Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Computational and Applied Mathematics 189 (2006) 34–50

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.com/locate/cam

Blended implicit methods for the numerical solution of DAE problems☆

Luigi Brugnano*, Cecilia Magherini, Filippo Mugnai

Dipartimento di Matematica "U. Dini", Viale Morgagni 67/A, 50134 Firenze, Italy

Received 10 March 2005; received in revised form 2 May 2005

Abstract

Recently, a new approach for solving the discrete problems, generated by the application of *block implicit methods* for the numerical solution of initial value problems for ODEs, has been devised [L. Brugnano, Blended block BVMs (B_3VMs): a family of economical implicit methods for ODEs, J. Comput. Appl. Math. 116 (2000) 41–62; L. Brugnano, C. Magherini, Blended implementation of block implicit methods for ODEs, Appl. Numer. Math. 42 (2002) 29–45; L. Brugnano, D. Trigiante, Block implicit methods for ODEs, in: D. Trigiante (Ed.), Recent Trends in Numerical Analysis, Nova Science Publishers, New York, 2001, pp. 81–105]. This approach is based on the so-called *blended* implementation of the methods, giving corresponding *blended implicit methods*. The latter have been implemented in the computational code BiM [L. Brugnano, C. Magherini, The BiM code for the numerical solution of ODEs, J. Comput. Appl. Math. 164–165 (2004) 145–158]. Blended implicit methods are here extended to handle the numerical solution of DAE problems, resulting in a straightforward generalization of the basic approach.
© 2005 Elsevier B.V. All rights reserved.

MSC: 65L05; 65L06; 65L80; 65H10

Keywords: Ordinary differential equations; Stiff problems; Differential algebraic equations; Numerical methods for ODEs; Numerical methods for DAEs; Iterative solution of algebraic systems; Blended implicit methods

☆ Research supported by I.N.d.A.M. and Italian M.I.U.R.

* Corresponding author. Tel.: +39 329 3603938; fax: +39 055 4222 695.

E-mail addresses: brugnano@math.unifi.it (L. Brugnano), magherini@math.unifi.it (C. Magherini), f.mugnai@libero.it (F. Mugnai).

0377-0427/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.cam.2005.05.005

1. Introduction

It is widely known that the use of implicit numerical methods is customary when solving *stiff* initial value problems for ODEs,

$$y' = f(t, y),\ t \in [t_0, T], \quad y(t_0) = y_0 \in \mathbb{R}^m. \tag{1}$$

Indeed, the use of explicit methods is avoided because of their poor stability properties. Moreover, usual linear multistep formulae (LMF), when used as initial value methods, do not permit the use of high order formulae, due to the well-known *second Dahlquist barrier*. As a consequence, the use of *block implicit methods* [6,9,20] is often considered. Such methods generate, at each step of numerical integration, a discrete problem in the form

$$A \otimes I_m \mathbf{y}_n - h_n B \otimes I_m \mathbf{f}_n = \boldsymbol{\eta}_n, \quad n = 0, 1, \ldots\ . \tag{2}$$

In the previous formula, I_m denotes the identity matrix of size m, the matrices $A, B \in \mathbb{R}^{r\times r}$ define the method, h_n is the current stepsize, the block vectors

$$\mathbf{y}_n = (y_{n1}, \ldots, y_{nr})^{\mathrm{T}}, \qquad \mathbf{f}_n = (f_{n1}, \ldots, f_{nr})^{\mathrm{T}},$$

$$f_{nj} = f(t_{nj}, y_{nj}), \quad y_{nj} \approx y(t_{nj}),\ t_{nj} = t_n + c_j h_n,\ j = 1, \ldots, r,$$

contain the discrete solution, and the vector $\boldsymbol{\eta}_n$ only depends on already known quantities. Hereafter, the two matrices A and B are assumed to be nonsingular, so that the method is implicit. Methods falling in this category are the majority of implicit Runge–Kutta methods, a number of general linear methods [11,14,15], and, more recently, block boundary value methods (BVMs) [8].

The key problem, when using methods in the form (2), is that, at each step of numerical integration, one has to solve a (generally nonlinear) system of rm algebraic equations, in order to compute the discrete (local) solution $\mathbf{y}_n$. Straightforward application of the simplified Newton iteration would require the factorization of the $rm \times rm$ matrix

$$A \otimes I_m - h_n B \otimes J_n,$$

where J_n denotes the Jacobian matrix of $f(t, y)$ evaluated at the last recently known point. The cost for this is usually too high for this approach to be of practical interest. Consequently, many attempts have been made, across the years, in order to get rid of this drawback. The first successful approach is due to Butcher [10], consisting, in this setting, in having the two matrices A and B essentially diagonalized by the same similarity transformation. By neglecting, for the sake of simplicity, the cost for functional evaluations (which are very problem dependent), such an approach reduces the complexity of the problem from $\mathrm{O}((rm)^3)$ to $\mathrm{O}(rm^3)$ *flops* (where, as usual, we count one of the basic algebraic binary operations as one *flop*), and it is successfully implemented in the computational codes RADAU5 and RADAU [15]. An alternative approach consists in defining suitable linear and/or nonlinear splittings [1,16,17], i.e., in looking for suitably simple structured matrices A^* and B^*, thus solving the sequence of simpler problems

$$\begin{aligned} A^* \otimes I_m \mathbf{y}_n^{(i)} - h_n B^* \otimes I_m \mathbf{f}_n^{(i)} = (A^* - A) \otimes I_m \mathbf{y}_n^{(i-1)} \\ -h_n (B^* - B) \otimes I_m \mathbf{f}_n^{(i-1)} + \boldsymbol{\eta}_n, \\ i = 1, 2, \ldots, \end{aligned} \tag{3}$$

with an obvious meaning of the used notation. Nevertheless, appropriate convergence properties are required for the iteration (3), in order not to introduce step limitations not due to the stability properties of the method (see also the next section). This approach, which has very good potentialities (a reduction of the complexity to $O(m^3)$ *flops* is, in principle, the target), is successfully implemented in the computational code GAM [18]. However, it is definitely not an easy task to derive suitably simple splitting matrices A^* and B^* defining corresponding iterations having satisfactorily convergence properties.

This task has been recently made much simpler by introducing the *blended implementation* of the block implicit method (2) (we also speak about a *blended implicit method*, in such a case). The basic idea, on which such schemes rely, consists in deriving a numerical method as the combination (*blending*) of two suitable component methods. This approach, at first defined for block BVMs [5], has then been deepened in [6,9], thus leading to the release of the code BiM [7], which is based on blended implicit methods. In this paper we extend this approach, defined for solving problem (1), to the case of linearly implicit DAE problems, that is, problems in the more general form

$$Ky' = f(t, y), \quad t \in [t_0, T], \qquad y(t_0) = y_0 \in \mathbb{R}^m, \tag{4}$$

where the *mass matrix* $K \in \mathbb{R}^{m\times m}$ may be singular. Such problems, indeed, are of great importance in the applications (see, e.g., [4,13]). For the eventual computation of y_0, we refer, for example, to [2–4,19].

The paper is organized as follows: in Section 2 the main facts about blended implicit methods are recalled for completeness and later reference; the extension of the approach to DAE problems is then described in Section 3; some numerical tests, obtained by using a modification of the code BiM, are reported in Section 4; finally, some concluding remarks are reported in Section 5.

2. Blended implicit methods

In order to conveniently introduce *blended implicit methods*, and to carry out a corresponding linear convergence analysis (according to [16,17]), it is customary to consider the application of the block method (2) to the usual test equation

$$y' = \mu y, \qquad y(t_0) = y_0, \qquad Re(\mu) \leqslant 0. \tag{5}$$

Moreover, for sake of simplicity, let us consider the very first application of the method, since the same arguments apply to each subsequent step of integration. This allows us to avoid, hereafter, the subscript n denoting the integration step. As a consequence, the discrete problem generated by the application of method (2) to problem (5) results in the following system of linear equations,

$$(A - qB)\mathbf{y} = \boldsymbol{\eta}, \tag{6}$$

where, as usual, we have set $q = h\mu$. We shall here consider the so-called *type* 1 *methods* as described in [6], which are the methods implemented in the code BiM [7]. In more detail, we observe that the nonsingularity of both matrices A and B implies that the discrete problem (6) is equivalent to the following two equations, where $C = A^{-1}B$, $\gamma > 0$, and I is the $r \times r$ identity matrix,

$$(I - qC)\mathbf{y} = A^{-1}\boldsymbol{\eta} \equiv \boldsymbol{\eta}_1, \tag{7}$$

$$\gamma(C^{-1} - qI)\mathbf{y} = \gamma B^{-1}\boldsymbol{\eta} \equiv \boldsymbol{\eta}_2. \tag{8}$$

Let us now introduce the *weighting function*

$$\theta(q) = (I - q\gamma I)^{-1}. \tag{9}$$

We observe that:

- $\theta(q)$ is analytical for $q \in \mathbb{C}^-$;
- $\theta(0) = I$;
- $\theta(q) \to O$, the zero matrix, as $q \to \infty$.

We then define the *blended implicit method* based on the block method (2) as the method generating the following discrete problem:

$$M(q)\mathbf{y} \equiv (A(q) - qB(q))\mathbf{y} = \hat{\boldsymbol{\eta}}(q), \tag{10}$$

where

$$\begin{aligned} A(q) &= \theta(q)I + (I - \theta(q))\gamma C^{-1}, \\ B(q) &= \theta(q)C + (I - \theta(q))\gamma I, \\ \hat{\boldsymbol{\eta}}(q) &= \theta(q)\boldsymbol{\eta}_1 + (I - \theta(q))\boldsymbol{\eta}_2, \end{aligned}$$

which is still equivalent to (6). We observe that

- $M(q) = I + \mathrm{O}(q)$, when $q \approx 0$;
- $M(q) = q(\gamma I + \mathrm{O}(q^{-1}))$, as $q \to \infty$.

This naturally induces the splitting matrix

$$N(q) = I - q\gamma I \equiv \theta(q)^{-1}, \tag{11}$$

since it coincides with $M(q)$ at $q = 0$ and at ∞, and the corresponding *blended iteration* associated with the blended method (10),

$$N(q)\mathbf{y}^{(i)} = (N(q) - M(q))\mathbf{y}^{(i-1)} + \hat{\boldsymbol{\eta}}(q), \quad i = 1, 2, \ldots\ . \tag{12}$$

The iteration will be convergent if and only if the spectral radius, say $\rho(q)$, of the iteration matrix,

$$Z(q) = I - N(q)^{-1}M(q),$$

is not larger than 1. From the previous arguments,

- $\rho(0) = 0$,
- $\rho(q) \to 0$, as $q \to \infty$,

for any choice of $\gamma > 0$. According to [16,17], and by using the maximum modulus principle, the iteration (12) is said to be L-convergent if the *maximum amplification factor*,

$$\rho^*_\gamma = \max_{x>0} \rho(\mathrm{i}x), \tag{13}$$

is smaller than 1, where, as usual, i denotes the imaginary unit. It is worth emphasizing that an L-convergent iteration is highly desirable, if the underlying block method is L-stable: conversely, step limitations, not due to the stability properties of the method, could result from requiring the convergence of the iteration (12). In the above definition (13), the subscript γ does denote the dependence of this factor from the choice of the positive parameter γ. For the methods implemented in the code BiM, and more in general for all block methods such that:

- the ith equation of the discrete problem is defined by a r-step LMF of order (at least) r,
- the spectrum of the matrix $C=A^{-1}B$ coincides (up to a variable scaling) with the reciprocal polynomial at the denominator of the (s,r) Padé approximation to the exponential, $s=r-2, r-1, r$,

the following result holds true [6].

Theorem 1. *The choice*

$$\gamma = \min_{\lambda\in\sigma(C)} |\lambda|$$

minimizes the value of the maximum amplification factor (13).

In particular, for all practical values of r one obtains L-convergent iterations, which are associated to corresponding L-stable methods, when $s<r$. By the way, we observe that Theorem 1 applies to Runge–Kutta Radau IIA methods.

We end this section by observing that, when the blended method is applied to the more general problem (1), then the iteration (10)–(12) becomes

$$\begin{aligned}\boldsymbol{\delta}^{(i)} &= N^{-1}(\theta((I-\gamma C^{-1})\otimes I_m\mathbf{y}^{(i-1)} - h(C-\gamma I)\otimes I_m\mathbf{f}^{(i-1)})\\ &\quad + \gamma(C^{-1}\otimes I_m\mathbf{y}^{(i-1)} - hI\otimes I_m\mathbf{f}^{(i-1)}) - \hat{\boldsymbol{\eta}}),\\ \mathbf{y}^{(i)} &= \mathbf{y}^{(i-1)} - \boldsymbol{\delta}^{(i)}, \quad i=1,2,\ldots,\end{aligned} \tag{14}$$

where by setting J the Jacobian of $f(t,y)$ evaluated at (t_0, y_0),

$$N^{-1} \equiv \theta = I\otimes\Omega^{-1}, \quad \Omega = I_m - h\gamma J. \tag{15}$$

As a consequence, if ℓ iterations are required to obtain convergence, the overall leading cost, to carry out the iteration (14), amounts to:

- 1 Jacobian evaluation,
- 1 factorization for the $m\times m$ matrix Ω,
- $r\ell$ function evaluations, and
- $2r\ell$ system solvings with the factors of Ω.

For more details, we refer to [7]. An additional, remarkable, feature of the iteration (14)–(15) is the block diagonal structure of the splitting matrix N.

For later reference, we observe that, in the case of the linear ODE

$$y' = Jy + g(t),$$

the iteration (14) induces the iteration matrix

$$Z = C^{-1}(C - \gamma I)^2 \otimes hJ(I_m - h\gamma J)^{-2}, \tag{16}$$

for which the previous linear convergence analysis, summarized by the result of Theorem 1, applies.

3. Extension to DAE problems

We now extend blended implicit methods for solving DAE problems in the form (4). Since we are interested in a linear analysis of convergence for the blended iteration, we confine ourselves to the case of linear equations in the form

$$Ky' = Jy + g(t), \tag{17}$$

where K and J are constant matrices and $g(t)$ is a vector valued function. The following notions (see, for example, [4,13]) are briefly recalled, for the sake of completeness. First of all, the *matrix pencil*

$$\mu K - J, \tag{18}$$

is associated with Eq. (17). The pencil is said to be *regular* if its determinant is not identically 0, as a function of μ. Moreover, Eq. (17) is solvable if and only if the pencil (18) is regular. This will be, therefore, assumed in the sequel. In such a case, the pencil (18) can be cast into its *Kronecker canonical form*,

$$PKQ = \begin{pmatrix} I_d & \\ & H \end{pmatrix} \equiv \hat{K}, \quad PJQ = \begin{pmatrix} G & \\ & I_a \end{pmatrix} \equiv \hat{J}, \tag{19}$$

where P and Q are nonsingular $m \times m$ matrices, $d+a=m$, I_d and I_a are the identity matrices of dimension d and a, respectively, $G \in \mathbb{R}^{d\times d}$, and,

$$H = \begin{pmatrix} H_1 & & \\ & \ddots & \\ & & H_k \end{pmatrix} \in \mathbb{R}^{a\times a}, \tag{20}$$

where

$$H_i = \begin{pmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & 0 \end{pmatrix} \in \mathbb{R}^{\nu_i \times \nu_i}, \quad i = 1, \dots, k.$$

Consequently,

$$H^\nu = O, \tag{21}$$

where the integer

$$\nu = \max_i \nu_i, \tag{22}$$

is known as the *index* (*of nilpotency*) of the DAE. It is also very well known that the higher the index, the more difficult the problem.

By means of the Kronecker normal form (19), the problem decouples as follows:

$$x' = Gx + \varphi(t), \quad Hz' = z + \psi(t), \tag{23}$$

where

$$Q^{-1}y \equiv \begin{pmatrix} x \\ z \end{pmatrix}, \quad Pg(t) \equiv \begin{pmatrix} \varphi(t) \\ \psi(t) \end{pmatrix}.$$

With this premise and notations, let us generalize the blended iteration (9)–(12), (14)–(15) for solving the DAE (17). The only differences with what were seen in Section 2 amount to the weighting function,

$$\theta = I \otimes K\Omega^{-1}, \quad \Omega = K - h\gamma J, \tag{24}$$

and the splitting matrix,

$$N = I \otimes \Omega, \tag{25}$$

in place of (15). We observe that we get exactly the formulae previously seen in Section 2, in the case $K = I_m$, i.e., when (17) reduces to an ODE.

Because of (17), (24)–(25), the blended iteration (14) now becomes,

$$\begin{aligned} \boldsymbol{\delta}^{(i)} &= N^{-1}(\theta((I - \gamma C^{-1}) \otimes K - h(C - \gamma I) \otimes J)\mathbf{y}^{(i-1)} + \gamma(C^{-1} \otimes K - hI \otimes J)\mathbf{y}^{(i-1)} - \hat{\boldsymbol{\eta}}), \\ \mathbf{y}^{(i)} &= \mathbf{y}^{(i-1)} - \boldsymbol{\delta}^{(i)}, \quad i = 1, 2, \ldots . \end{aligned} \tag{26}$$

The corresponding iteration matrix is then given by

$$Z = I \otimes I_m - N^{-1}(\theta((I - \gamma C^{-1}) \otimes K - h(C - \gamma I) \otimes J) + \gamma(C^{-1} \otimes K - hI \otimes J)). \tag{27}$$

Let now denote by $\hat{Z}$ the matrix obtained from Z with the following formal substitutions (see (19) and (24)–(25)):

$$K \leftarrow \hat{K}, \quad J \leftarrow \hat{J}. \tag{28}$$

The following result then holds true.

Lemma 1. $\hat{Z} = (I \otimes Q^{-1})Z(I \otimes Q)$.

Proof. Let us consider the Kronecker normal form (19) and define

$$\hat{\Omega} = P\Omega Q, \quad \hat{N} = (I \otimes P)N(I \otimes Q), \quad \hat{\theta} = (I \otimes P)\theta(I \otimes P^{-1}). \tag{29}$$

Moreover, one verifies that:

$$\begin{aligned} I \otimes \hat{K} - hC \otimes \hat{J} &= (I \otimes P)(I \otimes K - hC \otimes J)(I \otimes Q), \\ C^{-1} \otimes \hat{K} - hI \otimes \hat{J} &= (I \otimes P)(C^{-1} \otimes K - hI \otimes J)(I \otimes Q). \end{aligned}$$

Thus, by substituting in the corresponding formula (27)–(28) for $\hat{Z}$, we obtain:

$$\begin{aligned} \hat{Z} = I \otimes I_m - \hat{N}^{-1}(\hat{\theta}((I - \gamma C^{-1}) \otimes \hat{K} - h(C - \gamma I) \otimes \hat{J}) \\ + \gamma(C^{-1} \otimes \hat{K} - hI \otimes \hat{J})) = (I \otimes Q^{-1})\, Z\, (I \otimes Q). \quad \square \end{aligned} \tag{30}$$

Consequently, we can prove the following result.

Lemma 2. *The iteration matrix Z corresponding to* (26) *is similar to a block diagonal matrix*,

$$\begin{pmatrix} Z_d & \\ & Z_a \end{pmatrix}, \tag{31}$$

where see (19)–(23),

$$Z_d = C^{-1}(C-\gamma I)^2 \otimes hG(I_d - h\gamma G)^{-2}, \tag{32}$$
$$Z_a = C^{-1}(C-\gamma I)^2 \otimes hH(H - h\gamma I_a)^{-2}. \tag{33}$$

Proof. Let define the odd–even block permutation matrix,

$$\Pi = \begin{pmatrix} I_d & & & & & & & \\ & & I_d & & & & & \\ & & & & \ddots & & & \\ & & & & & & I_d & \\ & I_a & & & & & & \\ & & & I_a & & & & \\ & & & & & \ddots & & \\ & & & & & & & I_a \end{pmatrix}_{rm\times rm}.$$

From (30), one then obtains that $Z \sim \Pi\hat{Z}\Pi^{\mathrm{T}}$. The latter is a 2×2 block diagonal matrix in the form (31), with the diagonal blocks given by (see (24)–(25) and (29))

$$\begin{aligned} Z_d &= I\otimes I_d - I\otimes(I_d-h\gamma G)^{-1}[I\otimes(I_d-h\gamma G)^{-1}(I\otimes I_d \\ &\quad - hC\otimes G) - \gamma^2 hI\otimes G(I-h\gamma G)^{-1}(C^{-1}\otimes I_d - hI\otimes G)], \\ Z_a &= I\otimes I_a - I\otimes(H-h\gamma I_a)^{-1}[I\otimes H(H-h\gamma I_a)^{-1}(I\otimes H \\ &\quad - hC\otimes I_a) - \gamma^2 hI\otimes(H-h\gamma I_a)^{-1}(C^{-1}\otimes H - hI\otimes I_a)]. \end{aligned}$$

Let us prove that Z_a is given by (33). A similar proof holds for (32). One has

$$\begin{aligned} Z_a &= I\otimes I_a - I\otimes(H-h\gamma I_a)^{-2}(I\otimes H^2 - hC\otimes H - \gamma^2 hC^{-1}\otimes H + \gamma^2h^2 I\otimes I_a) \\ &= I\otimes(H-h\gamma I_a)^{-2}((-2h\gamma I + hC + \gamma^2 hC^{-1})\otimes H) \\ &= I\otimes(H-h\gamma I_a)^{-2}((hC^{-1}(C-\gamma I)^2)\otimes H) \\ &= C^{-1}(C-\gamma I)^2\otimes hH(H-h\gamma I_a)^{-2}. \qquad \square \end{aligned}$$

Remark 1. From the previous two lemmas, we obtain that the iteration matrix Z of the blended iteration (26) is similar to a matrix made up of two "pieces":

- one piece corresponding to the ODE in (23), i.e., Z_d,
- the other corresponding to the normalized DAE in the same equation, i.e., Z_a.

It is evident that for Z_d (compare (32) with (16)) the result of Theorem 1 applies. On the other hand, for Z_a the following result holds true.

Theorem 2. *The matrix Z_a is nilpotent of index ν.*

Proof. Obvious from (33), by taking into account (21)–(22). □

Remark 2. The previous statements imply that the convergence properties of the blended iteration (26) are essentially unaffected by the algebraic part of the problem. In fact, they depend only on the differential part Z_d, provided that at least v iterations are carried out, if v (see (22)) is the index of the DAE.

4. Numerical tests

The previous analysis has lead to the development of a preliminary release of the code BiM [7], extended to handle the numerical solution of DAE problems. The code, implementing a variable order-variable stepsize blended implicit method, has been modified, according to what was seen in Section 3, in order to allow the integration of problems in the form (4) of index up to 3. In the current version, only the methods of order 4-6-8-10 are considered in the variable order implementation (see [7,20], for details on

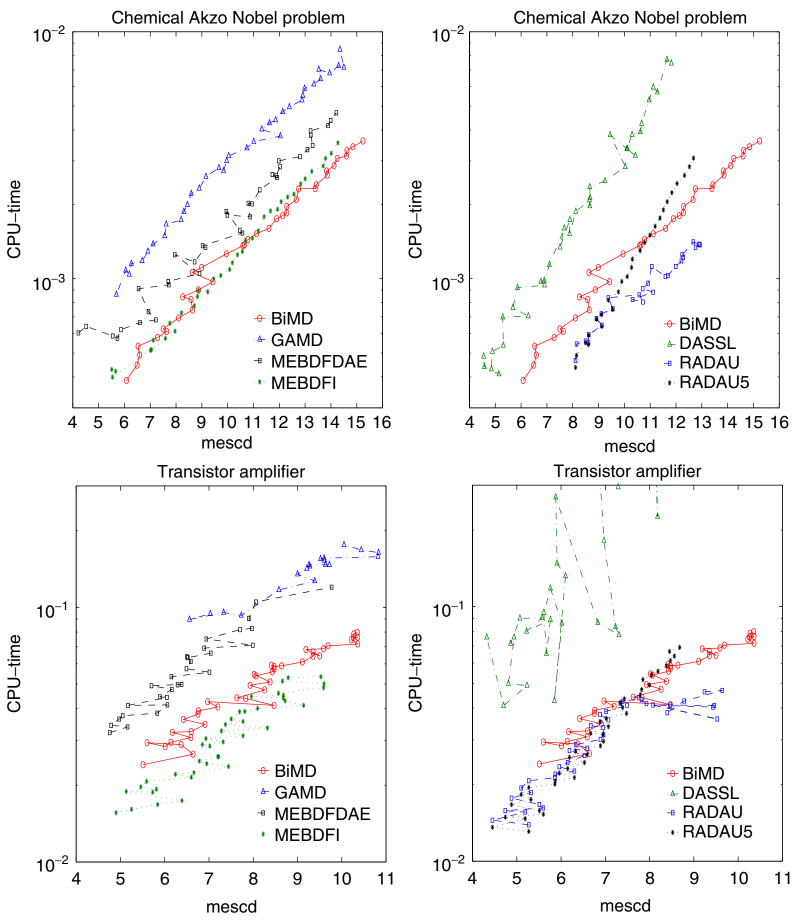

Fig. 1. Work-precision diagrams.

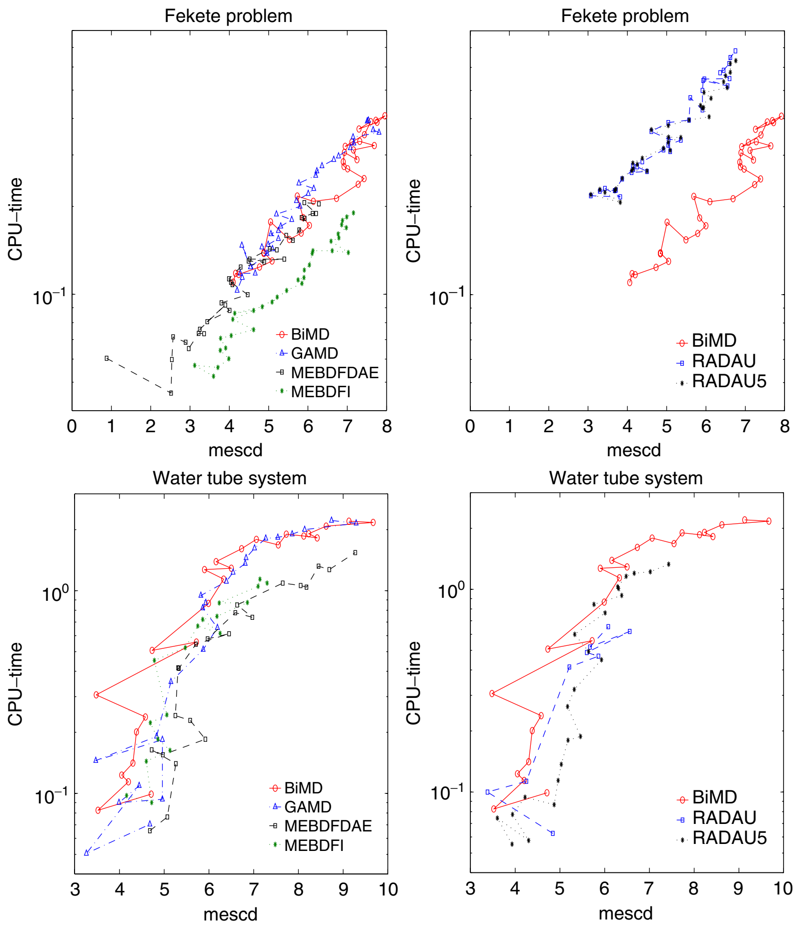

Fig. 2. Work-precision diagrams.

the methods implemented in the code BiM). Indeed, the use of higher order methods needs to address a few side problems: this research is still in progress. We shall refer to this new version of the code as BiMD. At the moment, the calling interface of the code BiMD is similar to that of most of the existing codes: it requires the variables to be sorted by increasing index [13] as, for example, for problems in Hessemberg form [4].

In this section we report a few numerical test comparing the code BiMD with the codes DASSL [4], GAMD [18], MEBDFDAE and MEBDFI [12], RADAU and RADAU5 [15], on a few test problems taken from the release 2.2 of the *Test Set for IVP Solvers* [21]. The release of each code, which has been used for the tests, is the one available in the above-mentioned version of the Test Set [21]. All the above codes are written in Fortran 77 with the exception of the code GAMD which is a Fortran 90 code. The numerical experiments have been done on an IBM SP Power 4 platform, by using the `IBM XL` Fortran compiler

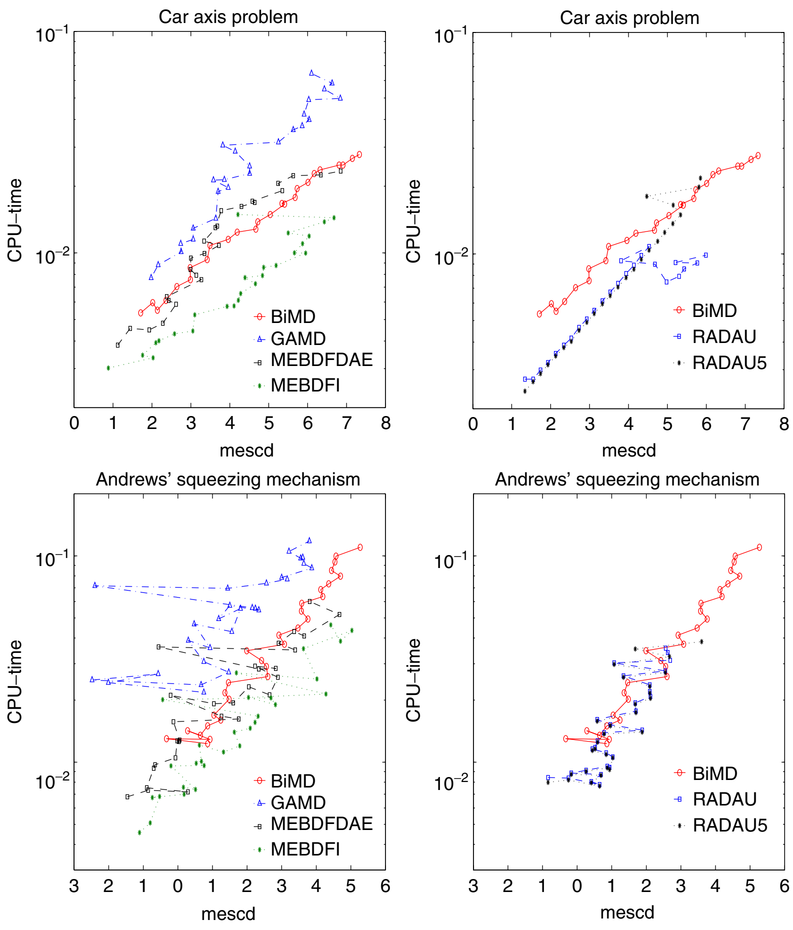

Fig. 3. Work-precision diagrams.

v.9.1, with the same optimization options, `-O5 -qstrict`, used for all the codes. The considered problems are:

- Chemical Akzo Nobel problem (index 1 DAE of dimension 6),
- Transistor amplifier (index 1 DAE of dimension 8),
- Fekete problem (index 2 DAE of dimension 160),
- Water tube system (index 2 DAE of dimension 49),
- Car axis problem (index 3 DAE of dimension 10),
- Andrews' squeezing mechanism (index 3 DAE of dimension 27).

The obtained results are summarized by corresponding *work precision diagrams*, plotted in Figs. 1–3. According to the standards of the Test Set (see [21]), the diagrams with respect to the *mescd* (mixed error

Table 1
Parameters used for the numerical tests

Problem	α	β	m^*
Chemical Akzo Nobel problem	4	4	36
Transistor amplifier	4	8	40
Fekete problem	2	8	32
Water tube system	4	4	24
Car axis problem	4	4	24
Andrews' squeezing mechanism	4	4	28

Table 2
Failed runs

Problem	Code	m
Chemical Akzo Nobel problem	MEBDFI	0
	RADAU	0–9
	RADAU5	0–9
Transistor amplifier	GAMD	0–9, 12, 15–23, 25, 27
	MEBDFDAE	24, 27, 29–32, 34, 35, 37–40
Water tube system	MEBDFDAE	16, 17, 22
	MEBDFI	18–24
	RADAU	0, 1, 3, 4, 6, 7, 9–14, 18, 22–24
	RADAU5	3
Andrews' squeezing mechanism	GAMD	1
	RADAU	53

significant correct digits) are plotted for each problem:

$$mescd = -\log_{10}(\|(y - y_{\text{ref}})./(\texttt{atol}/\texttt{rtol} + |y_{\text{ref}}|)\|_\infty).$$

In the previous formula, y is the vector with the computed solution at the end of the integration interval and y_{ref} is the vector containing the corresponding reference solution; $|\cdot|$ and ./ denote the component-wise absolute value and ratio, respectively; finally, `atol` and `rtol` are the absolute and relative input tolerances, respectively.

All codes are compared on each test problem, with the only exception of DASSL, which is used only on index 1 DAEs. For each problem, the following parameters have been used,

$$\texttt{atol} = \texttt{rtol} = \texttt{h0} = 10^{-(\alpha+m/\beta)}, \quad m = 0, \ldots, m^*,$$

where `atol` and `rtol` are the tolerances above specified, and `h0` is the initial stepsize (not required for DASSL). The specific values of the parameters α, β, m^* are listed in Table 1 , for all considered problems. A number of failed runs occurred for obtaining the data for the work precision diagrams: they are summarized in Table 2, where, for each problem, the codes and the corresponding values of m are listed. Finally, in Tables 3 and 4 some statistics for each problem are listed (*scd* denotes the computed significant correct digits [21]). From the numerical tests, it seems that the new, preliminary, version of the

Table 3
Run statistics

Code	atol rtol h_0	mescd	scd	steps	accept	#f	#Jac	#LU	CPU-time
Chemical Akzo Nobel problem									
BIMD	10^{-5}	7.28	4.72	22	21	331	21	22	$5.78 \cdot 10^{-4}$
	10^{-9}	11.60	9.57	35	35	936	35	35	$1.60 \cdot 10^{-3}$
	10^{-13}	15.24	12.81	60	60	2109	60	60	$3.61 \cdot 10^{-3}$
DASSL	10^{-5}	4.89	3.50	92	89	128	14		$5.07 \cdot 10^{-4}$
	10^{-9}	8.66	7.39	355	351	460	33		$1.97 \cdot 10^{-3}$
	10^{-13}	11.82	10.34	1439	1424	1688	74		$7.45 \cdot 10^{-3}$
GAMD	10^{-5}	6.69	4.63	16	16	449	16	16	$1.18 \cdot 10^{-3}$
	10^{-9}	11.01	9.24	29	29	1403	29	29	$3.59 \cdot 10^{-3}$
	10^{-13}	14.35	12.01	57	57	3395	57	57	$8.47 \cdot 10^{-3}$
MEBDFDAE	10^{-5}	5.87	4.03	74	74	115	12	12	$6.22 \cdot 10^{-4}$
	10^{-9}	10.87	8.55	205	205	300	27	27	$1.78 \cdot 10^{-3}$
	10^{-13}	14.21	12.67	539	539	764	56	56	$4.70 \cdot 10^{-3}$
MEBDFI	10^{-5}	7.01	4.87	74	73	250	12	12	$5.13 \cdot 10^{-4}$
	10^{-9}	10.57	8.46	201	201	677	24	24	$1.29 \cdot 10^{-3}$
	10^{-13}	14.27	12.60	543	542	1792	55	55	$3.55 \cdot 10^{-3}$
RADAU	10^{-5}	failed run							
	10^{-9}	10.58	8.11	52	52	539	42	52	$8.60 \cdot 10^{-4}$
	10^{-13}	12.93	10.61	53	53	1103	41	53	$1.37 \cdot 10^{-3}$
RADAU5	10^{-5}	failed run							
	10^{-9}	9.77	7.42	67	67	616	58	67	$8.84 \cdot 10^{-4}$
	10^{-13}	12.68	10.36	273	273	2520	133	153	$3.08 \cdot 10^{-3}$
Transistor amplifier									
BIMD	10^{-4}	5.51	5.26	484	419	8637	419	484	$2.41 \cdot 10^{-2}$
	10^{-7}	8.47	8.22	648	584	21241	584	648	$5.73 \cdot 10^{-2}$
DASSL	10^{-4}	4.32	3.89	9411	5882	17674	6943		$7.58 \cdot 10^{-2}$
	10^{-7}	7.17	6.45	46930	27692	88462	38249		$3.79 \cdot 10^{-1}$
GAMD	10^{-4}	failed run							
	10^{-7}	8.58	7.32	362	315	34589	356	362	$1.17 \cdot 10^{-1}$
MEBDFDAE	10^{-4}	4.76	4.50	1696	1586	3711	320	320	$3.22 \cdot 10^{-2}$
	10^{-7}	failed run							
MEBDFI	10^{-4}	4.90	4.47	1625	1527	6227	264	264	$1.56 \cdot 10^{-2}$
	10^{-7}	7.37	7.11	3620	3500	13297	431	431	$3.46 \cdot 10^{-2}$
RADAU	10^{-4}	5.27	4.95	681	499	6327	497	681	$1.39 \cdot 10^{-2}$
	10^{-7}	6.88	6.60	1765	1547	17554	1543	1765	$3.77 \cdot 10^{-2}$
RADAU5	10^{-4}	5.27	4.95	681	499	6327	497	681	$1.31 \cdot 10^{-2}$
	10^{-7}	6.88	6.60	1765	1547	17554	1543	1765	$3.54 \cdot 10^{-2}$
Fekete problem									
BIMD	10^{-2}	4.06	2.85	30	29	389	29	30	$1.10 \cdot 10^{-1}$
	10^{-4}	7.39	5.86	77	77	1054	77	77	$2.49 \cdot 10^{-1}$
	10^{-6}	7.69	6.23	78	78	2087	78	78	$3.94 \cdot 10^{-1}$

Table 3 (*continued*)

Code	atol rtol h$_0$	mescd	scd	steps	accept	#f	#Jac	#LU	CPU-time
GAMD	10^{-2}	4.16	2.99	26	24	526	24	26	$1.03 \cdot 10^{-1}$
	10^{-4}	5.76	4.45	38	38	1319	38	38	$2.00 \cdot 10^{-1}$
	10^{-6}	7.50	6.04	59	59	2865	59	59	$3.94 \cdot 10^{-1}$
MEBDFDAE	10^{-2}	0.84	−0.52	66	63	126	15	15	$6.04 \cdot 10^{-2}$
	10^{-4}	3.95	2.64	209	209	334	21	21	$1.13 \cdot 10^{-1}$
	10^{-6}	6.24	4.78	448	448	638	38	38	$2.04 \cdot 10^{-1}$
MEBDFDI	10^{-2}	3.56	2.10	60	57	192	15	15	$5.24 \cdot 10^{-2}$
	10^{-4}	5.81	4.81	218	216	707	23	23	$1.09 \cdot 10^{-1}$
	10^{-6}	7.12	5.66	441	441	1442	30	30	$1.90 \cdot 10^{-1}$
RADAU	10^{-2}	3.43	1.97	33	30	274	27	32	$2.31 \cdot 10^{-1}$
	10^{-4}	5.36	4.29	61	58	442	54	61	$3.38 \cdot 10^{-1}$
	10^{-6}	6.75	5.34	106	105	963	98	106	$6.83 \cdot 10^{-1}$
RADAU5	10^{-2}	3.43	1.97	33	30	274	27	32	$2.24 \cdot 10^{-1}$
	10^{-4}	5.36	4.29	61	58	442	54	61	$3.45 \cdot 10^{-1}$
	10^{-6}	6.76	5.31	128	127	919	116	122	$6.32 \cdot 10^{-1}$

code BiMD has a comparable performance with the existing codes; moreover, it appears to be slightly more robust (see Table 2).

5. Conclusions and future developments

In this paper we derived an extension of *blended implicit methods*, formerly defined for the efficient solution of stiff ODE-IVPs, for the numerical solution of DAE problems. Such methods have been recently introduced, in order to provide a discrete problem for which an efficient nonlinear splitting is readily available for its solution. The main goal of the paper has then been the generalization of this "mathematical technology" for handling linearly implicit DAE problems.

The generalization of the methods turns out to be very straightforward and, more than this, the corresponding *blended iteration* appears to be essentially unaffected by the algebraic part of the equation. This conclusion has been obtained by considering a linear analysis of convergence using linearly implicit DAEs in the form (17), with constant matrices K and J. This linear analysis of convergence naturally generalizes the corresponding analysis carried out in [6] for the ODE case.

The modified blended implicit methods have been used to obtain a preliminary release of the code BiM, called BiMD. Numerical tests carried out on test problems taken from the *Test Set for IVP Solvers* [21], show the potentialities of the new code, which well compares with some of the best codes currently available for the numerical solution of DAE problems. Moreover, it appears to be one of the more robust, among such codes.

Table 4
Run statistics

Code	atol rtol h_0	mescd	scd	steps	accept	#f	#Jac	#LU	CPU-time
Water tube system									
BIMD	10^{-4}	4.71	1.98	34	30	623	30	34	$9.91 \cdot 10^{-2}$
	10^{-7}	5.90	4.00	429	409	7463	409	429	1.27
GAMD	10^{-4}	4.68	2.41	25	21	498	21	25	$7.04 \cdot 10^{-2}$
	10^{-7}	5.93	3.37	219	197	8042	205	219	$8.71 \cdot 10^{-1}$
MEBDFDAE	10^{-4}	4.68	2.23	123	118	1355	23	23	$6.54 \cdot 10^{-2}$
	10^{-7}	5.97	3.05	1453	1404	11623	190	190	$5.78 \cdot 10^{-1}$
MEBDFDI	10^{-4}	4.16	2.50	139	126	2136	33	33	$9.75 \cdot 10^{-2}$
	10^{-7}	6.18	3.75	1572	1481	15917	213	213	$7.47 \cdot 10^{-1}$
RADAU	10^{-4}	failed run							
	10^{-7}	failed run							
RADAU5	10^{-4}	3.93	1.80	50	45	314	19	40	$5.53 \cdot 10^{-2}$
	10^{-7}	5.32	2.29	283	230	1898	109	223	$3.21 \cdot 10^{-1}$
Car axis problem									
BIMD	10^{-4}	1.71	0.20	121	121	1658	121	121	$5.36 \cdot 10^{-3}$
	10^{-6}	3.49	2.57	187	183	3507	183	187	$1.08 \cdot 10^{-2}$
	10^{-8}	5.68	3.94	213	210	5909	210	213	$1.78 \cdot 10^{-2}$
GAMD	10^{-4}	1.98	0.39	39	39	2169	39	39	$7.71 \cdot 10^{-3}$
	10^{-6}	3.96	1.50	81	77	5848	77	81	$1.97 \cdot 10^{-2}$
	10^{-8}	5.63	2.94	125	124	10588	124	125	$3.59 \cdot 10^{-2}$
MEBDFDAE	10^{-4}	1.12	−0.50	273	271	754	26	26	$3.83 \cdot 10^{-3}$
	10^{-6}	3.14	1.66	577	576	1578	65	65	$7.93 \cdot 10^{-3}$
	10^{-8}	3.78	1.65	1125	1115	2918	120	120	$1.55 \cdot 10^{-2}$
MEBDFI	10^{-4}	0.88	−0.23	280	278	1246	27	27	$3.02 \cdot 10^{-3}$
	10^{-6}	3.93	1.97	522	520	2277	34	34	$5.73 \cdot 10^{-3}$
	10^{-8}	5.19	3.21	813	812	3282	47	47	$8.77 \cdot 10^{-3}$
RADAU	10^{-4}	1.34	0.19	98	97	850	95	98	$2.72 \cdot 10^{-3}$
	10^{-6}	2.93	1.28	201	200	1742	196	201	$5.08 \cdot 10^{-3}$
	10^{-8}	4.53	2.74	418	417	3783	411	418	$1.08 \cdot 10^{-2}$
RADAU5	10^{-4}	1.34	0.19	98	97	850	95	98	$2.40 \cdot 10^{-3}$
	10^{-6}	2.93	1.28	201	200	1742	196	201	$4.91 \cdot 10^{-3}$
	10^{-8}	4.53	2.74	418	417	3783	411	418	$1.04 \cdot 10^{-2}$
Andrews' squeezing mechanism									
BIMD	10^{-4}	0.86	2.91	52	44	1114	44	52	$1.23 \cdot 10^{-2}$
	10^{-7}	2.56	4.64	119	117	2650	117	119	$2.91 \cdot 10^{-2}$
	10^{-10}	4.70	7.19	315	314	7265	314	315	$7.97 \cdot 10^{-2}$
GAMD	10^{-4}	−0.03	2.82	84	57	2248	57	84	$2.07 \cdot 10^{-2}$
	10^{-7}	1.81	4.00	161	126	5882	129	161	$5.55 \cdot 10^{-2}$
	10^{-10}	3.64	8.20	203	200	9206	200	203	$9.18 \cdot 10^{-2}$

Table 4 (*continued*)

Code	atol rtol h_0	mescd	scd	steps	accept	#f	#Jac	#LU	CPU-time
MEBDFDAE	10^{-4}	−1.46	−0.30	133	117	345	28	28	$6.80 \cdot 10^{-3}$
	10^{-7}	1.26	3.26	364	349	838	42	42	$1.67 \cdot 10^{-2}$
	10^{-10}	2.91	5.16	877	856	1790	94	94	$3.77 \cdot 10^{-2}$
MEBDFI	10^{-4}	−1.11	0.37	118	108	466	23	23	$4.56 \cdot 10^{-3}$
	10^{-7}	1.32	3.56	299	285	1193	38	38	$1.12 \cdot 10^{-2}$
	10^{-10}	4.02	6.50	673	664	2627	76	76	$2.53 \cdot 10^{-2}$
RADAU	10^{-4}	−0.84	1.36	96	56	810	54	96	$8.43 \cdot 10^{-3}$
	10^{-7}	0.52	4.45	114	95	1295	90	114	$1.18 \cdot 10^{-2}$
	10^{-10}	2.55	6.36	257	249	3094	241	256	$2.77 \cdot 10^{-2}$
RADAU5	10^{-4}	−0.84	1.36	96	56	810	54	96	$7.99 \cdot 10^{-3}$
	10^{-7}	0.52	4.45	114	95	1295	90	114	$1.16 \cdot 10^{-2}$
	10^{-10}	2.55	6.36	257	249	3094	241	256	$2.72 \cdot 10^{-2}$

In the near future, we plan to further improve the code BiMD: this will require the addressing of some side problems related to the efficient definition and implementation of the methods. Once this has been done, we shall made available on the web the new and improved release of the code, at the same URL of the code BiM [7].

References

[1] P. Amodio, L. Brugnano, A note on the efficient implementation of implicit methods for ODEs, J. Comput. Appl. Math. 87 (1997) 1–9.

[2] P. Amodio, F. Mazzia, Numerical solution of differential algebraic equations and computation of consistent initial/boundary conditions, J. Comput. Appl. Math. 87 (1997) 135–146.

[3] P. Amodio, F. Mazzia, An algorithm for the computation of consistent initial values for differential-algebraic equations, Numer. Alg. 19 (1998) 13–23.

[4] K.E. Brenan, S.L. Campbell, L.R. Petzold, Numerical solution of initial-value problems in differential-algebraic equations, Classics in Applied Mathematics, vol. 14, SIAM, Philadelphia, 1996. Code available at: http://www.netlib.org/ode/ddassl.f

[5] L. Brugnano, Blended block BVMs (B_3VMs): a family of economical implicit methods for ODEs, J. Comput. Appl. Math. 116 (2000) 41–62.

[6] L. Brugnano, C. Magherini, Blended implementation of block implicit methods for ODEs, Appl. Numer. Math. 42 (2002) 29–45.

[7] L. Brugnano, C. Magherini, The BiM code for the numerical solution of ODEs, J. Comput. Appl. Math. 164–165 (2004) 145–158 Code available at: http://math.unifi.it/~brugnano/BiM/index.html.

[8] L. Brugnano, D. Trigiante, Solving Differential Problems by Multistep Initial and Boundary Value Methods, Gordon and Breach, Amsterdam, 1998.

[9] L. Brugnano, D. Trigiante, Block implicit methods for ODEs, in: D. Trigiante (Ed.), Recent Trends in Numerical Analysis, Nova Science Publishers, New York, 2001, pp. 81–105.

[10] J.C. Butcher, On the implementation of implicit Runge–Kutta methods, BIT 6 (1976) 237–240.

[11] J.C. Butcher, Numerical Methods for Ordinary Differential Equations, Wiley, Chichester, 2003.

[12] J.R. Cash, S. Considine, An MEBDF code for stiff initial value problems, ACM Trans. Math. Software 18 (2) (1992) 142–158. Codes available at: http://www.ma.ic.ac.uk/~jcash/IVP_software/readme.html.

[13] E. Hairer, C. Lubich, M. Roche, The Numerical Solution of Differential-Algebraic Systems by Runge–Kutta Methods, Lecture Notes in Mathematics, vol. 1409, Springer, Berlin, 1989.

[14] E. Hairer, S.P. Nørsett, G. Wanner, Solving Ordinary Differential Equations I, second ed., Springer Series in Computational Mathematics, vol. 8, Springer, Berlin, 1993.

[15] E. Hairer, G. Wanner, Solving Ordinary Differential Equations II, Stiff and Differential-Algebraic Problems, second ed., Springer Series in Computational Mathematics, vol. 14, Springer, Berlin, 1996. Codes available at: http://www.unige.ch/math/folks/hairer/software.html.

[16] P.J. van der Houwen, J.J.B. de Swart, Triangularly implicit iteration methods for ODE-IVP solvers, SIAM J. Sci. Comput. 18 (1997) 41–55.

[17] P.J. van der Houwen, J.J.B. de Swart, Parallel linear system solvers for Runge–Kutta methods, Adv. Comput. Math. 7 (1–2) (1997) 157–181.

[18] F. Iavernaro, F. Mazzia, Solving ordinary differential equations by generalized Adams methods: properties and implementation techniques, Appl. Numer. Math. 28 (1998) 107–126 Code available at: http://pitagora.dm.uniba.it/~mazzia/ode/gamd.html.

[19] B. Leimkuhler, L.R. Petzold, C.W. Gear, Approximation methods for the consistent initialization of differential algebraic equations, SIAM J. Numer. Anal. 28 (1991) 205–226.

[20] H.A. Watts, L.F. Shampine, *A*-stable block one-step methods, BIT 12 (1972) 252–266.

[21] http://pitagora.dm.uniba.it/~testset/