SHORT COMMUNICATIONS

ZAMM · Z. angew. Math. Mech. 73 (1993) 11, 325–329 Akademie Verlag

JAY, L.

Dense Output for Extrapolation Based on the Semi-Implicit Midpoint Rule

MSC (1980): 65L05

1. Introduction

For the resolution of stiff and nonstiff differential equations, extrapolation methods are often used, in particular if a high precision of the solution is required. A serious drawback of these methods is that they are usually not equipped with a dense output formula. In [7], [8] HAIRER and OSTERMANN have provided some extrapolation methods with dense output formulas: the GBS (Gragg-Bulirsch-Stoer)-algorithm, the explicit, implicit and semi-implicit Euler methods.

The aim of this paper is to discuss the theoretical and algorithmic existence of dense output formulas for the extrapolation method based on the semi-implicit midpoint rule due to BADER and DEUFLHARD [1]. Some of the ideas of [7], [8] can be easily adapted to our situation. However, for stiff differential equations some modifications have to be achieved, and the proofs are completely different due to the fact that the remainder of an asymptotic expansion is not bounded independently of the stiffness.

After the presentation of the method (Section 2), we introduce in Section 3 an algorithm for the construction of a dense output formula, adapted to stiff problems. Next we give some theoretical results in relation with Rosenbrock-type methods (Section 4). In Section 5 we study the application of the algorithm to differential-algebraic systems of index 1 as a limit case of very stiff differential equations. Finally we consider some aspects of the implementation (Section 6).

2. The semi-implicit midpoint rule

We consider the system of differential equations

$$y' = f(y), \qquad y(x_0) = y_0 . \tag{2.1}$$

The application of the *semi-implicit midpoint rule* reads [1], [9, p. 146]:

$$(I - hJ)(y_1 - y_0) = hf(y_0), \tag{2.2a}$$

$$(I - hJ)(y_{i+1} - y_i) = -(I + hJ)(y_i - y_{i-1}) + 2hf(y_i),$$

$$i = 1, \dots, n, \tag{2.2b}$$

$$S_h(x_0 + H) = \frac{y_{n+1} + y_{n-1}}{2}, \tag{2.2c}$$

where J is an arbitrary matrix of the same dimension as the system (2.1), H is the basic stepsize, and $h := H/n$ with n even. The value $S_h(x_0 + H)$ is simply the result of one semi-implicit Euler step (2.2a), followed by n steps of the linearly implicit midpoint rule (2.2b), and by a smoothing step (2.2c). We choose an increasing sequence $\{n_j\}_{j \geq 1}$ of even integers, put $h_j := H/n_j$, and define

$$Y_{j1} := S_{h_j}(x_0 + H). \tag{2.3}$$

The error $S_h(x_0 + H) - y(x_0 + H)$ has an asymptotic h^2-expansion [1], [9, Section IV.9, Theorem 9.1], thus these values can be extrapolated with formulas [14, Section 7.2.14]

$$Y_{j,k+1} = Y_{jk} + \frac{Y_{jk} - Y_{j-1,k}}{(n_j/n_{j-k})^2 - 1}. \tag{2.4}$$

Each extrapolation eliminates one power of h^2 [6, Sections II.8 and II.9], so that

$$Y_{jk} - y(x_0 + H) = \begin{cases} O(H^{2k}) & \text{if } J \neq 0, \\ O(H^{2k+1}) & \text{if } J = 0. \end{cases} \tag{2.5}$$

3. Construction of dense output formulas

Assume that the value $y_1 := Y_{\varkappa\varkappa}$ has been accepted as an approximation to the solution at $x_0 + H$. To obtain a continuous solution we construct an Hermite interpolation polynomial by means of approximations to $y(x_0)$, $y(x_0 + H)$, $y'(x_0 + H)$, $y^{(k)}(x_0 + H/2)$, for $k = 0, \dots, \mu$. With $y_h(x)$ denoting the numerical solution y_i of (2.2) at $x = x_0 + ih$, the error $y_h(x) - y(x)$ has a different asymptotic expansion for i even or odd. For this reason, in the algorithm below, the indices involved in building up divided differences and their extrapolated values are required to be of the same parity. This implies that the indices $n_j/2$ have to be either all even or all odd and therefore

$$n_{j+1} - n_j \equiv 0 \bmod (4) \quad \text{for} \quad j = 1, 2, \dots . \tag{3.1}$$

In the implementation we have chosen the sequence (see [1], [2] for other reasons)

$$\{n_j\}_{j \geq 1} = \{2, 6, 10, 14, 22, 34, 50, 70, 98, \dots\} \tag{3.2}$$

which satisfies (3.1). A dense output can be constructed as follows.

Algorithm 1

Step 1: For each $j \in \{1, \dots, \varkappa\}$, we compute approximations to the derivatives of $y(x)$ at the midpoint $x = x_0 + H/2$ and to the first derivative at the endpoint $x = x_0 + H$:

$$d_j^{(k)} := \delta^k \tilde{y}^{(j)}_{n_j/2}/(2h_j)^k \quad \text{for} \quad k = 0, \dots, 2j - 1, \tag{3.3a}$$

$$r_j^{(1)} := \delta y^{(j)}_{n_j}/(2h_j), \tag{3.3b}$$

where $\tilde{y}_i^{(j)} := y_i^{(j)}$, $y_i^{(j)}$ is the approximation to $y(x_i)$ obtained during the computation of Y_{j1} and $\delta y_i := y_{i+1} - y_{i-1}$ denotes the central difference operator ($\delta^0 y_i := y_i$). We mention that no supplementary function evaluation is necessary and we insist on the fact that the values $f(y_i^{(j)})$ are not used, in contrast to the algorithm given in [7]. This is important for stiff problems.

Step 2: We extrapolate $d_j^{(2l-1)}$ $(\varkappa - l)$ times, $d_j^{(2l)}$ $(\varkappa - l - 1)$ times and $r_j^{(1)}$ $(\varkappa - 1)$ times. This yields $d^{(k)}$ and $r^{(1)}$.

Step 3: For given μ, $-1 \leq \mu \leq 2\varkappa - 1$, we define the vector-valued polynomial $P_\mu(\theta)$ of degree $\mu + 3$ by

$$P_\mu(0) := y_0, \qquad P_\mu(1) := y_1, \qquad P'_\mu(1) := Hr^{(1)}, \tag{3.4a, b, c}$$

$$P_\mu^{(k)}(1/2) := H^k d^{(k)} \quad \text{for} \quad k = 0, \dots, \mu. \tag{3.4d}$$

Remark: The above definition of $P_\mu(\theta)$ does not use the first derivative at the left-hand side. If one is interested in a solution which is globally C^1, one can add the condition $P'_\mu(0) := Hl^{(1)}$, where $l^{(1)}$ is the value of $r^{(1)}$ of the previous step. For stiff problems it is not recommended to take $l^{(1)} = f(y_0)$ because the large eigenvalues of the Jacobian of f would amplify errors in y_0.

Theorem 3.1: *If the step number sequence* $\{n_j\}_{j \geq 1}$ *satisfies* (3.1) *then the error of* $P_\mu(\theta)$ *verifies for* $\theta \in [0, 1]$

$$P_\mu(\theta) - y(x_0 + \theta H) = O(H^{2\varkappa}) \quad \textit{for} \quad \mu \geq 2\varkappa - 4. \tag{3.5}$$

Proof: To follow the proof for the GBS-algorithm as given in [8, Theorem 8], the error of y_i must possess an asymptotic expansion in h^2 with coefficients depending on the parity of the index i. Concerning the even indices, this result is contained in [9, Section IV.9, Theorem 9.1, Formula (9.20)]

$$y_h(x) - y(x) = \sum_{j=1}^{N} a_j(x)\, h^{2j} + h^{2N+2} A(x, h) \tag{3.6}$$

and we also have $a_j(x_0) = 0$. For the odd indices we easily obtain a similar result, with different coefficients $b_j(x)$ and $B(x, h)$, but generally $b_j(x_0) \neq 0$ [10, Lemma 4.2].

Since $P_\mu(\theta)$ is a polynomial of degree $\mu + 3$, the error due to the interpolation is of size $O(H^{\mu+4})$, which explains the restriction on μ in (3.5). The divided differences (3.3) use only even or odd indices i of y_i, from which it can be seen that the errors of $d_j^{(k)}$ and $r_j^{(1)}$ have an h_j^2-expansion.

$$d_j^{(k)} = y^{(k)}(x_0 + H/2) + h_j^2 d_{2,k}(x_0 + H/2) + h_j^4 d_{4,k}(x_0 + H/2) + \dots, \tag{3.7 a}$$

$$r_j^{(1)} = y^{(1)}(x_0 + H) + h_j^2 r_2(x_0 + H) + h_j^4 r_4(x_0 + H) + \dots. \tag{3.7b}$$

Thus the extrapolated values satisfy

$$H^k(d^{(k)} - y^{(k)}(x_0 + H/2)) = \begin{cases} O(H^{2\varkappa+1}) & \text{if } k \text{ odd}, \\ O(H^{2\varkappa}) & \text{if } k \text{ even}, \end{cases} \tag{3.8a}$$

$$H(r^{(1)} - y^{(1)}(x_0 + H)) = O(H^{2\varkappa+1}). \tag{3.8b}$$

Numerical example

We have provided the code SODEX of [9] with the above dense output. In Fig. 1 we present the application of this code to the nonstiff Van der Pol equation

$$\begin{pmatrix} y_1' \\ y_2' \end{pmatrix} = \begin{pmatrix} y_2 \\ (1 - y_1^2)\, y_2 - y_1 \end{pmatrix}, \quad \begin{pmatrix} y_1(0) \\ y_2(0) \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \end{pmatrix}, \tag{3.9}$$

with Algorithm 1 using TOL $= 10^{-7}$ (the tolerance), $\mu = 2\varkappa - 3$, and the control of the error of each derivative and of the error due to interpolation (see Section 6). We have plotted the continuous solution and its global error together with each final step-point for $0 \leqq x \leqq 10$.

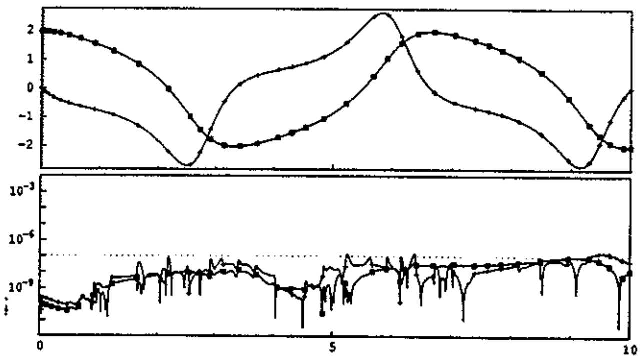

Fig. 1. Solution and error of (3.9)

4. Connection with Rosenbrock-type methods

The main application of the extrapolation method based on (2.2) is the solution of stiff differential equations. Unfortunately, the numerical solution of the semi-implicit midpoint rule (2.2) does not have an asymptotic expansion with smooth coefficients and a remainder which is bounded independently of the stiffness of the problem. In order to get convergence results for the above dense output we consider the entries of the extrapolation method as a Rosenbrock-type method (also named W-method, semi-implicit method or linearly implicit method).

Rosenbrock-type methods applied to (2.1) are defined by [9, Section IV.7]:

$$(I - \gamma_{ii} HJ)\, k_i = f\left(y_0 + H \sum_{j=1}^{i-1} \alpha_{ij} k_j\right) + HJ \sum_{j=1}^{i-1} \gamma_{ij} k_j, \quad i = 1, \dots, s, \tag{4.1a}$$

$$y_1 = y_0 + H \sum_{i=1}^{s} b_i k_i, \tag{4.1b}$$

where the coefficients α_{ij}, γ_{ij} and b_i characterize the method, H and J are as in (2.2). If $J = f'(y_0)$ we obtain *Rosenbrock methods*.

Every element Y_{jk} given by (2.3) and (2.4) is a linear combination of $Y_{j-k+1,1}, \dots, Y_{j1}$ and can be interpreted as numerical solution of a method of type (4.1). The coefficients corresponding to Y_{j1}, depending on $n = n_j$, are [3, Section 5]:

$$\alpha_{ij} = \begin{cases} 1/n & \text{if } j = 1 \text{ and } i \text{ even}, \\ 2/n & \text{if } 1 < j < i \text{ and } i - j \text{ odd}, \\ 0 & \text{else}; \end{cases} \tag{4.2a}$$

$$\gamma_{ij} = \begin{cases} (-1)^{i-j}/n & \text{if } j = 1 \text{ or } j = i, \\ 2(-1)^{i-j}/n & \text{if } 1 < j < i; \end{cases} \tag{4.2b}$$

$$\beta_{ij} = \begin{cases} 1/n & \text{if "} j = 1 \text{ and } i \text{ odd" or } i = j, \\ 2/n & \text{if } 1 < j < i \text{ and } i - j \text{ even}, \\ 0 & \text{else}; \end{cases} \tag{4.2c}$$

$$b_i = \begin{cases} 1/n & \text{if } i = 1 \text{ or } i = n + 1, \\ 2/n & \text{if } 1 < i < n + 1 \text{ and } i \text{ odd}, \\ 0 & \text{else}. \end{cases} \tag{4.2d}$$

Here we have included the expressions for $\beta_{ij} := \gamma_{ij} + \alpha_{ij}$, which are usually used in the presentation of the order conditions for the method (4.1) with $J = f'(y_0)$. The formulas (4.2) can easily be obtained from (2.2) by putting

$$k_1 := (y_1 - y_0)/h, \tag{4.3a}$$

$$k_{i+1} := (y_{i+1} - y_{i-1})/(2h), \qquad i = 1, \dots, n. \tag{4.3b}$$

A general expression of a continuous solution for our method written in the form (4.1) is

$$Y_{j1}(\theta) = y_0 + H \sum_{i=1}^{n_j+1} b_i(\theta, n_j)\, k_i \quad \text{for} \quad 0 \leqq \theta \leqq 1, \tag{4.4}$$

where $b_i(\theta, n)$ are polynomials in θ such that $b_i(0, n) = 0$, $b_i(1, n) = b_i$. The values $Y_{jk}(\theta)$ obtained by (4.4) and the recursion (2.4) are said to have *order p* if

$$Y_{jk}(\theta) - y(x_0 + \theta H) = O(H^{p+1}) \quad \text{for} \quad 0 \leqq \theta \leqq 1. \tag{4.5}$$

Necessary and sufficient conditions on $b_i(\theta, n)$ to obtain the same order as at the final value $Y_{jk}(1) = Y_{jk}$ in formula (2.5) have been derived in [10, Theorem 3.1]. These conditions are identical to those for the GBS-algorithm [8, Section 5], excepted the supplementary assumption $b_1(\theta, n) = b_1(\theta)/n$ for $k \geqq 2$, so that the remarks in the aforementioned article apply. We omit the proof which is long and technical. The main idea is the application of "simplifying assumptions". Details are given in [10, Chapter 3].

We are now interested in studying the error of the dense output formula for singularly perturbed problems

$$\left.\begin{aligned} y' &= f(y, z), & y(x_0) &= y_0, \\ \varepsilon z' &= g(y, z), & z(x_0) &= z_0. \end{aligned}\right\} \tag{4.6}$$

It seems to be very difficult to obtain detailed knowledge of the errors $y_h(x) - y(x)$ and $z_h(x) - z(x)$ which enter in the analysis of the error estimation for the finite differences (3.3). We therefore only analyse the limit case $\varepsilon = 0$ in the next section, which already gives much insight into the numerical behaviour of the method, but we mention that in practice the method is not well-suited for the resolution of differential-algebraic systems.

5. Dense output for differential-algebraic equations

We consider the differential-algebraic system

$$\begin{aligned} y' &= f(y, z), & y(x_0) &= y_0, \\ 0 &= g(y, z), & z(x_0) &= z_0, \end{aligned} \tag{5.1}$$

and we make the assumption that the initial values are consistent ($g(y_0, z_0) = 0$) and that g_z is invertible in the neighbourhood of the solution (index 1). The application of our method (2.2) to (5.1) can be achieved by first applying it to (4.6) with

$$J := \begin{pmatrix} f_y & f_z \\ \varepsilon^{-1} g_y & \varepsilon^{-1} g_z \end{pmatrix} \tag{5.2}$$

the Jacobian of the system (4.6) where all partial derivatives are evaluated at the initial value (y_0, z_0), and then by taking the limit

$\varepsilon \to 0$. It leads to

$$\begin{pmatrix} I - hf_y & -hf_z \\ -hg_y & -hg_z \end{pmatrix} \begin{pmatrix} y_1 - y_0 \\ z_1 - z_0 \end{pmatrix} = h \begin{pmatrix} f(y_0, z_0) \\ g(y_0, z_0) \end{pmatrix}, \tag{5.3a}$$

$$\begin{pmatrix} I - hf_y & -hf_z \\ -hg_y & -hg_z \end{pmatrix} \begin{pmatrix} y_{i+1} - y_i \\ z_{i+1} - z_i \end{pmatrix}$$
$$= - \begin{pmatrix} I + hf_y & hf_z \\ hg_y & hg_z \end{pmatrix} \begin{pmatrix} y_i - y_{i-1} \\ z_i - z_{i-1} \end{pmatrix} + 2h \begin{pmatrix} f(y_i, z_i) \\ g(y_i, z_i) \end{pmatrix},$$

$$i = 1, \ldots, n, \tag{5.3b}$$

$$S_h(x_0 + H) = \frac{1}{2} \begin{pmatrix} y_{n+1} + y_{n-1} \\ z_{n+1} + z_{n-1} \end{pmatrix}. \tag{5.3c}$$

We extrapolate the values $S_{h_j}(x_0 + H) = [Y_{j1}, Z_{j1}]^{\mathrm{T}}$ with (2.4) and we obtain

$$\begin{aligned} Y_{jk} - y(x_0 + H) &= O(H^{r_{jk}+1}), \\ Z_{jk} - z(x_0 + H) &= O(H^{s_{jk}}), \end{aligned} \tag{5.4}$$

where r_{jk} (resp. s_{jk}) is the *differential-algebraic order* of the y-component (resp. z). The global error in the two components is of size $O(H^{p_{jk}})$ where $p_{jk} = \min(r_{jk}, s_{jk})$.

For the convenience of the reader the results are only presented here with some indications for their proof. We do not present all the details.

Theorem 5.1 ([4]): *For the standard sequence* (3.2) *the differential-algebraic orders are given in the following Tables* I, II *and* III (* *indicates an unknown order between* 5 *and* 7):

1					
1	3				
1	3	5			
1	3	5	*		
1	3	5	*	*	
⋮	⋮	⋮	⋮	⋮	⋱

Table I. *Orders* r_{jk}

2					
2	4				
2	4	5			
2	4	5	5		
2	4	5	5	5	
⋮	⋮	⋮	⋮	⋮	⋱

Table II. *Orders* s_{jk}

1					
1	3				
1	3	5			
1	3	5	5		
1	3	5	5	5	
⋮	⋮	⋮	⋮	⋮	⋱

Table III. *Orders* p_{jk}.

Outline of the proof: We consider the scheme (5.3) as a Rosenbrock method whose coefficients are given by (4.2) and we verify the order conditions derived in [12] and [13]. For this purpose we need to calculate the matrix $W = (w_{ij}) := \beta^{-1} = (\beta_{ij})^{-1}$. As β is triangular we easily find

$$w_{ij} = n^2(-1)^{(i-j)/2}\beta_{ij}$$
$$= \begin{cases} n(-1)^{(i-j)/2} & \text{if "} j = 1 \text{ and } i \text{ odd" or } i = j, \\ 2n(-1)^{(i-j)/2} & \text{if } 1 < j < i \text{ and } i - j \text{ even}, \\ 0 & \text{else.} \end{cases} \tag{5.5}$$

□

For a dense output (abbreviated d.o.) we define the differential-algebraic orders $r^{\text{d.o.}}_{jk}$, $s^{\text{d.o.}}_{jk}$ and $p^{\text{d.o.}}_{jk}$ similarly as in (5.4). $p^{\text{d.o.}}_{jk}$ as high as possible is desirable and theoretically $p^{\text{d.o.}}_{jk} = [(p_{jk} + 1)/2]$ is at least attainable [11, Theorem 4], where the function $[\cdot]$ denotes "the integer part of".

Theorem 5.2: *Defining the dense output with Algorithm* 1, *we obtain for the sequence* (3.2) *the following differential-algebraic orders:*

1					
1	3				
1	3	3			
1	3	3	3		
1	3	3	3	3	
⋮	⋮	⋮	⋮	⋮	⋱

Table IV. *Orders* $r^{\text{d.o.}}_{jk}$.

2					
2	3				
2	3	3			
2	3	3	3		
2	3	3	3	3	
⋮	⋮	⋮	⋮	⋮	⋱

Table V. *Orders* $s^{\text{d.o.}}_{jk}$.

1					
1	3				
1	3	3			
1	3	3	3		
1	3	3	3	3	
⋮	⋮	⋮	⋮	⋮	⋱

Table VI. *Orders* $p^{\text{d.o.}}_{jk}$.

Remark: In contrast to the semi-implicit Euler method (see [5]), the numerical solution of the semi-implicit midpoint rule (5.3) does not have a perturbed asymptotic expansion, therefore the proof in [8] can not be extended.

Outline of the proof: Because of

$$r^{\text{d.o.}}_{jk} = \min \left[\min_{l=0,1} (l + \operatorname{order}(Y^{(l)}_{jk}(1))), \right.$$
$$\left. \min_{l=0,\ldots,\mu} (l + \operatorname{order}(Y^{(l)}_{jk}(1/2))) \right], \tag{5.6a}$$

$$s^{\text{d.o.}}_{jk} = \min \left[\min_{l=0,1} (l + \operatorname{order}(Z^{(l)}_{jk}(1))), \right.$$
$$\left. \min_{l=0,\ldots,\mu} (l + \operatorname{order}(Z^{(l)}_{jk}(1/2))) \right], \tag{5.6b}$$

it remains to find the orders of the approximations:

- $Y^{(l)}_{jk}(1/2)$ of $y^{(l)}(x_0 + H/2)$, for $l = 0, \ldots, \mu$;
- $Z^{(l)}_{jk}(1/2)$ of $z^{(l)}(x_0 + H/2)$, for $l = 0, \ldots, \mu$;
- $Y^{(1)}_{jk}(1)$ of $y^{(1)}(x_0 + H)$;
- $Z^{(1)}_{jk}(1)$ of $z^{(1)}(x_0 + H)$,

For a continuous solution $[Y(\theta), Z(\theta)]^{\mathrm{T}}$ the conditions to obtain a differential-algebraic oder r (resp. s) for the y-component (resp. z) are

$$\sum_i b_i(\theta)\, \Phi_i(t) = \frac{\theta^{\varrho(t)}}{\gamma(t)}, \tag{5.7}$$

for all $t \in \mathrm{DAT}_y$, $\varrho(t) \leqq r$, (resp for all $t \in \mathrm{DAT}_z$, $\varrho(t) \leqq s - 1$) (see [11], [12], [13], [9, Section VI.3] for the definitions of the DAT-trees t and the related quantities $\Phi_i(t)$, $\varrho(t)$ and $\gamma(t)$). As mentioned in [9, Section VI.3, Lemma 3.9], for Rosenbrock methods only a subset of the DAT-trees has to be considered. For $[Y^{(l)}(\theta), Z^{(l)}(\theta)]^{\mathrm{T}}$, $l \geqq 0$, the order conditions become

$$\frac{\mathrm{d}^l}{\mathrm{d}\theta^l} \left[\sum_i b_i(\theta)\, \Phi_i(t) \right] = \frac{\mathrm{d}^l}{\mathrm{d}\theta^l} \left(\frac{\theta^{\varrho(t)-l}}{\gamma(t)} \right), \tag{5.8a}$$

or equivalently

$$\sum_i b^{(l)}_i(\theta)\, \Phi_i(t)$$
$$= \begin{cases} \varrho(t) \ldots (\varrho(t) - l + 1) \dfrac{\theta^{\varrho(t)-l}}{\gamma(t)} & \text{if } \varrho(t) \geqq l, \\ 0 & \text{else.} \end{cases} \tag{5.8b}$$

Instead of directly verifying the order conditions (5.7) for $[Y_{jk}(\theta), Z_{jk}(\theta)]^{\mathrm{T}}$ with polynomials $b_{i,jk}(\theta)$, we can simply compute the corresponding coefficients $b^{(l)}_{i,j1}(1/2)$ for $l = 0, \ldots, \mu$, and $b^{(l)}_{i,j1}(1)$ for $l = 0, 1$, then substitute the values 1/2 and 1 respectively for θ in (5.8), and finally extrapolate the left side of (5.8b) according to (2.4). We have

$$b_{i,j1}(1/2) = \alpha_{n_j/2+1,i}, \tag{5.9}$$

and we do not need to verify the order conditions for the derivatives of the l-th order with $l \geqq 1$, because their contribution in (5.6) cannot decrease the differential-algebraic orders computed only with the remaining 0-th order derivatives. □

Compared to Table III we observe in Table VI a loss of accuracy and it is interesting to investigate whether a modification of the dense output of Section 3 can improve the behaviour. Several numerical tests have led to the following

Algorithm 2

The modifications with respect to Algorithm 1 (Section 3) are as follows: in *step* 1 we use

$$\tilde{y}^{(j)}_i := \frac{y^{(j)}_{i+1} + y^{(j)}_{i-1}}{2} \tag{5.10}$$

and we consider formula (3.3a) only for $k = 0, \ldots, 2j - 2$. In *step* 2 we extrapolate $d_j^{(2l-1)}$ $(\varkappa - l - 1)$ times. The restriction on μ in *step* 3 is $(-1 \leqq \mu \leqq 2\varkappa - 2)$ and if $\varkappa = 2$ we restrict μ in $\{-1, 0\}$.

Theorem 5.3: *For the sequence* (3.2) *and with Algorithm* 2 *the differential-algebraic orders of the dense output are given by*

1					
1	3				
1	3	4			
1	3	4	4		
1	3	4	4	4	
⋮	⋮	⋮	⋮	⋮	⋱

Table VII. *Orders* $r_{jk}^{\text{d.o.}}$.

2					
2	3				
2	3	4			
2	3	4	4		
2	3	4	4	4	
⋮	⋮	⋮	⋮	⋮	⋱

Table VIII. *Orders* $s_{jk}^{\text{d.o.}}$.

1					
1	3				
1	3	4			
1	3	4	4		
1	3	4	4	4	
⋮	⋮	⋮	⋮	⋮	⋱

Table IX. *Orders* $p_{jk}^{\text{d.o.}}$.

Outline of the proof: Instead of the result of Theorem 3.1, we get for Algorithm 2 applied to nonstiff problems and for $\varkappa \geqq 3$

$$P_\mu(\theta) - y(x_0 + \theta H) = O(H^{2\varkappa-1}) \quad \text{for} \quad \mu \geqq 2\varkappa - 5\,, \tag{5.11}$$

due to the fact that

$$H^k(d^{(k)} - y^{(k)}(x_0 + H/2)) = O(H^{2\varkappa-1}) \quad \text{if} \quad k \text{ odd}\,. \tag{5.12}$$

Now we can follow the proof of the previous Theorem 5.2 with coefficients

$$b_{i,j1}(1/2) = \frac{\alpha_{n_j/2+2,i} + \alpha_{n_j/2,i}}{2}\,, \tag{5.13a}$$

$$b_{i,j1}^{(1)}(1/2) = \frac{\alpha_{n_j/2+3,i} - \alpha_{n_j/2-1,i}}{4/n_j}\,, \tag{5.13b}$$

$$b_{i,j1}^{(1)}(1) = \begin{cases} 1 & \text{if} \quad i = n_j + 1\,, \\ 0 & \text{else}\,. \end{cases} \tag{5.13c}$$

□

Numerical example: We have applied the code SODEX to the pendulum problem in index 1 formulation

$$\begin{bmatrix} y_1' \\ y_2' \\ y_3' \\ y_4' \\ y_5' \end{bmatrix} = \begin{bmatrix} y_3 \\ y_4 \\ -y_1y_5 \\ -y_2y_5 - 1 \\ y_3^2 + y_4^2 - y_2 - y_5 \end{bmatrix}, \quad \begin{bmatrix} y_1(0) \\ y_2(0) \\ y_3(0) \\ y_4(0) \\ y_5(0) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{5.14}$$

with Algorithm 2 using TOL $= 10^{-7}$, $\mu = 2\varkappa - 4$, and the control of the error of each derivative and of the error due to interpolation (see Section 6). Fig. 2 shows the solution and the global errors of the components y_1, y_3 and y_5 for $0 \leqq x \leqq 10$.

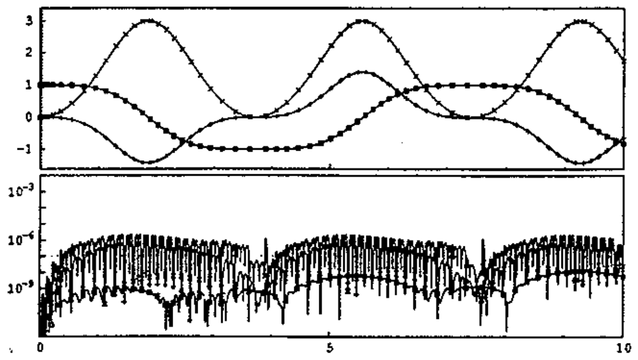

Fig. 2. Solution and error of (5.14)

6. Implementation

The code SODEX, based on the semi-implicit midpoint rule (2.2) and on the extrapolation formula (2.4), is written for problems of the form

$$My' = f(x, y)\,, \qquad y(x_0) = y_0\,, \tag{6.1}$$

where M is a constant square matrix, which may be singular. The implementation of the dense output is similar to the one described in [7] for the GBS-algorithm. The vector-valued polynomial $P_\mu(\theta)$, defined by (3.4), has the representation

$$P_\mu(\theta) = P_H(\theta) + \theta(1-\theta)^2\,(a_0 + a_1(\theta - \tfrac{1}{2}) + \ldots + a_\mu(\theta - \tfrac{1}{2})^\mu)\,, \tag{6.2}$$

where $P_H(\theta)$ is the Hermite polynomial of degree 2 defined by the first 3 conditions of (3.4). The a_j are calculated by building up the finite difference table for y_0, y_1, $Hr^{(1)}$, $d^{(0)}, \ldots, H^\mu d^{(\mu)}/\mu!$ and we find

$$a_0 = 8\left(d^{(0)} - P_H\left(\frac{1}{2}\right)\right), \tag{6.3a}$$

$$a_1 = 8\left(Hd^{(1)} - P_H^{(1)}\left(\frac{1}{2}\right) + \frac{1}{4}\,a_0\right), \tag{6.3b}$$

$$a_2 = 8\left(\frac{1}{2!}\left(H^2d^{(2)} - P_H^{(2)}\left(\frac{1}{2}\right)\right) + \frac{1}{4}\,a_1 + \frac{1}{2}\,a_0\right), \tag{6.3c}$$

$$a_k = 8\left(\frac{1}{k!}\,H^kd^{(k)} + \frac{1}{4}\,a_{k-1} + \frac{1}{2}\,a_{k-2} - a_{k-3}\right)$$

$$\text{for} \quad k = 3, \ldots, \mu\,. \tag{6.3d}$$

We can also take into account the error due to interpolation. The error of $P_\mu(\theta)$ is given by

$$\theta(1-\theta)^2\left(\theta - \frac{1}{2}\right)^{\mu+1} \frac{y^{(\mu+4)}(\xi)}{(\mu+4)!}\,H^{\mu+4}\,, \tag{6.4}$$

where $x_0 < \xi < x_0 + H$ (ξ may be different for each component of y). We estimate the norm of this error for $P_{\mu-1}(\theta)$ if $\mu \geqq 0$ by

$$\|P_\mu(\theta) - P_{\mu-1}(\theta)\| = \left|\theta(1-\theta)^2\left(\theta - \frac{1}{2}\right)^\mu\right| \cdot \|(a_\mu\|\,. \tag{6.5}$$

This function (6.5) is maximal at

$$\theta_{\mu,\text{int}} = \frac{1}{2} - \frac{1 + \sqrt{1 + 4\mu(\mu+3)}}{4(\mu+3)}\,. \tag{6.6}$$

We thus use

$$\text{errint} = \|P_\mu(\theta_{\mu,\text{int}}) - P_{\mu-1}(\theta_{\mu,\text{int}})\| \tag{6.7}$$

as error estimator, which together with (6.4) (μ replaced by $\mu - 1$) leads to the stepsize prediction formula (see [6, Section II.4])

$$H_{\text{int}} = H\left(\frac{\text{TOLINT}}{\text{errint}}\right)^{1/(\mu+3)}, \tag{6.8}$$

where TOLINT is the tolerance of the interpolation error. In SODEX we have chosen TOLINT $= 10 \cdot$ TOL. If $\mu = -1$ an estimated error is given by

$$|\theta(1-\theta)| \cdot \|y_1 - y_0 - Hr^{(1)}\| \tag{6.9}$$

which is maximal at $\theta_{-1,\text{int}} = 1/2$.

Also with control of the interpolation error, the numerical results were sometimes disappointing. The reason lies in the stiff accuracy of the method with $J = f'(y_0)$ (i.e. $\beta_{n+1,i} = b_i$; [9, pp. 448–450]). Stiffly accurate methods are tuned to give a highly accurate (even asymptotically exact for Prothero-Robinson-like examples) result at the end of the integration interval. Therefore it is impossible to get a dense output of the same accuracy without reducing the stepsize. As a remedy we propose to control also the error contribution of each k-th order derivative $D^{(k)}$ entering in the definition of the dense output. We collect the terms of (6.2) multiplying $D^{(k)}$ and we find that the worst error contribution is given by

$$\text{errder}\,(D^{(k)}) = \max_{\theta\in[0,1]} |P_{\mu,D^{(k)}}(\theta)| \cdot \frac{1}{k!} \cdot \text{err}\,(H^kD^{(k)})\,, \tag{6.10}$$

where $P_{\mu, D^{(k)}}(\theta)$ is a polynomial in θ different for each μ and $D^{(k)}$, and err $(H^k D^{(k)})$ is the norm of the error of $H^k D^{(k)}$. For example, if $D^{(k)} = r^{(1)}$ we find

$$P_{\mu, r^{(1)}}(\theta) = 2^{\mu+1}\theta(\theta - 1)\,(\theta - \tfrac{1}{2})^{\mu+1} \tag{6.11}$$

whose absolute value is maximal at

$$\theta_{\mu, r^{(1)}} = \frac{1}{2} \pm \frac{1}{2}\sqrt{1 - \frac{2}{\mu + 3}}\,. \tag{6.12}$$

If we do ($v \geqq 2$) extrapolations for the computation of $D_{vv}^{(k)}$, we can use the following error estimator for the m-th component of $H^k D_{v,v-1}^{(k)}$

$$H^k \cdot |D_{vv}^{(k),m} - D_{v,v-1}^{(k),m}|\,. \tag{6.13}$$

For small v it has been numerically observed that this estimator can be too pessimistic, and we have replaced it by

$$(H^k \cdot |D_{vv}^{(k),m} - D_{v,v-1}^{(k),m}|)^{(p+1)/(q+1)}\,, \tag{6.14}$$

where p (resp. q) is the order of $H^k D_{vv}^{(k)}$ (resp. $H^k D_{v,v-1}^{(k)}$). Here we have $p = q + 2$. This new quantity (6.14) now behaves (in terms of power of H) like the error of $H^k D_{vv}^{(k)}$.

As before a stepsize prediction formula is given by

$$H_{D^{(k)}} = H\left(\frac{\text{TOLDER}}{\text{errder}\,(D^{(k)})}\right)^{1/(p+1)}, \tag{6.15}$$

where TOLDER is the tolerance of the error contribution of a derivative, and formula (6.14) is used for the computation of err $(H^k D^{(k)})$ entering in (6.10). In SODEX we have chosen TOLDER $= 10 \cdot$ TOL.

Numerical example: In Fig. 3 we present the result of the code SODEX when applied to a stiff problem, the reaction equation of Robertson (1966) (see [9, Section IV.10])

$$\left.\begin{aligned} \begin{pmatrix} y_1' \\ y_2' \\ y_3' \end{pmatrix} &= \begin{pmatrix} -0.04y_1 + 10^4 y_2 y_3 \\ 0.04y_1 - 10^4 y_2 y_3 - 3 \cdot 10^7 y_2^2 \\ 3 \cdot 10^7 y_2^2 \end{pmatrix}, \\ \begin{pmatrix} y_1(0) \\ y_2(0) \\ y_3(0) \end{pmatrix} &= \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \end{aligned}\right\} \tag{6.16}$$

with Algorithm 2, TOL $= 10^{-9}$ using $\mu = 2\varkappa - 4$, and the control of the error and of each derivative and of the error due to interpolation. In the upper picture we have plotted the solution for $0 \leqq x \leqq 40$. The lower picture shows the global errors of each component.

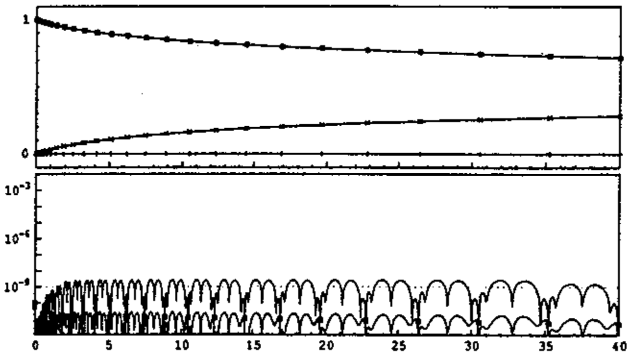

Fig. 3. Solution and error of (6.16)

The three numerical examples presented clearly show the behaviour of the method and of the associated dense output formulas. For stiff and differential-algebraic problems the dense output is less accurate than the solution at the grid points, but as mentioned before we can control its accuracy. It explains the "waves" in the global error of the second and third examples.

Acknowledgements: I wish to thank E. Hairer for many helpful remarks and ideas, A. Ostermann and G. Wanner for their advices, and W. Hundsdorfer for a careful reading of the manuscript.

References

1 Bader, G.; Deuflhard, P.: A semi-implicit mid-point rule for stiff systems of ordinary differential equations. Numer. Math. **41** (1983), 373–398.

2 Deuflhard, P.: Recent progress in extrapolation methods for ordinary differential equations. SIAM Review **27** (1985), 505–535.

3 Hairer, E.; Bader, G.; Lubich, Ch.: On the stability of semi-implicit methods for ordinary differential equations. BIT **22** (1982), 211–232.

4 Hairer, E.; Lubich, Ch.: On extrapolation methods for stiff and differential-algebraic equations. In Strehmel, K. (ed.): Numerical treatment of differential equations. Teubner-Texte zur Mathematik. Band 104. Leipzig 1987, pp. 64–73.

5 Hairer, E.; Lubich, Ch.: Extrapolation at stiff differential equations. Numer. Math. **52** (1988), 377–400.

6 Hairer, E.; Norsett, S. P.; Wanner, G.: Solving ordinary differential equations. I: Nonstiff problems. Springer-Verlag, Berlin 1987.

7 Hairer, E.; Ostermann, A.: Dense output for the GBS extrapolation method. In Cash, J. R.; Gladwell, I. (eds.): Computational ordinary differential equation. Clarendon Press, Oxford 1992, pp. 107–114.

8 Hairer, E.; Ostermann, A.: Dense output for extrapolation methods. Numer. Math. **58** (1990), 419–439.

9 Hairer, E.; Wanner, G.: Solving ordinary differential equations. II: Stiff and differential-algebraic problems. Computational Mathematics, Vol. 14, Springer-Verlag, Berlin 1991.

10 Jay, L.: Construction d'une solution continue pour une méthode d'extrapolation: la méthode du point-milieu semi-implicite, théorie et pratique. Diploma Thesis, Université de Genève 1990.

11 Ostermann, A.: Continuous extensions of Rosenbrock-type methods. Computing **44** (1990), 59–68.

12 Roche, M.: Rosenbrock methods for differential algebraic equations. Numer. Math. **52** (1988), 45–63.

13 Roche, M.: Méthodes de Runge-Kutta et Rosenbrock pour équations différentielles algébriques et systèmes différentiels raides. Doctoral thesis, Université de Genève 1988.

14 Stoer, J.; Bulirsch, R.: Introduction to numerical analysis. Springer-Verlag, New York 1980.

Received June 28, 1991

Address: Dr Laurent Jay, Université de Genève, Département de mathématiques, Rue du Lièvre 2–4, Case postale 240, CH-1211 Genève 24, Switzerland