BIT 35 (1995), 169–201.

LOOK–AHEAD IN BI–CGSTAB AND OTHER PRODUCT METHODS FOR LINEAR SYSTEMS *

C. BREZINSKI[1] and M. REDIVO–ZAGLIA[2]

[2] *Laboratoire d'Analyse Numérique et d'Optimisation, UFR IEEA - M3 Université des Sciences et Technologies de Lille, F–59655 Villeneuve d'Ascq Cedex, France. email: brezinsk@omega.univ-lille1.fr*

[2] *Dipartimento di Elettronica e Informatica, Università degli Studi di Padova via Gradenigo 6/a, I–35131 Padova, Italy. email: elen@elett1.dei.unipd.it*

Abstract.

The Lanczos method for solving $Ax = b$ consists in constructing the sequence of vectors x_k such that $r_k = b - Ax_k = P_k(A)r_0$ where P_k is the orthogonal polynomial of degree at most k with respect to the linear functional c whose moments are $c(\xi^i) = c_i = (y, A^i r_0)$.

In this paper we discuss how to avoid breakdown and near–breakdown in a whole class of methods defined by $r_k = Q_k(A)P_k(A)r_0$, Q_k being a given polynomial. In particular, the case of the Bi–CGSTAB algorithm is treated in detail. Some other choices of the polynomials Q_k are also studied.

AMS subject classification: 65F05, 65B05.

Key words: Linear equations, iterative methods.

1 Introduction.

We consider a system of n linear equations in n unknowns

$$Ax = b.$$

Let x_0 be an arbitrary vector. We set $r_0 = b - Ax_0$ and we define the numbers c_i by

$$c_i = (y, A^i r_0), \qquad i = 0, 1, \ldots$$

where y is an arbitrary nonzero vector and the linear functional c on the vector space of polynomials by

$$c(\xi^i) = c_i, \qquad i = 0, 1, \ldots$$

Let P_k be the polynomial (assumed to exist) of degree at most k, such that

$$P_k(0) = 1$$

*Received April 1994. Revised October 1994.

and

$$c(\xi^i P_k(\xi)) = 0, \qquad i = 0, \ldots, k-1.$$

The family $\{P_k\}$ is called the family of (formal) orthogonal polynomials with respect to c [5].

The Lanczos method [35, 36] for solving $Ax = b$ consists in constructing the sequence of vectors $r_k = b - Ax_k$ defined by

$$r_k = P_k(A)r_0.$$

Since $P_k(0) = 1$, we can write

$$P_k(\xi) = 1 + \xi S_{k-1}(\xi)$$

and thus

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

As we shall see below, the vectors r_k and x_k are usually computed recursively.

An important property of the Lanczos method is its finite termination (if no breakdown occurs), that is $\exists k \leq n$ such that $r_k = 0$ and $x_k = x$. Thus the Lanczos method can also be considered as a direct method for solving a system of linear equations.

The polynomials P_k can be recursively computed by various relations, thus giving rise to the methods known under the names of *Lanczos/Orthores, Lanczos/Orthomin, Lanczos/Orthodir, biores, biodir, biomin* or *biconjugate gradient* and so on, [22, 27, 32, 43]. See [16] for an extensive bibliography and for a unified derivation of all these algorithms based on formal orthogonal polynomials and some new algorithms as well. The connection of these algorithms with Padé approximants and continued fractions is treated in [27]. See also [28]. An account of the history of the subject can be found in [26].

The conjugate gradient squared (CGS) method was proposed by Sonneveld [41]. It consists in taking

$$r_k = P_k^2(A)r_0$$

and then computing x_k such that $r_k = b - Ax_k$.

The Bi-CGSTAB algorithm of van der Vorst [42] consists in taking

$$r_k = Q_k(A)P_k(A)r_0$$

with $Q_k(\xi) = (1 - a_{k-1}\xi)Q_{k-1}(\xi), Q_0(\xi) = 1$ and choosing a_{k-1} such that $(r_k, r_k) = \|r_k\|^2$ is minimum. Then x_k is obtained as before.

In this paper we shall consider some new methods of the same type where the residual vector r_k is defined by

$$r_k = Q_k(A)P_k(A)r_0,$$

Q_k being an arbitrary polynomial (not necessarily of degree k) such that $Q_k(0) = 1$ (this is a necessary condition for being able to compute x_k from r_k without using A^{-1}).

Such a method will be called *Biconjugate Gradient Multiplied*, in short BiCGM. Obviously this class of methods includes the CGS and the Bi–CGSTAB as particular cases. All these methods have the same property of finite termination as Lanczos' if no breakdown occurs. Such methods were first considered almost simultaneously in [6] and [29]. They are also called Lanczos–type product methods.

We shall first propose some choices for the polynomials Q_k. Of course, the only choices of practical interest are those where either the product P_kQ_k can be computed recursively without computing separately P_k and Q_k or where this product is easy to obtain and does not require the storage of too many auxiliary vectors. We shall consider the following methods

BiCG–Min	which uses least squares orthogonal polynomials
BiCG–Adj	which uses the adjacent family $\{P_k^{(1)}\}$
BiCG–Sti	which uses a weak form of Stieltjes polynomials
BiCG–SD	which combines conjugate gradient and steepest descent
BiCG–Ort	which uses a second family of orthogonal polynomials
BiCG–ε	which is based on the ε–algorithm
BiCG–S	which is Sonneveld's CGS method
Bi–CGSTAB	which is van der Vorst's method.

Then we shall show how to implement some of these methods of BiCGM type and how to avoid breakdowns and near–breakdowns in the corresponding algorithms when the polynomials Q_k are computed by a three–term recurrence relation of the most general form.

2 The BiCGM class.

We shall now propose some possible choices for the polynomials Q_k that seem of interest. Of course, other choices can also be studied. These polynomials must satisfy the condition $Q_k(0) = 1$ in order that x_k can be computed without knowing A^{-1}. Indeed, if this condition is satisfied we can write

$$P_k(\xi)Q_k(\xi) = 1 + \xi T_k(\xi).$$

Thus

$$r_k = b - Ax_k = P_k(A)Q_k(A)r_0 = (I + AT_k(A))(b - Ax_0)$$

and we have

$$x_k = x_0 - T_k(A)r_0.$$

These methods can be implemented either by some particular recurrence relations, as for the BiCG–S, or by implementing first the method of Lanczos, that is computing recursively the vectors $r'_k = P_k(A)r_0$, and then taking a linear combination of $r'_k, Ar'_k, \ldots$ with the coefficients of Q_k. Several numerical tests have already been performed with the methods described below. They show that these methods can exhibit, on some numerical examples, interesting convergence properties and are competitive with well established procedures. This

is the reason why we shall now present them even if they are not yet supported by theoretical results and even if the details of their implementation have not been completely worked out. Obviously, their theoretical and practical study has to be pursued.

2.1 BiCG–Min.

Let $Q_k^{(m)}$ be the polynomial of degree at most k such that

$$\sum_{i=0}^{m}\left[c(\xi^i Q_k^{(m)})\right]^2 = \min, \qquad m \geq k-1.$$

Such a polynomial, called a least squares orthogonal polynomial, was introduced by Brezinski and Matos [7].

We shall consider the polynomial $Q_1^{(k)}$ normalized by the condition $Q_1^{(k)}(0) = 1$. Writing it as

$$Q_1^{(k)}(\xi) = 1 - a_k\xi$$

we have

$$a_k = (c_0c_1 + \cdots + c_kc_{k+1})/(c_1^2 + \cdots + c_{k+1}^2).$$

Setting

$$\begin{aligned} \alpha_k &= c_1^2 + \cdots + c_{k+1}^2, \\ \beta_k &= c_0c_1 + \cdots + c_kc_{k+1}, \end{aligned}$$

we have $a_k = \beta_k/\alpha_k$ and

$$\begin{aligned} \alpha_{k+1} &= \alpha_k + c_{k+2}^2 \\ \beta_{k+1} &= \beta_k + c_{k+1}c_{k+2}, \qquad k = -1, 0, 1, \ldots \end{aligned}$$

with $\alpha_{-1} = \beta_{-1} = 0$.

The computation of P_k requires the knowledge of $c_0, \ldots, c_{2k-1}$. For the computation of α_{2k-2} and β_{2k-2} we need the same moments. Thus we shall take

$$r_k = Q_1^{(2k-2)}(A)P_k(A)r_0, \qquad k = 1, 2, \ldots$$

This method will be called *BiCG–Min* because of the minimization property of the least squares orthogonal polynomials $Q_1^{(k)}$. Since no simple recurrence relation exists for the polynomials $Q_k^{(m)}$ for arbitrary m and k, we restrict ourselves to the use of $Q_1^{(k)}$ which can be directly computed.

2.2 BiCG-Adj.

In this method, we shall make use of two adjacent families of orthogonal polynomials (which gives the name to the method) and take

$$Q_k(\xi) = \widetilde{P}_k^{(1)}(\xi) = \xi^k P_k^{(1)}(\xi^{-1}),$$

where $\{P_k^{(1)}\}$ are the monic orthogonal polynomials with respect to the linear functional $c^{(1)}$ defined by

$$c^{(1)}(\xi^i) = c(\xi^{i+1}) = c_{i+1} \qquad i = 0, 1, \ldots$$

Since $P_k^{(1)}$ is taken to be monic, $Q_k(0) = 1$.

These polynomials Q_k satisfy recurrence relations which are easily deduced from those of $P_k^{(1)}$. For example, the relation

$$P_{k+1}^{(1)}(\xi) = (\xi + B_{k+1})P_k^{(1)}(\xi) - C_{k+1}P_{k-1}^{(1)}(\xi)$$

gives

$$Q_{k+1}(\xi) = (1 + B_{k+1}\xi)Q_k(\xi) - C_{k+1}\xi^2 Q_{k-1}(\xi),$$

and the relation

$$P_{k+1}^{(1)}(\xi) = (\xi + A_{k+1})P_k^{(1)}(\xi) - D_{k+1}P_k(\xi)$$

gives

$$Q_{k+1}(\xi) = (1 + A_{k+1}\xi)Q_k(\xi) - D_{k+1}\xi\widetilde{P}_k(\xi)$$

where $\widetilde{P}_k(\xi) = \xi^k P_k(\xi^{-1})$.

2.3 BiCG-Sti.

The Stieltjes polynomial Q_{k+1} of degree $k+1$ is defined by

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

Usually such polynomials do not satisfy a recurrence relationship and they must be computed directly by solving a triangular system of linear equations [39]. This is the reason why we shall restrict ourselves to polynomials satisfying the above condition only for $i = 0$, that is so that

$$c(P_k Q_{k+1}) = 0.$$

Of course, these polynomials are not exactly those defined by Stieltjes. It is easy to check that they satisfy a recurrence relation of the form

$$Q_{k+1}(\xi) = (1 - a_k\xi)Q_k(\xi)\ ,\ Q_0(\xi) = 1.$$

Indeed, we have

$$c(P_k Q_{k+1}) = c((1 - a_k\xi)P_k Q_k) = c(P_k Q_k) - a_k c(\xi P_k Q_k) = 0.$$

That is

$$a_k = c(P_k Q_k)/c(\xi P_k Q_k).$$

Since $c_i = (y, A^i r_0)$ we have

$$a_k = (y, r_k)/(y, Ar_k), \qquad r_k = Q_k(A)P_k(A)r_0.$$

We can likewise consider the polynomials Q_{k+1} such that

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

It can be easily checked that such polynomials satisfy a recurrence relation of the form

$$Q_{k+1}(\xi) = (1 - a_k\xi)Q_k(\xi) - b_k\xi Q_{k-1}(\xi)$$

with $Q_0(\xi) = 1, Q_{-1}(\xi) = 0$ and

$$\begin{pmatrix} a_k \\ b_k \end{pmatrix} = \begin{pmatrix} (y, Ar_k) & (y, As_k) \\ (y, A^2 r_k) & (y, A^2 s_k) \end{pmatrix}^{-1} \begin{pmatrix} (y, r_k) \\ (y, Ar_k) \end{pmatrix},$$

where r_k is defined as before and $s_k = Q_{k-1}(A)P_k(A)r_0$.

2.4 BiCG-SD.

Let us assume that the matrix A is symmetric positive definite. As explained in [24], a projection method for solving $Ax = b$ consists in the iterations

$$x_{k+1} = x_k + a_k u_k,$$

where u_k is an arbitrary vector and where a_k is chosen such that $(r_{k+1}, u_k) = 0$. Since

$$r_{k+1} = b - Ax_{k+1} = b - Ax_k - a_k Au_k = r_k - a_k Au_k,$$

we have

$$a_k = (r_k, u_k)/(u_k, Au_k).$$

The steepest descent (SD) method consists in choosing $u_k = r_k$ and thus

$$r_{k+1} = (I - a_k A)r_k, \qquad a_k = (r_k, r_k)/(r_k, Ar_k).$$

Thus, if we set in this projection method $r_k = Q_k(A)r_0$, then the polynomials Q_k satisfy the recurrence relation

$$Q_{k+1}(\xi) = (1 - a_k\xi)Q_k(\xi) \quad \text{with } Q_0(\xi) = 1.$$

In analogy we can define a BiCGM method such that

$$r_k = Q_k(A)P_k(A)r_0$$

where Q_k is chosen such that $(r_{k+1}, r_k) = 0$. We obtain

$$r_{k+1} = (I - a_k A)Q_k(A)P_{k+1}(A)r_0.$$

But $P_{k+1} = P_k - \lambda_k \xi P_k^{(1)}$ with $\lambda_k = c(U_k P_k)/c(\xi U_k P_k^{(1)})$ where U_k is an arbitrary polynomial of exact degree k and thus

$$r_{k+1} = (I - a_k A)Q_k(A)\Big[P_k(A) - \lambda_k A P_k^{(1)}(A)\Big]r_0.$$

If we set $z_k = Q_k(A)P_k^{(1)}(A)r_0$ then

$$r_{k+1} = (I - a_k A)(r_k - \lambda_k A z_k).$$

Setting $\widetilde{r}_k = r_k - \lambda_k A z_k$ we have $r_{k+1} = \widetilde{r}_k - a_k A\widetilde{r}_k$ and

$$(r_{k+1}, r_k) = (\widetilde{r}_k, r_k) - a_k(A\widetilde{r}_k, r_k),$$

which gives

$$a_k = (\widetilde{r}_k, r_k)/(A\widetilde{r}_k, r_k).$$

This method can be applied even if A is not symmetric positive definite. This method must be compared with Bi-CGSTAB, where a_k is chosen so that (r_{k+1}, r_{k+1}) is minimum [42], and also with the conjugate gradient of Hestenes and Stiefel [31].

If A is not symmetric positive definite then $A^T A$ is non-negative definite and we can take

$$r_{k+1} = (I - a_k A^T A)Q_k(A)P_{k+1}(A)r_0,$$

which gives, with the same notations as above

$$r_{k+1} = \widetilde{r}_k - a_k A^T A\widetilde{r}_k, \qquad a_k = (\widetilde{r}_k, r_k)/(A\widetilde{r}_k, Ar_k).$$

This method must also be compared with Bi-CGSTAB [42].

REMARK 2.1. The iterates computed by the SD method (or its generalization if A is arbitrary) have the form $x_{k+1} = x_k + a_k r_k$ which shows that they fall within the framework defined by Mann [37]. Thus, as proved by Germain-Bonne [25], if $\forall k$, $0 < a_k < 1$, then a necessary and sufficient condition that (x_k) converges to x is that

$$\lim_{k\to\infty} a_k = 0 \quad \text{and} \quad \sum_{k=0}^{\infty} a_k \quad \text{diverges.}$$

2.5 BiCG-Ort.

Let d be a second linear functional on the space of polynomials. We can take for $\{Q_k\}$ the polynomials orthogonal with respect to d, normalized by the usual condition $Q_k(0) = 1$.

Of course, if d is identical to c, this method reduces to the CGS. A particular case is to take for Q_k the Chebyshev orthogonal polynomials which are known to possess acceleration properties, see [40] for a review. This method was already mentioned in [29].

2.6 BiCG-ε.

Let us now consider the choice

$$Q_k(\xi) = (1-\xi)^k = (1-\xi)Q_{k-1}(\xi) \quad \text{with} \quad Q_0(\xi) = 1.$$

As explained in [16], this method corresponds to the second generalization of Shanks transformation proposed in [4]. It can be implemented either by the second topological ε-algorithm (and thus the name of the method) or by a new algorithm generalizing the G-transformation [15]. These algorithms can be extended to systems of nonlinear equations.

2.7 BiCG-S.

For the choice

$$Q_k(\xi) = P_k(\xi)$$

we recover the CGS of Sonneveld [41] and we shall not discuss it further here. The problem of breakdown in this method was solved in [14] and that of near-breakdown in [9].

2.8 Bi-CGSTAB.

This method is due to van der Vorst [42]. It consists in taking

$$r_k = Q_k(A)P_k(A)r_0$$

with $Q_k(\xi) = (1-a_{k-1}\xi)Q_{k-1}(\xi)$ and $Q_0(\xi) = 1$. We have

$$P_{k+1} = P_k - \lambda_k \xi P_k^{(1)}$$

and

$$Q_{k+1}P_{k+1} = (1-a_k\xi)Q_kP_k - \lambda_k\xi(1-a_k\xi)Q_kP_k^{(1)}.$$

We set $z_k = Q_k(A)P_k^{(1)}(A)r_0$. Then

$$r_{k+1} = (I - a_kA)(r_k - \lambda_k A z_k).$$

a_k is chosen such that (r_{k+1}, r_{k+1}) is minimum, that is, if we set

$$\widetilde{r}_k = r_k - \lambda_k A z_k,$$

$$\begin{aligned} f(a_k) &= (r_{k+1}, r_{k+1}) = (\widetilde{r}_k - a_kA\widetilde{r}_k, \widetilde{r}_k - a_kA\widetilde{r}_k) \\ &= (\widetilde{r}_k, \widetilde{r}_k) - 2a_k(\widetilde{r}_k, A\widetilde{r}_k) + a_k^2(A\widetilde{r}_k, A\widetilde{r}_k). \end{aligned}$$

Thus $df/da_k = 0$ if and only if $a_k = (\widetilde{r}_k, A\widetilde{r}_k)/(A\widetilde{r}_k, A\widetilde{r}_k)$. Because of this minimization property, we have

$$(r_{k+1}, r_{k+1}) \leq (\widetilde{r}_k, \widetilde{r}_k)$$

and the Bi-CGSTAB appears as a particular case of the hybrid procedures introduced in [8]. This method must be compared with BiCG–SD, where a_k is such that $(r_{k+1}, r_k) = 0$. A possible variant of this method is to take for Q_k a polynomial of degree 1

$$Q_k(\xi) = 1 - a_k\xi,$$

to set $r_k = (I - a_kA)P_k(A)r_0$, $\tilde{r}_k = P_k(A)r_0$ and to choose a_k in order to minimize (r_k, r_k) that is

$$a_k = (\tilde{r}_k, A\tilde{r}_k)/(A\tilde{r}_k, A\tilde{r}_k)$$

as before.

3 Implementation of the BiCGM.

3.1 The algorithm.

In the Lanczos method, the polynomials P_k are recursively computed. There are several possibilities for this computation, thus leading to the various algorithms which are found in the literature.

Let us first assume that all the polynomials P_k (as defined in section 1) exist and are uniquely determined. A necessary and sufficient condition for that is $\forall k$

$$H_k^{(1)} = \begin{vmatrix} c_1 & c_2 & \dots & c_k \\ c_2 & c_3 & \dots & c_{k+1} \\ \vdots & \vdots & & \vdots \\ c_k & c_{k+1} & \dots & c_{2k-1} \end{vmatrix} \neq 0.$$

In this case the family of polynomials $\{P_k\}$ can be computed by its usual three–term recurrence relationship. Another possibility is to compute simultaneously the families $\{P_k\}$ and $\left\{P_k^{(1)}\right\}$ where the polynomials $P_k^{(1)}$ are the monic orthogonal polynomials with respect to the linear functional $c^{(1)}$ defined by

$$c^{(1)}(\xi^i) = c(\xi^{i+1}) = c_{i+1}, \qquad i = 0, 1, \dots$$

$P_k^{(1)}$ exists if and only if $H_k^{(1)} \neq 0$ which is the same condition as for the existence of P_k. We have [5]

$$P_{k+1}(\xi) = P_k(\xi) - \lambda_k\xi P_k^{(1)}(\xi)$$

with $P_0(\xi) = P_0^{(1)}(\xi) = 1$ and

$$\lambda_k = c(U_kP_k)/c(\xi U_kP_k^{(1)})$$

U_k being an arbitrary polynomial of degree k.

The polynomials $P_k^{(1)}$ can be computed either by their three–term recurrence relationship (see section 2.2) or by

$$P_{k+1}^{(1)}(\xi) = (\xi - \mu_k)P_k^{(1)}(\xi) - D_{k+1}P_k(\xi).$$

There are also other possibilities which are reviewed in [16].

If, for some value of k, the determinant $H_k^{(1)}$ vanishes, then the corresponding orthogonal polynomial P_k does not exist and a division by zero occurs in one of the recurrence relations used in the computation. This phenomenon is called a *breakdown*, more precisely, a *true breakdown* [13] or a *pivot breakdown* [18]. A division by zero can also occur in one of the recurrence relationships even if $H_k^{(1)}$ is different from zero, thus causing a so–called *ghost breakdown* [13] or *Lanczos breakdown* [18]. It is possible to avoid breakdowns in the algorithms for implementing the Lanczos method (and similarly in the BiCGM) by jumping over the non–existing polynomials and computing only the existing ones (called *regular*). This is the technique followed in [12] for avoiding breakdown in Lanczos type methods and in [14] for the CGS. Let us mention that, due to the different normalization of the polynomials $P_k^{(1)}$ which are assumed to be monic, P_k exists if and only if $P_k^{(1)}$ exists.

Now, in the recurrence relations, if a quantity in a denominator is not exactly zero but close to it, then a severe propagation of rounding errors can affect the algorithm, a phenomenon called *near–breakdown*. In order to avoid a near–breakdown, one can jump over the badly computed polynomials and compute directly the first regular polynomial following them. This was the procedure followed in [10, 11] for the Lanczos method and in [9] for the CGS. Let us mention that other look–ahead procedures for treating these problems exist in the literature [1, 18, 2, 23, 28, 30, 34, 38]. However, since these procedures were derived by purely linear algebra techniques, we wonder if they could be easily adapted to the algorithms of the BiCGM class. As we shall see now, our approach following formal orthogonal polynomials, provides a solution to the problems of breakdown and near–breakdown in BiCGM methods.

Let us first give the recurrence relations used in that case. For that, we shall change our notations and give a name only to the polynomials which are used in the recurrence relations. Thus, now, we shall denote by $P_k^{(1)}$ (instead of $P_{n_k}^{(1)}$) the monic polynomial of degree $n_k \geq k$ belonging to the family of orthogonal polynomials with respect to $c^{(1)}$. P_k (instead of P_{n_k}) will denote the polynomial of degree n_k at most belonging to the family of orthogonal polynomials with respect to c. Since similar jumps in the degrees will be considered for the auxiliary polynomials involved in the methods, we shall also denote by Q_k (instead of Q_{n_k}) the corresponding polynomial used in the BiCGM–type method.

Thus, at each step k, we first have to decide the length $m_k \geq 1$ of the jump. The tests for deciding when and how far to jump will be discussed in detail in the next section. We set $n_{k+1} = n_k + m_k$. Then, the following recurrence relations hold [10]

$$P_{k+1}(\xi) = (1 - \xi v_k(\xi))P_k(\xi) - \xi w_k(\xi)P_k^{(1)}(\xi), \tag{3.1}$$

$$P_{k+1}^{(1)}(\xi) = q_k(\xi)P_k^{(1)}(\xi) + t_k(\xi)P_k(\xi), \tag{3.2}$$

where w_k is a polynomial of degree $m_k - 1$ at most, v_k a polynomial of degree $m_k - 2$ at most if $n_k - m_k + 1 \geq 0$ and $n_k - 1$ otherwise, q_k a monic polynomial of degree m_k and t_k a polynomial of degree $m_k - 1$ at most if $n_k - m_k \geq 0$ and

$n_k - 1$ otherwise. In the case of breakdown, v_k and t_k are identically zero. The coefficients of these four polynomials are obtained as the solutions of systems of linear equations, see [10, 11]. In order that these relations hold, it must be assumed that $c(\xi^{n_k} P_k) \neq 0$.

In order to show how to avoid breakdown and near–breakdown in BiCGM methods, we shall now assume that the polynomials Q_k satisfy a recurrence relationship of the form

$$Q_{k+1}(\xi) = E_k(\xi)Q_k(\xi) + \xi F_k(\xi)Q_{k-1}(\xi) \tag{3.3}$$

with $Q_{-1}(\xi) = 0$ and $Q_0(\xi) = 1$. E_k and F_k are arbitrary polynomials of arbitrary degrees. Since we need to have $Q_{k+1}(0) = 1$, we must have $E_k(0) = 1$.

In many of the methods suggested in section 2, the polynomials Q_k satisfy a recurrence of this type. We shall now see how the product $Q_k P_k$ can be recursively obtained.

Multiplying (3.1) by (3.3) gives (dropping the variable ξ when unnecessary)

$$\begin{aligned} Q_{k+1}P_{k+1} &= E_k(1 - \xi v_k)Q_k P_k - \xi E_k \omega_k Q_k P_k^{(1)} \\ &+ \xi F_k(1 - \xi v_k)Q_{k-1}P_k - \xi^2 F_k \omega_k Q_{k-1} P_k^{(1)}. \end{aligned} \tag{3.4}$$

Thus, we also need to compute recursively the products $Q_k P_k^{(1)}$, $Q_{k-1}P_k$ and $Q_{k-1}P_k^{(1)}$. Multiplying (3.2) by (3.3), we have

$$\begin{aligned} Q_{k+1}P_{k+1}^{(1)} &= E_k q_k Q_k P_k^{(1)} + E_k t_k Q_k P_k + \xi F_k q_k Q_{k-1} P_k^{(1)} \\ &\quad + \xi t_k F_k Q_{k-1} P_k. \end{aligned} \tag{3.5}$$

Multiplying (3.1) and (3.2) by Q_k leads to

$$Q_k P_{k+1} = (1 - \xi v_k)Q_k P_k - \xi \omega_k Q_k P_k^{(1)}, \tag{3.6}$$

$$Q_k P_{k+1}^{(1)} = q_k Q_k P_k^{(1)} + t_k Q_k P_k. \tag{3.7}$$

Thus if we set

$$\begin{aligned} r_k &= P_k(A)Q_k(A)r_0, & z_k &= Q_k(A)P_k^{(1)}(A)r_0 \\ s_k &= Q_{k-1}(A)P_k(A)r_0, & p_k &= Q_{k-1}(A)P_k^{(1)}(A)r_0 \end{aligned}$$

the relations (3.4), (3.5), (3.6) and (3.7) give immediately

$$\begin{aligned} r_{k+1} &= E_k(A)(I - Av_k(A))r_k - AE_k(A)w_k(A)z_k \\ &+ AF_k(A)(I - Av_k(A))s_k - A^2 F_k(A)w_k(A)p_k \end{aligned} \tag{3.8}$$

$$\begin{aligned} z_{k+1} &= E_k(A)q_k(A)z_k + E_k(A)t_k(A)r_k + AF_k(A)q_k(A)p_k \\ &+ At_k(A)F_k(A)s_k \end{aligned} \tag{3.9}$$

$$s_{k+1} = (I - Av_k(A))r_k - Aw_k(A)z_k \tag{3.10}$$

$$p_{k+1} = q_k(A)z_k + t_k(A)r_k. \tag{3.11}$$

Let us now see how to compute x_{k+1} from r_{k+1}. Since $E_k(0) = 1$, we can write

$$E_k(\xi) = 1 + \xi R_k(\xi).$$

Thus

$$E_k(1 - \xi v_k) = (1 + \xi R_k)(1 - \xi v_k) = 1 + \xi(R_k - v_k - \xi R_k v_k)$$

and it follows, since $\xi R_k = E_k - 1$

$$\begin{aligned} x_{k+1} = \quad & x_k - (R_k(A) - E_k(A)v_k(A))r_k + E_k(A)w_k(A)z_k \\ & -F_k(A)(I - Av_k(A))s_k + AF_k(A)w_k(A)p_k. \end{aligned}$$

Let us mention that, in (3.3), the variable ξ is needed in the second term of the right hand side in order to be able to compute x_{k+1} from r_{k+1} without using A^{-1}.

In the case of CGS, the formulae given in [9] are recovered. Thus we are able to avoid breakdown and near–breakdown in all the methods of the BiCGM–type and, in particular, in the Bi–CGSTAB method of van der Vorst [42] which seems to be very efficient and promising.

At each step, the number of matrix–by–vector and inner products and the storage requirements depend on the degree of the polynomial Q_k. For example, in the case of CGS, we need $6m_k - 3$ matrix–by–vector products if $m_k \leq n_k + 1$, and $5m_k + n_k - 2$ products otherwise and the storage overhead is $\mathcal{O}(m_k n)$.

3.2 Avoiding A^T.

One of the main interest of CGS [41] and Bi–CGSTAB [42] is that their implementation avoids the use of A^T. In [14] and [9], it was respectively shown how to avoid the breakdown and the near–breakdown in the CGS without using A^T. We shall now explain how to avoid the use of A^T when curing breakdown and near–breakdown in the algorithms of the BiCGM class in the particular case where the polynomials F_k are identically zero which is a case covering Bi–CGSTAB. We shall not give here all the details since they are quite similar to those given in [9] where they can be found.

The coefficients of the linear systems giving the coefficients of the polynomials v_k, w_k, q_k and t_k involved in (3.1) and (3.2) have the form

$$c(\xi^{n_k+i}P_k) \quad \text{and} \quad c^{(1)}\left(\xi^{n_k+i}P_k^{(1)}\right)$$

for $i = 0, \ldots, 2m_k - 1$. We have, from the definition of the moments of the linear functional c given in section 1,

$$c(\xi^{n_k+i}P_k) = (y, A^{n_k+i}P_k(A)r_0), \quad c^{(1)}(\xi^{n_k+i}P_k^{(1)}) = (y, A^{n_k+i+1}P_k^{(1)}(A)r_0).$$

In order to avoid the computation of quantities like $(y, A^{n_k+j}z)$ for high value of $n_k + j$, such scalar products could be rewritten as $(A^{T^{n_k}}y, A^j z)$, but such

expressions imply using A^T. We shall now see how to avoid this drawback, even if a relation such as (3.3) does not occur among the polynomials Q_k.

Let us assume now that $\forall k, Q_k$ has exactly the degree n_k. We set

$$Q_k(\xi) = 1 + a_1^{(k)}\xi + \cdots + a_{n_k}^{(k)}\xi^{n_k}$$

and recall that (3.1) holds, that is

$$\begin{aligned} P_{k+1}(\xi) &= P_k(\xi) - \xi\, w_k(\xi)\, P_k^{(1)}(\xi) - \xi\, v_k(\xi)\, P_k(\xi) \\ &= (1 - \xi\, v_k(\xi))\, P_k(\xi) - \xi\, w_k(\xi)\, P_k^{(1)}(\xi) \end{aligned} \tag{3.12}$$

where w_k is a polynomial of degree $m_k - 1$ at most and v_k a polynomial of degree $m_k - 2$ at most (if $m_k \leq n_k$) and of degree $n_k - 1$ at most (if $m_k > n_k$).

Let us now explain how to obtain the coefficients of the polynomials v_k and w_k without having to use A^T. Two cases have to be considered according to the respective values of n_k and m_k.

- Case $n_k - m_k + 1 \geq 0$.

  –We set

$$w_k(\xi) = \gamma_0 + \cdots + \gamma_{m_k-1}\xi^{m_k-1}.$$

There exists a polynomial U_1 of degree $n_k - 1$ and a constant β_{m_k-1} such that

$$\xi^{n_k-m_k+1} w_k(\xi) = U_1(\xi) + \beta_{m_k-1} Q_k(\xi).$$

Identifying the coefficients of ξ^{n_k} in both sides of this relation, we obtain

$$\gamma_{m_k-1} = \beta_{m_k-1}\, a_{n_k}^{(k)}.$$

There exists a polynomial U_2 of degree $n_k - 1$ and two constants β_{m_k-1} and β_{m_k-2} such that

$$\xi^{n_k-m_k+2} w_k(\xi) = U_2(\xi) + \beta_{m_k-2} Q_k(\xi) + \beta_{m_k-1}\xi Q_k(\xi).$$

Identifying the coefficients of ξ^{n_k} leads to

$$\gamma_{m_k-2} = \beta_{m_k-2} a_{n_k}^{(k)} + \beta_{m_k-1} a_{n_k-1}^{(k)}$$

and identifying the coefficients of ξ^{n_k+1} we have the same relation as above for γ_{m_k-1}, which proves that β_{m_k-1} is the same as above.

And so on, there exists a polynomial U_{m_k} of degree $n_k - 1$ and constants $\beta_0, \ldots, \beta_{m_k-1}$ such that

$$\xi^{n_k} w_k(\xi) = U_{m_k}(\xi) + \beta_0 Q_k(\xi) + \cdots + \beta_{m_k-1}\xi^{m_k-1} Q_k(\xi).$$

Identifying, we obtain one more new relation

$$\gamma_0 = \beta_0 a_{n_k}^{(k)} + \beta_1 a_{n_k-1}^{(k)} + \cdots + \beta_{m_k-1} a_{n_k-m_k+1}^{(k)}$$

with $a_0^{(k)} = 1$.

–We set

$$v_k(\xi) = \gamma'_0 + \cdots + \gamma'_{m_k-2}\xi^{m_k-2}.$$

Now, similarly, there exists a polynomial V_1 of degree $n_k - 1$ and a constant β'_{m_k-2} such that

$$\xi^{n_k-m_k+2}v_k(\xi) = V_1(\xi) + \beta'_{m_k-2}Q_k(\xi).$$

By identification, we have

$$\gamma'_{m_k-2} = \beta'_{m_k-2}a_{n_k}^{(k)}.$$

There exists a polynomial V_2 of degree $n_k - 1$ and constants β'_{m_k-2} and β'_{m_k-3} such that

$$\xi^{n_k-m_k+3}v_k(\xi) = V_2(\xi) + \beta'_{m_k-3}Q_k(\xi) + \beta'_{m_k-2}\xi Q_k(\xi).$$

Identifying, we have

$$\gamma'_{m_k-3} = \beta'_{m_k-3}a_{n_k}^{(k)} + \beta'_{m_k-2}a_{n_k-1}^{(k)}.$$

And so on, there exists a polynomial V_{m_k-1} of degree $n_k - 1$ and constants $\beta'_0, \ldots, \beta'_{m_k-2}$ such that

$$\xi^{n_k}v_k(\xi) = V_{m_k-1}(\xi) + \beta'_0Q_k(\xi) + \cdots + \beta'_{m_k-2}\xi^{m_k-2}Q_k(\xi).$$

By identification, we obtain

$$\gamma'_0 = \beta'_0a_{n_k}^{(k)} + \beta'_1a_{n_k-1}^{(k)} + \cdots + \beta'_{m_k-2}a_{n_k-m_k+2}^{(k)}.$$

The preceding relations allow us to compute the γ's and the γ''s from the β's and the β''s. Let us now show how to compute the β's and the β''s.

We apply the orthogonality conditions of P_{k+1} with respect to ξ^i for $i = n_k - m_k+1, \ldots, n_k-1$. Writing down the orthogonality condition $c(\xi^{n_k-m_k+1}P_{k+1}) = 0$, we obtain

$$\begin{aligned} c\left(\xi^{n_k-m_k+1}P_{k+1}\right) &= c\left(\xi^{n_k-m_k+1}P_k\right) - c^{(1)}\left(\xi^{n_k-m_k+1}w_kP_k^{(1)}\right) \\ &\quad -c\left(\xi^{n_k-m_k+2}v_kP_k\right) = 0. \end{aligned}$$

Expressing $w_kP_k^{(1)}$ and v_kP_k in terms of the U_i's and the V_i's by means of the relations given above and using the orthogonality of P_k and $P_k^{(1)}$, we get

$$c^{(1)}\left(U_1P_k^{(1)}\right) + \beta_{m_k-1}c^{(1)}\left(Q_k\,P_k^{(1)}\right) + c\left(V_1P_k\right) + \beta'_{m_k-2}c\left(Q_k\,P_k\right) = 0.$$

But, since U_1 and V_1 have degree $n_k - 1$, then $c^{(1)}\left(U_1P_k^{(1)}\right) = c\left(V_1P_k\right) = 0$, and it follows

$$\beta_{m_k-1}c^{(1)}\left(Q_kP_k^{(1)}\right)+\beta'_{m_k-2}c\left(Q_k\,P_k\right)=0.$$

Writing down the orthogonality condition $c\left(\xi^{n_k-m_k+2}P_{k+1}\right)=0$ and using $c^{(1)}\left(U_2P_k^{(1)}\right)=c\left(V_2P_k\right)=0$, we obtain

$$\begin{aligned}&\beta_{m_k-2}c^{(1)}\left(Q_kP_k^{(1)}\right)+\beta_{m_k-1}c^{(1)}\left(\xi Q_kP_k^{(1)}\right)+\beta'_{m_k-3}c\left(Q_kP_k\right)\\&+\beta'_{m_k-2}c\left(\xi Q_kP_k\right)=0.\end{aligned}$$

And so on until $c(\xi^{n_k-1}P_{k+1})=0$ which gives

$$\begin{aligned}&\beta_1c^{(1)}\left(Q_kP_k^{(1)}\right)+\cdots+\beta_{m_k-1}c^{(1)}\left(\xi^{m_k-2}Q_kP_k^{(1)}\right)+\beta'_0c\left(Q_kP_k\right)+\cdots\\&+\beta'_{m_k-2}c\left(\xi^{m_k-2}Q_kP_k\right)=0.\end{aligned}$$

Thus we have obtained m_k-1 relations for determining the $2m_k-1$ unknown coefficients β_i and β'_i.

We shall now write the remaining orthogonality conditions of P_{k+1} as

$$c(\xi^{i-n_k}Q_kP_{k+1})=0\qquad\text{for}\quad i=n_k,\ldots,n_k+m_k-1.$$

Thus we obtain for $i=0,\ldots,m_k-1$

$$c^{(1)}\left(\xi^iw_kQ_kP_k^{(1)}\right)+c\left(\xi^{i+1}v_kQ_kP_k\right)=c\left(\xi^iQ_kP_k\right),$$

that is

$$\begin{aligned}&\gamma_0c^{(1)}\left(\xi^iQ_kP_k^{(1)}\right)+\cdots+\gamma_{m_k-1}c^{(1)}\left(\xi^{i+m_k-1}Q_kP_k^{(1)}\right)+\\&\gamma'_0c\left(\xi^{i+1}Q_kP_k\right)+\cdots+\gamma'_{m_k-2}c\left(\xi^{i+m_k-1}Q_kP_k\right)=c\left(\xi^iQ_kP_k\right).\end{aligned}$$

Replacing, in these relations, the γ's and the γ''s by their expressions above, we obtain m_k additional relations for the β's and the β''s. Setting

$$c_i=c^{(1)}\left(\xi^iQ_kP_k^{(1)}\right)\qquad d_i=c\left(\xi^iQ_kP_k\right)$$

these m_k relations become (for $i=0,\ldots,m_k-1$)

$$\gamma_0c_i+\cdots+\gamma_{m_k-1}c_{i+m_k-1}+\gamma'_0d_{i+1}+\cdots+\gamma'_{m_k-2}d_{i+m_k-1}=d_i,$$

that is

$$\begin{aligned}&c_i\left(\beta_0a_{n_k}^{(k)}+\cdots+\beta_{m_k-1}a_{n_k-m_k+1}^{(k)}\right)+\cdots+c_{i+m_k-1}\left(\beta_{m_k-1}a_{n_k}^{(k)}\right)+\\&d_{i+1}\left(\beta'_0a_{n_k}^{(k)}+\cdots+\beta'_{m_k-2}a_{n_k-m_k+2}^{(k)}\right)+\cdots+d_{i+m_k-1}\left(\beta'_{m_k-2}a_{n_k}^{(k)}\right)=d_i,\end{aligned}$$

or

$$\begin{aligned}
&\beta_0\left(c_i a_{n_k}^{(k)}\right)+\beta_1\left(c_i a_{n_k-1}^{(k)}+c_{i+1}a_{n_k}^{(k)}\right)+\cdots\\
&\quad+\beta_{m_k-1}\left(c_i a_{n_k-m_k+1}^{(k)}+\cdots+c_{i+m_k-1}a_{n_k}^{(k)}\right)+\\
&\beta_0'\left(d_{i+1}a_{n_k}^{(k)}\right)+\beta_1'\left(d_{i+1}a_{n_k-1}^{(k)}+d_{i+2}a_{n_k}^{(k)}\right)+\cdots\\
&\quad+\beta_{m_k-2}'\left(d_{i+1}a_{n_k-m_k+2}^{(k)}+\cdots+d_{i+m_k-1}a_{n_k}^{(k)}\right)=d_i.
\end{aligned}$$

Thus, we have the same number of unknowns and relations.

- Case $n_k - m_k + 1 < 0$.

We have again

$$w_k(\xi)=\gamma_0+\cdots+\gamma_{m_k-1}\xi^{m_k-1}.$$

Since $n_k < m_k - 1$ and w_k is a polynomial of degree at most $m_k - 1$, then there exists a polynomial U_1 of degree $n_k - 1$ and constants $\beta_{n_k},\ldots,\beta_{m_k-1}$ such that

$$w_k(\xi)=U_1(\xi)+\beta_{n_k}Q_k(\xi)+\cdots+\beta_{m_k-1}\xi^{m_k-n_k-1}Q_k(\xi).$$

Identifying the coefficients of the successive terms in ξ^i for $i=n_k,\ldots,m_k-1$, we obtain (with the convention that $a_i^{(k)}=0$ for $i<0$)

$$\begin{aligned}
&\gamma_{n_k}=\beta_{n_k}a_{n_k}^{(k)}+\beta_{n_k+1}a_{n_k-1}^{(k)}+\cdots+\beta_{m_k-1}a_{2n_k-m_k+1}^{(k)},\\
&\gamma_{n_k+1}=\beta_{n_k+1}a_{n_k}^{(k)}+\beta_{n_k+2}a_{n_k-1}^{(k)}+\cdots+\beta_{m_k-1}a_{2n_k-m_k+2}^{(k)},\\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\
&\gamma_{m_k-1}=\beta_{m_k-1}a_{n_k}^{(k)}.
\end{aligned}$$

There exists a polynomial U_2 of degree $n_k - 1$ and constants $\beta_{n_k-1},\beta_{n_k},\ldots,$ β_{m_k-1} such that

$$\xi w_k(\xi)=U_2(\xi)+\beta_{n_k-1}Q_k(\xi)+\cdots+\beta_{m_k-1}\xi^{m_k-n_k}Q_k(\xi).$$

By identification we obtain

$$\gamma_{n_k-1}=\beta_{n_k-1}a_{n_k}^{(k)}+\beta_{n_k}a_{n_k-1}^{(k)}+\cdots+\beta_{m_k-1}a_{2n_k-m_k}^{(k)}$$

and the same relations as above for $\gamma_{n_k},\ldots,\gamma_{m_k-1}$.

And so on, there exists a polynomial U_{n_k+1} of degree $n_k - 1$ and constants $\beta_0,\ldots,\beta_{m_k-1}$ such that

$$\xi^{n_k}w_k(\xi)=U_{n_k+1}(\xi)+\beta_0Q_k(\xi)+\cdots+\beta_{m_k-1}\xi^{m_k-1}Q_k(\xi).$$

Identifying, we obtain

$$\gamma_0=\beta_0a_{n_k}^{(k)}+\beta_1a_{n_k-1}^{(k)}+\cdots+\beta_{m_k-1}a_{n_k-m_k+1}^{(k)}.$$

–In this case we shall take for v_k a polynomial of degree at most $n_k - 1$ (for $k = 0$, $n_0 = 0$ and there is no polynomial v_0) and we set

$$v_k(\xi) = \gamma'_0 + \cdots + \gamma'_{n_k-1}\xi^{n_k-1}.$$

Similarly, there exists a polynomial V_1 of degree $n_k - 1$ and a constant β'_{n_k-1} such that

$$\xi v_k(\xi) = V_1(\xi) + \beta'_{n_k-1}Q_k(\xi).$$

Identifying, we get

$$\gamma'_{n_k-1} = \beta'_{n_k-1}a^{(k)}_{n_k}.$$

And so on, there exists constants $\beta'_0, \ldots, \beta'_{n_k-1}$ and a polynomial V_{n_k} of degree $n_k - 1$ such that

$$\xi^{n_k}v_k(\xi) = V_{n_k}(\xi) + \beta'_0Q_k(\xi) + \cdots + \beta'_{n_k-1}\xi^{n_k-1}Q_k(\xi),$$

which gives, by identification

$$\gamma'_0 = \beta'_0a^{(k)}_{n_k} + \beta'_1a^{(k)}_{n_k-1} + \cdots + \beta'_{n_k-1}a^{(k)}_1.$$

Writing down the orthogonality conditions $c\left(\xi^i P_{k+1}\right) = 0$ for $i = 0, \ldots, n_k - 1$, and using $c^{(1)}\left(U_1P^{(1)}_k\right) = c(V_1P_k) = 0$, we obtain

$$\begin{aligned}
&\beta_{n_k}c^{(1)}\left(Q_kP^{(1)}_k\right) + \cdots + \beta_{m_k-1}c^{(1)}\left(\xi^{m_k-n_k-1}Q_kP^{(1)}_k\right) + \\
&\quad \beta'_{n_k-1}c\left(Q_kP_k\right) = 0,\\
&\beta_{n_k-1}c^{(1)}\left(Q_kP^{(1)}_k\right) + \cdots + \beta_{m_k-1}c^{(1)}\left(\xi^{m_k-n_k}Q_kP^{(1)}_k\right) + \\
&\quad \beta'_{n_k-2}c\left(Q_kP_k\right) + \beta'_{n_k-1}c\left(\xi Q_kP_k\right) = 0,\\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\
&\beta_1c^{(1)}(Q_kP^{(1)}_k) + \cdots + \beta_{m_k-1}c^{(1)}(\xi^{m_k-2}Q_kP^{(1)}_k) + \beta'_0c(Q_kP_k) + \cdots + \\
&\quad \beta'_{n_k-1}c(\xi^{n_k-1}Q_kP_k) = 0.
\end{aligned}$$

Thus, we have already obtained n_k relations.

Now, writing down the conditions

$$c\left(\xi^iQ_kP_{k+1}\right) = 0$$

for $i = 0, \ldots, m_k - 1$, we obtain

$$\begin{aligned}
&\gamma_0c^{(1)}\left(\xi^iQ_kP^{(1)}_k\right) + \cdots + \gamma_{m_k-1}c^{(1)}\left(\xi^{i+m_k-1}Q_kP^{(1)}_k\right) + \\
&\qquad \gamma'_0c\left(\xi^{i+1}Q_kP_k\right) + \cdots + \gamma'_{n_k-1}c\left(\xi^{i+n_k}Q_kP_k\right) = c\left(\xi^iQ_kP_k\right).
\end{aligned}$$

Replacing, as above, the γ_i and the γ'_i by their expressions, and using the c_i's and the d_i's previously defined we obtain the following m_k additional relations for the β_i and the β'_i

$$\gamma_0c_i + \cdots + \gamma_{m_k-1}c_{i+m_k-1} + \gamma'_0d_{i+1} + \cdots + \gamma'_{n_k-1}d_{i+n_k} = d_i,$$

that is

$$c_i\left(\beta_0 a_{n_k}^{(k)}+\cdots+\beta_{m_k-1}a_{n_k-m_k+1}^{(k)}\right)+\cdots+c_{i+m_k-1}\left(\beta_{m_k-1}a_{n_k}^{(k)}\right)+$$
$$d_{i+1}\left(\beta'_0 a_{n_k}^{(k)}+\cdots+\beta'_{n_k-1}a_1^{(k)}\right)+\cdots+d_{i+n_k}\left(\beta'_{n_k-1}a_{n_k}^{(k)}\right)=d_i,$$

or

$$\beta_0\left(c_i a_{n_k}^{(k)}\right)+\beta_1\left(c_i a_{n_k-1}^{(k)}+c_{i+1}a_{n_k}^{(k)}\right)+\cdots+$$
$$\beta_{m_k-1}\left(c_i a_{n_k-m_k+1}^{(k)}+\cdots+c_{i+m_k-1}a_{n_k}^{(k)}\right)+\beta'_0\left(d_{i+1}a_{n_k}^{(k)}\right)+$$
$$\beta'_1\left(d_{i+1}a_{n_k-1}^{(k)}+d_{i+2}a_{n_k}^{(k)}\right)+\cdots+\beta'_{n_k-1}\left(d_{i+1}a_1^{(k)}+\cdots+d_{i+n_k}a_{n_k}^{(k)}\right)=d_i.$$

Thus, finally, we have n_k+m_k relations with the same number of unknowns.

Let us now try to compute recursively $P_{k+1}^{(1)}$. We shall look for $P_{k+1}^{(1)}$ under the form

$$P_{k+1}^{(1)}(\xi) = q_k(\xi)\,P_k^{(1)}(\xi)+t_k(\xi)\,P_k(\xi) \tag{3.13}$$

where q_k is a monic polynomial of degree m_k and t_k a polynomial of degree m_k-1 at most (if $m_k\leq n_k$) or of degree n_k-1 at most (if $m_k>n_k$). For computing the coefficients of the polynomials q_k and t_k two cases have to be considered according to the respective values of n_k and m_k.

- Case $n_k-m_k\geq 0$.
  We set

$$q_k(\xi)=\eta_0+\cdots+\eta_{m_k-1}\xi^{m_k-1}+\xi^{m_k}.$$

There exists a polynomial U_1 of degree n_k-1 and a constant α_{m_k} such that

$$\xi^{n_k-m_k}q_k(\xi)=U_1(\xi)+\alpha_{m_k}Q_k(\xi).$$

Identifying the coefficients of ξ^{n_k} in both sides of this relation, we obtain

$$1=\alpha_{m_k}a_{n_k}^{(k)}.$$

There exists a polynomial U_2 of degree n_k-1 and two constants α_{m_k} and α_{m_k-1} such that

$$\xi^{n_k-m_k+1}q_k(\xi)=U_2(\xi)+\alpha_{m_k-1}Q_k(\xi)+\alpha_{m_k}\xi Q_k(\xi).$$

Identifying the coefficients of ξ^{n_k} leads to

$$\eta_{m_k-1}=\alpha_{m_k-1}a_{n_k}^{(k)}+\alpha_{m_k}a_{n_k-1}^{(k)}$$

and identifying the coefficients of ξ^{n_k+1} we have the same relation as above which proves that α_{m_k} is the same as above. And so on, there exists a polynomial U_{m_k+1} of degree n_k-1 and constants $\alpha_0,\ldots,\alpha_{m_k}$ such that

$$\xi^{n_k} q_k(\xi) = U_{m_k+1}(\xi) + \alpha_0 Q_k(\xi) + \cdots + \alpha_{m_k} \xi^{m_k} Q_k(\xi).$$

Identifying, we obtain one more new relation

$$\eta_0 = \alpha_0 a_{n_k}^{(k)} + \alpha_1 a_{n_k-1}^{(k)} + \cdots + \alpha_{m_k} a_{n_k-m_k}^{(k)}.$$

–We set

$$t_k(\xi) = \eta'_0 + \cdots + \eta'_{m_k-1} \xi^{m_k-1}.$$

Now, similarly, there exists a polynomial V_1 of degree $n_k - 1$ and a constant α'_{m_k-1} such that

$$\xi^{n_k-m_k+1} t_k(\xi) = V_1(\xi) + \alpha'_{m_k-1} Q_k(\xi).$$

By identification, we have

$$\eta'_{m_k-1} = \alpha'_{m_k-1} a_{n_k}^{(k)}.$$

And so on, there exists a polynomial V_{m_k} of degree $n_k - 1$ and constants $\alpha'_0, \ldots, \alpha'_{m_k-1}$ such that

$$\xi^{n_k} t_k(\xi) = V_{m_k}(\xi) + \alpha'_0 Q_k(\xi) + \cdots + \alpha'_{m_k-1} \xi^{m_k-1} Q_k(\xi).$$

By identification, we obtain

$$\eta'_0 = \alpha'_0 a_{n_k}^{(k)} + \alpha'_1 a_{n_k-1}^{(k)} + \cdots + \alpha'_{m_k-1} a_{n_k-m_k+1}^{(k)}.$$

The preceding relations allow us to compute the η's and the η''s from the α's and the α''s. Let us now show how to compute the α's and the α''s.

We apply the orthogonality conditions of $P_{k+1}^{(1)}$ with respect to ξ^i for $i = n_k - m_k, \ldots, n_k - 1$. Writing down the orthogonality condition $c^{(1)}\left(\xi^{n_k-m_k} P_{k+1}^{(1)}\right) = 0$, we obtain

$$c^{(1)}\left(\xi^{n_k-m_k} P_{k+1}^{(1)}\right) = c^{(1)}\left(\xi^{n_k-m_k} q_k P_k^{(1)}\right) + c\left(\xi^{n_k-m_k+1} t_k P_k\right) = 0.$$

Expressing $q_k P_k^{(1)}$ and $t_k P_k$ in terms of the U_i's and the V_i's by means of the relations given above and using the orthogonality of P_k and $P_k^{(1)}$, we get

$$c^{(1)}\left(U_1 P_k^{(1)}\right) + \alpha_{m_k} c^{(1)}\left(Q_k P_k^{(1)}\right) + c\left(V_1 P_k\right) + \alpha'_{m_k-1} c\left(Q_k P_k\right) = 0.$$

But, since U_1 and V_1 have degree $n_k - 1$, then $c^{(1)}\left(U_1 P_k^{(1)}\right) = c\left(V_1 P_k\right) = 0$, and it follows

$$\alpha'_{m_k-1} c\left(Q_k P_k\right) = -\alpha_{m_k} c^{(1)}\left(Q_k P_k^{(1)}\right).$$

Writing down the orthogonality condition $c^{(1)}\left(\xi^{n_k-m_k+1}P_{k+1}^{(1)}\right)=0$ and using $c^{(1)}\left(U_2P_k^{(1)}\right)=c\left(V_2P_k\right)=0$, we obtain

$$\alpha_{m_k-1}c^{(1)}\left(Q_kP_k^{(1)}\right)+\alpha'_{m_k-2}c\left(Q_kP_k\right)+\alpha'_{m_k-1}c\left(\xi Q_kP_k\right)=-\alpha_{m_k}c^{(1)}\left(\xi Q_kP_k^{(1)}\right),$$

and so on until $c^{(1)}\left(\xi^{n_k-1}P_{k+1}^{(1)}\right)=0$, which gives

$$\begin{aligned}\alpha_1c^{(1)}\left(Q_kP_k^{(1)}\right)+\cdots+\alpha_{m_k-1}c^{(1)}\left(\xi^{m_k-2}Q_kP_k^{(1)}\right)+\alpha'_0c\left(Q_kP_k\right)+\cdots\\ +\alpha'_{m_k-1}c\left(\xi^{m_k-1}Q_kP_k\right)=-\alpha_{m_k}c^{(1)}\left(\xi^{m_k-1}Q_kP_k^{(1)}\right).\end{aligned}$$

Thus we have obtained m_k-1 relations for determining the unknown coefficients α_i and α'_i.

We shall now write the remaining orthogonality conditions of $P_{k+1}^{(1)}$ as

$$c^{(1)}\left(\xi^iQ_kP_{k+1}^{(1)}\right)=0\qquad\text{for }\ i=0,\ldots,m_k-1.$$

Thus we obtain

$$c^{(1)}\left(\xi^iq_kQ_kP_k^{(1)}\right)+c\left(\xi^{i+1}t_kQ_kP_k\right)=0$$

that is

$$\begin{aligned}\eta_0c^{(1)}\left(\xi^iQ_kP_k^{(1)}\right)+\cdots+\eta_{m_k-1}c^{(1)}\left(\xi^{i+m_k-1}Q_kP_k^{(1)}\right)+\\ \eta'_0c\left(\xi^{i+1}Q_kP_k\right)+\cdots+\eta'_{m_k-1}c\left(\xi^{i+m_k}Q_kP_k\right)=-c^{(1)}\left(\xi^{i+m_k}Q_kP_k^{(1)}\right).\end{aligned}$$

Replacing, in these relations, the η's and the η''s by their expressions above, we obtain m_k additional relations for the α's and the α''s, that is

$$\begin{aligned}c_i\left(\alpha_0a_{n_k}^{(k)}+\cdots+\alpha_{m_k}a_{n_k-m_k}^{(k)}\right)+\cdots+c_{i+m_k-1}\left(\alpha_{m_k-1}a_{n_k}^{(k)}+\alpha_{m_k}a_{n_k-1}\right)\\ +d_{i+1}\left(\alpha'_0a_{n_k}^{(k)}+\cdots+\alpha'_{m_k-1}a_{n_k-m_k+1}^{(k)}\right)+\cdots+d_{i+m_k}\left(\alpha'_{m_k-1}a_{n_k}^{(k)}\right)=\\ -c_{i+m_k},\end{aligned}$$

or

$$\begin{aligned}\alpha_0\left(c_ia_{n_k}^{(k)}\right)+\alpha_1\left(c_ia_{n_k-1}^{(k)}+c_{i+1}a_{n_k}^{(k)}\right)+\cdots+\alpha_{m_k-1}\left(c_ia_{n_k-m_k+1}^{(k)}+\cdots\right.\\ \left.+c_{i+m_k-1}a_{n_k}^{(k)}\right)+\alpha'_0\left(d_{i+1}a_{n_k}^{(k)}\right)+\alpha'_1\left(d_{i+1}a_{n_k-1}^{(k)}+d_{i+2}a_{n_k}^{(k)}\right)+\cdots\\ +\alpha'_{m_k-1}\left(d_{i+1}a_{n_k-m_k+1}^{(k)}+\cdots+d_{i+m_k}a_{n_k}^{(k)}\right)=\\ -\alpha_{m_k}\left(c_ia_{n_k-m_k}^{(k)}+\cdots+c_{i+m_k}a_{n_k}^{(k)}\right).\end{aligned}$$

Thus, we have the same number of unknowns and relations.

• Case $n_k - m_k < 0$
–We have again

$$q_k(\xi) = \eta_0 + \cdots + \eta_{m_k-1}\xi^{m_k-1} + \xi^{m_k}.$$

There exists $\alpha_0, \ldots, \alpha_{m_k-1}$ such that the η_i are given by

$$\begin{aligned}
&\eta_{m_k-1} = \alpha_{m_k-1}a^{(k)}_{n_k} + \alpha_{m_k}a^{(k)}_{n_k-1}\\
&\eta_{m_k-2} = \alpha_{m_k-2}a^{(k)}_{n_k} + \alpha_{m_k-1}a^{(k)}_{n_k-1} + \alpha_{m_k}a^{(k)}_{n_k-2}\\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\
&\eta_0 = \alpha_0 a_{n_k} + \alpha_1 a^{(k)}_{n_k-1} + \cdots + \alpha_{m_k-1}a^{(k)}_{n_k-m_k+1} + \alpha_{m_k}a^{(k)}_{n_k-m_k}.
\end{aligned}$$

Moreover, there exists a polynomial U_1 of degree $n_k - 1$ such that

$$q_k(\xi) = U_1(\xi) + \alpha_{n_k}Q_k(\xi) + \cdots + \alpha_{m_k}\xi^{m_k-n_k}Q_k(\xi).$$

Identifying the coefficients of ξ^{m_k} in both sides of this relation, we obtain again

$$1 = \alpha_{m_k}a^{(k)}_{n_k}.$$

–We shall take for t_k a polynomial of degree at most $n_k - 1$ (for $k = 0$, $n_0 = 0$ and there is no polynomial t_0)

$$t_k(\xi) = \eta'_0 + \cdots + \eta'_{n_k-1}\xi^{n_k-1}.$$

We have

$$\begin{aligned}
&\eta'_{n_k-1} = \alpha'_{n_k-1}a^{(1)}_{n_k}\\
&\cdots\cdots\cdots\cdots\cdots\cdots\\
&\eta'_0 = \alpha'_0 a^{(k)}_{n_k} + \alpha'_1 a^{(k)}_{n_k-1} + \cdots + \alpha'_{n_k-1}a^{(k)}_1.
\end{aligned}$$

Applying the orthogonality conditions we have

$$\begin{aligned}
&\alpha_{n_k}c^{(1)}\left(Q_kP^{(1)}_k\right) + \cdots + \alpha_{m_k}c^{(1)}\left(\xi^{m_k-n_k}Q_kP^{(1)}_k\right) + \alpha'_{n_k-1}c(Q_kP_k) = 0,\\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\\
&\alpha_1 c^{(1)}\left(Q_kP^{(1)}_k\right) + \cdots + \alpha_{m_k}c^{(1)}\left(\xi^{m_k-1}Q_kP^{(1)}_k\right) +\\
&\qquad \alpha'_0 c\left(Q_kP_k\right) + \cdots + \alpha'_{n_k-1}c\left(\xi^{n_k-1}Q_kP_k\right) = 0.
\end{aligned}$$

We have also

$$\begin{aligned}
&\eta_0 c^{(1)}\left(\xi^i Q_kP^{(1)}_k\right) + \cdots + \eta_{m_k-1}c^{(1)}\left(\xi^{i+m_k-1}Q_kP^{(1)}_k\right) +\\
&\quad c^{(1)}\left(\xi^{i+n_k}Q_kP^{(1)}_k\right) + \eta'_0 c\left(\xi^{i+1}Q_kP_k\right) + \cdots + \eta'_{n_k-1}c\left(\xi^{i+n_k}Q_kP_k\right) = 0
\end{aligned}$$

for $i = 0, \ldots, m_k - 1$.

Replacing, in these relations, the η's and the η''s by their expressions above, we obtain

$$c_i\left(\alpha_0 a_{n_k}^{(k)}+\cdots+\alpha_{m_k}a_{n_k-m_k}^{(k)}\right)+\cdots+c_{i+m_k-1}\left(\alpha_{m_k-1}a_{n_k}^{(k)}+\alpha_{m_k}a_{n_k-1}^{(k)}\right)$$
$$+\,d_{i+1}\left(\alpha'_0 a_{n_k}^{(k)}+\cdots+\alpha'_{n_k-1}a_1^{(k)}\right)+\cdots+d_{i+n_k}\left(\alpha'_{n_k-1}a_{n_k}^{(k)}\right)=-c_{i+m_k},$$

that is

$$\alpha_0\left(c_i a_{n_k}^{(k)}\right)+\alpha_1\left(c_i a_{n_k-1}^{(k)}+c_{i+1}a_{n_k}^{(k)}\right)+\cdots$$
$$+\,\alpha_{m_k-1}\left(c_i a_{n_k-m_k+1}^{(k)}+\cdots+c_{i+m_k-1}a_{n_k}^{(k)}\right)+\alpha'_0\left(d_{i+1}a_{n_k}^{(k)}\right)+$$
$$\alpha'_1\left(d_{i+1}a_{n_k-1}^{(k)}+d_{i+2}a_{n_k}^{(k)}\right)+\cdots+\alpha'_{n_k-1}\left(d_{i+1}a_1^{(k)}+\cdots+d_{i+n_k}a_{n_k}^{(k)}\right)=$$
$$-\,\alpha_{m_k}\left(c_i a_{n_k-m_k}^{(k)}+\cdots+c_{i+m_k-1}a_{n_k-1}^{(k)}\right)-c_{i+m_k},$$

or

$$\alpha_0\left(c_i a_{n_k}^{(k)}\right)+\alpha_1\left(c_i a_{n_k-1}^{(k)}+c_{i+1}a_{n_k}^{(k)}\right)+\cdots$$
$$+\,\alpha_{m_k-1}\left(c_i a_{n_k-m_k+1}^{(k)}+\cdots+c_{i+m_k-1}a_{n_k}^{(k)}\right)+\alpha'_0\left(d_{i+1}a_{n_k}^{(k)}\right)+$$
$$\alpha'_1\left(d_{i+1}a_{n_k-1}^{(k)}+d_{i+2}a_{n_k}^{(k)}\right)+\cdots+\alpha'_{n_k-1}\left(d_{i+1}a_1^{(k)}+\cdots+d_{i+n_k}a_{n_k}^{(k)}\right)=$$
$$-\,\alpha_{m_k}\left(c_i a_{n_k-m_k}^{(k)}+\cdots+c_{i+m_k}a_{n_k}^{(k)}\right).$$

In what precedes, we assumed that the polynomial Q_k has the exact degree n_k. A quite similar treatment can be made if Q_k has the degree $q_k \le n_k$. This will be the case, for example, in the line minimization described in section 4 for the Bi-CGSTAB, where $q_k = k$. In that case, writing

$$Q_k(\xi) = 1 + a_1^{(k)}\xi+\cdots+a_{q_k}^{(k)}\xi^{q_k}$$

we obtain, when $n_k - m_k + 1 \ge 0$,

$$\gamma_{m_k-1} = \beta_{m_k-1}a_{q_k}^{(k)}$$
$$\gamma_{m_k-i} = \beta_{m_k-i}a_{q_k}^{(k)}+\cdots+\beta_{m_k-1}a_{q_k-i+1}^{(k)} \quad \text{for } i=2,\ldots,m_k$$

with the convention that $a_i = 0$ for $i \le 0$ and

$$\gamma'_{m_k-2} = \beta'_{m_k-2}a_{q_k}^{(k)}$$
$$\gamma'_{m_k-i} = \beta'_{m_k-i}a_{q_k}^{(k)}+\cdots+\beta'_{m_k-1}a_{q_k-i+2}^{(k)} \quad \text{for } i=3,\ldots,m_k.$$

The β_i and the β'_i are the solution of the system

$$\beta_{m_k-1}c_{n_k-q_k}+\beta'_{m_k-2}d_{n_k-q_k}=0,$$
$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$$
$$\beta_1 c_{n_k-q_k}+\cdots+\beta_{m_k-1}c_{n_k-q_k+m_k-2}+\beta'_0 d_{n_k-q_k}+\cdots+$$
$$\beta'_{m_k-2}d_{n_k-q_k+m_k-2}=0,$$
$$\gamma_0 c_i+\cdots+\gamma_{m_k-1}c_{i+m_k-1}+\gamma'_0 d_{i+1}+\cdots+\gamma'_{m_k-2}d_{i+m_k-1}=d_i,$$
$$i=n_k-q_k,\ldots,n_k-q_k+m_k-1,$$

after replacing, in the last m_k relations, the γ_i and the $\gamma_i^{'}$ by their expressions given above.

The case $n_k - m_k + 1 < 0$ can be treated similarly and the recurrence relationship (3.13) as well.

However, it must be noticed that these relations involve the quantities

$$c(\xi^{n_k-q_k+i}Q_kP_k), \qquad c^{(1)}(\xi^{n_k-q_k+i}Q_kP_k^{(1)}), \qquad i=0,\ldots,2m_k-1,$$

which means that the vectors

$$A^{n_k-q_k+i}Q_k(A)P_k(A)r_0, \qquad A^{n_k-q_k+i+1}Q_k(A)P_k^{(1)}(A)r_0,$$

will have to be computed for $i=0,\ldots,2m_k-1$. Thus, if the difference n_k-q_k is large the procedure will be costly or will necessitate the use of A^T. In particular, if $q_k=k$, as in the line minimization described in section 4, and if many jumps or long ones occur, the discrepancy n_k-k will be large and the procedure expensive.

3.3 Summary of the algorithm.

Let us now briefly summarize the computations to be performed in our algorithm. At the step k, after having decided the value of m_k (see the next Subsection), we have to

- Compute the β's and the β''s,
- Compute the γ's (coefficients of w_k) and the γ''s (coefficients of v_k),
- Compute r_{k+1} by (3.8) and control its value for deciding to continue or not,
- If the solution has not been obtained
  - Compute s_{k+1} by (3.10),
  - Compute the α's and the α''s,
  - Compute the η's (coefficients of q_k) and the η''s (coefficients of t_k),
  - Compute z_{k+1} by (3.9) and p_{k+1} by (3.11).

3.4 Tests for near-breakdown.

Let us first consider the case where some of the Hankel determinants defined in section 3.1 vanish. It was proved by Draux [21] that

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

and that

$$c^{(1)}(\xi^{n_k+m_k-1}P_k^{(1)})\neq 0.$$

These relations define the value of m_k in the case of breakdown. Thus, it seems that the near–breakdown could be defined by replacing 0 by some threshold ε that is

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

and

$$\left|c^{(1)}\left(\xi^{n_k+m_k-1} P_k^{(1)}\right)\right| > \varepsilon.$$

However, as already observed with the CGS [9], such a test is not satisfactory and it must be replaced by a more suitable one. Following [9], let us set

$$\sigma_{k+1}^{(1)} = c^{(1)}(Q_k P_{k+1}).$$

It is easy to see that

$$\sigma_{k+1}^{(1)} = a_{n_k}^{(k)} c(\xi^{n_{k+1}} P_{k+1}).$$

Thus $\sigma_{k+1}^{(1)}$ is zero (or close to it) in three different cases

1. when the solution is about to be obtained, since $\sigma_{k+1}^{(1)} = 0$ if $P_{k+1}(A)r_0 = 0$; that is if $x_{k+1} = x$,
2. when $a_{n_k}^{(k)} = 0$,
3. when $c(\xi^{n_{k+1}} P_{k+1}) = 0$.

From the last two cases, we see that $\sigma_{k+1}^{(1)} \neq 0$ if and only if $a_{n_k}^{(k)} \neq 0$ and $c(\xi^{n_{k+1}} P_{k+1}) \neq 0$, which are the two conditions for not having a breakdown (or a near–breakdown) at the next iteration of the algorithm. Thus, after testing if the solution has been obtained, a jump has to begin if $\left|\sigma_{k+1}^{(1)}\right| \leq \varepsilon$. So the value of m_k is set to 2 and the system giving the coefficients $\beta_i, \beta_i', \alpha_i$ and α_i' is solved. If this system is singular (pivot= 0) or nearly singular ($|\text{pivot}| \leq \varepsilon_1$) then m_k is changed to $m_k + 1$ and the procedure is repeated until a non–nearly singular system has been found. But, it is also necessary to check whether or not a near–breakdown will occur at the beginning of the next step in order to avoid it (otherwise it will be too late). For that, we set

$$\sigma_{k+1}^{(m_k)} = c(\xi^{m_k} Q_k P_{k+1}).$$

We have

$$\sigma_{k+1}^{(m_k)} = a_{n_k}^{(k)} c(\xi^{n_{k+1}} P_{k+1}).$$

Thus, if the solution has not been obtained, the length m_k of the jump has still to be increased in the current step, the system giving the coefficients of the polynomials has again to be solved and the whole procedure has to be repeated until all the conditions be satisfied. It must be remarked that $\sigma_{k+1}^{(m_k)}$ is computed by

$$\sigma_{k+1}^{(m_k)} = (y, A^{m_k} s_{k+1}),$$

with $s_{k+1} = Q_k(A)P_{k+1}(A)r_0$.

The other tests mentioned in [9] also have to be performed.

Of course, the tests we used are only based on heuristics and there is no guarantee that they will work well in all situations nor that the systems giving the coefficients of the polynomials involved in the recurrence relations will be well-conditioned (although we test the pivots in the Gaussian elimination procedure used for their solution). It may be possible that better tests could be based on the results obtained by Beckermann [3].

4 A look-ahead Bi-CGSTAB.

In the Bi-CGSTAB introduced by van der Vorst [42], the polynomial F_k is identically zero and, when $m_k = 1$, the polynomial E_k is written as

$$E_k(\xi) = 1 - b\xi,$$

and the parameter b is chosen so that (r_{k+1}, r_{k+1}) is minimized that is

$$b = (s_{k+1}, As_{k+1})/(As_{k+1}, As_{k+1}), \qquad s_{k+1} = Q_k(A)P_{k+1}(A)r_0.$$

Let us now explain how to compute the coefficients of E_k when $m_k \geq 1$. There are three different possibilities. The first one consists in taking for E_k a polynomial of degree one, even if $m_k > 1$, that is defining it as van der Vorst [42]. In the two other cases, E_k will be a polynomial of the degree m_k at most.

1. *Line minimization*

$$E_k(\xi) = 1 - b\xi$$

For $x_0 = 0$ and $y = (0, 0, 0, -1, 1, \ldots, -(-1)^{n-1}, 0)^{\mathrm{T}}$, a breakdown occurs at the first iteration when n is even since $(y, r_0) = (y, Ar_0) = 0$.

For $n = 400, x_0 = 0, \varepsilon = 10^{-30}$ we have a jump of length 2 at the first iteration to avoid the breakdown and we obtain $n_{54} = 55$ and $\|r_{54}\| = 0.66 \times 10^{-11}$ which coincides with the actual residual (that is the residual computed from x_k by $r_k = b - Ax_k$). When $\varepsilon = 10^{-20}$ and $\varepsilon_2 = 10^{-14}$, we have again the jump of length 2 at the first iteration but we also have several jumps (from $n_{34} = 35$ to $n_{35} = 37$, from $n_{35} = 37$ to $n_{36} = 40$, from $n_{36} = 40$ to $n_{37} = 42$ and, finally, a jump from $n_{37} = 42$ to $n_{38} = 48$) and we obtain $\|r_{38}\| = 0.42 \times 10^{-14}$ and an actual residual of 0.62×10^{-14}.

The behavior of the algorithm for $n = 400$ and these two different values of ε s shown in Fig. 1.

$$e_1(A^i s_{k+1}, As_{k+1}) + \cdots + e_{m_k}(A^i s_{k+1}, A^{m_k} s_{k+1}) = -(A^i s_{k+1}, s_{k+1})$$

for $i = 1, \ldots, m_k$, where s_{k+1} is defined as above.

3. *Step-by-step minimization*

We set

$$E_k(\xi) = (1 - b_{m_k}\xi) \cdot \ldots \cdot (1 - b_1\xi) = 1 + e_1\xi + \cdots + e_{m_k}\xi^{m_k}.$$

We also set

$$\begin{aligned} r_{k+1}^{(0)} &= s_{k+1}, \\ r_{k+1}^{(i)} &= (1 - b_i A) r_{k+1}^{(i-1)}, \quad \text{for } i = 1, \ldots, m_k. \end{aligned}$$

We choose b_i which minimizes $(r_{k+1}^{(i)}, r_{k+1}^{(i)})$, that is

$$b_i = (r_{k+1}^{(i-1)}, A r_{k+1}^{(i-1)}) / (A r_{k+1}^{(i-1)}, A r_{k+1}^{(i-1)}),$$

and we obtain $r_{k+1} = r_{k+1}^{(m_k)}$.

A crucial point which must be noticed is that the vector s_{k+1} depends on the value of m_k. Thus, when changing m_k, the new vector s_{k+1} has to be computed before beginning the minimization process.

5 Numerical results.

In this section we shall present some numerical results obtained with our implementation of the look-ahead Bi–CGSTAB of section 4. The program, written in Fortran 77, uses the theory explained in the previous sections and the breakdowns and near-breakdowns are detected using the tests described in the subsection 3.3 and those already reported in [9]. The results were obtained on a SUN SparcStation IPC. Our results were compared with those obtained by implementing the usual Bi–CGSTAB algorithm proposed by van der Vorst in [42].

As in [9], we use three different thresholds

- ε for testing the quantities involved in the jumps,
- ε_1 for testing the pivots and the determinant in Gaussian elimination and jumping,
- ε_2 for testing if $\|r_k\| \leq \varepsilon_2$ and stopping.

The numerical results show that, in the case of the look-ahead Bi–CGSTAB, the choice of ε_1 is not an important item and that a very small value for this threshold (i.e. $\varepsilon_1 = 10^{-50}$) can be chosen without any loss. Thus we shall not report this value. In all the examples, r_k denotes the residual computed recursively by the algorithm.

Our numerical experiments seem to indicate that the global minimization is better than the step–by–step minimization. Thus, except in the example 1, we shall only give the results obtained by the global minimization.

Our heuristics for the jumps are sensitive to the choice of ε and of ε_2 because the first value is related to the beginning of the jump and to its length and too large a value can produce too long a jump. Moreover, when $\left|\sigma_{k+1}^{(m_k)}\right| \leq \varepsilon$, the condition $\|r_{k+1}\| \leq \varepsilon_2$ is tested and thus, too small a value of ε_2 could increase the length of the jump up to the maximum allowed dimension.

The sensitivity of our heuristics is more important than in the algorithm proposed in [9] where the software CADNA [19, 20] insures that the tests for detecting numerical instabilities produce a quite stable program.

EXAMPLE 5.1. The first example was proposed by Joubert [33]. He considered the following matrix of dimension 4 that leads to a breakdown in the biconjugate gradient method, for almost every starting vector x_0

$$A = \begin{pmatrix} 1 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 3 & -1 \\ 0 & 0 & 1 & 3 \end{pmatrix}.$$

For $b = (0, 2, 2, 4)$, the system has the solution $x = (1, 1, 1, 1)$. If we take $y = (1, 1, 1, 1)$, the Bi–CGSTAB breaks down at the iteration 2. Our look–ahead version, with $\varepsilon = 10^{-20}$, jumps from $n_0 = 0$ to $n_1 = 2$ due to $\sigma_1^{(1)} = 0$ and when $n_3 = 4$, we obtain, with the step–by–step minimization, $\|r_3\| = 0.17 \times 10^{-13}$ and, with the global minimization, $\|r_3\| = 0.33 \times 10^{-14}$. If we take $y = r_0$, the Bi–CGSTAB, at the iteration 4, gives $\|r_4\| = 0.47$ and the solution is obtained only at the iteration 7 with $\|r_7\| = 0.13 \times 10^{-15}$. With $\varepsilon = 10^{-6}$, our program has a jump from $n_1 = 1$ to $n_2 = 3$ due to $\left|\sigma_2^{(1)}\right| \leq \varepsilon$ and, for $n_3 = 4$, we obtain, with the step–by–step minimization, $\|r_3\| = 0.7 \times 10^{-14}$ and, with the global minimization, $\|r_3\| = 0.1 \times 10^{-14}$.

EXAMPLE 5.2. Let us now consider the system (Gutknecht [29])

$$\begin{pmatrix} 2 & 1 & & & & & & \\ 0 & 2 & 1 & & & & & \\ 1 & 0 & 2 & \ddots & & & & \\ & 1 & \ddots & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & \ddots & & \\ & & & 1 & 0 & 2 & 1 & \\ & & & & 1 & 0 & 2 & \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ \vdots \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 3 \\ 3 \\ 4 \\ \vdots \\ \vdots \\ 4 \\ 3 \end{pmatrix}.$$

For $x_0 = 0$ and $y = (0, 0, 0, -1, 1, \ldots, -(-1)^{n-1}, 0)^{\mathrm{T}}$, a breakdown occurs at the first iteration when n is even since $(y, r_0) = (y, Ar_0) = 0$.

For $n = 400, x_0 = 0, \varepsilon = 10^{-30}$ we have a jump of length 2 at the first iteration to avoid the breakdown and we obtain $n_{54} = 55$ and $\|r_{54}\| = 0.66 \times 10^{-11}$ which coincides with the actual residual (that is the residual computed from x_k by $r_k = b - Ax_k$). When $\varepsilon = 10^{-20}$ and $\varepsilon_2 = 10^{-14}$, we have again the jump of length 2 at the first iteration but we also have several jumps (from $n_{34} = 35$ to $n_{35} = 37$, from $n_{35} = 37$ to $n_{36} = 40$, from $n_{36} = 40$ to $n_{37} = 42$ and, finally, a jump from $n_{37} = 42$ to $n_{38} = 48$) and we obtain $\|r_{38}\| = 0.42 \times 10^{-14}$ and an actual residual of 0.62×10^{-14}.

The behavior of the algorithm for $n = 400$ and these two different values of ε is shown in Fig. 1.

Fig. 1

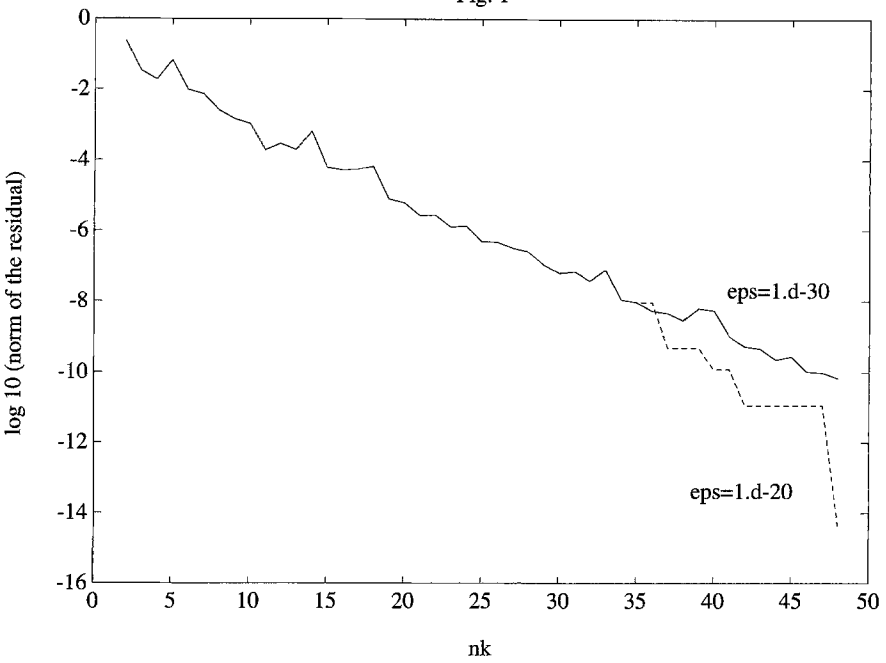

Figure 5.1: Behavior of algorithm for system in Example 5.2, $n = 400$ and $\varepsilon = 10^{-20}, 10^{-30}$.

EXAMPLE 5.3. Let us now consider the following system (Brown [17])

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

When $a = 0$, a breakdown occurs every odd step of the Lanczos method. If we take $n = 400, a = 1.1, x_0 = 0$ and $y = r_0$, the Bi–CGSTAB converges to the solution quite regularly and at the iteration 61, we have $\|r_{61}\| = 0.84 \times 10^{-14}$. With $\varepsilon = 10^{-14}$, our program detects some numerical instabilities in the computation and it jumps several time (the first one from $n_{19} = 19$ to $n_{20} = 21$). When $n_{26} = 40$, $\|r_{26}\| = 0.37 \times 10^{-13}$ which coincides with the actual residual. These results are presented in Fig. 2.

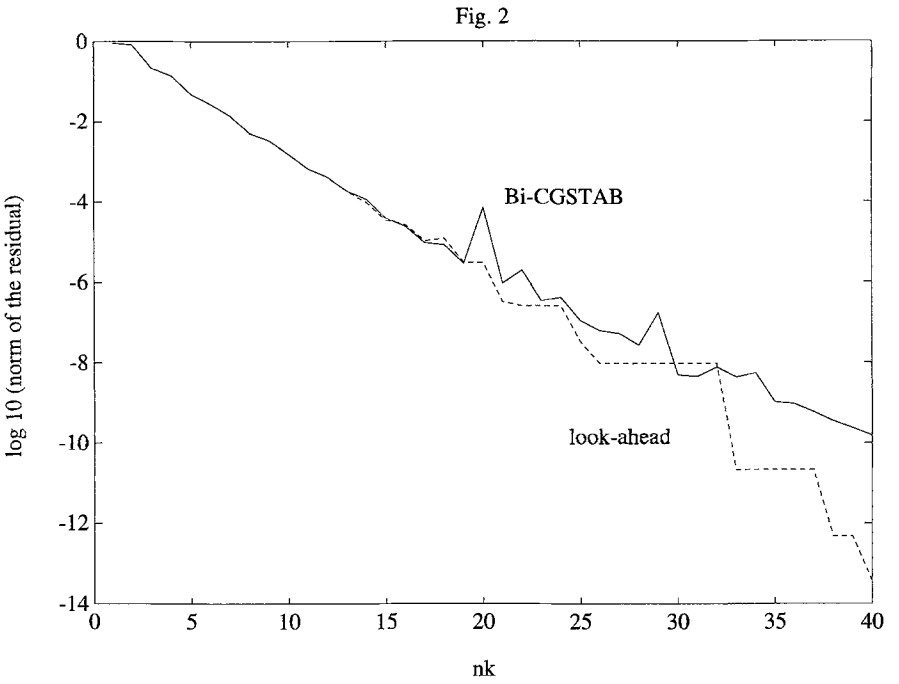

Figure 5.2: Behavior of algorithm for system in Example 5.3, $n = 400$ and $\varepsilon = 10^{-14}$.

EXAMPLE 5.4. Let us consider the $n \times n$ matrix

$$A = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ a_1 & 1 & 1 & \cdots & 1 & 1 \\ a_1 & a_2 & 1 & \cdots & 1 & 1 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ a_1 & a_2 & a_3 & \cdots & a_{n-1} & 1 \end{pmatrix}.$$

Its determinant is equal to $(1 - a_1)(1 - a_2) \cdots (1 - a_{n-1})$. Thus the matrix A is singular if and only if $a_i = 1$ for some i. With $a_i = 1 + i\delta, \delta = 10^{-2}, x_0 = 0$, $y = r_0$ and $n = 100$, the Bi-CGSTAB does not converge and, at the iteration 100, we get $\|r_{100}\| = 0.4 \times 10^{-2}$. For $\varepsilon = 10^{-6}$, our look-ahead program exhibits several jumps (from $n_7 = 7$ and with small values of m_k) and, when $n_{46} = 100$, we obtain $\|r_{46}\| = 0.6 \times 10^{-8}$ which coincides with the actual residual. The results are shown in Fig. 3.

Fig. 3

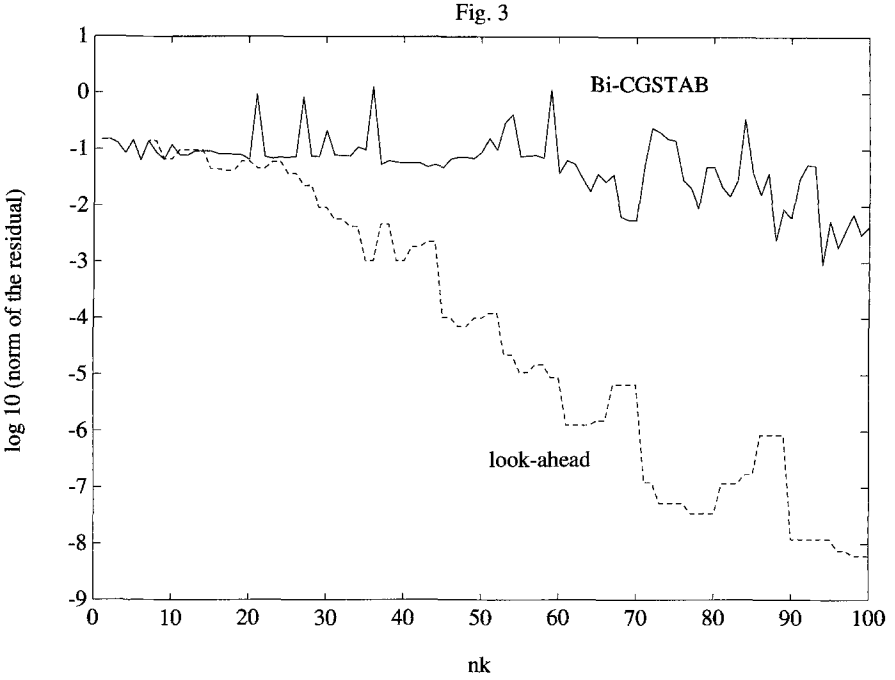

Figure 5.3: Behavior of algorithm for system in Example 5.4, $n = 100$ and $\varepsilon = 10^{-6}$.

6 Conclusions.

In this paper we

- discussed a class of methods based on Lanczos polynomials. This class, called BiCGM, contains the CGS and the Bi–CGSTAB as particular cases,
- gave a recursive algorithm for their implementation in the case where the auxiliary polynomials Q_k appearing in these methods satisfy a three–term recurrence relationship,
- showed how to cure the breakdowns and near–breakdowns arising from the polynomials P_k and $P_k^{(1)}$ of the Lanczos method. The case of the Bi–CGSTAB algorithm was treated in details.

A lot of work remains to be done on these methods. We have

- to investigate the theory of the methods of the BiCGM class and, in particular, their convergence properties and the choice of the polynomials Q_k. More numerical experiments with the methods of the BiCGM class proposed in section 2 have to be conducted,
- to study the heuristics for deciding when and how far to jump for avoiding near–breakdowns. Our numerical results are quite sensitive to the choice of the

various ε appearing in the algorithms which leads to thinking that our strategy for the jumps could be improved,

• to study, on a theoretical basis, the three minimization strategies we proposed. In particular, our line minimization assumes that the polynomials Q_k are of the degree n_k and the formulae we gave are no longer valid for this minimization. However, as we saw above, it is quite easy to derive the recurrence relationships to be used when it is assumed that Q_k has a degree $q_k \leq n_k$. But, when the difference $n_k - q_k$ is large, these formulae need the computation of many matrix–by–vector products or the use of A^T and the method loses one of its main advantages. This is, in particular, the case when $q_k = k$ since, then, the difference $n_k - k$ depends on the sum of the lengths of all the previous jumps,

• to study how to deal with the breakdowns and near–breakdowns arising from the polynomials Q_k since our treatment is only able to cure those coming out from the polynomials P_k and $P_k^{(1)}$. In particular, when $s_k = 0$, our algorithm has to be stopped since the systems giving the coefficients of the polynomials E_k are all singular.

These questions will be discussed in subsequent publications.

Acknowledgements.

We would like to thank Hassane Sadok for several interesting discussions. This paper was completed while the second author was an invited professor at the Laboratoire MASI of the Université Pierre et Marie Curie in Paris. She would like to thank Professor René Alt for his kind invitation and for providing her all the working facilities. Finally, we also acknowledge the help of the referee whose comments helped us to clarify some points and to improve the presentation of the paper.

REFERENCES

1. R. E. Bank and T. F. Chan, *An analysis of the composite step biconjugate gradient method*, Numer. Math., 66 (1993), pp. 295–319.
2. R. E. Bank and T. F. Chan, *A composite step bi–conjugate gradient algorithm for solving nonsymmetric systems*, Numerical Algor., 7 (1994), pp. 1–16.
3. B. Beckermann, *The stable computation of formal orthogonal polynomials*, Ann. Numer. Math., to appear.
4. C. Brezinski, *Généralisations de la transformation de Shanks, de la table de Padé et de l'ε–algorithme*, Calcolo, 12 (1975), pp. 317–360.
5. C. Brezinski, *Padé–Type Approximation and General Orthogonal Polynomials*, ISNM vol. 50, Birkhäuser, Basel, 1980.
6. C. Brezinski, *CGM: a whole class of Lanczos–type solvers for linear systems*, Publication ANO–253, Université des Sciences et Technologies de Lille, November 1991.
7. C. Brezinski and A. C. Matos, *Least–squares orthogonal polynomials*, J. Comput. Appl. Math., 46 (1993), pp. 229–239.

8. C. Brezinski and M. Redivo–Zaglia, *Hybrid procedures for solving linear systems*, Numer. Math., 67 (1994), pp. 1–19.

9. C. Brezinski and M. Redivo–Zaglia, *Treatment of near–breakdown in the CGS algorithm*, Numerical Algor., 7 (1994).

10. C. Brezinski, M. Redivo–Zaglia and H. Sadok, *Avoiding breakdown and near–breakdown in Lanczos type algorithms*, Numerical Algor., 1 (1991), pp. 207–221.

11. C. Brezinski, M. Redivo–Zaglia and H. Sadok, *Addendum to "Avoiding breakdown and near–breakdown in Lanczos type algorithms"*, Numerical Algor., 2 (1992), pp. 133–136.

12. C. Brezinski, M. Redivo–Zaglia and H. Sadok, *A breakdown–free Lanczos type algorithm for solving linear systems*, Numer. Math., 63 (1992), pp. 29–38.

13. C. Brezinski, M. Redivo–Zaglia and H. Sadok, *Breakdowns in the implementation of the Lánczos method for solving linear systems*, Comp. & Math. with Applics., to appear.

14. C. Brezinski and H. Sadok, *Avoiding breakdown in the CGS algorithm*, Numerical Algor., 1 (1991), pp. 199–206.

15. C. Brezinski and H. Sadok, *Some vector sequence transformations with applications to systems of equations*, Numerical Algor., 3 (1992), pp. 75–80.

16. C. Brezinski and H. Sadok, *Lanczos type methods for solving systems of linear equations*, Appl. Numer. Math., 11 (1993), pp. 443–473.

17. P. N. Brown, *A theoretical comparison of the Arnoldi and GMRES algorithms*, SIAM J. Sci. Stat. Comput., 12 (1991), pp. 58–78.

18. T. F. Chan and T. Szeto, *A composite step conjugate gradients squared algorithm for solving nonsymmetric linear systems*, Numerical Algor., 7 (1994), pp. 17–32.

19. J. M. Chesneaux, *Stochastic arithmetic properties, in Computational and Applied Mathematics*, C. Brezinski and U. Kulisch eds., North–Holland, Amsterdam, 1992, pp. 81–91.

20. J. M. Chesneaux and J. Vignes, *L'algorithme de Gauss en arithmétique stochastique*, C.R. Acad. Sci. Paris, II, 316 (1993), pp. 171–176.

21. A. Draux, *Polynômes Orthogonaux Formels. Applications*, LNM 974, Springer Verlag, Berlin, 1983.

22. H. C. Elman, *Iterative Methods for Large, Sparse, Nonsymmetric Systems of Linear Equations*, Ph.D. Thesis, Yale University, 1982.

23. R. W. Freund, M. H. Gutknecht and N. M. Nachtigal, *An implementation of the look–ahead Lanczos algorithm for non–Hermitian matrices*, SIAM J. Sci. Stat. Comput., 14 (1993), pp. 137–158.

24. N. Gastinel, *Analyse Numérique Linéaire*, Hermann, Paris, 1966.

25. B. Germain–Bonne, *Estimation de la Limite de Suites et Formalisation de Procédés d'Accélération de Convergence*, Thèse d'Etat, Université de Lille I, 1978.

26. G. H. Golub and D. P. O'Leary, *Some history of the conjugate and Lanczos algorithms*, SIAM Rev., 31 (1989), pp. 50–102.

27. M. H. Gutknecht, *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, 1990.

28. M. H. Gutknecht, *A completed theory of the unsymmetric Lanczos process and related algorithms, part I*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 594–639.

29. M. H. Gutknecht, *Variants of BICGSTAB for matrices with complex spectrum*, IPS Research Report 91–14, August 1991, Revised June 1992, to appear in SIAM J. Sci. Comput.

30. Cs. J. Hegedüs, *Generating conjugate directions for arbitrary matrices by matrix equations*, I, II, Computers Math. Applic., 21 (1991), pp. 71–85, 87–94.

31. M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. Natl. Bur. Stand., 49 (1952), pp. 409–436.

32. K. C. Jea and D. M. Young, *On the simplification of generalized conjugate-gradient methods for nonsymmetrizable linear systems*, Linear Algebra Appl., 52/53 (1983), pp. 399–417.

33. W. Joubert, *Generalized Conjugate Gradient and Lanczos Methods for the Solution of Nonsymmetric Systems of Linear Equations*, Ph.D. Thesis, University of Texas at Austin, Austin, 1990.

34. M. Khelifi, *Lanczos maximal algorithm for unsymmetric eigenvalue problems*, Appl. Numer. Math., 7 (1991), pp. 179–193.

35. C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Res. Natl. Bur. Stand., 45 (1950), pp. 255–282.

36. C. Lanczos, *Solution of systems of linear equations by minimized iterations*, J. Res. Natl. Bur. Stand., 49 (1952), pp. 33–53.

37. W. R. Mann, *Mean value methods in iteration*, Proc. Am. Math. Soc., 4 (1953), pp. 506–510.

38. B. N. Parlett, D.R. Taylor, Z.A. Liu, *A look-ahead Lanczos algorithm for unsymmetric matrices*, Math. Comput., 44 (1985), pp. 105–124.

39. M. Prévost, *Stieltjes- and Geronimus-type polynomials*, J. Comput. Appl. Math., 21 (1988), pp. 133–144.

40. Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, Manchester University Press, Manchester, 1992.

41. P. Sonneveld, *CGS: a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 10 (1989), pp. 36–52.

42. H. A. Van der Vorst, *Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 631–644.

43. D. M. Young and K. C. Jea, *Generalized conjugate-gradient acceleration of nonsymmetrizable iterative methods*, Linear Algebra Appl., 34 (1980), pp. 159–194.