Journal of Computational and Applied Mathematics 125 (2000) 117–130

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

Modified extended backward differentiation formulae for the numerical solution of stiff initial value problems in ODEs and DAEs

J.R. Cash

Department of Mathematics, Imperial College of Science, South Kensington, London SW7 2BZ, UK

Received 20 August 1999; received in revised form 24 November 1999

Abstract

For many years the methods of choice for the numerical solution of stiff initial value problems and certain classes of differential algebraic equations have been the well-known backward differentiation formulae (BDF). More recently, however, new classes of formulae which can offer some important advantages over BDF have emerged. In particular, some recent large-scale independent comparisons have indicated that modified extended backward differentiation formulae (MEBDF) are particularly efficient for general stiff initial value problems and for linearly implicit DAEs with index $\leqslant 3$. In the present paper we survey some of the more important theory associated with these formulae, discuss some of the practical applications where they are particularly effective, e.g., in the solution of damped highly oscillatory problems, and describe some significant recent extensions to the applicability of MEBDF codes. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Stiff differential equations; Differential algebraic equations; MEBDF

1. Introduction

In the 1950s Curtiss and Hirschfelder [12] published one of the first papers which identified clearly the difficulties of solving stiff initial value problems of the form

$$\boldsymbol{y}' = \boldsymbol{f}(x, \boldsymbol{y}), \quad \boldsymbol{y}(x_0) = \boldsymbol{y}_0, \quad \boldsymbol{y} \in \mathbb{R}^s. \tag{1}$$

Since that time a whole variety of methods have been proposed for the numerical solution of (1). The fact that this class of problems has remained so challenging is not at all surprising given the fact

E-mail address: j.cash@ic.ac.uk (J.R. Cash).

that it is still not clear exactly what is meant by the term stiffness. Although numerous attempts have been made to give a rigorous definition of this concept, it is probably fair to say that none of these definitions is entirely satisfactory. Indeed the authoritative book of Hairer and Wanner [18] deliberately avoids trying to define stiffness and relies instead on an entirely pragmatic definition given in [12]. What is clear, however, is that numerical methods for solving stiff initial value problems have to satisfy much more stringent stability requirements than is the case for methods intended for nonstiff problems. One of the first, and still one of the most important, stability requirements particularly for linear multistep methods is that of A-stability which was proposed in [13]. However, the requirement of A-stability puts a severe limitation on the choice of suitable linear multistep methods. This is articulated in the so-called Dahlquist second barrier which says, among other things, that the order of an A-stable linear multistep method must be $\leqslant 2$ and that an A-stable linear multistep method must be implicit.

This pessimistic result has encouraged researchers to seek other classes of numerical methods for solving stiff equations. For Runge–Kutta methods, for example, the situation regarding stability is much more satisfactory. In fact, there exist A-stable Runge–Kutta methods of arbitrarily high order. In particular, the s-stage Gauss Runge–Kutta methods have order $2s$ and are A-stable for all s. However, as is well known, these fully implicit Runge–Kutta methods can be very expensive to implement.

We are therefore faced with the classic dilemma that, generally speaking, linear multistep methods are relatively cheap to implement but suffer severe degradation of stability as their order increases while implicit Runge–Kutta methods can have excellent stability properties but tend to be expensive to implement. Most attempts to ‘get around’ the Dahlquist barrier have involved either lessening the requirement of A-stability to something which is less restrictive, but which is still appropriate for some classes of stiff equations, or proposing a totally different class of formulae. However, the desirability of having methods which are A-stable is sufficiently well documented that it seems the only way to achieve real efficiency for general stiff problems is to consider formulae other than linear multistep methods.

One very successful proposal in this direction was by Hairer and Wanner [18] who developed a code Radau5 based on Radau Runge–Kutta formulae. We will return to this code at a later stage. A second proposal that we wish to describe is to use what have become known as boundary value methods. These methods are able to achieve excellent stability by using information at advanced step points (also known in the literature as superfuture points). As with Runge–Kutta methods it is the efficiency of implementation of these methods rather than their stability which is the real challenge and we will discuss this in the next section.

2. Boundary value methods

To introduce the general class of boundary value methods that we will be interested in we consider again the linear multistep method

$$y_{n+k} + \sum_{j=0}^{k-1} \bar{a}_j y_{n+j} = h\bar{b}_k f(x_{n+k},\ y_{n+k}). \tag{2}$$

In the limit $h=0$ this formula reduces to the linear recurrence relation

$$y_{n+k}+\sum_{j=0}^{k-1}\bar{a}_j y_{n+j}=0. \tag{3}$$

Eq. (3) can be regarded as a linear multistep method ‘integrating forward’ with a step $h=0$. It is clear that the required solution of (3) is

$$y_n=y_{n+1}=\cdots=y_{n+s}=c. \tag{4}$$

If, however, we appeal to the theory of linear recurrence relations it is well known that if we solve (3) by direct forward recurrence starting with the initial conditions

$$y_{n+i}=c, \quad 0\leqslant i\leqslant k-1 \tag{5}$$

in an attempt to compute the solution

$$y_{n+m}=c, \quad m>k-1, \tag{6}$$

then this process is stable if and only if

(i) $r=1$ is the root of largest modulus of

$$\sum_{j=0}^{k}\bar{a}_j r^j=0 \tag{7}$$

and

(ii) all roots of (7) of unit modulus are distinct.

If these conditions are not satisfied then forward recurrence is unstable. In the parlance of the theory of linear recurrence relations, requirements (i) and (ii) simply impose the condition that $r=1$ is the dominant zero of (7) so that $y_n=c$, for all n, is the dominant solution of (3). In essence the theory tells us that only the dominant solution of (3) can be generated in a stable manner by forward recurrence. However, conditions (i) and (ii) are precisely the conditions for (2) to be zero-stable. Thus, an alternative way of looking at this is to realize that we have to impose the condition of zero-stability on (2) precisely because we demand that we should solve (2) by forward recurrence; that is we solve for y_{n+k} from given values $y_n, y_{n+1}, y_{n+2}, \ldots, y_{n+k-1}$. If we were to interpret (2) not as a prescription for y_{n+k} but as an equation which, if satisfied for $k=0,1,2,\ldots,$ determines the sequence of approximations we are interested in then the relevant question becomes how we solve the resulting simultaneous equations in a stable way without excessive cost. If we were to solve (2) in a different way then we would no longer need to impose the condition of zero-stability and this in turn offers us the possibility of obtaining high order A-stable formulae.

One possible alternative way of solving (3) is to rewrite it as a boundary value problem. (This is the basis of some celebrated algorithms for finding nondominant solutions of linear recurrence relations [23,22,8,9].) To describe one variant of this approach we consider the third order, linear 2 step method:

$$y_{n+2}+4y_{n+1}-5y_n=h(4f_{n+1}+2f_n). \tag{8}$$

It is well known that this formula does not satisfy the condition of zero-stability and so is unsuitable for the solution of initial value problems. It was shown in [10] that if we apply (8) to the linear scalar equation

$$\frac{\mathrm{d}y}{\mathrm{d}x} = \lambda y, \quad \lambda \in \mathbb{R} \tag{9}$$

to give

$$y_{n+2} + (4 - 4\lambda h)y_{n+1} - (5 + 2h\lambda)y_n = 0, \tag{10}$$

then the required solution of (10) is subdominant for all $h\lambda$. This fact suggests that, instead of solving (10) by forward recurrence starting from two initial conditions on y which would be an unstable process, we should instead generate the required solution using the boundary value formulation

$$\begin{aligned} &y_0 = y(x_0),\\ &y_{n+2} + 4y_{n+1} - 5y_n = h(4f_{n+1} + 2f_n), \quad n = 0, 1, 2, \ldots, N-2,\\ &y_N = 0 \end{aligned} \tag{11}$$

for some large N. Note, in particular, that this defines a tridiagonal system of linear algebraic equations of size $N+1$ for the $N+1$ unknowns. It was shown in [10] that this formulation produces an A-stable algorithm which has order 3. Theoretically, this approach is a very successful one due to the high-order A-stability that we have achieved. In fact, we have used a linear multistep method and achieved A-stability with order > 2. However computationally this algorithm is not, in general, satisfactory for solving initial value problems since it does not allow easy change of stepsize or order and for large systems the storage requirement can be prohibitive. Even more important is the problem that there may be a lack of convergence in the solution of the simultaneous nonlinear algebraic equations. Ideally, what we need is a special kind of boundary value approach which shares the improved stability obtained by (11) but which does allow variable stepsize and order. One of the easiest ways of achieving this is to develop special classes of formulae to which a boundary value approach can be applied.

An early attempt in this direction was in [9] where such a method was derived from a standard Adams–Moulton formula. This early work has since been extended in several ways. In particular, Brugnano and Trigiante [2] have developed a whole theory of boundary value methods suitable for several important classes of initial value problems, such as stiff problems, Hamiltonian problems and differential algebraic equations. However, a major difference is that we set up a 'local' boundary value problem so that when computing y_{n+k} the boundary condition is imposed at x_{n+k+1}. The approach of Brugnano and Trigiante can be regarded as a more conventional one where the boundary condition is imposed at a large distance from the initial point. Their results are too extensive to quote here and we refer the reader to [2].

In what follows, we will extend the approach suggested by (11) to formulae which are particularly suitable for stiff problems. There are numerous ways in which this could be done and in the next section we will describe one particular approach which is based on modified extended backward differentiation formulae and which has proved to be particularly efficient for general stiff initial value problems and differential algebraic equations.

3. Modified extended backward differentiation formulae

Modified extended backward differentiation formulae (MEBDF) were originally proposed as a class of formulae to which an efficient variable order, variable step boundary value approach could easily be applied. The precise form taken by the general k step MEBDF is

$$y_{n+k} + \sum_{j=0}^{k-1} \hat{a}_j y_{n+j} = h[\hat{b}_{k+1} f_{n+k+1} + \hat{b}_k f_{n+k}], \tag{12}$$

where the coefficients are chosen so that this formula has order $k+1$. This order requirement uniquely specifies the coefficients of (12). Starting from given data $y_n, y_{n+1}, \ldots, y_{n+k-1}$, a predictor is first used to predict y_{n+k+1}, the derivative approximation y'_{n+k+1} is then computed and finally y_{n+k} is computed from $y_n, y_{n+1}, \ldots, y_{n+k-1}, y'_{n+k+1}$. Of course, the accuracy and stability of this method is critically dependent on the predictor used to compute y_{n+k+1} and in particular this predictor must be of order at least k if the whole process is to be of order $k+1$. A natural kth-order predictor is the k-step BDF and this leads to the so-called EBDF algorithm

Stage 1: Use a standard BDF to compute $\bar{y}_{n+k}$:

$$\bar{y}_{n+k} + \sum_{j=0}^{k-1} \bar{a}_j y_{n+j} = h\bar{b}_k f(x_{n+k}, \bar{y}_{n+k}). \tag{13}$$

Stage 2: Use a standard BDF to compute $\bar{y}_{n+k+1}$:

$$\bar{y}_{n+k+1} + \bar{a}_{k-1}\bar{y}_{n+k} + \sum_{j=0}^{k-2} \bar{a}_j y_{n+j+1} = h\bar{b}_k f(x_{n+k+1}, \bar{y}_{n+k+1}). \tag{14}$$

Stage 3: Compute a corrected solution of order $k+1$ at x_{n+k} using

$$y_{n+k} + \sum_{j=0}^{k-1} \hat{a}_j y_{n+j} = h[\hat{b}_{k+1} \bar{f}_{n+k+1} + \hat{b}_k f_{n+k}]. \tag{15}$$

Note that at each of these three stages a nonlinear set of equations must be solved in order that the desired approximations can be computed. The boundary value nature of this approach can be seen from Stage 3 where y_{n+k} is computed from past values y_{n+i} as well as from the future value $\bar{y}_{n+k+1}$.

One of the main drawbacks of this approach is the need to compute and factorize the two iteration matrices arising in the application of a modified Newton iteration at each stage. To avoid this, the EBDF approach described above can be modified (to give the so-called MEBDF approach [5]) by changing Stage 3 to

Stage 3*:

$$y_{n+k} + \sum_{j=0}^{k-1} \hat{a}_j y_{n+j} = h[\hat{b}_{k+1} \bar{f}_{n+k+1} + \bar{b}_k f_{n+k} + (\hat{b}_k - \bar{b}_k)\bar{f}_{n+k}]. \tag{16}$$

Fortunately, this modification not only improves the computational efficiency of this approach, it also improves its stability. The stability properties of the MEBDF together with the reasons for their computational efficiency are fully described in [6,18] and a code MEBDFDAE based on the MEBDF approach is available from NETLIB and from the author's web page. In particular, the MEBDF are

A-stable for order up to 4 and $A(\alpha)$-stable for order up to and including 9 and this is considerably better than the stability achieved by BDF. In the next section we will consider one particular class of problems for which this enhanced stability is particularly appropriate.

4. Damped stiff highly oscillatory problems

As was explained earlier, attempts to give a precise general mathematical definition of stiffness have been largely unsuccessful. However, some important insights into the concept of stiffness can be gained by considering a suitably restricted class of problems. Normally, the aim of considering these problems has been to define a new stability concept which is appropriate for dealing with stiff problems rather than to define stiffness per se. Often these new definitions have evolved from consideration of problems for which there is a contractivity property for the solutions and, for a survey of this, the interested reader is referred to [3]. The most straightforward problem to analyse for linear methods is the constant coefficient equation

$$\frac{\mathrm{d}y}{\mathrm{d}x} = Ay. \tag{17}$$

A definition of stiffness for this problem has been given, for example, in [21]. This definition is not the whole story, however, because other quantities such as the initial conditions and the interval of integration also need to be considered. However if we assume for the time being that all components of the general solution of (17) are in the solution that we require, and that we are integrating over a sufficiently long interval of x, then we can say something about the likely performance of standard codes on (17). In particular if all the eigenvalues of A are real, or lie close to the real axis, then we can expect codes based on BDF and MEBDF to normally perform well since they have linear stability properties appropriate for dealing with such problems. However, if some of the eigenvalues of A have relatively large imaginary part then we would expect BDF to perform rather poorly since they are A-stable only up to order 2. In the case where there exist eigenvalues lying close to or on the imaginary axis whose imaginary part is large in modulus then there are two distinct classes of problems that we need to distinguish between rather carefully. The first is where the large eigenvalues are purely imaginary so that there is no damping in the components of the solution corresponding to these eigenvalues. In this case it is necessary to follow the oscillations exactly and we would expect nonstiff methods to be more efficient than (implicit) stiff methods for this problem. However if the large eigenvalues lie close to the imaginary axis but not on it, so that the rapid oscillations are damped, then it is only necessary to follow them for a short time and in this case highly stable implicit methods normally perform very well. We can conveniently characterize highly oscillatory problems of the form (17) as ones where the eigenvalues λ_j of A satisfy

$$\lambda_j = \mu_j + \mathrm{i}v_j, \tag{18}$$

where $\mu_j < 0$ for all j, $\max_{1\leqslant j\leqslant n}|\mu_j| \gg \min_{1\leqslant j\leqslant n}|\mu_j|$ and $|\mu_j| \ll |v_j|$ for at least one pair of eigenvalues of large modulus. In their famous DETEST test set [15], Enright et al. devised a problem denoted by $B5$ for which one component of the solution is of the form

$$\exp(-10x)(A\cos\omega x + B\sin\omega x) \quad \text{where } \omega = 50. \tag{19}$$

This problem was specially designed to trap BDF-based methods (and succeeded in doing so!). However, it is important to realize that the performance of BDF codes on these problems can be very different depending on which code is used. For example, when faced with stiff oscillatory problems, LSODE [20] will often use high-order methods with small stepsizes rather than correctly reducing the order to allow the possibility of large stepsize increases. However, the BDF code DASSL is geared towards selecting lower order and this code will often reduce the order to 2 (so that A-stable formulae are being selected) when faced with such problems [1]. We feel that this strategy of DASSL of being biased towards lower order is an extremely important one and the incorporation of this strategy into the MEBDF code MEBDFDAE has had a strong positive influence on its performance. This strategy is fully described in [11]. For a long time these stiff highly oscillatory problems were regarded as being rather intractable mainly because of the relatively poor performance of BDF. Indeed writing in 1984 Gaffney [16] concluded that none of the currently available codes was satisfactory for dealing with these problems. However, recently some excellent codes have become available and this allows these problems to be solved very routinely. In particular, we mention the codes MEBDFDAE, Radau5 and DESI [4] which all have excellent stability properties and can be recommended for these problems. For a survey of the performance of these codes on the highly oscillatory problem the reader is referred to [11].

5. Extensions to the MEBDF approach

In the particular formulation of MEBDF that was considered in Section 3 we set up the boundary value approach using just one superfuture point. The main theoretical result that is indicated by numerical experiment for this particular algorithm is that it is A-stable for order up to and including 4. It would, of course, be valuable to have a proof of this result. It is possible to develop this approach in various directions. For example, we could use more super future points. In particular, it is of interest to see what can be achieved by using two superfuture points. The natural way to define such an algorithm would be to have three predictor steps based on BDF as described in Section 3 and then to apply a corrector of the general form

$$y_{n+k} + \sum_{j=0}^{k-1} \hat{a}_j y_{n+j} = h[\hat{b}_{k+2}\bar{f}_{n+k+2} + \hat{b}_{k+1}\bar{f}_{n+k+1} + \hat{b}_k f_{n+k}]. \tag{20}$$

Here the coefficients are chosen so that the corrector has order $k+1$ and this defines a one parameter family of coefficients for (20). The stability properties of this approach were investigated in detail in [24]. He found that by using this approach it is possible to find A-stable formulae with order up to and including 6. However, rather disappointingly, it is not possible to achieve this stability and still retain the property that only one iteration matrix needs to be factorized. There is, however, a very important theoretical result that comes out of this investigation. This concerns the conjecture that a well-known result of Norsett and Wolfbrandt [25] for one-step methods carries over to the multistep case. Basically, this conjecture is that the order p of an A-stable general linear method whose characteristic function has s poles, all of which are real, satisfies $p \leqslant s+1$. Remarkably, the order 6 MEBDF, with two advanced step points, has 4 real poles and $p=6$. This serves as a rather surprising counterexample to this conjecture. For more details on this the reader is referred to [26].

If we summarize what can be achieved in the way of stability using superfuture points we note that

(1) For linear multistep methods with no superfuture points we have that A-stability implies the order is $\leqslant 2$.
(2) With one superfuture point we have that A-stability implies the order is $\leqslant 4$.
(3) With two superfuture points we have that A-stability implies the order is $\leqslant 6$

It is tempting to conjecture that with k superfuture points we have that A-stability implies that the order p satisfies $p \leqslant 2k+2$. This would be an interesting and important result but, based on the difficulty of finding A-stable methods of order 6 with $k=2$, we expect this conjecture to be false although a proof of this is elusive.

Due to the fact that when using 2 superfuture points we need to factorize two iteration matrices in order to obtain A-stability with order 6 it seems unlikely that the 'two superfuture points' approach will be competitive with the standard MEBDF for the solution of general stiff problems. It may however have a role to play in the highly oscillatory case where high accuracy is requested. However for parallel implementation the situation is quite different. There are many ways in which the MEBDF approach can be parallelized and, in particular, in a parallel environment the need to factorize two iteration matrices is no longer a problem. One possible way of deriving a parallel MEBDF code was investigated in [24]. He developed an approach whereby all predicted solutions can be computed simultaneously and he showed that there is a significant gain in efficiency using this approach. A different and rather ingenious method of parallelization was proposed in [27]. He modified the EBDF approach with two superfuture points to obtain new classes of formulae which are immediately parallelizable. His results indicate that he is able to achieve significant speed ups using this approach and it seems likely that this will be one of the most effective of all parallel algorithms for the solution of general stiff initial value problems.

The second extension we wish to consider in this section is where extra derivative terms are introduced. We illustrate this by considering the one-step case which is of order 2. Here the standard MEBDF is replaced by the three stages:

Stage 1:

$$\bar{y}_{n+1} = y_n + h[\theta f(x_{n+1}, \bar{y}_{n+1}) + (1-\theta) f(x_n, y_n)]. \tag{21}$$

Stage 2:

$$\bar{y}_{n+2} = \bar{y}_{n+1} + h[\theta f(x_{n+2}, \bar{y}_{n+2}) + (1-\theta) f(x_{n+1}, \bar{y}_{n+1})]. \tag{22}$$

Stage 3:

$$\begin{aligned} y_{n+1} = y_n + h[&(c - \tfrac{1}{2}) f(x_{n+2}, \bar{y}_{n+2}) + (\tfrac{3}{2} - 2c - \theta) f(x_{n+1}, \bar{y}_{n+1}) \\ &+ \theta f(x_{n+1}, y_{n+1}) + c f(x_n, y_n)]. \end{aligned}$$

Note that the standard MEBDF is of this form with $\theta = 1$, $c = 0$. Applying these three stages to the standard scalar test equation $y' = \lambda y$ we obtain an expression of the form

$$\frac{y_{n+1}}{y_n} = R(q), \quad q = \lambda h. \tag{23}$$

In order to get the correct asymptotic behaviour, that is

$$\lim_{q \to -\infty} R(q) = 0, \tag{24}$$

we require

$$(c - \tfrac{1}{2})(1-\theta)^2 - (\tfrac{3}{2} - 2c - \theta)(1-\theta)\theta + c\theta^2 = 0. \tag{25}$$

This defines c in terms of θ and leaves θ as a free parameter to improve the accuracy and/or stability of the method. In general, for a k-step formulation, we will again have two free parameters, one of which will be used to give the correct asymptotic behaviour and the other will be used to improve stability. Research is at present in progress to see by how much this approach improves stability and, in particular, whether it is possible to obtain A-stability with $k=4$. However as k increases the situation becomes very complicated since there are many ways in which the extra derivative terms can be added. What is really needed is some theory linking the order and stability of these methods to the step number k.

6. Differential algebraic equations

One of the important properties of MEBDF is that, in common with BDF, they can be extended to the solution of differential algebraic equations in a straightforward way. The extension to linearly implicit DAEs is the most natural and we consider this first of all. Many important classes of differential algebraic equations can be written in the linearly implicit form

$$M\frac{\mathrm{d}y}{\mathrm{d}x} = f(x,y), \tag{26}$$

where the coefficient matrix M is singular. In particular, the constrained system

$$\frac{\mathrm{d}y}{\mathrm{d}x} = F(x,y,z), \quad 0 = g(y,z) \tag{27}$$

can be rewritten as

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} y' \\ z' \end{pmatrix} = \begin{pmatrix} F(x,y,z) \\ g(y,z) \end{pmatrix}, \tag{28}$$

which is of the form (26). The MEBDF approach described in the previous sections is very straightforward to extend for (26), from ODEs to linearly implicit DAEs. This can be done simply by using the algorithm of Section 3 and being careful to arrange the computation so that we never call for the inverse of the singular matrix M. The one-step MEBDF, for example, would be expressed in a completely analogous way to the one-step BDF as

$$M(y_{n+1} - y_n) = h(-\tfrac{1}{2}f(x_{n+2}, y_{n+2}) + \tfrac{3}{2}f(x_{n+1}, y_{n+1})). \tag{29}$$

An important concept when dealing with DAEs is that of the index. Perhaps the most widely used definitions are of differentiation index and perturbation index. In particular, the differentiation index is i, if i is the minimum number of analytic differentiations that need to be performed on the system to allow an explicit ODE to be extracted. For more on this important concept the reader is referred to [18, p. 445]. The major change from the ODE case is the way in which the errors (i.e., both the local truncation error and the error in the Newton iteration) are controlled. Following the approach described in [19], the error E that we control when using a steplength h is defined as

$$E = \mathrm{Error}_{\mathrm{Index1}} + h\,\mathrm{Error}_{\mathrm{Index2}} + h^2\,\mathrm{Error}_{\mathrm{Index3}}. \tag{30}$$

Here we use the obvious notation that, for example, $\text{Error}_{\text{Index1}}$ is the error in the index 1 variables [19, p. 124]. As explained in [19] this approach is needed essentially to deal with the near singularity (for small h) of the iteration matrix. Numerical results presented on the author's web page [7] indicate the good performance of MEBDFDAE on a variety of linearly implicit test problems of indices 1–3 and this has recently been confirmed by some extensive independent comparisons [14].

It would perhaps be valuable to extend the MEBDF approach to more general equations such as the fully implicit equation

$$F(x,y,y') = 0. \tag{31}$$

This problem could be solved by rewriting (31) in the linearly implicit form

$$y' = z, \quad F(x,y,z) = 0, \tag{32}$$

but at the cost of increasing the size and more importantly the index of the system since one more analytic differentiation now has to be performed to allow an explicit ODE to be extracted.

Another problem we may wish to deal with directly is

$$C(y)\frac{\mathrm{d}y}{\mathrm{d}x} = g(y), \tag{33}$$

where $C(y)$ is singular. This can be rewritten in the form (28) where the left-hand side of this equation remains the same and the right-hand side is now

$$f(x,y) = \begin{pmatrix} z \\ C(y)z - g(y) \end{pmatrix},$$

providing that the matrix $C(y)$ satisfies some rather general conditions [18, p. 445]. One way of dealing with (33) is by adding extra variables so making it again of the form (26). If the linear algebra is carried out in a careful way [18, p. 576] then the computational effort is increased by relatively little. However, users may not be prepared to change their problems to fit into a more restrictive framework and a more satisfactory approach may be to develop MEBDFDAE to deal directly with these more general equations. The necessary theory to allow this is relatively straightforward to develop and MEBDF codes for the direct solution of (33) and (31) are now available on the author's web page.

A second major problem concerning MEBDF follows from the well-known phenomenon of order reduction. There are at present no theoretical results concerning the order of MEBDF when applied to DAEs. However, the numerical results that have been obtained are highly suggestive. This leads us to make the following conjecture:

A $(p-1)$th step MEBDF when applied to a DAE of index i has order $p+1-i$. The MEBDF code is implemented on the assumption that this is indeed the correct behaviour and it is a serious gap in the theory that we do not have a proof of this result. Finally, we note that this order reduction is particularly serious when dealing with damped highly oscillatory problems of index 3 in the case where it is not required to follow the oscillations. If our conjecture is correct, and the order is indeed reduced by 2, then we in effect have A-stability only up to order 2. In this case the extensions described in the previous section where either two superfuture points are used or possibly where extra derivatives are used will be potentially very important.

7. Numerical results

In this section we will present some numerical results to illustrate the performance of the code MEBDFDAE. The drivers used to obtain these results are available on the web page of the author [7]. The example we consider is that of a simple constrained mechanical system, namely the pendulum. This is a particularly nice example since it is straightforward to derive systems of indices 1, 2 and 3 which describe the equations of motion. In what follows, we will consider the equations of motion of a point mass, m, at the end of a massless rod of length 1 oscillating under the influence of gravity g in the cartesian coordinates (p,q). The first four equations of motion are

$$\begin{aligned} p' &= u, \\ q' &= v, \\ mu' &= -p\lambda, \\ mv' &= -q\lambda - g. \end{aligned} \tag{34}$$

Here u and v are velocities and λ is the rod tension. The fifth equation which completes index 3 formulation is

$$0 = p^2 + q^2 - l^2. \tag{35}$$

To obtain index 2 formulation we differentiate constraint (35) to obtain

$$0 = pu + qv. \tag{36}$$

Eqs. (34)–(36) give index 2 formulation. If we differentiate (36) again we obtain

$$0 = m(u^2 + v^2) - qg - l^2\lambda. \tag{37}$$

Eqs. (34) together with (37) give index 1 formulation. We note that, starting from the original index 3 formulation, there are several ways of rewriting the constraint to reduce the index. In particular, the process of differentiating the constraint may result in the original constraint not being satisfied. This 'drift off' phenomenon is described for example in [18, p. 468; 19, p. 7]. In an attempt to avoid this problem Gear et al. [17] proposed adding in the original constraint via a Lagrange multiplier which vanishes on the exact solution. Thus, index 2 reformulation of index 2 problem (34)–(36) proposed in [17] is

$$\begin{aligned} p' &= u - p\mu, \\ q' &= v - q\mu, \\ mu' &= -p\lambda, \\ mv' &= -q\lambda - g, \\ 0 &= p^2 + q^2 - l^2, \\ 0 &= pu + qv. \end{aligned} \tag{38}$$

In Table 1 we present the results for (34), (35); (34), (36); and (34), (37). We normalize the equations by taking $m = g = l = 1$. The initial conditions are

$$p(0) = v(0) = \lambda(0) = 1, \quad q(0) = u(0) = 0 \tag{39}$$

Table 1
MEBDF results for index 1 pendulum problem

Tol	Fn	Jac	Steps	Time	Figs	3	2	1
10^{-2}	18	4	13	0.01	2.78	0.3d−3	0.6d−3	0.5d−2
10^{-3}	23	5	16	0.01	3.89	0.1d−3	0.2d−3	0.2d−3
10^{-4}	30	4	21	0.02	5.49	0.8d−5	0.5d−5	0.4d−5
10^{-5}	43	5	27	0.03	6.31	0.1d−6	0.1d−6	0.3d−6
10^{-6}	51	7	34	0.04	7.70	0.1d−7	0.5d−8	0.3d−7
10^{-7}	77	11	53	0.06	8.30	0.5d−8	0.5d−8	0.1d−8
10^{-8}	86	10	61	0.06	8.88	0.1d−10	0.4d−9	0.4d−8
10^{-9}	119	12	77	0.10	10.25	0.1d−10	0.6d−10	0.3d−10

Table 2
MEBDF results for index 2 pendulum problem

Tol	Fn	Jac	Steps	Time	Figs	3	2	1
10^{-2}	18	4	13	0.02	2.42	0.3d−2	0.1d−3	0.8d−2
10^{-3}	23	4	16	0.02	3.29	0.9d−4	0.9d−3	0.1d−2
10^{-4}	31	4	21	0.02	4.54	0.1d−6	0.2d−5	0.3d−4
10^{-5}	52	4	28	0.03	5.10	0.1d−5	0.2d−5	0.3d−4
10^{-6}	60	6	39	0.04	5.93	0.4d−7	0.8d−7	0.3d−5
10^{-7}	69	8	45	0.05	7.50	0.6d−8	0.1d−8	0.8d−7
10^{-8}	117	11	68	0.08	8.40	0.8d−10	0.2d−9	0.9d−8
10^{-9}	138	15	84	0.10	9.41	0.2d−10	0.3d−10	0.1d−8

and the range of integration is [0,1]. The results given in Table 1 should be largely self-explanatory. In particular, Tol is the specified local tolerance, Fn is the number of function evaluations, Jac is the number of Jacobian evaluations, Steps is the number of integration steps, Time is the time in seconds taken on an IBM RS6000 and Figs is the number of correct figures at $x=1$. Under columns 3, 2, 1 we also give the amounts by which the constraints (35), (36), (37), which are index 3, index 2 and index 1, respectively, are not satisfied at $x=1$. We see from Table 1 that the code performs well for all three problems. As the index is increased the code obtains less accuracy, as would be expected, but is still satisfactory (see Tables 2 and 3).

8. Conclusions

These results back up the claims made in this paper regarding the promise of MEBDF. In particular, it is clear that the MEBDF have better theoretical properties than the BDF methods. The MEBDF are also excellently suited to stiff oscillatory ODEs. The results presented in [7,14], particularly on the FEKETE problem, indicate that MEBDF perform well on some difficult DAE systems although there are still some gaps in the theory which have been highlighted in this paper and which need to be filled in. However, the BDF codes and Radau5 are powerful codes in their own right. In

Table 3
MEBDF results for index 3 pendulum problem

Tol	Fn	Jac	Steps	Time	Figs	3	2	1
10^{-2}	15	3	9	0.01	1.85	0.5d−3	0.1d−2	0.4d−1
10^{-3}	25	4	14	0.01	2.23	0.2d−2	0.4d−4	0.2d−1
10^{-4}	52	5	25	0.03	2.73	0.1d−4	0.5d−4	0.5d−2
10^{-5}	99	12	38	0.05	3.67	0.8d−9	0.5d−5	0.7d−3
10^{-6}	73	9	41	0.04	4.57	0.2d−8	0.1d−6	0.5d−4
10^{-7}	100	9	55	0.06	5.01	0.1d−7	0.4d−6	0.3d−4
10^{-8}	130	14	81	0.08	6.42	0.3d−9	0.1d−6	0.1d−4
10^{-9}	154	15	96	0.11	6.78	0.1d−10	0.3d−8	0.4d−6

particular, BDF codes are often well suited to large ODE/DAE systems and it remains to be seen how competitive MEBDFDAE is compared to BDF on such problems.

Acknowledgements

The author is grateful to two referees for their careful reading of this manuscript.

References

[1] K. Brenan, S.L. Campbell, L. Petzold, Numerical Solution of Initial Value Problems in Differential-Algebraic Equations, North-Holland, New York, 1989.
[2] L. Brugnano, D. Trigiante, Solving Differential Problems by Multistep Initial and Boundary Value Methods, Gordon and Breach, London, 1997.
[3] J. Butcher, The Numerical Analysis of Ordinary Differential Equations, Wiley, New York, 1987.
[4] J. Butcher, J.R. Cash, M.T. Diamantakis, DESI methods for stiff initial value problems, ACM Trans. Math. Software 22 (1996) 401–422.
[5] J.R. Cash, The integration of stiff initial value problems in ODEs using modified extended backward differentiation formulae, Comput. Math. Appl. 9 (1983) 645–660.
[6] J.R. Cash, S. Considine, An MEBDF code for stiff initial value problems, ACM Trans. Math. Software 18 (1992) 142–160.
[7] http://www.ma.ic.ac.uk/˜jcash/IVP_ Software/readme.html.
[8] J.R. Cash, An extension of Olver's method for the numerical solution of linear recurrence relations, Math. Comp. 32 (1978) 497–510.
[9] J.R. Cash, Stable Recursions with Applications to Stiff Differential Equations, Academic Press, London, 1979.
[10] J.R. Cash, Stability concepts in the numerical solution of difference and differential equations, Comput. Math. Appl. 28 (1994) 45–53.
[11] J.R. Cash, A comparison of some codes for the stiff oscillatory problem, Comp. Math. Appl. 36 (1998) 51–57.
[12] C.F. Curtiss, J.D. Hirschfelder, Integration of stiff equations, Proc. Nat. Acad. Sci. 38 (1952) 235–243.
[13] G. Dahlquist, A special stability problem for linear multistep methods, BIT 3 (1963) 27–43.
[14] W.M. Lioen, J.J.B. de Swart, Test set for initial value problem solvers, Report MAS-R9832, CWI, Amsterdam, 1998 and http://www.cwi.nl/cwi/projects/IVPtestset.
[15] W.H. Enright, T.E. Hull, B. Lindberg, Comparing numerical methods for stiff systems of ODEs, BIT 15 (1975) 10–48.
[16] P.W. Gaffney, A performance evaluation of some FORTRAN subroutines for the solution of stiff oscillatory ordinary differential equations, ACM Trans. Math. Software 10 (1984) 58–72.

[17] C.W. Gear, G.K. Gupta, B. Leimkuhler, Automatic integration of Euler–Lagrange equations with constraints, J. Comput. Appl. Math. 12 (1985) 77–90.
[18] E. Hairer, G. Wanner, Solving Ordinary Differential Equations II Stiff and Differential Algebraic Problems, Springer, Berlin, 1996.
[19] E. Hairer, C. Lubich, M. Roche, The Numerical Solution of Differential-Algebraic Systems by Runge–Kutta Methods, Lecture Notes in Maths, Vol. 1409, Springer, Berlin, 1989.
[20] A.C. Hindmarsh, LSODE and LSODI, two initial value ordinary differential equation solvers ACM SIGNUM Newslett. 15 (1980).
[21] J.D. Lambert, Computational Ordinary Differential Equations, Wiley, New York, 1973.
[22] F.W.J. Olver, Numerical solution of second order linear difference equations, J. Res. Nat. Bur. Standards Math. Math. Phys. 71B (1967) 111–129.
[23] J.C.P. Miller, Bessel Functions, Part II, Math. Tables, Vol. X, British Association for the Advancement of Sciences, CUP, 1952.
[24] G. Psihoyios, Advanced step-point methods for the solution of initial value problems, Ph.D. Thesis, University of London, Imperial College, 1995.
[25] S.P. Norsett, A. Wolfbrandt, Attainable orders of rational approximations to the exponential function with only real poles, BIT 17 (1977) 200–208.
[26] G. Psihoyios, J.R. Cash, A stability result for general linear methods with characteristic function having real poles only, BIT 38 (1998) 612–617.
[27] P. Vander Houwen, private communication, 1999.