Numer. Math. (1999) 83: 53–85

Numerische Mathematik

New look-ahead Lanczos-type algorithms for linear systems

C. Brezinski[1], **M. Redivo Zaglia**[2], **H. Sadok**[3]

[1] Laboratoire d'Analyse Numérique et d'Optimisation, Université des Sciences et Technologies de Lille, F-59655 Villeneuve d'Ascq Cedex, France; e-mail: Claude.Brezinski@univ-lille1.fr
[2] Dipartimento di Elettronica e Informatica, Università degli Studi di Padova, via Gradenigo 6/a, I-35131 Padova, Italy; e-mail: michela@dei.unipd.it
[3] Laboratoire de Mathematiques Appliquées, Université du Littoral, Centre Universitaire de la Mi-Voix, 50 rue F. Buisson, F-62228 Calais, France; e-mail: sadok@lma.univ-littoral.fr

Received September 2, 1997 / Revised version received July 24, 1998

Summary. A breakdown (due to a division by zero) can arise in the algorithms for implementing Lanczos' method because of the non-existence of some formal orthogonal polynomials or because the recurrence relationship used is not appropriate. Such a breakdown can be avoided by jumping over the polynomials involved. This strategy was already used in some algorithms such as the MRZ and its variants.

In this paper, we propose new implementations of the recurrence relations of these algorithms which only need the storage of a fixed number of vectors, independent of the length of the jump. These new algorithms are based on Horner's rule and on a different way for computing the coefficients of the recurrence relationships. Moreover, these new algorithms seem to be more stable than the old ones and they provide better numerical results.

Numerical examples and comparisons with other algorithms will be given.

Mathematics Subject Classification (1991): 65F10, 65F25

Lanczos method is among the most popular methods for solving a system of linear equations. It is a projection method on a Krylov subspace and it can be implemented via various recurrence relationships. Unfortunately, when computing the coefficients of these relations, a division by zero, called a *breakdown*, can occur and the process has normally to be stopped. So, a quite important problem is to be able to continue the implementation of Lanczos method in such a situation. Several procedures for that purpose appeared in

the literature these last few years. They consist either in jumping over the sigularities, a procedure called *look-ahead* [30], or in going around them, which is called *look-around* [21], or in filling the gaps by biorthogonal polynomials, a technique called ALA (Avoiding Look-Ahead) [1]. In this paper, we will propose new algorithms for treating this problem by a look-ahead procedure. They are based on the algorithms given in [10, 12]. The main property of these new look-ahead algorithms is that they need to store only a fixed number of vectors instead of a number of vectors depending on the length of the jump over the singularities. All the other algorithms known before have this drawback.

Although solving a system of linear equations by Lanczos method is a purely linear algebra problem, we will make use, in the sequel, of the theory of formal orthogonal polynomials (FOP). Of course, one can like or not such an approach but, anyway, FOP enter into the business in a very natural way and they even already appear in the papers of Lanczos himself [28, 29]. Moreover, they lead to a quite simple solution of the problems related to breakdowns and, obviously, some algorithms could not have been obtained by pure linear algebra techniques. Some authors have also obtained quite similar algorithms on the basis of known results on Padé approximants and continued fractions. However, this approach presents an unnecessary complication since only the denominators of the Padé approximants (which are, in fact, the FOP, after reversing the numbering of the coefficients) are used [22–24].

Let us now relate the algorithms given below with other algorithms. As explained above, there exist several recurrence relationships for implementing Lanczos method. They can all be derived, in a very natural way, from the theory of FOP. In these algorithms, we make use of two families of FOP: $\{P_k\}$ and $\{P_k^{(1)}\}$ (or the family of proportional polynomials $\{Q_k\}$). The polynomials P_k will be related to the residuals $r_k = b - Ax_k$ of Lanczos method by $r_k = P_k(A)r_0$, while the polynomials $P_k^{(1)}$ (or the polynomials Q_k) will define the auxiliary vectors $z_k = P_k^{(1)}(A)r_0$ (or $z_k = Q_k(A)r_0$).

Let us begin by the case where no breakdown occurs. In the algorithm Lanczos/Orthores, P_{k+1} is computed from P_k and P_{k-1}. In Lanczos/Orthodir, P_{k+1} is obtained from P_k and $P_k^{(1)}$, and $P_{k+1}^{(1)}$ from $P_k^{(1)}$ and $P_{k-1}^{(1)}$. In Lanczos/Orthomin (equivalent to BCG), P_{k+1} is computed from P_k and Q_k, and Q_{k+1} from P_{k+1} and Q_k. Other (new) algorithms can be found in [14].

When a breakdown occurs, it means that either some orthogonal polynomials do not exist (a situation called *true* breakdown) or that they cannot be computed by the recurrence relationship used (which is called a *ghost* breakdown) [9]. The remedy consists of jumping over these polynomials (a strategy called *look-ahead*) and computing only the other polynomials by means of slightly more complicated recurrence relationships. The only al-

gorithm where ghost breakdowns cannot occur is Lanczos/Orthodir. Using a look-ahead strategy in this algorithm gives rise to the algorithm called MRZ [12].

In this algorithm, it is necessary to compute scalar products of the form $(y, p(A)q(A)u_k)$ where u_k is either r_k or z_k, p is a polynomial of degree equal to the dimension of the Krylov subspace related to r_k, and q is a polynomial of degree depending on the length of the jump. We also have to compute vectors of the form $s(A)u_k$ where s is also a polynomial whose degree depends on the jump. Usually, $s(A)u_k$ is computed as a combination of the inner vectors $A^i u_k$. Using Horner's rule allows to compute the coefficients of s one by one instead of computing them simultaneously. So, the linear combination $s(A)u_k$ is obtained recursively which leads to an algorithm, called HMRZ, where only a fixed number of vectors have to be stored instead of a number of vectors depending on the degree of s. For reducing the amount of work in computing the preceding scalar products, we write them under the form $(p(A^{\mathrm{T}})y, q(A)u_k)$, where $p(A^{\mathrm{T}})y$ is updated at each iteration. We also remark that, now, $p(A^{\mathrm{T}})$ is applied to a vector independent of the iteration. Choosing $p \equiv P_k^{(1)}$, we obtain a more stable variant of the MRZ, called the MRZ-stab, where the vectors $P_k^{(1)}(A^{\mathrm{T}})y$ are computed recursively.

Combining this procedure with Horner's rule gives the HMRZ-stab. In the case of breakdowns, using the polynomials $P_k^{(1)}$ instead of the polynomials Q_k in Lanczos/Orthomin is called the BMRZ [10]. Doing the same kind of treatment transforms it into the algorithms respectively called HBMRZ and HBMRZ-stab below. Finally, if P_{k+1} and $P_{k+1}^{(1)}$ are both obtained from P_k and $P_k^{(1)}$, we have, in the case of breakdowns, an algorithm called SMRZ [10]. Treating it similarly leads to the HSMRZ and the HSMRZ-stab.

The difference between the approach of [9–13] and that of [22–24] consists in the fact that, in the first case, the polynomial q (and p in the non-stab versions) are expressed in the power basis while, in the second case the product pq is written in a basis formed by polynomials satisfying a three-term recurrence relationship (see Formula (2.10) and Theorem 2.7 of [23]).

In Sect. 1, we give the necessary background on Lanczos-type algorithms and formal orthogonal polynomials. Section 2 deals with breakdowns. In Sect. 3, the algorithhm MRZ for treating exact breakdowns is presented. Our new algorithms are explained in Sects. 4 to 7. In Sect. 8, these algorithms are compared with other methods in terms of storage and cost. The paper ends by a section with numerical results showing the efficiency of our algorithms.

1. Lanczos type algorithms and orthogonal polynomials

Let us consider a system of linear equations in $\mathbb{C}^n$.

$$Ax = b$$

where A is a nonsingular matrix.

Lanczos' method [28,29] for solving this system consists of constructing a sequence of vectors (x_k) as follows

- choose two arbitrary nonzero vectors x_0 and y in $\mathbb{C}^n$
- set $r_0 = b - Ax_0$
- determine x_k such that

$$x_k - x_0 \in E_k = \text{span}(r_0, Ar_0, \ldots, A^{k-1}r_0) \tag{1}$$
$$r_k = b - Ax_k \perp F_k = \text{span}(y, A^*y, \ldots, A^{*^{k-1}}y)$$

where A^* is the conjugate transpose of A.

$x_k - x_0$ can be written as

$$x_k - x_0 = -a_1^{(k)}r_0 - \cdots - a_k^{(k)}A^{k-1}r_0.$$

Multyplying both sides by A, adding and subtracting b, we obtain

$$r_k = r_0 + a_1^{(k)}Ar_0 + \cdots + a_k^{(k)}A^k r_0.$$

The orthogonality conditions above give

$$(A^{*^i}y, r_k) = 0 \qquad \text{for } i = 0, \ldots, k-1 \tag{2}$$

which is a system of k linear equations in the k unknowns $a_1^{(k)}, \ldots, a_k^{(k)}$. This system is nonsingular only if r_0, Ar_0,..., $A^{k-1}r_0$ are linearly independent and y, A^*y,..., $A^{*^{k-1}}y$ also, or, equivalently, if the following Hankel determinant, denoted $H_k^{(1)}$, is different from zero

$$H_k^{(1)} = \begin{vmatrix} (y, Ar_0) & (y, A^2r_0) & \cdots & (y, A^k r_0) \\ (y, A^2r_0) & (y, A^3r_0) & \cdots & (y, A^{k+1}r_0) \\ \vdots & \vdots & & \vdots \\ (y, A^k r_0) & (y, A^{k+1}r_0) & \cdots & (y, A^{2k-1}r_0) \end{vmatrix}.$$

Under the assumption that $\forall k$, $H_k^{(1)} \neq 0$, the two conditions (1) uniquely determine x_k.

An interesting property of Lanczos' method is its finite termination, namely that $\exists k \leq n$ such that $r_k = 0$ and $x_k = x = A^{-1}b$ [14].

If we set

$$P_k(\xi) = 1 + a_1^{(k)}\xi + \ldots + a_k^{(k)}\xi^k = 1 + \xi R_{k-1}(\xi)$$

then we have

$$r_k = P_k(A)r_0 \tag{3}$$

and

$$x_k = x_0 - R_{k-1}(A)r_0.$$

Moreover, if we define the linear functional c on the space of polynomials by

$$c(\xi^{\mathrm{i}}) = (y, A^i r_0) \;=\; c_i, \qquad i = 0, 1, \ldots \tag{4}$$

then, the orthogonality conditions (2) can be written as

$$c(\xi^{\mathrm{i}} P_k(\xi)) = 0 \quad \text{for } i = 0, \ldots, k-1.$$

These relations show that P_k is the polynomial of degree at most k belonging to the family of formal orthogonal polynomials with respect to c [7]. This polynomial is defined apart from a multiplying factor which is chosen, in our case, such that $P_k(0) = 1$. With this normalization, the polynomial P_k can be written as a ratio of two determinants as follows

$$P_k(\xi) = \begin{vmatrix} 1 & \xi & \cdots & \xi^k \\ c_0 & c_1 & \cdots & c_k \\ \vdots & \vdots & & \vdots \\ c_{k-1} & c_k & \cdots & c_{2k-1} \end{vmatrix} \Bigg/ \begin{vmatrix} c_1 & c_2 & \cdots & c_k \\ c_2 & c_3 & \cdots & c_{k+1} \\ \vdots & \vdots & & \vdots \\ c_k & c_{k+1} & \cdots & c_{2k-1} \end{vmatrix}.$$

Then we have

$$r_k = \begin{vmatrix} r_0 & Ar_0 & \cdots & A^k r_0 \\ c_0 & c_1 & \cdots & c_k \\ \vdots & \vdots & & \vdots \\ c_{k-1} & c_k & \cdots & c_{2k-1} \end{vmatrix} \Bigg/ \begin{vmatrix} c_1 & c_2 & \cdots & c_k \\ c_2 & c_3 & \cdots & c_{k+1} \\ \vdots & \vdots & & \vdots \\ c_k & c_{k+1} & \cdots & c_{2k-1} \end{vmatrix},$$

and

$$x_k - x_0 = \begin{vmatrix} 0 & r_0 & \cdots & A^{k-1} r_0 \\ c_0 & c_1 & \cdots & c_k \\ \vdots & \vdots & & \vdots \\ c_{k-1} & c_k & \cdots & c_{2k-1} \end{vmatrix} \Bigg/ \begin{vmatrix} c_1 & c_2 & \cdots & c_k \\ c_2 & c_3 & \cdots & c_{k+1} \\ \vdots & \vdots & & \vdots \\ c_k & c_{k+1} & \cdots & c_{2k-1} \end{vmatrix}.$$

Since the determinants in the denominators of P_k, r_k and $x_k - x_0$ are in fact the preceding Hankel determinant $H_k^{(1)}$, and P_k exists and is unique if

and only if this determinant is not zero, the existence of the polynomial P_k guarantees the existence and the uniqueness of r_k and x_k.

Let us now consider the monic polynomial $P_k^{(1)}$ of degree k belonging to the family of formal orthogonal polynomials with respect to the functional $c^{(1)}$ defined by $c^{(1)}\left(\xi^{\mathrm{i}}\right) = c\left(\xi^{\mathrm{i}+1}\right)$. It satisfies the orthogonality conditions

$$c^{(1)}(\xi^{\mathrm{i}} P_k^{(1)}(\xi)) = 0 \qquad \text{for } i = 0, \ldots, k-1$$

and it can be written as a ratio of two determinants

$$P_k^{(1)}(\xi) = \begin{vmatrix} c_1 & c_2 & \cdots & c_{k+1} \\ \vdots & \vdots & & \vdots \\ c_k & c_{k+1} & \cdots & c_{2k} \\ 1 & \xi & \cdots & \xi^k \end{vmatrix} \Bigg/ \begin{vmatrix} c_1 & c_2 & \cdots & c_k \\ c_2 & c_3 & \cdots & c_{k+1} \\ \vdots & \vdots & & \vdots \\ c_k & c_{k+1} & \cdots & c_{2k-1} \end{vmatrix}.$$

Since it has the same denominator as P_k, then $P_k^{(1)}$ exists under the same condition and conversely.

The linear functionals c and $c^{(1)}$ will always act on the variable ξ which will be suppressed when unnecessary.

The recursive computation of the polynomials P_k, needed in Lanczos' method, can be achieved in several ways. For instance, we can use the usual three-term recurrence relation, or relations involving also polynomials of the family $\{P_k^{(1)}\}$ or polynomials proportional to $P_k^{(1)}$. Using such recurrence relationships, leads to all the known algorithms for implementing the method of Lanczos and also to new ones. Such algorithms will be called *Lanczos-type algorithms*. See [14] for a unified presentation of all these methods based on the theory of FOP (Formal Orthogonal Polynomials) and [2] for more details.

2. Breakdown in Lanczos-type algorithms

The recursive computation of P_k involves the computation of some scalar products which appear as denominators and numerators of the coefficients of the recurrence relationships. Thus, if one of these scalar products is zero, in a denominator, then a *breakdown* occurs in the algorithm which has to be stopped. This can be due to the non-existence of some of the polynomials P_k and the breakdown can be avoided by jumping over these polynomials and by computing only the existing ones (called *regular*). This kind of breakdown is called *true breakdown* [9, 13] or *pivot breakdown* [17]. There is another possible breakdown in Lanczos-type algorithms. It is not due to the non-existence of some orthogonal polynomials of the family $\{P_k\}$, but to the recurrence relationship under consideration which cannot be used for computing P_k, for some k. For instance, in Lanczos/Orthores [26] and Lanczos/Orthomin [37, 38], the supplementary assumption that P_k has the exact

degree k is needed. This condition, unnecessary in the theory of Lanczos' method, produces a so-called *ghost breakdown* (see [9,13]), also named *Lanczos breakdown* [17].

The unique Lanczos-type algorithm that cannot have a ghost breakdown is the Lanczos/Orthodir algorithm where P_{k+1} is computed from P_k and $P_k^{(1)}$, and the polynomial $P_{k+1}^{(1)}$ is computed from $P_{k-1}^{(1)}$ and $P_k^{(1)}$.

For avoiding a true breakdown, only the regular polynomials of the family $\{P_k\}$, have to be computed and the algorithm has to jump over the non-existing polynomials. This is possible by using recurrence relationships where some coefficients are polynomials of a degree higher than usual. For avoiding a ghost breakdown, we should be able to compute any regular polynomial of a family by a relation not using necessarily the immediately preceding regular polynomials. That is, we need a rule for jumping over the existing and the non-existing polynomials of the family. Such recurrence relationships are also used for avoiding another problem that affects Lanczos-type algorithms. It is the so-called *near-breakdown* which appears when a scalar product in a denominator is different from zero but close to it, thus leading to an important propagation of rounding errors.

Jumping over breakdows and near-breakdowns is a technique known as *look-ahead*. It was introduced by Taylor [35] and Parlett, Taylor and Liu [30]. Since then, several look-ahead techniques were proposed. However, let us point out that not all the recurrence relationships used for curing breakdowns can be extended to the case of near-breakdowns. For example, it was shown in [10,11] that the MRZ and the SMRZ can be generalized but not the BMRZ, a breakdown-free version of an algorithm similar to Lanczos/Orthomin.

Let us briefly mention the various contributions to this problem. An unnormalized version of BIORES for curing true breakdowns, but not the ghost ones, was proposed by Gutknecht [22]. He also gave algorithms quite similar to ours [23,24]. Another implementation is discussed in [20]. A different procedure for treating breakdowns in the classical Lanczos algorithm is described in [6]. Zero divisor-free Hestenes-Stiefel type conjugate direction algorithms can be found in [25]. Another strategy for avoiding true breakdows in Lanczos/Orthomin was proposed by Bank and Chan [3,5]. It is similar to the technique proposed in [35,30] and improved in [27]. It consists of a 2×2 composite step. The problem of breakdown can also be treated by introducing new vectors into Krylov subspaces [31] or by an adaptative block Lanczos algorithm [32]. Another scheme, based on a modified Krylov subspace approach, is presented in [36]. Avoiding breakdowns in the other algorithms for implementing Lanczos' method described in [14] was studied in [2]. Finally, necessary and sufficient conditions for look-ahead versions of the block conjugate gradient algorithm to be free from serious and incurable

breakdowns were given in [16]. Thus, unstable versions of the algorithms can be identified and stable ones proposed.

Let us also mention that, instead of jumping over polynomials responsible for breakdowns, it is possible to go around the square blocks where they stand. Such a technique was introduced by Graves-Morris [21] and it is called *look-around*. Another possibility is to fill the gap between two successive orthogonal polynomials by biorthogonal polynomials, a technique due to Ayachour [1] and called ALA (Avoiding Look-Ahead).

In the following Section, we will recall the original MRZ (*Method of Recursive Zoom*) [12] which is a breakdown-free version of Lanczos/Orthodir. This method can only suffer from an incurable hard breakdown if $(y, A^n z_k) = 0$, where n is the dimension of the system to be solved. In this case, it is necessary to restart the algorithm by changing x_0 and/or y.

In the subsequent Sections, we will propose some modifications of the MRZ and its variants, the SMRZ and the BMRZ [10, 11]. We will first propose a modification based on the use of Horner's rule for the computation of the polynomials. Contrarily to the initial versions, the algorithms obtained need the storage of a fixed number of vectors, independent of the lengths of the jumps. More stable variants will also be given but at the cost of additional products by A^{T}.

The techniques described in this paper could possibly be extended to the treatment of near-breakdowns. This problem will be studied in a forthcoming paper.

3. The MRZ

As explained in the preceding Section, for avoiding a true breakdown we will consider only the existing orthogonal polynomials. Thus, let us change slightly our notations and denote by P_k the kth regular polynomial of the family. Its degree will be at most $n_k \geq k$ and it will satisfy the orthogonality relations

$$c\left(\xi^{\mathrm{i}} P_k\right) = 0 \qquad \text{for } i = 0, \ldots, n_k - 1$$

with $P_k(0) = 1$, Similarly, $P_k^{(1)}$ will denote the kth regular monic polynomial of degree $n_k \geq k$ of the family of formal orthogonal polynomials with respect to $c^{(1)}$.

If $P_k^{(1)}$ satisfies

$$c^{(1)}\left(\xi^{\mathrm{i}} P_k^{(1)}\right) = 0 \qquad \text{for } i = 0, \ldots, n_k + m_k - 2$$

$$\text{and } c^{(1)}\left(\xi^{n_k+m_k-1} P_k^{(1)}\right) \neq 0 \tag{5}$$

then, as stated by Struble [34] but first proved by Draux in [18] (see also [8] for an easier proof), the next regular polynomial $P_{k+1}^{(1)}$ will have the degree $n_{k+1} = n_k + m_k$ and it can be computed by

$$P_{k+1}^{(1)}(\xi) = q_k(\xi)\, P_k^{(1)}(\xi) - C_{k+1}\, P_{k-1}^{(1)}(\xi) \tag{6}$$

with $P_{-1}^{(1)}(\xi) = 0$, $C_1 = 0$ and q_k a monic polynomial of degree m_k.

In [12], we proved that the corresponding polynomial P_{k+1} can be computed by the relation

$$P_{k+1}(\xi) = P_k(\xi) - \xi\, w_k(\xi)\, P_k^{(1)}(\xi) \tag{7}$$

with $P_0^{(1)}(\xi) = 1$ and w_k a polynomial of degree $m_k - 1$ at most.

The constant m_k represents the length of the jump between two successive regular polynomials of the families $\{P_k\}$ and $\{P_k^{(1)}\}$.

The coefficients of the polynomials $q_k(\xi)$ and $w_k(\xi)$, and the constant C_{k+1} are determined by imposing the orthogonality relations to P_{k+1} and $P_{k+1}^{(1)}$ as we will see in the sequel.

If we set

$$\begin{aligned} r_k &= P_k(A)\, r_0 \\ z_k &= P_k^{(1)}(A)\, r_0 \end{aligned}$$

then the preceding recurrence relationships lead to the MRZ

$$\begin{aligned} r_{k+1} &= r_k - A\, w_k(A)\, z_k \\ x_{k+1} &= x_k + w_k(A)\, z_k \\ z_{k+1} &= q_k(A)\, z_k - C_{k+1}\, z_{k-1} \end{aligned} \tag{8}$$

with $z_0 = r_0$, $z_{-1} = 0$ and $C_1 = 0$.

By (4) and since $c^{(1)}(\xi^i P_k^{(1)}) = c(\xi^{i+1} P_k^{(1)})$, the conditions (5) can be rewritten

$$\begin{aligned} &\left(y, A^{i+1} z_k\right) = 0\,, \qquad \text{for } i = 0, \ldots, n_k + m_k - 2 \\ &\text{and } \left(y, A^{n_k+m_k} z_k\right) \neq 0. \end{aligned} \tag{9}$$

We set

$$\begin{aligned} w_k(\xi) &= \beta_0^{(k)} + \cdots + \beta_{m_k-1}^{(k)} \xi^{m_k-1} \\ q_k(\xi) &= \alpha_0^{(k)} + \cdots + \alpha_{m_k-1}^{(k)} \xi^{m_k-1} + \xi^{m_k}. \end{aligned} \tag{10}$$

and

$$\begin{aligned}
d_i^{(k)} &= c\left(\xi^{n_k+i}P_k\right) = \left(y, A^{n_k+i}P_k(A)r_0\right)\\
&= \left(y, A^{n_k+i}r_k\right) \quad \text{for } i = 0,\ldots,m_k-1\\
b_i^{(k)} &= c^{(1)}\left(\xi^{n_k+m_k+i-1}P_k^{(1)}\right) = \left(y, A^{n_k+m_k+i}P_k^{(1)}(A)r_0\right)\\
&= \left(y, A^{n_k+m_k+i}z_k\right) \quad \text{for } i = 0,\ldots,m_k\\
p_i^{(k)} &= c^{(1)}\left(\xi^{n_k+i-1}P_{k-1}^{(1)}\right) = \left(y, A^{n_k+i}P_{k-1}^{(1)}(A)r_0\right)\\
&= \left(y, A^{n_k+i}z_{k-1}\right) \quad \text{for } i = 0,\ldots,m_k.
\end{aligned}$$

Since the value of m_k is determined by (9), imposing the orthogonality conditions $c\left(\xi^{\mathrm{i}}P_{k+1}\right) = 0$, for $i = n_k,\ldots,n_k+m_k-1$ to (7) and $c^{(1)}\left(\xi^{\mathrm{i}}P_{k+1}^{(1)}\right) = 0$, for $i = n_k,\ldots n_k+m_k-1$ to (6), leads to the two following linear systems for the $\beta_i^{(k)}$'s, the $\alpha_i^{(k)}$'s and C_{k+1}

$$\text{(11)} \qquad \begin{cases}
\beta_{m_k-1}^{(k)}b_0^{(k)} = d_0^{(k)}\\
\beta_{m_k-2}^{(k)}b_0^{(k)} + \beta_{m_k-1}^{(k)}b_1^{(k)} = d_1^{(k)}\\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
\beta_0^{(k)}b_0^{(k)} + \beta_1^{(k)}b_1^{(k)} + \cdots + \beta_{m_k-1}^{(k)}b_{m_k-1}^{(k)} = d_{m_k-1}^{(k)}
\end{cases}$$

and

$$\text{(12)} \qquad \begin{cases}
b_0^{(k)} = C_{k+1}\,p_0^{(k)}\\
\alpha_{m_k-1}^{(k)}b_0^{(k)} + b_1^{(k)} = C_{k+1}\,p_1^{(k)}\\
\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
\alpha_0^{(k)}b_0^{(k)} + \alpha_1^{(k)}b_1^{(k)} + \cdots + \alpha_{m_k-1}^{(k)}b_{m_k-1}^{(k)} + b_{m_k}^{(k)} = C_{k+1}\,p_{m_k}^{(k)}.
\end{cases}$$

Computing C_{k+1} first, these two systems have the same lower triangular matrix but two different right hand sides. Using the notations above, we have

System of m_k equations for the $\beta_i^{(k)}$

System of m_k equations for the $\alpha_i^{(k)}$ ($C_{k+1} = b_0^{(k)}/p_0^{(k)} = b_0^{(k)}/b_0^{(k-1)}$)

$$
\begin{array}{cccccccc}
\beta_{m_k-1}^{(k)} & \beta_{m_k-2}^{(k)} & \cdots & \cdots & \cdots & \beta_0^{(k)} & \text{for} & \text{for} \\
\alpha_{m_k-1}^{(k)} & \alpha_{m_k-2}^{(k)} & \cdots & \cdots & \cdots & \alpha_0^{(k)} & \beta_i^{(k)} & \alpha_i^{(k)} \\
\downarrow & \downarrow & & & & \downarrow & \downarrow & \downarrow
\end{array}
$$

$$
\left[\begin{array}{ccccccc|cc}
b_0^{(k)} & & & & & & & d_0^{(k)} & -b_1^{(k)}+p_1^{(k)}C_{k+1} \\
b_1^{(k)} & b_0^{(k)} & & & & 0 & & d_1^{(k)} & -b_2^{(k)}+p_2^{(k)}C_{k+1} \\
\vdots & b_1^{(k)} & \ddots & & & & & \vdots & \vdots \\
\vdots & \vdots & \ddots & \ddots & & & & \vdots & \vdots \\
b_{m_k-2}^{(k)} & b_{m_k-1}^{(k)} & \cdots & \ddots & b_0^{(k)} & & & d_{m_k-2}^{(k)} & -b_{m_k-1}^{(k)}+p_{m_k-1}^{(k)}C_{k+1} \\
b_{m_k-1}^{(k)} & b_{m_k-2}^{(k)} & \cdots & \cdots & b_1^{(k)} & b_0^{(k)} & & d_{m_k-1}^{(k)} & -b_{m_k}^{(k)}+p_{m_k}^{(k)}C_{k+1}
\end{array}\right]
$$

This system is always nonsingular since m_k is determined by the condition $(y, A^{n_k+m_k}z_k) = b_0^{(k)} \neq 0$.

In the first implementation of the MRZ [10], these systems were solved by the usual formula for triangular systems.

Then the relations (8) give

$$r_{k+1} = r_k - [\beta_0^{(k)} A z_k + \beta_1^{(k)} A^2 z_k + \cdots + \beta_{m_k-1}^{(k)} A^{m_k} z_k]$$

$$x_{k+1} = x_k + [\beta_0^{(k)} z_k + \beta_1^{(k)} A z_k + \cdots + \beta_{m_k-1}^{(k)} A^{m_k-1} z_k]$$

$$z_{k+1} = \alpha_0^{(k)} z_k + \alpha_1^{(k)} A z_k + \cdots + \alpha_{m_k-1}^{(k)} A^{m_k-1} z_k + A^{m_k} z_k - C_{k+1} z_{k-1}.$$

If, at the iteration k, we store the vector z_{k-1} and compute and store the $m_k + 1$ vectors of $\mathbb{R}^n$

$$z_{k,i} = A^{\mathrm{i}} z_k \text{ for } i = 0, \ldots, m_k,$$

then the new vectors will be computed by

$$r_{k+1} = r_k - [\beta_0^{(k)} z_{k,1} + \beta_1^{(k)} z_{k,2} + \cdots + \beta_{m_k-1}^{(k)} z_{k,m_k}]$$

$$x_{k+1} = x_k + [\beta_0^{(k)} z_{k,0} + \beta_1^{(k)} z_{k,1} + \cdots + \beta_{m_k-1}^{(k)} z_{k,m_k-1}]$$

$$z_{k+1} = \alpha_0^{(k)} z_{k,0} + \alpha_1^{(k)} z_{k,1} + \cdots + \alpha_{m_k-1}^{(k)} z_{k,m_k-1} + z_{k,m_k} - C_{k+1} z_{k-1}.$$

Thus, this algorithm needs the storage of the matrix A, $m + 7$ vectors of $\mathbb{R}^n$ and 5 vectors of $\mathbb{R}^{m+1}$, where $m = \max_k m_k$. It is expected that, in general, the values of m_k are small compared to n.

For reducing the number of matrix-by-vector products in the algorithm, we will now use some properties and some peculiarities of the method.

At each iteration, for $i = 0, \ldots, m_k$, we need to compute the quantities $p_i^{(k)} = \left(y, A^{n_k+i} z_{k-1}\right)$, which correspond, when $m_k \leq m_{k-1}$, to the quantities $b_i^{(k-1)} = \left(y, A^{n_{k-1}+m_{k-1}+i} z_{k-1}\right)$ for $i = 0, \ldots m_{k-1}$ of the previous iteration. Then, we can save the $b_i^{(k-1)}$ in the vector that contains the $p_i^{(k)}$.

In the next iteration, we will compute the $p_i^{(k)}$ for $i = m_{k-1}+1, \ldots, m_k$, only if $m_k > m_{k-1}$.

Another gain in the number of matrix-by-vector products is obtained by using the property

$$\left(y, A^{n_k+i} u\right) = \left(A^{\mathrm{T}^{n_k}} y, A^i u\right).$$

Since the computation of the $d_i^{(k)}$'s, $b_i^{(k)}$'s and $p_i^{(k)}$'s involves only powers of A greater than n_k, then, at each iteration, we left multiply m_k times the vector y by A^{T} and thus we will start the next iteration with a vector $\hat{y} = A^{\mathrm{T}^{n_{k+1}}} y$.

Doing so, we have only $2m$ matrix-by-vector products and, moreover, the number of inner products becomes smaller than $4m$.

The algorithm can be written, in a short version not considering all the tests for incurable hard breakdown and for the length of the jump, as follows

Algorithm MRZ (A, b, x_0, y)

1. **Initializations:**
   $z_{-1} \leftarrow 0$; $\hat{y} \leftarrow y$;
   $r_0 \leftarrow b - A\, x_0$; $z_0 \leftarrow r_0$
   $n_0 \leftarrow 0$; $C_1 \leftarrow 0$; $m_{-1} \leftarrow 0$
   $k \leftarrow 0$
2. **While** $r_k \neq 0$ **do**
   $z_{k,0} \leftarrow z_k$
   $z_{k,1} \leftarrow A\, z_k$
   $d_0^{(k)} \leftarrow (\hat{y}, r_k)$
   $m_k \leftarrow 1$
   $b_0^{(k)} \leftarrow (\hat{y}, z_{k,1})$
3. **While** $b_0^{(k)} = 0$ **do**
   $m_k \leftarrow m_k + 1$
   $z_{k,m_k} \leftarrow A\, z_{k,m_k-1}$
   $b_0^{(k)} \leftarrow (\hat{y}, z_{k,m_k})$
   end while
4. $\beta_{m_k-1}^{(k)} \leftarrow d_0^{(k)} / b_0^{(k)}$
   If $k \neq 0$ **then** $C_{k+1} \leftarrow b_0^{(k)} / p_0^{(k)}$
5. **For** $i = 1, \ldots, m_k$ **do**

$\hat{y} \leftarrow A^{\mathrm{T}}\,\hat{y}$
$b_i^{(k)} \leftarrow (\hat{y}, z_{k,m_k})$
If $i \neq m_k$ **then**
$d_i^{(k)} \leftarrow (\hat{y}, r_k)$
compute $\beta_{m_k-i-1}^{(k)}$
end if
If $i > m_{k-1}$ **then** $p_i^{(k)} \leftarrow (\hat{y}, z_{k-1})$
compute $\alpha_{m_k-i}^{(k)}$
end for

6. $x_{k+1} = x_k + [\beta_0^{(k)} z_{k,0} + \beta_1^{(k)} z_{k,1} + \cdots + \beta_{m_k-1}^{(k)} z_{k,m_k-1}]$
$r_{k+1} = r_k - [\beta_0^{(k)} z_{k,1} + \beta_1^{(k)} z_{k,2} + \cdots + \beta_{m_k-1}^{(k)} z_{k,m_k}]$
$z_{k+1} = \alpha_0^{(k)} z_{k,0} + \alpha_1^{(k)} z_{k,1} + \cdots + \alpha_{m_k-1}^{(k)} z_{k,m_k-1} + z_{k,m_k} - C_{k+1} z_{k-1}$
7. $n_{k+1} \leftarrow n_k + m_k$
For $i = 0, \ldots, m_k$ **do** $p_i^{(k+1)} \leftarrow b_i^{(k)}$
$k \leftarrow k + 1$
end while

4. MRZ-stab

In [10], we showed how compute differently the scalar products needed in the MRZ and in its variants. Since $P_k^{(1)}$ has the degree n_k exactly, the orthogonality relations $c\big(\xi^{\mathrm{i}} P_{k+1}\big) = 0$, for $i = n_k, \ldots, n_k + m_k - 1$ can be replaced by $c\Big(\xi^{\mathrm{i}} P_k^{(1)} P_{k+1}\Big) = 0$, for $i = 0, \ldots, m_k - 1$. Thus, using these relations in (7) and since $P_k^{(1)}$ is orthogonal to any polynomial of degree at most $n_k + m_k - 2$ (that is $c^{(1)}(\xi^{\mathrm{i}} {P_k^{(1)}}^2) = 0$, for $i = 0, \ldots, m_k - 2$), we obtain again a lower triangular system similar to (11).

In fact, if we set

$$z_k = P_k^{(1)}(A)\, r_0$$
$$\tilde{z}_k = P_k^{(1)}(A^{\mathrm{T}})\, y$$

it is sufficient to replace the coefficients $b_i^{(k)}$ and $d_i^{(k)}$ in (11) by the quantities

$$\begin{aligned}
\tilde{d}_i^{(k)} &= c\left(\xi^{\mathrm{i}} P_k^{(1)} P_k\right) = \left(y, A^{\mathrm{i}} P_k^{(1)}(A) P_k(A) r_0\right) \\
&= \left(P_k^{(1)}(A^{\mathrm{T}}) y, A^{\mathrm{i}} P_k(A) r_0\right) = \left(\tilde{z}_k, A^{\mathrm{i}} r_k\right) \\
&= \left(A^{\mathrm{T}^i} \tilde{z}_k, r_k\right) \quad \text{for } i = 0, \ldots, m_k - 1
\end{aligned}$$

$$\begin{aligned}\tilde{b}_i^{(k)} &= c^{(1)}\left(\xi^{\mathrm{i}+m_k-1}{P_k^{(1)}}^2\right) = \left(y, A^{\mathrm{i}+m_k}P_k^{(1)}(A)P_k^{(1)}(A)r_0\right)\\ &= \left(P_k^{(1)}(A^{\mathrm{T}})y, A^{\mathrm{i}+m_k}P_k^{(1)}(A)r_0\right) = \left(\tilde{z}_k, A^{\mathrm{i}+m_k}z_k\right)\\ &= \left(A^{\mathrm{T}^{\mathrm{i}+m_k}}\tilde{z}_k, z_k\right) \quad \text{for } i = 0,\ldots,m_k-1.\end{aligned}$$

Similarly, replacing the orthogonality conditions $c^{(1)}(\xi^{\mathrm{i}}P_{k+1}^{(1)}) = 0$, for $i = n_k,\ldots n_k+m_k-1$ by $c^{(1)}(\xi^{\mathrm{i}}P_k^{(1)}P_{k+1}^{(1)}) = 0$, for $i = 0,\ldots,m_k-1$ and using these relations in (6), we obtain the system

$$\text{(13)}\qquad \begin{cases} \tilde{b}_0^{(k)} = C_{k+1}\,\tilde{p}_0^{(k)}\\ \alpha_{m_k-1}^{(k)}\tilde{b}_0^{(k)} + \tilde{b}_1^{(k)} = 0\\ \cdots\cdots\cdots\cdots\cdots\cdots\\ \alpha_0^{(k)}\tilde{b}_0^{(k)} + \alpha_1^{(k)}\tilde{b}_1^{(k)} + \cdots + \alpha_{m_k-1}^{(k)}\tilde{b}_{m_k-1}^{(k)} + \tilde{b}_{m_k}^{(k)} = 0. \end{cases}$$

with $\tilde{p}_0^{(k)} = c^{(1)}\left(\xi^{m_{k-1}-1}{P_{k-1}^{(1)}}^2\right) = \tilde{b}_0^{(k-1)}$ and $C_{k+1} = \tilde{b}_0^{(k)}/\tilde{b}_0^{(k-1)}$. Thus now we again have to solve two systems of m_k equations having the same lower triangular matrix and two different right hand sides

$$\text{(14)}\qquad \begin{array}{cccccc|cc} \beta_{m_k-1}^{(k)} & \beta_{m_k-2}^{(k)} & \cdots & \cdots & \cdots & \beta_0^{(k)} & \text{for} & \text{for}\\ \alpha_{m_k-1}^{(k)} & \alpha_{m_k-2}^{(k)} & \cdots & \cdots & \cdots & \alpha_0^{(k)} & \beta_i^{(k)} & \alpha_i^{(k)}\\ \downarrow & \downarrow & & & & \downarrow & \downarrow & \downarrow \\ \hline \tilde{b}_0^{(k)} & & & & & & \tilde{d}_0^{(k)} & -\tilde{b}_1^{(k)}\\ \tilde{b}_1^{(k)} & \tilde{b}_0^{(k)} & & & & & \tilde{d}_1^{(k)} & -\tilde{b}_2^{(k)}\\ \vdots & \tilde{b}_1^{(k)} & \ddots & & 0 & & \vdots & \vdots\\ \vdots & \vdots & \ddots & \ddots & & & \vdots & \vdots\\ \tilde{b}_{m_k-2}^{(k)} & \tilde{b}_{m_k-1}^{(k)} & \cdots & \ddots & \tilde{b}_0^{(k)} & & \tilde{d}_{m_k-2}^{(k)} & -\tilde{b}_{m_k-1}^{(k)}\\ \tilde{b}_{m_k-1}^{(k)} & \tilde{b}_{m_k-2}^{(k)} & \cdots & \cdots & \tilde{b}_1^{(k)} & \tilde{b}_0^{(k)} & \tilde{d}_{m_k-1}^{(k)} & -\tilde{b}_{m_k}^{(k)} \end{array}$$

With respect to the MRZ, we have to define the additional vectors $\tilde{z}_k$. Using the recurrence relationship of the polynomials $P_k^{(1)}$, these vectors can be computed by

$$\tilde{z}_{k+1} = q_k(A^{\mathrm{T}})\,\tilde{z}_k - C_{k+1}\,\tilde{z}_{k-1},$$

with $\tilde{z}_0 = y$ and $\tilde{z}_{-1} = 0$.

Obviously, this modified version of the MRZ algorithm needs the storage of the vector $\tilde{z}_{k-1}$ and the computation of the $m_k + 1$ additional vectors of $\mathbb{R}^n$

$$\tilde{z}_{k,i} = A^{\mathrm{T}^i} \tilde{z}_k \text{ for } i = 0, \ldots, m_k$$

and, thus, it is not really interesting. However, this different procedure for computing the polynomials q_k and w_k will be of interest in the sequel.

Let us mention that computing the coefficients in that way, leads to a method similar to the BIODIR method given by Gutknecht [23].

5. HMRZ

We will show now that, using a different way for computing the polynomials w_k and q_k given by (10), we will be able to greatly reduce the number of vectors of $\mathbb{R}^n$ needed in the algorithm in the case of breakdown. Let us mention that a quite similar technique is also described in [1].

Let us consider the monic polynomial ν_k given by

$$\nu_k(\xi) = \gamma_0^{(k)} + \cdots + \gamma_{m_k-1}^{(k)} \xi^{m_k-1} + \xi^{m_k}. \tag{15}$$

It can be computed by Horner's rule

$$\begin{aligned} &\nu_k^{(0)}(\xi) = 1 \\ &\nu_k^{(i)}(\xi) = \xi \nu_k^{(i-1)}(\xi) + \gamma_{m_k-i}^{(k)}, \qquad \text{for } i = 1, \ldots, m_k \end{aligned}$$

and we obtain $\nu_k(\xi) = \nu_k^{(m_k)}(\xi)$. The coefficients $\gamma_i^{(k)}$ are solution of the following lower triangular system which has the same matrix as the system of Sect. 3

$$\begin{pmatrix} b_0^{(k)} & & & & \\ b_1^{(k)} & b_0^{(k)} & & 0 & \\ \vdots & b_1^{(k)} & \ddots & & \\ \vdots & \vdots & \ddots & \ddots & \\ b_{m_k-2}^{(k)} & b_{m_k-1}^{(k)} & \cdots & \ddots & b_0^{(k)} \\ b_{m_k-1}^{(k)} & b_{m_k-2}^{(k)} & \cdots & \cdots & b_1^{(k)} \; b_0^{(k)} \end{pmatrix} \begin{pmatrix} \gamma_{m_k-1}^{(k)} \\ \gamma_{m_k-2}^{(k)} \\ \vdots \\ \vdots \\ \gamma_1^{(k)} \\ \gamma_0^{(k)} \end{pmatrix} = - \begin{pmatrix} b_1^{(k)} \\ b_2^{(k)} \\ \vdots \\ \vdots \\ b_{m_k-1}^{(k)} \\ b_{m_k}^{(k)} \end{pmatrix}. \tag{16}$$

We denote this system as

$$T_{m_k}^{(k)} s_{m_k}^{(k)} = -v_{m_k}^{(k)}$$

where the lower index represents the dimension.

Let $T_i^{(k)} \in \mathbb{R}^{\mathrm{i} \times i}$ be the principal submatrix of $T_{m_k}^{(k)}$ and $v_i^{(k)} \in \mathbb{R}^i$ the vector consisting in the first i components of $v_{m_k}^{(k)}$. The following result holds

Lemma 1.

$$\left(\xi^{\mathrm{i}-1},\ldots,1\right) T_i^{(k)^{-1}} = \frac{1}{b_0^{(k)}} \left(\nu_k^{(i-1)}(\xi),\ldots,\nu_k^{(0)}(\xi)\right) \quad \textit{for } i=1,\ldots,m_k$$

Proof. By induction. The result is obvious for $i=1$. Let us assume that the relation holds for $i-1$.

We have

$$T_i^{(k)^{-1}} = \left(\begin{array}{c|c} \dfrac{1}{b_0^{(k)}} & 0^{\mathrm{T}} \\ \hline -\dfrac{1}{b_0^{(k)}} T_{i-1}^{(k)^{-1}} v_{i-1}^{(k)} & T_{i-1}^{(k)^{-1}} \end{array}\right).$$

Thus

$$\left(\xi^{\mathrm{i}-1},\xi^{\mathrm{i}-2},\ldots,1\right) T_i^{(k)^{-1}} = \left(\frac{1}{b_0^{(k)}} \left(\xi^{\mathrm{i}-1} - \left(\xi^{\mathrm{i}-2},\ldots,1\right) T_{i-1}^{(k)^{-1}} v_{i-1}^{(k)}\right), \left(\xi^{\mathrm{i}-2},\ldots,1\right) T_{i-1}^{(k)^{-1}}\right).$$

We have

$$\begin{aligned} \frac{1}{b_0^{(k)}} \left(\xi^{\mathrm{i}-1} - \left(\xi^{\mathrm{i}-2},\ldots,1\right) T_{i-1}^{(k)^{-1}} v_{i-1}^{(k)}\right) = \\ \frac{1}{b_0^{(k)}} \left(\xi^{\mathrm{i}-1} + \gamma_{m_k-1}^{(k)} \xi^{\mathrm{i}-2} + \cdots + \gamma_{m_k-i-1}^{(k)}\right) = \\ \frac{1}{b_0^{(k)}} \left(\xi \nu_k^{(i-2)}(\xi) + \gamma_{m_k-i-1}^{(k)}\right) = \frac{1}{b_0^{(k)}} \left(\nu_k^{(i-1)}(\xi)\right). \end{aligned}$$

Since the relation holds for $i-1$, we immediately obtain the lemma. □

The main result is given in the following theorem.

Theorem 1.

$$w_k(\xi) = \frac{1}{b_0^{(k)}} \sum_{j=0}^{m_k-1} d_{m_k-j-1}^{(k)} \nu_k^{(j)}(\xi)$$

Proof. We first consider Horner's rule for computing w_k. We have

$$\begin{aligned} & w_k^{(0)}(\xi) = \beta_{m_k-1}^{(k)} \\ & w_k^{(i)}(\xi) = \xi w_k^{(i-1)}(\xi) + \beta_{m_k-i-1}^{(k)}, \quad \text{for } i=1,\ldots,m_k-1 \end{aligned} \tag{17}$$

and we obtain $w_k(\xi) = w_k^{(m_k-1)}(\xi)$.

From (17) and (11), we have

$$\begin{aligned} w_k^{(i)}(\xi) &= \left(\xi^{\mathrm{i}}, \ldots, 1\right) \left(\beta_{m_k-1}^{(k)}, \ldots, \beta_{m_k-i-1}^{(k)}\right)^{\mathrm{T}} \\ &= \left(\xi^{\mathrm{i}}, \ldots, 1\right) T_{i+1}^{(k)^{-1}} \left(d_0^{(k)}, \ldots, d_i^{(k)}\right)^{\mathrm{T}} \\ &= \frac{1}{b_0^{(k)}} \left(\nu_k^{(i)}(\xi), \ldots, \nu_k^{(0)}(\xi)\right) \left(d_0^{(k)}, \ldots, d_i^{(k)}\right)^{\mathrm{T}} \\ &= \frac{1}{b_0^{(k)}} \sum_{j=0}^{\mathrm{i}} d_{i-j}^{(k)} \nu_k^{(j)}(\xi). \quad \square \end{aligned}$$

Similarly, the polynomial q_k can be computed by Horner's rule

$$\begin{aligned} & q_k^{(0)}(\xi) = 1 \\ (18) \qquad & q_k^{(i)}(\xi) = \xi q_k^{(i-1)}(\xi) + \alpha_{m_k-i}^{(k)}, \qquad \text{for } i = 1, \ldots, m_k \end{aligned}$$

and we obtain $q_k(\xi) = q_k^{(m_k)}(\xi)$. We have the following theorem which proof is the same as the preceding one.

Theorem 2.

$$q_k(\xi) = \nu_k(\xi) + \frac{1}{p_0^{(k)}} \sum_{j=0}^{m_k-1} p_{m_k-j}^{(k)} \nu_k^{(j)}(\xi)$$

With this approach, we no longer need to store all the $b_i^{(k)}$'s, but only $b_0^{(k)}$. Obviously, for each iteration, the values of the $p_i^{(k)}$'s, for $i = 1, \ldots, m_k$ have all to be computed since the $b_i^{(k-1)}$'s, are not computed at all. Moreover we don't need to store the coefficients of the polynomials w_k and q_k, and thus the number of vectors of $\mathbb{R}^{m+1}$ to be stored reduces to 2. Therefore, the most important feature of this variant is that the number of vectors of $\mathbb{R}^n$ to be stored does not depend of the length of the jump m_k, but is always equal to 8.

Algorithm HMRZ (A, b, x_0, y)

1. **Initializations:**
   $z_{-1} \leftarrow 0$; $\hat{y} \leftarrow y$
   $r_0 \leftarrow b - A\,x_0$; $z_0 \leftarrow r_0$
   $n_0 \leftarrow 0$; $C_1 \leftarrow 0$; $\alpha \leftarrow 0$; $b_0^{(-1)} = 0$
   $k \leftarrow 0$
2. **While** $r_k \neq 0$ **do**
   $d_0^{(k)} \leftarrow (\hat{y}, r_k)$
   $m_k \leftarrow 1$

$\hat{y} \leftarrow A^{\mathrm{T}}\hat{y}$
$p_0^{(k)} \leftarrow b_0^{(k-1)}$
$p_1^{(k)} \leftarrow (\hat{y}, z_{k-1})$
$b_0^{(k)} \leftarrow (\hat{y}, z_k)$

3. **While** $b_0^{(k)} = 0$ **do**
$m_k \leftarrow m_k + 1$
$d_{m_k-1}^{(k)} \leftarrow (\hat{y}, r_k)$
$\hat{y} \leftarrow A^{\mathrm{T}}\hat{y}$
$b_0^{(k)} \leftarrow (\hat{y}, z_k)$
$p_{m_k}^{(k)} \leftarrow (\hat{y}, z_{k-1})$
end while

4. **If** $k \neq 0$ **then** $C_{k+1} \leftarrow b_0^{(k)}/p_0^{(k)}$
$t_k \leftarrow z_k$
$z_{k+1} \leftarrow -C_{k+1}\, z_{k-1}$
$x_{k+1} \leftarrow x_k$
$r_{k+1} \leftarrow r_k$

5. **For** $i = 1, \ldots, m_k$ **do**
$u_k \leftarrow At_k$
$\beta \leftarrow d_{m_k-i}^{(k)}/b_0^{(k)}$
$x_{k+1} \leftarrow x_{k+1} + \beta t_k$
$r_{k+1} \leftarrow r_{k+1} - \beta u_k$
If $k \neq 0$ **then** $\alpha \leftarrow p_{m_k-i+1}^{(k)}/p_0^{(k)}$
$z_{k+1} \leftarrow z_{k+1} + \alpha t_k$
$\gamma \leftarrow -(\hat{y}, u_k)/b_0^{(k)}$
$t_k \leftarrow u_k + \gamma z_k$
end for

6. $z_{k+1} \leftarrow z_{k+1} + t_k$
$n_{k+1} \leftarrow n_k + m_k$
$k \leftarrow k + 1$
end while

6. HMRZ-stab

Horner's rule can also be applied to the MRZ-stab of Sect. 4 and we obtain an algorithm which seems to be more stable than the HMRZ.

If we denote by $\tilde{\gamma}_i^{(k)}$'s the solution of the system (16), when the $b_i^{(k)}$'s are replaced by the $\tilde{b}_i^{(k)}$'s, computed as in Sect. 4, the coefficients $\alpha_i^{(k)}$'s of the polynomial $q_k(\xi)$ coincide exactly with the $\tilde{\gamma}_i^{(k)}$'s and thus the algorithm simplifies. That is, if we set $\tilde{\nu}_k^{(i)}(\xi) = \xi\tilde{\nu}_k^{(i-1)}(\xi) + \tilde{\gamma}_{m_k-i}^{(k)}$, for

$i = 1, \ldots, m_k$, we have $\tilde{\nu}_k(\xi) = \tilde{\nu}_k^{(m_k)}(\xi) = q_k(\xi)$. The coefficients $\beta_i^{(k)}$ of the polynomial $w_k(\xi)$ will be computed, by Theorem 1, as $w_k(\xi) = \frac{1}{\tilde{b}_0^{(k)}} \sum_{j=0}^{m_k-1} \tilde{d}_{m_k-j-1}^{(k)} \tilde{\nu}_k^{(j)}(\xi)$.

Hence we obtain the following algorithm

Algorithm HMRZ-stab (A, b, x_0, y)

1. **Initializations:**
   $z_{-1} \leftarrow 0$; $\tilde{z}_{-1} \leftarrow 0$
   $r_0 \leftarrow b - A\,x_0$
   $z_0 \leftarrow r_0$; $\tilde{z}_0 \leftarrow y$
   $n_0 \leftarrow 0$; $C_1 \leftarrow 0$; $\tilde{b}_0^{(-1)} = 0$
   $k \leftarrow 0$
2. **While** $r_k \neq 0$ **do**
   $\tilde{d}_0^{(k)} \leftarrow (\tilde{z}_k, r_k)$
   $m_k \leftarrow 1$
   $\tilde{y}_k \leftarrow A^{\mathrm{T}}\,\tilde{z}_k$
   $\tilde{u}_k \leftarrow \tilde{y}_k$
   $\tilde{b}_0^{(k)} \leftarrow (\tilde{y}_k, z_k)$
3. **While** $\tilde{b}_0^{(k)} = 0$ **do**
   $m_k \leftarrow m_k + 1$
   $\tilde{d}_{m_k-1}^{(k)} \leftarrow (\tilde{y}_k, r_k)$
   $\tilde{y}_k \leftarrow A^{\mathrm{T}}\,\tilde{y}_k$
   $\tilde{b}_0^{(k)} \leftarrow (\tilde{y}_k, z_k)$
   end while
4. **If** $k \neq 0$ **then** $C_{k+1} \leftarrow \tilde{b}_0^{(k)} / \tilde{b}_0^{(k-1)}$
   $t_k \leftarrow z_k$
   $z_{k+1} \leftarrow -C_{k+1}\,z_{k-1}$
   $\tilde{t}_k \leftarrow \tilde{z}_k$
   $\tilde{z}_{k+1} \leftarrow -C_{k+1}\,\tilde{z}_{k-1}$
   $x_{k+1} \leftarrow x_k$
   $r_{k+1} \leftarrow r_k$
5. **For** $i = 1, \ldots, m_k$ **do**
   $u_k \leftarrow A t_k$
   $\tilde{\beta} \leftarrow \tilde{d}_{m_k-i}^{(k)} / \tilde{b}_0^{(k)}$
   $x_{k+1} \leftarrow x_{k+1} + \tilde{\beta} t_k$
   $r_{k+1} \leftarrow r_{k+1} - \tilde{\beta} u_k$
   $\tilde{\gamma} \leftarrow -(\tilde{y}_k, u_k) / \tilde{b}_0^{(k)}$
   $t_k \leftarrow u_k + \tilde{\gamma} z_k$
   If $i \neq 1$ **then** $\tilde{u}_k \leftarrow A^{\mathrm{T}} \tilde{t}_k$

$\tilde{t}_k \leftarrow \tilde{u}_k + \tilde{\gamma}\tilde{z}_k$
end for
6. $z_{k+1} \leftarrow z_{k+1} + t_k$
$\tilde{z}_{k+1} \leftarrow \tilde{z}_{k+1} + \tilde{t}_k$
$n_{k+1} \leftarrow n_k + m_k$
$k \leftarrow k + 1$
end while

This algorithm requires the storage of 12 vectors of $\mathbb{R}^n$ (independently of the lengths m_k of the jumps) instead of 8 for the HMRZ. But, as can be seen from the numerical examples, the gain in the numerical stability seems to be quite important.

7. Variants of the MRZ

In this Section, we will consider the two variants of the MRZ proposed in [10, 11], where P_{k+1} is still computed by the same recurrence relationship but where $P_{k+1}^{(1)}$ is obtained in two different ways. It must be noted that these two variants can suffer from ghost breakdowns. In such a case, they cannot be used.

For these algorithms we will propose again the modification based on the use of Horner's rule and also the correspondent stable versions.

7.1. HSMRZ and HSMRZ-stab

In this variant, $P_{k+1}^{(1)}$ is expressed as

$$P_{k+1}^{(1)}(\xi) = t_k(\xi)P_k^{(1)}(\xi) - D_{k+1}P_k(\xi), \tag{19}$$

where t_k is a monic polynomial of degree m_k and D_{k+1} is a constant, with $D_1 = 0$.

The coefficients of this polynomial are solution of the system (12), with a different right hand side corresponding to $(D_{k+1}d_0^{(k)}, \ldots, D_{k+1}d_{m_k}^{(k)})^{\mathrm{T}}$.

Therefore, as in Theorem 2, we have

$$t_k(\xi) = \nu_k(\xi) + \frac{1}{d_0^{(k)}} \sum_{j=0}^{m_k-1} d_{m_k-j}^{(k)} \nu_k^{(j)}(\xi).$$

Using this relation we have the following algorithm

Algorithm HSMRZ (A, b, x_0, y)

1. **Initializations:**
   $\hat{y} \leftarrow y$;
   $r_0 \leftarrow b - A\,x_0$; $z_0 \leftarrow r_0$
   $n_0 \leftarrow 0$; $D_1 \leftarrow 0$; $\alpha \leftarrow 0$
   $k \leftarrow 0$
2. **While** $r_k \neq 0$ **do**
   $d_0^{(k)} \leftarrow (\hat{y}, r_k)$
   If $d_0^{(k)} = 0$ **then** Impossible to use the HSMRZ. Stop.
   $m_k \leftarrow 1$
   $\hat{y} \leftarrow A^{\mathrm{T}}\,\hat{y}$
   $b_0^{(k)} \leftarrow (\hat{y}, z_k)$
3. **While** $b_0^{(k)} = 0$ **do**
   $m_k \leftarrow m_k + 1$
   $d_{m_k-1}^{(k)} \leftarrow (\hat{y}, r_k)$
   $\hat{y} \leftarrow A^{\mathrm{T}}\,\hat{y}$
   $b_0^{(k)} \leftarrow (\hat{y}, z_k)$
   end while
   $d_{m_k}^{(k)} \leftarrow (\hat{y}, r_k)$
4. **If** $k \neq 0$ **then** $D_{k+1} \leftarrow b_0^{(k)}/d_0^{(k)}$
   $t_k \leftarrow z_k$
   $z_{k+1} \leftarrow -D_{k+1}\,r_k$
   $x_{k+1} \leftarrow x_k$
   $r_{k+1} \leftarrow r_k$
5. **For** $i = 1, \ldots, m_k$ **do**
   $u_k \leftarrow At_k$
   $\beta \leftarrow d_{m_k-i}^{(k)}/b_0^{(k)}$
   $x_{k+1} \leftarrow x_{k+1} + \beta t_k$
   $r_{k+1} \leftarrow r_{k+1} - \beta u_k$
   If $k \neq 0$ **then** $\alpha \leftarrow d_{m_k-i+1}^{(k)}/d_0^{(k)}$
   $z_{k+1} \leftarrow z_{k+1} + \alpha t_k$
   $\gamma \leftarrow -\,(\hat{y}, u_k)\,/b_0^{(k)}$
   $t_k \leftarrow u_k + \gamma z_k$
   end for
6. $z_{k+1} \leftarrow z_{k+1} + t_k$
   $n_{k+1} \leftarrow n_k + m_k$
   $k \leftarrow k + 1$
   end while

By considering the orthogonality conditions of the Sect. 4, as in the HMRZ-stab, we have $w_k(\xi) = \dfrac{1}{\tilde{b}_0^{(k)}} \displaystyle\sum_{j=0}^{m_k-1} \tilde{d}_{m_k-j-1}^{(k)} \tilde{\nu}_k^{(j)}(\xi)$. The polynomial t_k is now computed by $t_k(\xi) = \tilde{\nu}_k(\xi) + \dfrac{1}{\tilde{d}_0^{(k)}} \displaystyle\sum_{j=0}^{m_k-1} \tilde{d}_{m_k-j}^{(k)} \tilde{\nu}_k^{(j)}(\xi)$. If we set $\tilde{r}_k = P(A^{\mathrm{T}})y$, using the relation (19), these additionals vectors can be computed by

$$\tilde{r}_{k+1} = \tilde{r}_k - A^{\mathrm{T}} w_k(A^{\mathrm{T}}) \tilde{z}_k. \tag{20}$$

Thus we obtain another algorithm called HSMRZ-stab.

Algorithm HSMRZ-stab (A, b, x_0, y)

1. **Initializations:**
   $r_0 \leftarrow b - A\,x_0$
   $z_0 \leftarrow r_0$; $\tilde{z}_0 \leftarrow y$; $\tilde{r}_k \leftarrow y$
   $n_0 \leftarrow 0$; $D_1 \leftarrow 0$
   $k \leftarrow 0$
2. **While** $r_k \neq 0$ **do**
   $\tilde{d}_0^{(k)} \leftarrow (\tilde{z}_k, r_k)$
   If $\tilde{d}_0^{(k)} = 0$ **then** Impossible to use the HSMRZ-stab. Stop.
   $m_k \leftarrow 1$
   $\tilde{y}_k \leftarrow A^{\mathrm{T}}\,\tilde{z}_k$
   $\tilde{u}_k \leftarrow \tilde{y}_k$
   $\tilde{b}_0^{(k)} \leftarrow (\tilde{y}_k, z_k)$
3. **While** $\tilde{b}_0^{(k)} = 0$ **do**
   $m_k \leftarrow m_k + 1$
   $\tilde{d}_{m_k-1}^{(k)} \leftarrow (\tilde{y}_k, r_k)$
   $\tilde{y}_k \leftarrow A^{\mathrm{T}}\,\tilde{y}_k$
   $\tilde{b}_0^{(k)} \leftarrow (\tilde{y}_k, z_k)$
   end while
4. **If** $k \neq 0$ **then** $D_{k+1} \leftarrow \tilde{b}_0^{(k)} / \tilde{d}_0^{(k)}$
   $t_k \leftarrow z_k$
   $z_{k+1} \leftarrow -D_{k+1}\,r_k$
   $\tilde{t}_k \leftarrow \tilde{z}_k$
   $\tilde{z}_{k+1} \leftarrow -D_{k+1}\,\tilde{r}_k$
   $x_{k+1} \leftarrow x_k$
   $r_{k+1} \leftarrow r_k$
   $\tilde{r}_{k+1} \leftarrow \tilde{r}_k$
5. **For** $i = 1, \ldots, m_k$ **do**
   $u_k \leftarrow A t_k$

$\tilde{\beta} \leftarrow \tilde{d}_{m_k-i}^{(k)} / \tilde{b}_0^{(k)}$
$x_{k+1} \leftarrow x_{k+1} + \tilde{\beta} t_k$
$r_{k+1} \leftarrow r_{k+1} - \tilde{\beta} u_k$
If $k \neq 0$ **then** $\alpha \leftarrow \tilde{d}_{m_k-i+1}^{(k)} / \tilde{d}_0^{(k)}$
$z_{k+1} \leftarrow z_{k+1} + \alpha t_k$
$\tilde{z}_{k+1} \leftarrow \tilde{z}_{k+1} + \alpha \tilde{t}_k$
$\tilde{\gamma} \leftarrow -(\tilde{y}_k, u_k) / \tilde{b}_0^{(k)}$
$t_k \leftarrow u_k + \tilde{\gamma} z_k$
If $i \neq 1$ **then** $\tilde{u}_k \leftarrow A^{\mathrm{T}} \tilde{t}_k$
$\tilde{t}_k \leftarrow \tilde{u}_k + \tilde{\gamma} \tilde{z}_k$
$\tilde{r}_{k+1} \leftarrow \tilde{r}_{k+1} - \tilde{\beta} \tilde{u}_k$
end for

6. $z_{k+1} \leftarrow z_{k+1} + t_k$
$\tilde{z}_{k+1} \leftarrow \tilde{z}_{k+1} + \tilde{t}_k$
$n_{k+1} \leftarrow n_k + m_k$
$k \leftarrow k + 1$
end while

7.2. The HBMRZ and the HBMRZ-stab

In this variant, the polynomial $P_{k+1}^{(1)}$ satisfy the following relation

$$P_{k+1}^{(1)}(\xi) = A_{k+1} P_{k+1}(\xi) + B_{k+1} P_k^{(1)}(\xi), \tag{21}$$

where

$$A_{k+1} = -\frac{c\left(\xi^{n_k+m_k} P_k^{(1)}\right)}{c\left(\xi^{n_k} P_k\right)} \quad \text{and} \quad B_{k+1} = \frac{c\left(\xi^{n_k+m_k} P_{k+1}\right)}{c\left(\xi^{n_k} P_k\right)}.$$

With this choice, we recover exactly the BCG, written under an algorithmic form by Fletcher [19] (see also [7, p.91]). Using this relation, and computing the polynomial $w_k(\xi)$ as in Theorem 1, we obtain the following algorithm

Algorithm HBMRZ (A, b, x_0, y)

1. **Initializations:**
$\hat{y} \leftarrow y$;
$r_0 \leftarrow b - A\, x_0$; $z_0 \leftarrow r_0$
$n_0 \leftarrow 0$
$d_0^{(0)} \leftarrow (\hat{y}, r_0)$
$k \leftarrow 0$
2. **While** $r_k \neq 0$ **do**

If $d_0^{(k)} = 0$ **then** Impossible to use the HBMRZ. Stop.
$m_k \leftarrow 1$
$\hat{y} \leftarrow A^{\mathrm{T}}\,\hat{y}$
$b_0^{(k)} \leftarrow (\hat{y}, z_k)$

3. **While** $b_0^{(k)} = 0$ **do**
$m_k \leftarrow m_k + 1$
$d_{m_k-1}^{(k)} \leftarrow (\hat{y}, r_k)$
$\hat{y} \leftarrow A^{\mathrm{T}}\,\hat{y}$
$b_0^{(k)} \leftarrow (\hat{y}, z_k)$
end while

4. $t_k \leftarrow z_k$
$x_{k+1} \leftarrow x_k$
$r_{k+1} \leftarrow r_k$

5. **For** $i = 1, \ldots, m_k$ **do**
$u_k \leftarrow A t_k$
$\beta \leftarrow d_{m_k-i}^{(k)} / b_0^{(k)}$
$x_{k+1} \leftarrow x_{k+1} + \beta t_k$
$r_{k+1} \leftarrow r_{k+1} - \beta u_k$
$\gamma \leftarrow -\left(\hat{y}, u_k\right) / b_0^{(k)}$
$t_k \leftarrow u_k + \gamma z_k$
end for

6. $A_{k+1} \leftarrow -b_0^{(k)} / d_0^{(k)}$
$d_0^{(k+1)} \leftarrow (\hat{y}, r_{k+1})$
$B_{k+1} \leftarrow d_0^{(k+1)} / d_0^{(k)}$
$z_{k+1} \leftarrow A_{k+1} r_{k+1} + B_{k+1} z_k$
$n_{k+1} \leftarrow n_k + m_k$
$k \leftarrow k + 1$

end while

As for the HMRZ-stab and the HSMRZ-stab, it is possible to obtain, using the orthogonality conditions of the Sect. 4, another algorithm. Now we have

$$A_{k+1} = -\frac{c^{(1)}\left(\xi^{m_k-1} P_k^{(1)^2}\right)}{c\left(P_k^{(1)} P_k\right)} \quad \text{and} \quad B_{k+1} = \frac{c\left(\xi^{n_k+m_k} P_{k+1}\right)}{c\left(P_k^{(1)} P_k\right)}$$

and, if we set $\tilde{r}_k = P(A^{\mathrm{T}})y$, computed as in (20), we obtain from (21) $\tilde{z}_{k+1} = A_{k+1}\tilde{r}_{k+1} + B_{k+1}\tilde{z}_k$. The corresponding algorithm follows.

Algorithm HBMRZ-stab (A, b, x_0, y)

1. **Initializations:**
$r_0 \leftarrow b - A\,x_0$

$z_0 \leftarrow r_0$; $\tilde{z}_0 \leftarrow y$; $\tilde{r}_k \leftarrow y$
$n_0 \leftarrow 0$
$k \leftarrow 0$

2. **While** $r_k \neq 0$ **do**
$\tilde{d}_0^{(k)} \leftarrow (\tilde{z}_k, r_k)$
If $\tilde{d}_0^{(k)} = 0$ **then** Impossible to use the HBMRZ-stab. Stop.
$m_k \leftarrow 1$
$\tilde{y}_k \leftarrow A^{\mathrm{T}}\, \tilde{z}_k$
$\tilde{u}_k \leftarrow \tilde{y}_k$
$\tilde{b}_0^{(k)} \leftarrow (\tilde{y}_k, z_k)$
3. **While** $\tilde{b}_0^{(k)} = 0$ **do**
$m_k \leftarrow m_k + 1$
$\tilde{d}_{m_k-1}^{(k)} \leftarrow (\tilde{y}_k, r_k)$
$\tilde{y}_k \leftarrow A^{\mathrm{T}}\, \tilde{y}_k$
$\tilde{b}_0^{(k)} \leftarrow (\tilde{y}_k, z_k)$
end while
4. $t_k \leftarrow z_k$
$\tilde{t}_k \leftarrow \tilde{z}_k$
$x_{k+1} \leftarrow x_k$
$r_{k+1} \leftarrow r_k$
$\tilde{r}_{k+1} \leftarrow \tilde{r}_k$
5. **For** $i = 1, \ldots, m_k$ **do**
$u_k \leftarrow A t_k$
$\tilde{\beta} \leftarrow \tilde{d}_{m_k-i}^{(k)} / \tilde{b}_0^{(k)}$
$x_{k+1} \leftarrow x_{k+1} + \tilde{\beta} t_k$
$r_{k+1} \leftarrow r_{k+1} - \tilde{\beta} u_k$
$\tilde{\gamma} \leftarrow -\left(\tilde{y}_k, u_k\right) / \tilde{b}_0^{(k)}$
$t_k \leftarrow u_k + \tilde{\gamma} z_k$
If $i \neq 1$ **then** $\tilde{u}_k \leftarrow A^{\mathrm{T}} \tilde{t}_k$
$\tilde{t}_k \leftarrow \tilde{u}_k + \tilde{\gamma} \tilde{z}_k$
$\tilde{r}_{k+1} \leftarrow \tilde{r}_{k+1} - \tilde{\beta} \tilde{u}_k$
end for
6. $A_{k+1} \leftarrow -\tilde{b}_0^{(k)} / \tilde{d}_0^{(k)}$
$B_{k+1} \leftarrow \left(\tilde{y}_k, r_{k+1}\right) / \tilde{d}_0^{(k)}$
$z_{k+1} \leftarrow A_{k+1} r_{k+1} + B_{k+1} z_k$
$\tilde{z}_{k+1} \leftarrow A_{k+1} \tilde{r}_{k+1} + B_{k+1} \tilde{z}_k$
$n_{k+1} \leftarrow n_k + m_k$
$k \leftarrow k + 1$
end while

Table 1. Computational costs and storage requirements

Case without breakdown, $m_k = 1$

Method	Inner product	SAXPY	Matrix-vector product$^{(*)}$	Storage reqmts
MRZ	3	4	$1/1$	$matrix + 8n$
HMRZ	4	6	$1/1$	$matrix + 8n$
HMRZ-stab	3	8	$1/1$	$matrix + 12n$
HSMRZ	3	6	$1/1$	$matrix + 7n$
HSMRZ-stab	3	11	$1/1$	$matrix + 11n$
HBMRZ	3	4	$1/1$	$matrix + 7n$
HBMRZ-stab	4	7	$1/1$	$matrix + 11n$
GMRES$^{(a)}$	$k+1$	$k+1$	$1/0$	$matrix + (k+5)n$
QMR	2	$8 + 4^{(b)(c)}$	$1/1$	$matrix + 16n^{(c)}$
CGS	2	6	$2/0$	$matrix + 11n$
Bi-CGStab	4	6	$2/0$	$matrix + 10n$

Case with breakdown, $m_k > 1$

Method	Inner product	SAXPY	Matrix-vector product$^{(*)}$	Storage reqmts$^{(+)}$
MRZ	$4m_k - m_{k-1}$	$3m_k + 1$	m_k/m_k	$matrix + (M+7)n + 5M$
HMRZ	$4m_k$	$4m_k + 2$	m_k/m_k	$matrix + 8n + 2M$
HMRZ-stab	$3m_k$	$4m_k + 4$	$m_k/2m_k - 1$	$matrix + 12n + 1M$
HSMRZ	$3m_k$	$4m_k + 2$	m_k/m_k	$matrix + 7n + 1M$
HSMRZ-stab	$3m_k$	$7m_k + 4$	$m_k/2m_k - 1$	$matrix + 11n + 1M$
HBMRZ	$3m_k$	$3m_k + 1$	m_k/m_k	$matrix + 7n + 1M$
HBMRZ-stab	$3m_k + 1$	$5m_k + 2$	$m_k/2m_k - 1$	$matrix + 11n + 1M$

$(*)$ i/j means i multiplications with the matrix A and j multiplications with its transpose A^{T}

(a) k denotes the iteration

(b) True SAXPY operations + vector scaling

(c) Less for implementations that do not recursively update the residual.

$(+)$ $M = \sup_k m_k$

8. Comparison with other methods

Our algorithms, in the case without breakdown, compare well with other known algorithms with respect to the computational kernel and to the storage requirements. In Table 1, we will show a summary of operations for iteration k for selected algorithms (see [4]) and our algorithms and also the storage requirements for each algorithm (without preconditioning).

For the case of breakdown, the exact summary of operations and storage requirements needed in some look-ahead versions given in the literature on the subject is not known. Thus we will show, in Table 1, only a summary for our algorithms.

9. Numerical results

In this section, we will present some numerical results obtained using the MATLAB versions of our algorithms. For testing the true breakdown we used, instead of zero, a threshold ε and, in all the examples, r_k denotes the residual computed recursively by the algorithm and, in the case of breakdown, r_k corresponds to a Krylov subspace of dimension n_k. Since our algorithms only cure exact breakdowns, we have tested them on examples specially built for presenting such a feature and also used by other authors. Up to our knowledge, examples coming out from PDEs do not present exact breakdowns, but only near-breakdowns.

9.1. Example 1

We consider the system given in [6].

$$\begin{pmatrix} 0 & 0 & 0 & \cdots & 0 & -1 \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 3 \\ \vdots \\ n \end{pmatrix} = \begin{pmatrix} -n \\ 1 \\ 2 \\ \vdots \\ n-1 \end{pmatrix}.$$

If we choose $y = (1, \ldots, 1)^{\mathrm{T}}$, in order to obtain the solution, we have to make a jump from $n_3 = 3$ to $n_4 = n - 3$. Choosing different ε for testing the breakdown, we remarked that both MRZ and HMRZ are very sensitive to these values, but not the HMRZ-stab that always detects the correct jump.

For instance, if we consider a system of dimension $n = 100$ and $x_0 = 0$ and we choose $\varepsilon = 10^{-5}$, then all the three algorithms jump from $n_3 = 3$ to $n_4 = 97$ and, when $n_7 = 100$, we have $\|r_7\| \simeq 0.4 \cdot 10^{-3}$ (see Fig. 1). Therefore when $\varepsilon = 10^{-10}$, only the HMRZ-stab makes the correct jump. The other algorithms do not converge and they give a norm of the residual of the order of 10^{17} (see Fig. 2).

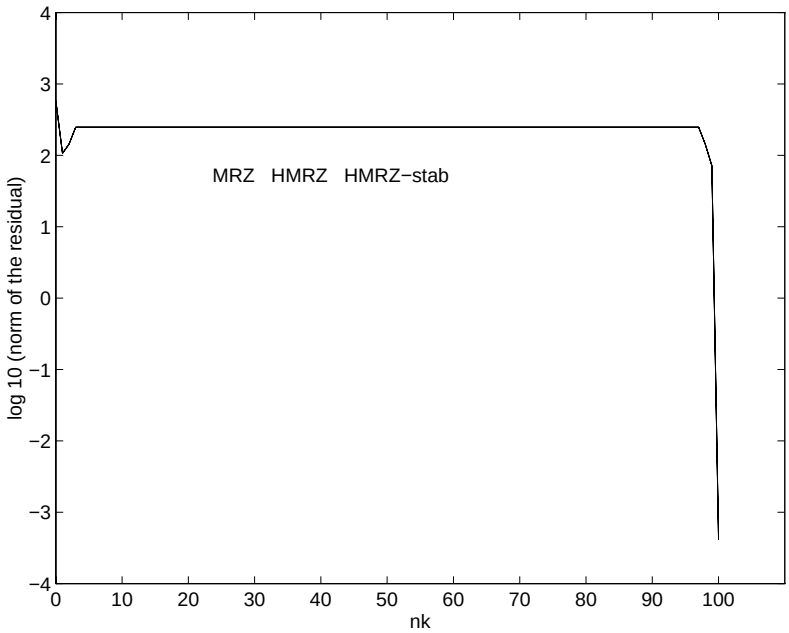

Fig. 1. Example 1. $n = 100, \varepsilon = 10^{-5}$

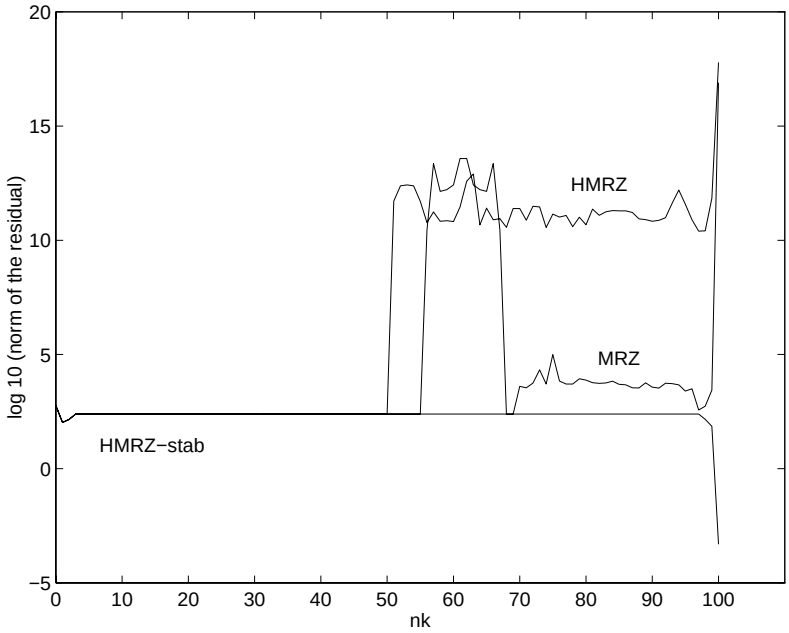

Fig. 2. Example 1. $n = 100, \varepsilon = 10^{-10}$

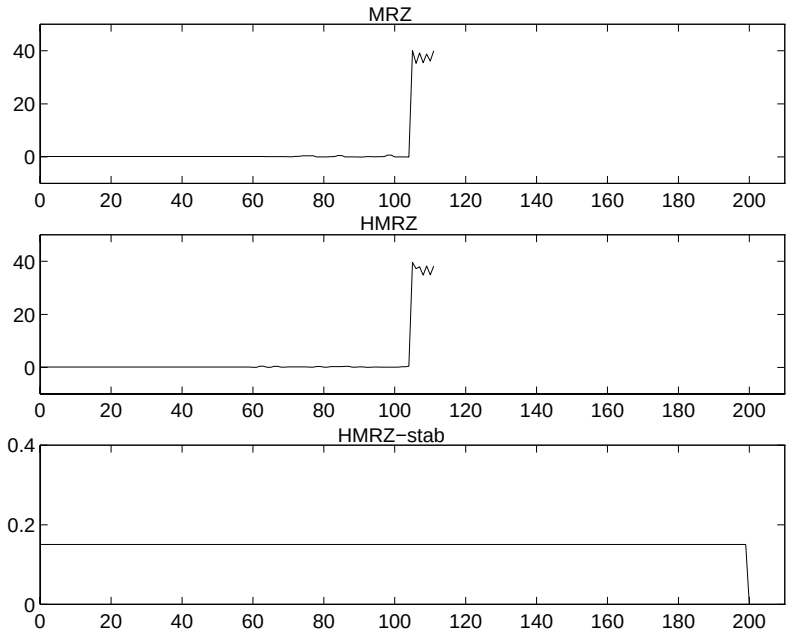

Fig. 3. Example 2. $n = 200, \varepsilon = 10^{-8}$

9.2. Example 2

Let us now consider the following system studied by Brown [15]

$$\begin{pmatrix} a & 1 & & & & & \\ -1 & a & 1 & & & & \\ & -1 & a & \ddots & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & -1 & a & 1 \\ & & & & & -1 & a \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ \vdots \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} a+1 \\ a \\ a \\ \vdots \\ \vdots \\ a \\ a-1 \end{pmatrix}.$$

When $a = 0$, and $y = r_0$, a breakdown occurs every odd step in Lanczos' method, thus we have to have several jumps of lenght 2 starting from the beginning for obtaining the exact solution at $n_k = n$. For $n = 200$, $x_0 = 0$ and $\varepsilon = 10^{-8}$, only the HMRZ-stab detects the correct jumps and obtains the solution at $n_{100} = 200$ and we have $\|r_{100}\| = 0$. The MRZ and the HMRZ have an overflow at $n_k = 112$ (Fig. 3). For the same x_0 and y, but $n = 2000$ and $\varepsilon = 10^{-6}$, the HMRZ-stab again obtains the solution at $n_{1000} = 2000$ with $\|r_{1000}\| = 0$ (Fig. 4). The HSMRZ-stab and the HBMRZ-stab have the same behaviour. For the same example, the FORTRAN 77 version of the HMRZ-stab gives $\|r_{1000}\| = 0.59 \cdot 10^{-5}$ and an actual residual of $0.35 \cdot 10^{-10}$.

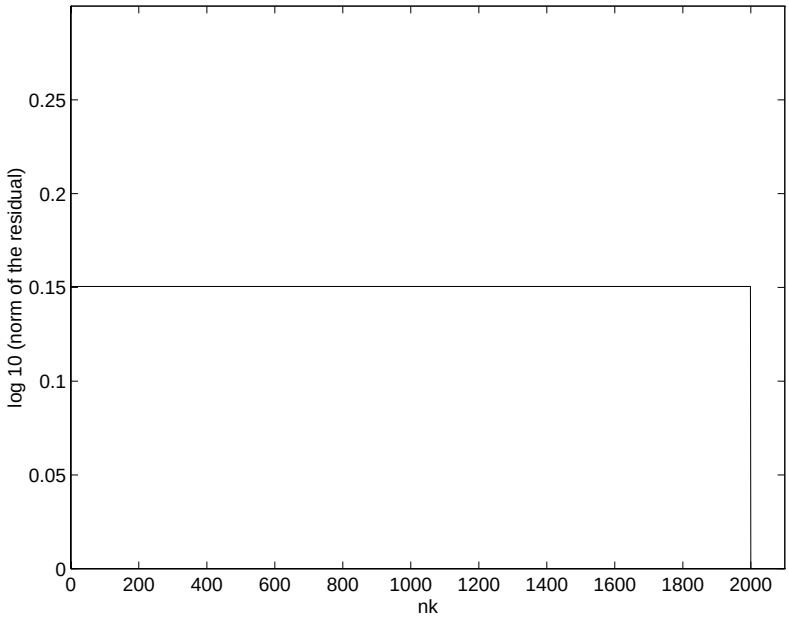

Fig. 4. Example 2. $n = 2000, \varepsilon = 10^{-6}$

9.3. Example 3

The last example is taken from [33]. We consider the 40×40 matrix

$$A = \begin{pmatrix} B & -I & & & \\ -I & B & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & B & -I \\ & & & -I & B \end{pmatrix}, \quad B = \begin{pmatrix} 2 & \alpha & 0 & 0 \\ \beta & 2 & \alpha & 0 \\ 0 & \beta & 2 & \alpha \\ 0 & 0 & \beta & 2 \end{pmatrix},$$

where $\alpha = -1 + \delta$, $\beta = -1 - \delta$ and $\delta = 1.1$.

When $x_0 = 0, y = r_0$ and $\varepsilon = 10^{-8}$, there are no jumps in the algorithms HMRZ, HSMRZ and HBMRZ and we obtain the results of Fig. 5. The best computed residuals are obtained for the HMRZ at $n_k = 25$ with $\|r_{25}\| = 9.4 \cdot 10^{-4}$, for the HSMRZ at $n_k = 29$ with $\|r_{29}\| = 3.2 \cdot 10^{-5}$ and for the HBMRZ at $n_k = 30$ with $\|r_{30}\| = 2.8 \cdot 10^{-5}$. All the stab versions of these algorithms have only one jump starting at $n_k = 20$ and we obtain the results of Fig. 6. The length of the jump varies according to the algorithm ($m_k = 13$ for the HMRZ, $m_k = 11$ for the HSMRZ and $m_k = 9$ for the HBMRZ) and, for these algorithms, smaller residuals are obtained. For $n_k = 40$ we have $\|r_{28}\| = 3.6 \cdot 10^{-11}$ for the HMRZ, $\|r_{30}\| = 2.7 \cdot 10^{-10}$ for the HSMRZ and $\|r_{32}\| = 2.5 \cdot 10^{-11}$ for the HBMRZ.

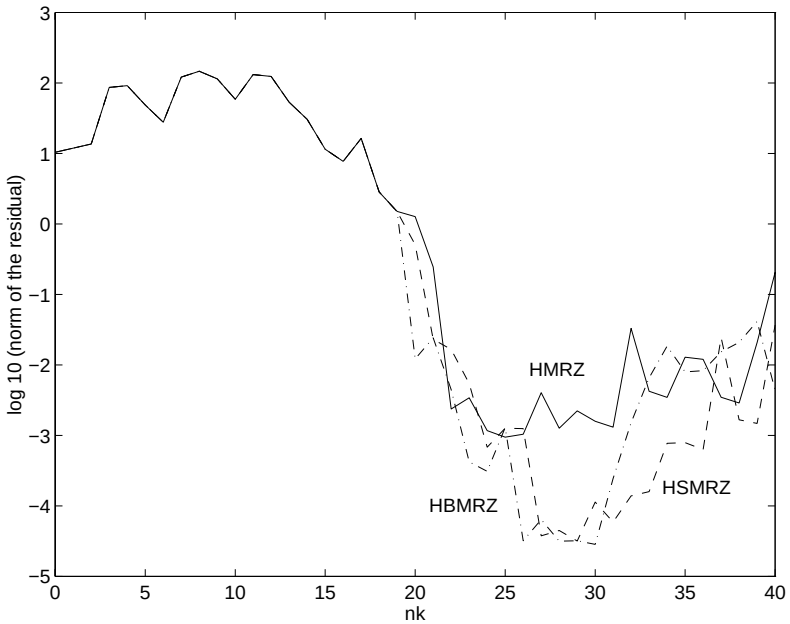

Fig. 5. Example 3. $n = 40, \varepsilon = 10^{-8}$

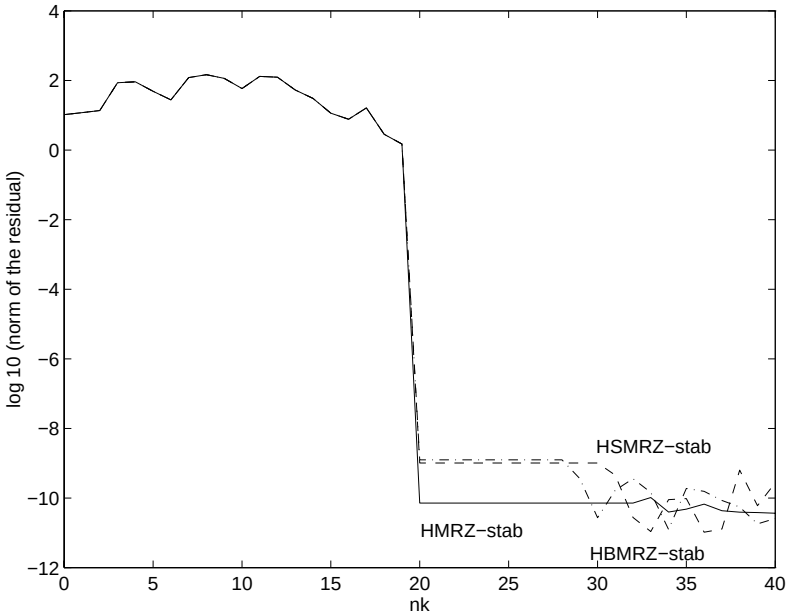

Fig. 6. Example 3. $n = 40, \varepsilon = 10^{-8}$

Final remark

We are in the process of extending the techniques developed in this paper, to the case of near-breakdowns. Doing so, ghost breakdowns (corresponding to the message *Impossible to use the...* in the preceding pseudo-codes) will also be avoided.

References

1. Ayachour, E.H.: Avoiding look-ahead in Lanczos method and Padé approximation, Applicationes Mathematicæ, to appear
2. Baheux, C. (1995): New implementations of Lanczos method, J. Comput. Appl. Math. **57**, 3–15
3. Bank, R.E., Chan, T.F. (1993): An analysis of the composite step bi–conjugate gradient algorithm for solving nonsymmetric systems, Numer. Math. **66**, 295–319
4. Barret, R., Berry, M., et al. (1994): Templates for the Solution of Linear Systems: Building blocks for Iterative Methods, SIAM, Philadelphia, PA
5. Bank, R.E., Chan, T.F. (1994): A composite step bi–conjugate gradient algorithm for solving nonsymmetric systems, Numerical Algorithms **7**, 1–16
6. Boley, D.L., Elhay, S., Golub, G.H., Gutknecht, M.H. (1991): Nonsymmetric Lanczos and finding orthogonal polynomials associated with indefinite weights, Numerical Algorithms **1**, 21–44
7. Brezinski, C. (1980): Padé–Type Approximation and General Orthogonal Polynomials, ISNM vol. 50, Birkhäuser-Verlag, Basel
8. Brezinski, C., Redivo-Zaglia, M. (1991): A new presentation of orthogonal polynomials with applications to their computation, Numerical Algorithms **1**, 207–221
9. Brezinski, C., Redivo-Zaglia, M. (1994): Breakdowns in the computation of orthogonal polynomials, in Nonlinear Numerical Methods and Rational Approximation, A. Cuyt ed., Kluwer, Dordrecht, pp. 49–59
10. Brezinski, C., Redivo-Zaglia, M., Sadok, H. (1991): Avoiding breakdown and near-breakdown in Lanczos type algorithms, Numerical Algorithms **1**, 261–284
11. Brezinski, C., Redivo-Zaglia, M., Sadok, H. (1992): Addendum to "Avoiding breakdown and near-breakdown in Lanczos type algorithms", Numerical Algorithms **2**, 133–136
12. Brezinski, C., Redivo-Zaglia, M., Sadok, H. (1992): A breakdown-free Lanczos type algorithm for solving linear systems, Numer. Math. **63**, 29–38
13. Brezinski, C., Redivo-Zaglia, M., Sadok, H. (1997): Breakdowns in the implementation of the Lánczos method for solving linear systems, Comput. & Math. with Applics. **33**, 31–44
14. Brezinski, C., Sadok, H. (1993): Lanczos type methods for solving systems of linear equations, Appl. Numer. Math. **11**, 443–473
15. Brown, P.N. (1991): A theoretical comparison of the Arnoldi and GMRES algorithms, SIAM J. Sci. Stat. Comput. **12**, 58–78
16. Broyden, C.G.: Look-ahead block-CG algorithms, Optim. Methods and Soft., to appear
17. Chan, T.F., Szeto, T. (1994): A composite step conjugate gradients squared algorithm for solving nonsymmetric linear systems, Numerical Algorithms **7**, 17–32
18. Draux, A. (1983): Polynômes Orthogonaux Formels. Applications, LNM 974, Springer Verlag, Berlin

19. Fletcher, R. (1976): Conjugate gradient methods for indefinite systems, in Numerical Analysis, G.A. Watson ed., LNM 506 (Springer, Berlin) pp. 73–89
20. Freund, R.W., Gutknecht, M.H., Nachtigal, N.M. (1993): An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices, SIAM J. Sci. Comput. **14**, 137–158
21. Graves–Morris, P.R. (1997): A "Look-around Lanczos" algorithm for solving a system of linear equations, Numerical Algorithms **15**, 247–274
22. Gutknecht, M.H. (1990): The unsymmetric Lanczos algorithms and their relations to Padé approximation, continued fractions and the qd algorithm, in Proceedings of the Copper Mountain Conference on Iterative Methods, unpublished
23. Gutknecht, M.H. (1992): A completed theory of the unsymmetric Lanczos process and related algorithms, Part I, SIAM J. Matrix Anal. Appl. **13**, 594–639
24. Gutknecht, M.H. (1994): A completed theory of the unsymmetric Lanczos process and related algorithms, Part II, SIAM J. Matrix Anal. Appl. **15**, 15–58
25. Hegedüs, Cs.J. (1991): Generating conjugate directions for arbitrary matrices by matrix equations, Computers Math. Applic. **21**, 71–85; 87–94
26. Jea, K.C., Young, D.M. (1983): On the simplification of generalized conjugate gradient methods for nonsymmetrizable linear systems, Linear Alg. Appl. **52/53**, 399–417
27. Khelifi, M. (1991): Lanczos maximal algorithm for unsymmetric eigenvalue problems, Appl. Numer. Math. **7**, 179–193
28. Lanczos, C. (1950): An iteration method for the solution of the eigenvalue problem of linear differential and integral operators, J. Res. Natl. Bur. Stand. **45**, 255–282
29. Lanczos, C. (1952): Solution of systems of linear equations by minimized iterations, J. Res. Natl. Bur. Stand. **49**, 33–53
30. Parlett, B.N., Taylor, D.R., Liu, Z.A. (1985): A look-ahead Lanczos algorithm for unsymmetric matrices, Math. Comput. **44**, 105–124
31. Qiang Ye (1994): A breakdown-free variation of the nonsymmetric Lanczos algorithm, Math. Comput. **62**, 179–207
32. Qiang Ye (1996): An adaptative block Lanczos algorithm, Numerical Algorithms **12**, 97–110
33. Saunders, M.A., Simon, H.D., Yip, E.L. (1988): Two conjugate-gradient-type methods for unsymmetric linear equations, SIAM J. Numer. Anal. **25**, 927–940
34. Struble, G.W. (1963): Orthogonal polynomials: variable–signed Weight Functions, Numer. Math. **5**, 88–94
35. Taylor, D.R. (1982): Analysis of the Look–Ahead Lanczos Algorithm, Ph.D. Thesis, Dept. of Mathematics, University of California, Berkeley
36. Tong, C.H., Qiang Ye (1996): A linear system solver based on a modified Krylov subspace method for breakdown recovery, Numerical Algorithms **12**, 233–251
37. Vinsome, P.K.W. (1976): Orthomin, an iterative method for solving sparse sets of simultaneous linear equations, in Proc. 4th Symposium on Reservoir Simulation, Society of Petroleum Engineers of AIME, pp. 149–159
38. Young, D.M., Jea, K.C. (1980): Generalized conjugate–gradient acceleration of nonsymmetrizable iterative methods, Linear Algebra Appl. **34**, 159–194