Pergamon

Computers Math. Applic. Vol. 33, No. 1/2, pp. 31–44, 1997

0898-1221/97 $17.00 + 0.00

PII: S0898-1221(96)00217-9

Breakdowns in the Implementation of the Lánczos Method for Solving Linear Systems

C. BREZINSKI
Laboratoire d'Analyse Numérique et d'Optimisation, UFR IEEA - M3
Université des Sciences et Technologies de Lille, F-59655 Villeneuve d'Ascq Cedex, France
brezinsk@omega.univ-lille1.fr

M. REDIVO-ZAGLIA
Dipartimento di Elettronica e Informatica, Università degli Studi di Padova
via Gradenigo 6/a, I-35131 Padova, Italy
michela@coco.dei.unipd.it

H. SADOK
Laboratoire de Mathématiques Appliqueés
Université du Littoral, Centre Universitaire de la Mi-Voix
Bât. H Poincaré, 50 rue F. Buisson, BP 699, 62228 Calais Cedex, France
sadok@lma.univ-littoral.fr

Abstract—The Lánczos method for solving systems of linear equations is based on formal orthogonal polynomials. Its implementation is realized via some recurrence relationships between polynomials of a family of orthogonal polynomials or between those of two adjacent families of orthogonal polynomials. A division by zero can occur in such recurrence relations, thus causing a breakdown in the algorithm which has to be stopped. In this paper, two types of breakdowns are discussed. The true breakdowns which are due to the nonexistence of some polynomials and the ghost breakdowns which are due to the recurrence relationship used. Among all the recurrence relationships which can be used and all the algorithms for implementing the Lánczos method which came out from them, the only reliable algorithm is Lánczos/Orthodir which can only suffer from true breakdowns. It is shown how to avoid true breakdowns in this algorithm. Other algorithms are also discussed and the case of near-breakdown is treated. The same treatment applies to other methods related to Lánczos'.

Keywords—Linear equations, Lánczos method, Orthogonal polynomials.

1. INTRODUCTION

In 1950, C. Lánczos [1] proposed a biorthogonalization procedure for transforming any matrix into a similar tridiagonal one. Then, the characteristic polynomial of the tridiagonal matrix can be computed by a three-term recurrence relationship and the eigenvalues of the initial matrix can be obtained as the zeros of the characteristic polynomial.

In an interview given in 1974 [2], only some time before he died, Lánczos was asked:

> What would you say has been the most important, the most fundamental and essential aspect of your sixty years of work?

And he answered,

> I believe my most important contribution was in the fields of mathematics, to be precise, in numerical analysis—my discovery of a method now known as the Lánczos method. It is very little used today, because there are now a number of other methods, but it was particularly interesting in that the analysis of the matrix could be carried out, that is, all the eigenvectors could be obtained by a simple procedure.

Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX

Of course, since the computation of the eigenelements of a matrix and the solution of a system of linear equations are equivalent problems, Lánczos [3] soon proposed a procedure for the second problem. When the matrix of the system is symmetric and positive definite, the Lánczos procedure is equivalent to the conjugate gradients algorithm obtained independently by Hestenes and Stiefel [4] around the same period. Extensions to the nonsymmetric case were given in [5], but the method only became widely known in 1975 with the biconjugate gradient algorithm of Fletcher [6].

An enormous literature on the Lánczos method exists and it is not our purpose here either to give a list of references or to describe its connections with other questions. A quite complete account of the history of the subject and an annotated bibliography can be found in [7]. See also [8] for more details.

Let us now describe the Lánczos method. We consider the system of linear equations in $\mathbf{C}^n$, $Ax = b$. Let y and x_0 be two arbitrary nonzero vectors and let $K_k(A,u) = \mathrm{span}(u, Au, \dots, A^{k-1}u)$. The Lánczos method consists of constructing the sequence of vectors (x_k) defined by

$$x_k - x_0 \in K_k(A, r_0), \tag{1}$$

$$r_k = b - Ax_k \perp K_k(A^*, y), \tag{2}$$

where A^* denotes the conjugate transpose of the matrix A.

From (1), we have

$$x_k - x_0 = -\alpha_1 r_0 - \cdots - \alpha_k A^{k-1} r_0,$$

that is, multiplying by A and adding and subtracting b

$$r_k = r_0 + \alpha_1 A r_0 + \cdots + \alpha_k A^k r_0 \tag{3}$$

$$= P_k(A) r_0, \tag{4}$$

with

$$P_k(\xi) = 1 + \alpha_1 \xi + \cdots + \alpha_k \xi^k.$$

The orthogonality condition (2) is equivalent to

$$\left(A^{*^i} y, r_k\right) = 0, \qquad \text{for } i = 0, \dots, k-1.$$

Setting

$$c_i = \left(A^{*^i} y, r_0\right) = \left(y, A^i r_0\right), \qquad i = 0, 1, \dots,$$

equation (2) is also equivalent to

$$c_i + \alpha_1 c_{i+1} + \cdots + \alpha_k c_{i+k} = 0, \qquad \text{for } i = 0, \dots, k-1. \tag{5}$$

If we define the linear functional c on the space of complex polynomials by

$$c\left(\xi^i\right) = c_i, \qquad i = 0, 1, \dots,$$

then the preceding relations (5) can be written as

$$c\left(\xi^i P_k(\xi)\right) = 0, \qquad \text{for } i = 0, \dots, k-1.$$

These conditions show that P_k is the polynomial of degree k at most belonging to the family of formal orthogonal polynomials with respect to the linear functional c. Such polynomials are defined apart from a multiplying factor chosen, in our case, so that the normalization condition $P_k(0) = 1$ holds. Formal orthogonal polynomials satisfy all the algebraic properties of the usual orthogonal polynomials (which correspond to the case where the linear functional c is given as the integral on the real line of a positive measure) except some of the properties about their zeros; see [9].

Since the constant term of P_k is equal to 1, it can be written as

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

and it follows that

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

which shows that x_k can be computed from $r_k = b - Ax_k$ without using A^{-1}. It is well known that the Lánczos method terminates in a finite number of steps not greater than the dimension of the system to be solved, that is $\exists k \leq n$ such that $r_k = 0$ and $x_k = x = A^{-1}b$.

In practice, the Lánczos method is implemented by computing recursively the residual vectors r_k, that is the polynomials P_k. We shall now give some indications about this computation. A quite complete exposition can be found in [10].

2. ORTHOGONAL POLYNOMIALS

The orthogonal polynomials P_k defined in the previous section are given by the determinantal formula

$$P_k(\xi) = \frac{\begin{vmatrix} 1 & \cdots & \xi^k \\ c_0 & \cdots & c_k \\ \vdots & & \vdots \\ c_{k-1} & \cdots & c_{2k-1} \end{vmatrix}}{\begin{vmatrix} c_1 & \cdots & c_k \\ \vdots & & \vdots \\ c_k & \cdots & c_{2k-1} \end{vmatrix}}. \tag{6}$$

Obviously, P_k exists if and only if the Hankel determinant

$$H_k^{(1)} = \begin{vmatrix} c_1 & \cdots & c_k \\ \vdots & & \vdots \\ c_k & \cdots & c_{2k-1} \end{vmatrix}$$

is different from zero. Thus, P_{k+1} exists if and only if $H_{k+1}^{(1)} \neq 0$.

It is well known that a family of orthogonal polynomials satisfies a three-term recurrence relationship. Thus, the easiest procedure for computing the polynomials P_k is to use this relation which is

$$P_{k+1}(\xi) = \left(A_{k+1}\xi + B_{k+1}\right) P_k(\xi) - C_{k+1} P_{k-1}(\xi), \tag{7}$$

for $k = 0, 1, \ldots,$ with $P_{-1}(\xi) = 0$ and $P_0(\xi) = 1$.

Writing the orthogonality conditions, we obtain

$$A_{k+1} c\left(\xi^k P_k(\xi)\right) - C_{k+1} c\left(\xi^{k-1} P_{k-1}(\xi)\right) = 0, \tag{8}$$

$$A_{k+1} c\left(\xi^{k+1} P_k(\xi)\right) + B_{k+1} c\left(\xi^k P_k(\xi)\right) - C_{k+1} c\left(\xi^k P_{k-1}(\xi)\right) = 0. \tag{9}$$

The normalization condition $P_k(0) = 1$ provides a third equation which is

$$B_{k+1} - C_{k+1} = 1.$$

Thus, solving this system of three equations gives the three unknown coefficients A_{k+1}, B_{k+1}, and C_{k+1} of the recurrence relationship. The determinant d_k of this system is

$$d_k = -c\left(\xi^k P_k(\xi)\right) \left[c\left(\xi^k P_k(\xi)\right) - c\left(\xi^k P_{k-1}(\xi)\right)\right] - c\left(\xi^{k-1} P_{k-1}(\xi)\right) c\left(\xi^{k+1} P_k(\xi)\right).$$

Thus, if $d_k = 0$, a *breakdown* will occur in the recurrence relationship due to a division by zero and the algorithm will have to be stopped.

We see, from the above expression, that d_k can be zero even if $H_{k+1}^{(1)} \neq 0$, that is, even if P_{k+1} exists. So, such a breakdown is not due to the nonexistence of an orthogonal polynomial of the family but to the recurrence relationship we are trying to use. A breakdown of this type is called a *ghost breakdown* [11]. The corresponding algorithm for implementing the Lánczos method, based on this recurrence relationship, will also suffer from a breakdown, called a *Lánczos breakdown* [12]. So, this algorithm, known as *Lánczos/Orthores* [13] or *BIORES* [14] is not reliable.

Let us now define the linear functional $c^{(1)}$ on the space of complex polynomials by

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

and let $\{P_k^{(1)}\}$ be the family of orthogonal polynomials with respect to $c^{(1)}$. These polynomials are taken to be monic. They are given by the determinantal formula

$$P_k^{(1)}(\xi) = \frac{\begin{vmatrix} c_1 & \cdots & c_{k+1} \\ \vdots & & \vdots \\ c_k & \cdots & c_{2k} \\ 1 & \cdots & \xi^k \end{vmatrix}}{\begin{vmatrix} c_1 & \cdots & c_k \\ \vdots & & \vdots \\ c_k & \cdots & c_{2k-1} \end{vmatrix}}. \tag{10}$$

Thus, $P_k^{(1)}$ exists if and only if $H_k^{(1)} \neq 0$, which is also the condition for the existence of P_k.

There exist many recurrence relations between the two adjacent families of polynomials $\{P_k\}$ and $\{P_k^{(1)}\}$. Each of them gives rise to a different algorithm for implementing the Lánczos method. They have been reviewed in [10] and studied in details in [15]. They are all subject to possible ghost breakdowns except two of them that will now be considered.

It can be proved that it holds

$$P_{k+1}(\xi) = P_k(\xi) - \lambda_k \, \xi \, P_k^{(1)}(\xi), \tag{11}$$

with $P_0(\xi) = P_0^{(1)}(\xi) = 1$. The coefficient λ_k is given by

$$\lambda_k = \frac{c\left(U_k(\xi)P_k(\xi)\right)}{c\left(\xi U_k(\xi)P_k^{(1)}(\xi)\right)},$$

where U_k is an arbitrary polynomial of the exact degree k [16].

Thus a breakdown occurs in this relation if and only if

$$c\left(\xi U_k(\xi)P_k^{(1)}(\xi)\right) = c^{(1)}\left(U_k(\xi)P_k^{(1)}(\xi)\right) = 0.$$

Thanks to the orthogonality conditions of $P_k^{(1)}$, this is equivalent to

$$c^{(1)}\left(\xi^k P_k^{(1)}(\xi)\right) = 0.$$

Thus, by (10), we see that a breakdown occurs if and only if $H_{k+1}^{(1)} = 0$ or, in other words, if and only if $P_{k+1}^{(1)}$ and P_{k+1} do not exist. Such a breakdown, due to the nonexistence of the polynomial which is to be computed and not to the recurrence relationship used, is called a *true* *breakdown* [11]. It will give rise to a breakdown, called a *pivot breakdown* [12], in any algorithm for implementing the Lánczos method.

The polynomials $\{P_k^{(1)}\}$ satisfy the usual three-term recurrence relationship which becomes, since these polynomials are monic,

$$P_{k+1}^{(1)}(\xi) = (\xi - a_{k+1})\, P_k^{(1)}(\xi) - b_{k+1} P_{k-1}^{(1)}(\xi), \tag{12}$$

with $P_0^{(1)}(\xi) = 1$ and $P_{-1}^{(1)}(\xi) = 0$. The coefficients a_{k+1} and b_{k+1} are given by

$$b_{k+1} = \frac{c^{(1)}\left(\xi U_{k-1}(\xi) P_k^{(1)}(\xi)\right)}{c^{(1)}\left(U_{k-1}(\xi) P_{k-1}^{(1)}(\xi)\right)},$$

$$a_{k+1} = \frac{c^{(1)}\left(\xi U_k(\xi) P_k^{(1)}(\xi)\right) - b_{k+1} c^{(1)}\left(U_k(\xi) P_{k-1}^{(1)}(\xi)\right)}{c^{(1)}\left(\xi U_k(\xi) P_k^{(1)}(\xi)\right)},$$

where $\{U_k\}$ is an auxiliary family of polynomials such that $\forall k, U_k$ has the exact degree k. We see that, for the same reason as above, the recurrence relationship (12) can only be the subject of true breakdowns.

Thus, using alternately the relations (11) and (12) allows us to compute simultaneously the two families $\{P_k\}$ and $\{P_k^{(1)}\}$. Setting

$$r_k = P_k(A) r_0 \qquad \text{and} \qquad z_k = P_k^{(1)}(A) r_0,$$

these two relations give

$$\begin{aligned} r_{k+1} &= r_k - \lambda_k A z_k, \\ x_{k+1} &= x_k + \lambda_k z_k, \\ z_{k+1} &= A z_k - a_{k+1} z_k - b_{k+1} z_{k-1}. \end{aligned}$$

This algorithm is known under the names of *Lánczos/Orthodir* [13] and also *BIODIR* [14] when $U_k \equiv P_k^{(1)}$. Among all the recursive algorithms for implementing the Lánczos method, it is the only one which can suffer only from true breakdowns. This algorithm cannot be the seat of ghost breakdowns and thus it is the only reliable algorithm for the implementation of the Lánczos method. Moreover, as pointed out in [17] (see also [18]), its convergence properties make it a more interesting algorithm than the others.

The *classical Lánczos algorithm* [19,20] (also called the *non-Hermitian Lánczos algorithm* [21]), uses the three-term recurrence relationship of the polynomials $P_k^{(1)}$.

Another possibility for implementing Lánczos method is to compute P_{k+1} from P_k and $P_k^{(1)}$ (or, more precisely, a polynomial proportional to it), and $P_{k+1}^{(1)}$ from $P_k^{(1)}$ and P_{k+1}. The corresponding algorithm is called *Lánczos/Orthomin*. It is essentially due to Vinsome [22]. Another implementation, corresponding to a different choice of the auxiliary polynomials U_k, is the *biconjugate gradient algorithm* (BCG) due to Lánczos [1,3], but which only became known after having being put under a more algorithmic form by Fletcher [6].

We shall now explain how to avoid breakdowns in the recursive algorithms for the Lánczos method.

3. AVOIDING TRUE BREAKDOWNS

The treatment of a true breakdown consists in the following operations:

1. to be able to recognize the occurrence of such a breakdown, that is, that the next orthogonal polynomial does not exist,

2. to be able to determine the degree of the next existing (that is, regular) orthogonal polynomial,
3. to be able to jump over the nonexisting orthogonal polynomials and to have a recurrence relationship which makes only use of the regular ones.

This problem was completely treated by Draux [23] in the case of monic orthogonal polynomials. Since the polynomials $\{P_k^{(1)}\}$ are monic, we shall use his results. But before that, we shall slightly change our notations to simplest ones.

Up to now, the k^{th} polynomial of the family had exactly the degree k, and thus, it was denoted by $P_k^{(1)}$. Now, since some of the polynomials of the family may not exist, we shall only give an index to the existing ones. Thus, the k^{th} regular polynomial of the family will still be denoted by $P_k^{(1)}$, but now, its degree will be equal to n_k with $n_k \geq k$. The next regular polynomial will be denoted by $P_{k+1}^{(1)}$ and its degree n_{k+1} will be $n_{k+1} = n_k + m_k$. Thus, m_k is the *jump* in the degrees between the regular polynomial $P_k^{(1)}$ and the next one. This change in the notations means that $P_k^{(1)}$ is, in fact, the polynomial previously denoted by $P_{n_k}^{(1)}$. Since the polynomials of the degrees $n_k + 1, \dots, n_k + m_k - 1$ do not exist, we are not giving them a name. The same change of notations will be made for the family $\{P_k\}$.

It was proved by Draux [23] that m_k is defined by the conditions

$$c^{(1)}\left(\xi^i P_k^{(1)}\right) = 0, \qquad \text{for } i = 0, \dots, n_k + m_k - 2 \tag{13}$$

$$\neq 0, \qquad \text{for } i = n_k + m_k - 1. \tag{14}$$

Moreover, these polynomials can be recursively computed by the relationship

$$P_{k+1}^{(1)}(\xi) = \left(\alpha_0 + \cdots + \alpha_{m_k-1}\xi^{m_k-1} + \xi^{m_k}\right) P_k^{(1)}(\xi) - C_{k+1}P_{k-1}^{(1)}(\xi), \tag{15}$$

for $k = 0, 1, \dots$, with $P_{-1}^{(1)}(\xi) = 0$, $P_0^{(1)}(\xi) = 1$, $C_1 = 0$, and

$$C_{k+1} = \frac{c^{(1)}\left(\xi^{n_k+m_k-1}P_k^{(1)}\right)}{c^{(1)}\left(\xi^{n_k-1}P_{k-1}^{(1)}\right)},$$

$$\alpha_{m_k-1}c^{(1)}\left(\xi^{n_k+m_k-1}P_k^{(1)}\right) + c^{(1)}\left(\xi^{n_k+m_k}P_k^{(1)}\right) = C_{k+1}c^{(1)}\left(\xi^{n_k}P_{k-1}^{(1)}\right),$$

$$\vdots$$

$$\alpha_0 c^{(1)}\left(\xi^{n_k+m_k-1}P_k^{(1)}\right) + \cdots + \alpha_{m_k-1}c^{(1)}\left(\xi^{n_k+2m_k-2}P_k^{(1)}\right) + c^{(1)}\left(\xi^{n_k+2m_k-1}P_k^{(1)}\right)$$
$$= C_{k+1}c^{(1)}\left(\xi^{n_k+m_k-1}P_{k-1}^{(1)}\right).$$

Since, by definition of m_k, $c^{(1)}(\xi^{n_k+m_k-1}P_k^{(1)})$ is different from zero, then this system is nonsingular, which shows that no breakdown (true or ghost) can occur in (15).

For implementing the Lánczos method by the algorithm *Lánczos/Orthodir*, we also need to compute P_{k+1} from P_k and $P_k^{(1)}$. As proved in [24], we have

$$P_{k+1}(\xi) = P_k(\xi) - \xi\left(\beta_0 + \cdots + \beta_{m_k-1}\xi^{m_k-1}\right) P_k^{(1)}(\xi), \tag{16}$$

where the β_i's are given by the system

$$\beta_{m_k-1}c^{(1)}\left(\xi^{n_k+m_k-1}P_k^{(1)}\right) = c\left(\xi^{n_k}P_k\right)$$

$$\vdots$$

$$\beta_0 c^{(1)}\left(\xi^{n_k+m_k-1}P_k^{(1)}\right) + \cdots + \beta_{m_k-1}c^{(1)}\left(\xi^{n_k+2m_k-2}P_k^{(1)}\right) = c\left(\xi^{n_k+m_k-1}P_k\right).$$

This relation generalizes (11).

Thus, using alternately (15) and (16) gives a breakdown-free algorithm for implementing the Lánczos method. This algorithm, given in [24], was called the MRZ where the initials stand for *Method of Recursive Zoom*. It can only suffer from an *incurable hard breakdown* which occurs when $c^{(1)}(\xi^{n-1}P_k^{(1)}) = 0$, where n is the dimension of the linear system to be solved. Quite similar breakdown-free algorithms were also obtained by Gutknecht [25,26].

The *Conjugate Gradient Squared* algorithm (CGS) was obtained by Sonneveld [27]. It consists of considering the residual vectors given by

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

with P_k as defined above. By computing recursively the polynomials P_k^2 and not the polynomials P_k, one avoids the use of A^*, a drawback of the Lánczos method. This is possible by squaring the recurrence relationships used for implementing the Lánczos method. Thus, true and ghost breakdowns can appear in the recursive algorithms for implementing the CGS for the same reasons as explained above. This is, in particular, the case for the algorithm given by Sonneveld [27] which consists of squaring the recurrence relationships of Lánczos/Orthomin. Since *Lánczos/Orthodir* can only suffer from true breakdowns, then squaring (15) and (16) leads to a breakdown-free algorithm for the CGS called the MRZS [28].

Another strategy for avoiding true breakdowns in Lánczos/Orthomin was proposed by Bank and Chan [29,30]. It is similar to the technique proposed in [31,32] and improved in [33]. It consists of a 2×2 composite step and the corresponding algorithm was called the CSBCG. This technique was extended to the CGS by Chan and Szeto [12] and the algorithm was named CSCGS.

Let us now consider the other recurrence relationships between orthogonal polynomials. They can all be used for implementing the Lánczos method. However, as we saw above, they can be the subject of ghost breakdowns. We shall now explain how to avoid these ghost breakdowns.

4. AVOIDING GHOST BREAKDOWNS

Let us, for example, try to compute $P_{k+1}^{(1)}$ from $P_k^{(1)}$ and P_k. As proved in [34], we have the relation

$$P_{k+1}^{(1)}(\xi) = \left(\delta_0 + \cdots + \delta_{m_k-1}\xi^{m_k-1} + \xi^{m_k}\right) P_k^{(1)}(\xi) - D_{k+1}P_k(\xi), \tag{17}$$

where m_k is defined as above.

This relation can be used in conjunction with (16) for computing recursively the families $\{P_k\}$ and $\{P_k^{(1)}\}$. Imposing the orthogonality conditions, we have

$$\begin{aligned}
D_{k+1}c\left(\xi^{n_k}P_k\right) &= c^{(1)}\left(\xi^{n_k+m_k-1}P_k^{(1)}\right)\\
D_{k+1}c\left(\xi^{n_k+1}P_k\right) - \delta_{m_k-1}c^{(1)}\left(\xi^{n_k+m_k-1}P_k^{(1)}\right) &= c^{(1)}\left(\xi^{n_k+m_k}P_k^{(1)}\right)\\
&\vdots\\
D_{k+1}c\left(\xi^{n_k+m_k}P_k\right) - \delta_0 c^{(1)}\left(\xi^{n_k+m_k-1}P_k^{(1)}\right) - \cdots - \delta_{m_k-1}c^{(1)}\left(\xi^{n_k+2m_k-2}P_k^{(1)}\right)&\\
= c^{(1)}\left(\xi^{n_k+2m_k-1}P_k^{(1)}\right).&
\end{aligned}$$

Since $c^{(1)}(\xi^{n_k+m_k-1}P_k^{(1)})$ is different from zero by definition of m_k, the preceding system is regular if and only if $c(\xi^{n_k}P_k) \neq 0$. If this condition is not satisfied, then a ghost breakdown will occur in the algorithm. The corresponding algorithm for implementing the Lánczos method was called the SMRZ and it is discussed at length in [34]. Squaring its recurrence relationships leads to the SMRZS for implementing the CGS [28].

It is possible to avoid such a ghost breakdown by jumping farther, until polynomials P_k and $P_k^{(1)}$ satisfying, in addition, the condition $c(\xi^{n_k}P_k) \neq 0$ have been found. Thus, now, we must be able to jump not only over nonexisting orthogonal polynomials but also over regular ones. The same phenomenon arises when trying to compute $P_{k+1}^{(1)}$ from P_{k+1} and $P_k^{(1)}$. The corresponding algorithm for the Lánczos method was called the BMRZ [34] and the BMRZS for the CGS [28].

For jumping over regular polynomials, it is necessary to use special recurrence relationships. They can be obtained by the technique explained in [16] and their coefficients are found by imposing the orthogonality conditions to both sides of the relations. For example, (16) becomes in that case

$$P_{k+1}(\xi) = (1 - \xi v_k(\xi))\, P_k(\xi) - \xi w_k(\xi) P_k^{(1)}(\xi), \tag{18}$$

where w_k is a polynomial of the degree $m_k - 1$ at most and v_k a polynomial of the degree $m_k - 2$ at most. For computing the coefficients of these polynomials, it is necessary to consider two cases according whether or not $n_k - m_k + 1$ is greater or equal to zero. The corresponding relations can be found in [34].

For computing the two families of polynomials $\{P_k\}$ and $\{P_k^{(1)}\}$, a second recurrence relationship is needed. The first possible choice is to use the three-term recurrence relationship (15) which now becomes

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

where q_k is a monic polynomial of the degree m_k and p_k a polynomial of the degree $m_k - 1$ at most. Their coefficients are given in [34]. The corresponding algorithm for implementing the Lánczos method uses alternately (18) and (19) and is called the GMRZ. It is a generalization of the MRZ.

The second choice consists in generalizing the relation (17) which becomes

$$P_{k+1}^{(1)}(\xi) = s_k(\xi) P_k^{(1)}(\xi) + t_k(\xi) P_k(\xi), \tag{20}$$

where s_k is a monic polynomial of the degree m_k and t_k a polynomial of the degree $m_k - 1$ at most whose coefficients can be computed as explained in [34]. Making use alternately of the relations (18) and (20) for implementing the Lánczos method leads to an algorithm named the BSMRZ which generalizes the SMRZ. Squaring the recurrence relationships of this algorithm produces the algorithm called BSMRZS for implementing the CGS [35]. In the BSMRZS, the most difficult point was to find out how to avoid the use of A^*. It was possible to overcome this problem by expressing the orthogonal polynomials on a basis different from the canonical one and then imposing the orthogonality conditions with respect to a suitably chosen family of auxiliary polynomials U_k. A simpler version of this algorithm, which makes use of A^*, can be found in [36].

As shown in [34], it is impossible to generalize the BMRZ.

FORTRAN subroutines corresponding to some of these algorithms can be found in [34] together with numerical examples; see also [35,37].

Let us mention that Gutknecht proposed an unnormalized version of the BIORES algorithm for curing ghost breakdowns in the BIORES by using a three-term recurrence relationship and, by squaring it, he obtained the unnormalized BIORES^2 for treating ghost breakdowns in the CGS [14] (these algorithms will be, respectively, denoted by uBIORES and uBIORES^2 in Table 1 below). Another procedure for treating breakdowns in the classical Lánczos algorithm is described in [38].

5. NEAR-BREAKDOWNS

As explained above, a breakdown occurs in a recurrence relationship when a quantity arising in the denominator of one of its coefficients is equal to zero. If such a quantity is not exactly zero, but close to it, then the corresponding coefficient can become very large and badly computed and roundoff errors can affect seriously the algorithm. This situation is called a *near-breakdown*. In order to avoid such a numerical instability, it is necessary to jump over all the polynomials which could be badly computed and to compute directly the first regular polynomial which follows. Such procedures, which consist in jumping over polynomials which do not exist or could be badly computed, were first introduced by Taylor [31] and Parlett, Taylor and Liu [32] under the name of *look-ahead* techniques (see [33] for an improvement). They are based on recurrence relationships allowing to jump over existing polynomials. Such relations were already given in the preceding section as well as the corresponding algorithms for the implementation of the Lánczos method. These algorithms are the GMRZ and the BSMRZ [34,37]. A look-ahead technique for avoiding breakdowns and near-breakdowns in the three-term recurrence relationship satisfied by the polynomials P_k was also proposed in [20] under the name of *look-ahead Lánczos algorithm*. It reduces to the classical Lánczos algorithm (that is Lánczos/Orthores) when no jump occurs; see also [19,39]. For the CGS, we have the algorithm called the BSMRZS [35]. The case of other algorithms where the residual vector r_k is defined as $r_k = V_k(A)P_k(A)r_0$, with V_k a polynomial satisfying some recurrence relationship, was investigated in [40]. Algorithms of this type are called *Conjugate Gradient Multiplied* (CGM). This class of methods includes, as a particular case, the Bi-CGSTAB of Van der Vorst [41].

In all these algorithms, the main point (which is quite difficult) is the definition of the near-breakdown itself. In other words, it is difficult to decide when and how far to jump. Changing the definition can lead to very different numerical results. Let us now explain how we finally solved this problem.

We saw above that, in the case of a true breakdown, the length m_k of the jump is given by the conditions (13) and (14). Of course, in practice, it is impossible to check a strict equality to zero. So, in our first implementation of the algorithms [34,37], we chose, for treating the near-breakdown, a threshold value ε and defined the value of m_k by the conditions

$$\begin{aligned} \left|c^{(1)}\left(\xi^i P_k^{(1)}\right)\right| &\le \varepsilon, \qquad \text{for } i = 0, \ldots, n_k + m_k - 2, \\ &> \varepsilon, \qquad \text{for } i = n_k + m_k - 1. \end{aligned}$$

Obviously, these conditions force themselves from (13) and (14). However, the beginning and the length of the jumps were quite sensitive to the choice of ε and so were also the numerical results. It meant that it was not the proper way of jumping and that our test had to be changed for a more appropriate one.

Let us explain how this problem was solved in the case of the CGS [35].

In that case, since $P_k^{(1)}$ has exactly the degree n_k, the preceding inequalities can be replaced by the equivalent ones

$$\begin{aligned} \left|c^{(1)}\left(\xi^i P_k^{(1)^2}\right)\right| &\le \varepsilon, \qquad \text{for } i = 0, \ldots, m_k - 2 \quad \text{and} \\ \left|c^{(1)}\left(\xi^i P_k^{(1)^2}\right)\right| &> \varepsilon, \qquad \text{for } i = m_k - 1. \end{aligned}$$

This test was used in many examples but the results obtained were also very sensitive to the value of ε (see [36]). The reason was that the beginning of the jump was correctly defined by the condition

$$\left|c^{(1)}\left(P_k^{(1)^2}\right)\right| \le \varepsilon$$

(more precisely, the ratio of this quantity to $|c(P_k^{(1)}P_k)|$ and, for the first step, to $|c^{(1)}(\xi P_k^{(1)^2})|$ also), but that the end of the jump (that is the value of m_k) was not properly given by the condition

$$\left|c^{(1)}\left(\xi^{m_k-1} P_k^{(1)^2}\right)\right| > \varepsilon.$$

Defining m_k in that way, led, in some examples, to a value of m_k which was too large, thus producing a numerical instability because we jumped over polynomials which were well computed.

The remedy we used consists in replacing the above condition giving the value of m_k by a test on the near singularity of the system for computing the coefficients of the recurrence relationships. We continue to jump until a nonnearly singular system has been found, which gives the value of m_k.

This type of near-breakdown is clearly related, by (13) and (14), to a true breakdown and thus it can be called a *true near-breakdown*.

But there is also a second type of near-breakdown which can be called a *ghost near-breakdown* since it is related to the ghost breakdown as defined above. Indeed, since our algorithm was obtained by squaring the relations (18) and (20), a ghost breakdown, due to $c(\xi^{n_k} P_k) = 0$, could also occur. So, the ghost near-breakdown which arises when this quantity is close to zero, has also to be avoided. In the program given in [36], it was not tried curing this type of ghost near-breakdown and the program stopped in that case, which can explain its numerical instability since we could divide by quantities close to zero. Let us now explain how we treated this problem in the program given in [35].

It was observed numerically that the quantity

$$\sigma_{k+1}^{(1)} = c\left(\xi P_k^{(1)} P_k\right) - c\left(P_k^{(1)} P_k\right) \cdot \frac{c^{(1)}\left(\xi P_k^{(1)^2}\right)}{c^{(1)}\left(P_k^{(1)^2}\right)}$$

was close to zero in two cases:

1. when it is necessary to jump, and
2. when the exact solution will be obtained at the end of the current iteration.

Let us now explain the theoretical reasons for this observation. It is easy to see that

$$\sigma_{k+1}^{(1)} = c^{(1)}\left(P_k^{(1)} P_{k+1}\right) = c\left(\xi^{n_{k+1}} P_{k+1}\right).$$

By definition of the linear functional $c^{(1)}$, we have

$$\sigma_{k+1}^{(1)} = \left(y, A P_k^{(1)}(A) P_{k+1}(A) r_0\right),$$

which shows that $\sigma_{k+1}^{(1)} = 0$ if $P_{k+1}(A) r_0 = 0$, that is, if the exact solution will be obtained at the end of the current iteration.

The quantity $\sigma_{k+1}^{(1)}$ can also be zero if the orthogonality conditions of P_{k+1} are satisfied farther than $n_{k+1} - 1$. In that case, as explained above, a ghost breakdown will occur at the next iteration and thus it is necessary to jump farther during the current iteration. Obviously a ghost near-breakdown occurs if this quantity is close to zero and we also have to jump in this case.

Thus we now have to decide how far to jump. The value of m_k is set to 2 and the systems giving the coefficients of the recurrence relationships are solved. If these systems are singular (pivot $= 0$) or nearly singular ($|\text{pivot}| \le \varepsilon_1$, where ε_1 is some threshold value) then m_k is changed to $m_k + 1$ and the procedure is repeated until a nonnearly singular system has been obtained. Then, we have to check if a ghost breakdown (or a ghost near-breakdown) could occur at the beginning of the next iteration. The quantity by which we shall have to divide at the beginning of the next iteration is

$$\sigma_{k+1}^{(m_k)} = c^{(1)}\left(\xi^{m_k - 1} P_k^{(1)} P_{k+1}\right) = c\left(\xi^{n_{k+1}} P_{k+1}\right).$$

If it is equal to zero, it can mean, as before, that the solution will be obtained at the end of the current iteration. It is not the case and if $|\sigma_{k+1}^{(m_k)}|$ is zero (or close to it), then we shall have a ghost breakdown (or a ghost near-breakdown) at the next iteration. Thus, the preceding jump was not long enough and we have still to increase by 1 the value of m_k. This procedure has to be repeated until polynomials satisfying all the previous conditions have been obtained.

Thus finally, for avoiding a near-breakdown, it is necessary, first to decide when to jump and then to find the value m_k of the length of the jump. For obtaining m_k, two different tests have to be performed:

1. the singularity (or the near-singularity) of the systems giving the coefficients of the polynomials, and
2. the value of the quantity $|\sigma_{k+1}^{(m_k)}|$.

These new tests have been incorporated in our codes for curing true and ghost near-breakdowns in the CGS [35] and in the Bi-CGSTAB [40]. Other tests for checking small quantities have also been included. The preceding tests have not yet been implemented in our codes for the SMRZ, the BMRZ and the BSMRZ.

6. CONCLUSIONS

The analysis and remedy for breakdowns and near-breakdowns presented in this paper came out from the theory of formal orthogonal polynomials which forms the foundations for procedures based on the Lánczos method. As shown above, these orthogonal polynomials occur not only in the case where the matrix of the system is symmetric positive definite (which is the usual well-known case where the linear functionals can be represented as an integral with respect to a positive Borel measure on the real line; see [42,43], for example) but also in the case of an arbitrary nonsymmetric matrix (which corresponds to an indefinite inner product; see [10,21], for example). In our opinion, this approach, based on orthogonal polynomials, is simple and powerful and could possibly be extended to other algorithms related to Lánczos method such as those using biorthogonal polynomials [44,45]. It could also possibly be useful in implementing the extensions of these methods to nonlinear systems [46]. The classical approach to the Lánczos method and to Krylov subspace methods by linear algebra techniques can be found in [47,48], and the problems of breakdown and near-breakdown are discussed in [49,50].

We do not pretend that the techniques summarized in this paper are able to cure all the possible near-breakdowns, nor that our codes are for all seasons. But, from the numerical examples performed, it seems that they are, at least, able to bring some more numerical stability to the algorithms. Another important and open question concerns the optimal choice (in the sense of numerical stability) of the auxiliary polynomials $\{U_k\}$ appearing in the computation of the coefficients of the various recurrence relationships used for the orthogonal polynomials.

As mentioned in [51], roundoff errors can appear as a scaling (or normalization) problem. So, instead of working with the polynomials P_k and $P_k^{(1)}$, it is possible to use the polynomials $\hat{P}_k = \hat{a}_k P_k$ and $\widehat{P_k^{(1)}} = \hat{b}_k P_k^{(1)}$, where $\hat{a}_k$ and $\hat{b}_k$ are suitably chosen scaling factors. A Lánczos/Orthodir algorithm modified along this idea was proposed in [52]. However, it must be noticed that changing the normalization could also change the nature of the breakdowns and near-breakdowns in an algorithm.

Thanks to the close connection between formal orthogonal polynomials and Padé approximants [9,14,25,26], any stable algorithm for computing recursively the sequence $([k-1/k])$ of Padé approximants could possibly be adapted to our problem. Such an algorithm was recently derived by Cabay and Meleshko [53]. It consists in computing scaled polynomials $\hat{P}_k$ by their recurrence relationship and jumping over the polynomials which could be badly computed, according to some criteria. The criteria are, in fact, a measure of the conditioning of the Hankel system which gives the coefficients of the orthogonal polynomials. Only the orthogonal polynomials corresponding to well-conditioned Hankel systems are computed, the other ones being skipped over. Thus a weakly stable algorithm is obtained. It computes a sequence of Padé approximants along the main subdiagonal of the Padé table. When the threshold value used in the criteria for the jumps is set to zero, then an algorithm previously proposed by Cabay and Choi [54] for jumping over the nonexisting polynomials $\hat{P}_k$ is recovered. So, the ideas used in [53,54] (see also [55]) are quite similar to ours. They still have to be applied to the treatment of breakdowns and near-breakdowns in Lánczos algorithms. In particular, using the algorithms given in [53,54] will produce, respectively, procedures for curing breakdowns and near-breakdowns in Lánczos/Orthores.

Let us also mention that a minimal residual smoothing technique [56–58] or an hybrid procedure [59] can be incorporated into the algorithms in order to improve their numerical stability and to smooth their convergence. Numerical examples could be found in [12,59].

The algorithms discussed in this paper are summarized in Table 1.

Table 1.

recurrence relations	algorithm's name	Lánczos method		CGS method	
		breakdown	near-break.	breakdown	near-break.
$P_{k+1} \leftarrow P_k, P_k^{(1)}$ $P_{k+1}^{(1)} \leftarrow P_k^{(1)}, P_{k-1}^{(1)}$	Lánczos/Orthodir	MRZ	GMRZ	MRZS	
$P_{k+1} \leftarrow P_k, P_{k-1}$	Lánczos/Orthores	look-ahead uBIORES	look-ahead	uBIORES^2	
$P_{k+1} \leftarrow P_k, P_k^{(1)}$ $P_{k+1}^{(1)} \leftarrow P_k^{(1)}, P_{k+1}$	Lánczos/Orthomin	BMRZ CSBCG	does not exist	BMRZS CSCGS	does not exist
$P_{k+1} \leftarrow P_k, P_k^{(1)}$ $P_{k+1}^{(1)} \leftarrow P_k^{(1)}, P_k$	A8/B8	SMRZ	BSMRZ	SMRZS	BSMRZS

We are currently programming the GMRZ and also filling up the void places in Table 1. In particular, it will be interesting to derive the algorithm based on Lánczos/Orthodir for treating near-breakdowns in the CGS, since this algorithm is the only reliable one for implementing Lánczos method.

The ideas developed in this paper only became clear after programming the algorithms and testing them on many numerical examples, which shows once more, if necessary, that, as stated by Wynn [60]

... numerical analysis is very much an experimental science.

REFERENCES

1. C. Lánczos, An iteration method for the solution of the eigenvalue problem of linear differential and integral operators, *J. Res. Natl. Bur. Stand.* **45**, 255–282 (1950).
2. I. Kardos, *Scientists Face to Face*, Corvina Kidó, Budapest, (1978).
3. C. Lánczos, Solution of systems of linear equations by minimized iterations, *J. Res. Natl. Bur. Stand.* **49**, 33–53 (1952).
4. M.R. Hestenes and E. Stiefel, Methods of conjugate gradients for solving linear systems, *J. Res. Natl. Bur. Stand.* **49**, 409–436 (1952).
5. M.R. Hestenes, The conjugate-gradient method for solving linear systems, In *Proceedings of the Sixth Symposium on Applied Mathematics*, (Edited by J. Curtiss), pp. 83–102, Amer. Math. Soc., Providence, RI, (1956).
6. R. Fletcher, Conjugate gradient methods for indefinite systems, In *Numerical Analysis*, LNM Vol. 506, (Edited by G.A. Watson), pp. 73–89, Springer-Verlag, Berlin, (1976).
7. G.H. Golub and D.P. O'Leary, Some history of the conjugate and Lanczos algorithms, *SIAM Rev.* **31**, 50–102 (1989).
8. M.R. Hestenes, *Conjugate Direction Methods in Optimization*, Springer-Verlag, Berlin, (1980).
9. C. Brezinski, *Padé-Type Approximation and General Orthogonal Polynomials*, ISNM Vol. 50, Birkhäuser-Verlag, Basel, (1980).
10. C. Brezinski and H. Sadok, Lanczos-type algorithms for solving systems of linear equations, *Appl. Numer. Math.* **11**, 443–473 (1993).

11. C. Brezinski and M. Redivo-Zaglia, Breakdowns in the computation of orthogonal polynomials, In *Nonlinear Numerical Methods and Rational Approximation, II*, (Edited by A. Cuyt), pp. 49–59, Kluwer, Dordrecht, (1994).
12. T.F. Chan and T. Szeto, A composite step conjugate gradients squared algorithm for solving nonsymmetric linear systems, *Numerical Algorithms* **7**, 17–32 (1994).
13. D.M. Young and K.C. Jea, Generalized conjugate-gradient acceleration of nonsymmetrizable iterative methods, *Linear Algebra Appl.* **34**, 159–194 (1980).
14. M.H. Gutknecht, The unsymmetric Lanczos algorithms and their relations to Padé approximation, continued fractions, and the qd algorithm, In *Preliminary Proceedings of the Copper Mountain Conference on Iterative Methods*, April 1–5, 1990.
15. C. Baheux, New implementations of Lanczos method, *J. Comput. Appl. Math.* **57**, 3–15 (1995).
16. C. Brezinski and M. Redivo-Zaglia, A new presentation of orthogonal polynomials with applications to their computation, *Numerical Algorithms* **1**, 207–221 (1991).
17. K.C. Jea, Generalized conjugate gradient acceleration of iterative methods, Ph.D. Thesis, Dept. of Mathematics, University of Texas at Austin, (1982).
18. K.C. Jea and D.M. Young, On the simplification of generalized conjugate gradient methods for nonsymmetrizable linear systems, *Linear Algebra Appl.* **52**, 299–317 (1983).
19. R.W. Freund, G.H. Golub and N.M. Nachtigal, Iterative solution of linear systems, *Acta Numerica* **1**, 57–100 (1991).
20. 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. Comput.* **14**, 137–158 (1993).
21. Y. Saad, The Lanczos biorthogonalization algorithm and other oblique projection methods for solving large unsymmetric systems, *SIAM J. Numer. Anal.* **19**, 485–506 (1982).
22. P.K.W. Vinsome, Orthomin, an iterative method for solving sparse sets of simultaneous equations, In *Proceedings 4^{th} Symposium on Reservoir Simulation*, pp. 149–159, Society of Petroleum Engineers of AIME, (1976).
23. A. Draux, *Polynômes Orthogonaux Formels. Applications*, LNM Vol. 974, Springer-Verlag, Berlin, (1983).
24. C. Brezinski, M. Redivo-Zaglia and H. Sadok, A breakdown-free Lanczos type algorithm for solving linear systems, *Numer. Math.* **63**, 29–38 (1992).
25. M.H. Gutknecht, A completed theory of the unsymmetric Lanczos process and related algorithms, Part I, *SIAM J. Matrix Anal. Appl.* **13**, 594–639 (1992).
26. M.H. Gutknecht, A completed theory of the unsymmetric Lanczos process and related algorithms, Part II, *SIAM J. Matrix Anal. Appl.* **15**, 15–58 (1994).
27. P. Sonneveld, A fast Lanczos-type solver for nonsymmetric linear systems, *SIAM J. Sci. Stat. Comput.* **10**, 35–52 (1989).
28. C. Brezinski and H. Sadok, Avoiding breadkown in the CGS algorithm, *Numerical Algorithms* **1**, 199–206 (1991).
29. R.E. Bank and T.F. Chan, A composite step bi-conjugate gradient algorithm for solving nonsymmetric systems, *Numerical Algorithms* **7**, 1–16 (1994).
30. R.E. Bank and T.F. Chan, An analysis of the composite step bi-conjugate gradient algorithm for solving nonsymmetric systems, *Numer. Math.* **66**, 295–319 (1993).
31. D.R. Taylor, Analysis of the look-ahead Lanczos algorithm, Ph.D. Thesis, Dept. of Mathematics, University of California, Berkeley, (November 1982).
32. B.N. Parlett, D.R. Taylor and Z.A. Liu, A look-ahead Lanczos algorithm for unsymmetric matrices, *Math. Comput.* **44**, 105–124 (1985).
33. M. Khelifi, Lanczos maximal algorithm for unsymmetric eigenvalue problems, *Appl. Numer. Math.* **7**, 179–193 (1991).
34. C. Brezinski, M. Redivo-Zaglia and H. Sadok, Avoiding breakdown and near-breakdown in Lanczos type algorithms, *Numerical Algorithms* **1**, 261–284 (1991).
35. C. Brezinski and M. Redivo-Zaglia, Treatment of near-breakdown in the CGS algorithm, *Numerical Algorithms* **7**, 33–73 (1994).
36. C. Brezinski and M. Redivo-Zaglia, Treatment of near-breakdown in the CGS algorithm, Publication ANO 257, Université des Sciences et Technologies de Lille, (November 1991).
37. C. Brezinski, M. Redivo-Zaglia and H. Sadok, Addendum to "Avoiding breakdown and near-breakdown in Lanczos type algorithms", *Numerical Algorithms* **2**, 133–136 (1992).
38. D.L. Boley, S. Elhay, G.H. Golub and M.H. Gutknecht, Nonsymmetric Lanczos and finding orthogonal polynomials associated with indefinite weights, *Numerical Algorithms* **1**, 21–44 (1991).
39. R.W. Freund, Solution of shifted linear systems by quasi-minimal residual iterations, In *Numerical Linear Algebra*, (Edited by L. Reichel, A. Ruttan and R.S. Varga), pp. 101–121, W. de Gruyter, Berlin, (1993).
40. C. Brezinski and M. Redivo-Zaglia, Look-ahead in Bi-CGSTAB and other methods for linear systems, *BIT* **35**, 169–201 (1995).
41. 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. Comp.* **13**, 631–644 (1992).
42. B.N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice Hall, Englewood Cliffs, NJ, (1980).
43. D.S. Watkins, Some perspectives on the eigenvalue problem, *SIAM Rev.* **35**, 430–471 (1993).
44. C. Brezinski, *Biorthogonality and Its Applications to Numerical Analysis*, Marcel Dekker, New York, (1992).

45. C. Brezinski, Biorthogonality and conjugate gradient-type algorithms, In *Contributions in Numerical Mathematics*, WSSIAA Vol. 2, (Edited by R.P. Agarwal), pp. 55–70, World Scientific, Singapore, (1993).
46. C. Brezinski and H. Sadok, Some vector sequence transformations with applications to systems of equations, *Numerical Algorithms* **3**, 75–80 (1992).
47. Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, Manchester University Press, Manchester, (1992).
48. Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publ. Co., Boston, (1996).
49. 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, (1990).
50. N.M. Nachtigal, A look-ahead variant of the Lanczos algorithm and its application to the quasi-minimal residual method for non-Hermitian linear systems, Ph.D. Thesis, Massachusetts Institute of Technology, (1991).
51. Y. Saad and M.H. Schultz, Conjugate gradient-like algorithms for solving non-symmetric linear systems, *Math. Comput.* **44**, 417–424 (1985).
52. T.-Z. Mai, Modified Lanczos method for solving large sparse linear systems, *Commun. Numer. Meth. Engn.* **9**, 67–79 (1993).
53. S. Cabay and R. Meleshko, A weakly stable algorithm for Padé approximants and the inversion of Hankel matrices, *SIAM J. Matrix Anal. Appl.* **14**, 735–765 (1993).
54. S. Cabay and D.-K. Choi, Algebraic computations of scaled Padé fractions, *SIAM J. Comput.* **15**, 243–270 (1986).
55. R. Meleshko and S. Cabay, On computing Padé approximants quickly and accurately, *Congr. Numerantium* **80**, 245–255 (1991).
56. W. Schönauer, *Scientific Computing on Vector Computers*, North-Holland, Amsterdam, (1987).
57. R. Weiss, Convergence behavior of generalized conjugate gradient methods, Thesis, University of Karlsruhe, (1990).
58. L. Zhou and H.F. Walker, Residual smoothing techniques for iterative methods, *SIAM J. Sci. Comput.* **15**, 297–312 (1994).
59. C. Brezinski and M. Redivo-Zaglia, Hybrid procedures for solving linear systems, *Numer. Math.* **67**, 1–19 (1994).
60. P. Wynn, On some recent developments in the theory and application of continued fractions, *SIAM J. Numer. Anal.* Ser. B **1**, 177–197 (1964).