurately as κ is smaller. For the problems (4.1), (4.2), (4.3) and (4.4), 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. $m = 5$, $m = 10$, $m = 20$ and $m = 10$, respectively. In problem (4.2), where no exact solution is available, we used the Radau starting method with $m = 10$. Tables 4.5–4.9 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 stepsize is adjusted to the required accuracy. On the basis of the above results, we can derive theoretical speedup factors for the efficiency of the iteration part of the methods. Table 4.10 presents such efficiency speedup factors by comparing M-values (averaged over the *scd*-values) for sequential MEBDF and diagonal EBDF(2).

Table 4.5: Values of M for problem (4.1) with $\kappa = 0.1$.

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 4.6: Values of M for problem (4.2) with $\kappa = 0.1$.

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

Table 4.7: Values of M for problem (4.3) with $\kappa = 0.1$.

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 4.8: Values of M for problem (4.4) on $[0, 1]$ with $\kappa = 0.1$.

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 4.9: Values of M for problem (4.4) on $[0, 10]$ with $\kappa = 0.1$.

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

Table 4.10: Theoretical iteration speedup of diagonal EBDF(2).

Problem	Speedup
(4.1)	1.3
(4.2)	1.7
(4.3)	2.7
(4.4)	2.2

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 shared memory machine, in this case a Cray C916. Parallel speedups in this section were obtained using the Autotasking Expert analysis tool [4] available on Cray computer systems, which estimates the speedup that would be obtained by a program run on a dedicated multiprocessor system, based on the observed performance on an arbitrarily loaded system. Since the ATExpert tool measures speedup with respect to the same code run on a single processor, it is important for obtaining 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 timestep 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 startup procedure.

There is, of course, a certain amount of parallelism available in sequential MEBDF. For each of the three relations in $\{(2.2a),(2.2c)\}$, a nonlinear 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 righthand 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.

These tasks all contain a number of independent operations which is proportional to the problem dimension d (*parallelism across the space*, in the classification of Gear[5]) 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 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 diagonal EBDF.

4.3.1 LU-decompositions. Since the computation of LU-decompositions are 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 4.11 shows for $N = 1280$ time steps the speedup 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 (4.1)–(4.4), the parallel computation of the LU decompositions does not lead to a substantial speedup, even for the 8-dimensional HIRES problem (4.2). Of course, for higher-dimensional problems, the speedup will *increase*. On the other hand, a more sophisticated implementation, where the Jacobian is only updated every few steps, will *decrease* the speedup attained by concurrent decomposition of Jacobians. Therefore, a substantial speedup of a parallel implementation of diagonal EBDF should not be expected from the parallel computation of the LU-decompositions alone.

Table 4.11: Speedups attained by concurrent decomposition of Jacobians in diagonal EBDF.

Problem	$m = 2$	$m = 3$	$m = 4$	$m = 5$
(4.1)	0.97	0.97	1.00	1.00
(4.2)	1.18	1.14	1.11	1.10
(4.3)	1.03	1.02	1.02	1.02
(4.4)	1.04	1.03	1.03	1.02

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 (2.6), whereas these are constant components of the residue functions if the subsystems are solved in sequence. 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 {(2.2a),(2.2b)} are solved sequentially. An indication of the significance of this overhead is provided in Table 4.12, 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.

Table 4.12: Ratio of serial CPU times for sequential and diagonal Newton.

Problem	$m = 2$	$m = 3$	$m = 4$	$m = 5$
(4.1)	0.94	0.91	0.89	0.89
(4.2)	0.98	0.97	0.97	0.96
(4.3)	1.01	0.99	0.97	0.97
(4.4)	0.98	0.96	0.95	0.94

4.3.3 Overall speedup factors. Table 4.13 shows the ATExpert observed speedup 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 speedup is essentially independent of the number of Newton iterations. In the table we have also listed the dimension of each system (the nonautonomous terms of problems (4.3) and (4.4) have been implemented as an extra dimension). The attainable speedup 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 speedup factors in Table 4.13 are also an indication of the speedup of *efficiency* (that is, CPU speedup under the condition of equal accuracies).

Table 4.13: Speedup of diagonal EBDF on 3 processors.

Problem	d	$m = 5$
(4.1)	2	1.8
(4.2)	8	2.3
(4.3)	4	2.0
(4.4)	4	2.0

References

1. P.N. Brown, A.C. Hindmarsh, and G.D. Byrne. VODE: A variable coefficient ODE solver. Available at `http://www.netlib.org./ode/vode.f`, 1992.
2. J.R. Cash. On the integration of stiff ODEs using extended backward differentiation formulae. *Numer. Math.*, 34:235–246, 1980.
3. J.R. Cash. The integration of stiff initial value problems in ODEs using modified extended backward differentiation formulae. *Comput. Math. Appl.*, 5:645–657, 1983.
4. Cray Research Inc. *CF77 Commands and directives, SR-3771*, 6.0 edition, 1994.
5. C.W. Gear. Massive parallelism across time in ODEs. *Appl. Numer. Math.*, 11:27–44, 1993. Proceedings of the International Conference on Parallel Methods for Ordinary Differential Equations, Grado (It), Sept. 10–13, 1991.
6. E. Hairer and G. Wanner. *Solving ordinary differential equations, II. Stiff and differential-algebraic problems.* Springer-Verlag, Berlin, 1991.
7. E. Hairer and G. Wanner. RADAU. Available at `ftp://ftp.unige.ch/pub/doc/math/stiff/radau.f`, 1998.
8. P. Kaps. Rosenbrock-type methods. In G. Dahlquist and R. Jeltsch, editors, *Numerical methods for stiff initial value problems, Bericht nr. 9.* Inst. für Geometrie und Praktische Mathematik der RWTH Aachen, 1981.
9. W.M. Lioen and J.J.B. de Swart. Test set for IVP solvers, Release 2.0. Available at `http://www.cwi.nl/cwi/projects/IVPtestset/`, 1998.
10. L.R. Petzold. DASSL: A differential/algebraic system solver. Available at `http://www.netlib.org/ode/ddassl.f`, 1991.
11. H.H. Robertson. The solution of a set of reaction rate equations. In J. Walsh, editor, *Numerical Analysis, an Introduction*, pages 178–182. Academ. Press, 1966.