Applied Numerical Mathematics 11 (1993) 443–473
North-Holland

APNUM 405

Lanczos-type algorithms for solving systems of linear equations

Claude Brezinski and Hassane Sadok

Laboratoire d'Analyse Numérique et d'Optimisation, U.F.R. I.E.E.A. — M3,
Université des Sciences et Technologies de Lille, 59655 Villeneuve d'Ascq Cedex, France

Abstract

Brezinski, C. and H. Sadok, Lanczos-type algorithms for solving systems of linear equations, Applied Numerical Mathematics 11 (1993) 443–473.

In this paper, a synthesis of the various Lanczos-type algorithms for solving systems of linear equations is given. It is based on formal orthogonal polynomials and the various algorithms consist in using various recurrence relations for computing these orthogonal polynomials. Moreover new algorithms are easily obtained from the theory. New formulae and a new interpretation of the conjugate gradient squared (CGS) algorithm are also derived and a new formula for the second topological ε-algorithm. The theory of orthogonal polynomials also enables us to avoid breakdown in Lanczos-type methods and in the CGS. The case of near-breakdown can be treated similarly.

Keywords. Lanczos method; projection; orthogonal polynomials; extrapolation.

1. Introduction and definitions

In 1950, Lanczos [38] proposed an algorithm for transforming a matrix into a similar tridiagonal matrix. Lanczos method can also be used for solving a system of linear equations and it gives rise to an iterative method which gives the exact solution in a finite number of steps not greater than the dimension of the system [39]. As explained in [8], Lanczos method is connected to the method of moments and that of Galerkin. Thanks to its numerous advantages, Lanczos method was the subject of very many investigations (see [27] for an analysis of the literature), and many algorithms for implementing Lanczos method were obtained—among them, the famous conjugate gradient method of Hestenes and Stiefel [32] when the matrix is symmetric positive-definite. Other special types of matrices have also been considered (see [18,22,23] and [24] for a review). As explained in [43], all these algorithms can be put into the common framework of the Petrov–Galerkin oblique projection method on Krylov subspaces.

Correspondence to: C. Brezinski, Laboratoire d'Analyse Numérique et d'Optimisation, U.F.R. I.E.E.A.—M3, Université des Sciences et Technologies de Lille, 59655 Villeneuve d'Ascq Cedex, France. Telephone: (+33-20) 434296. Fax: (+33-20) 436869. E-mail: brezinsk@citil.citilille.fr.

Since the problem to be solved is a problem in linear algebra, most of the papers on the subject use linear algebra techniques. However, as shown in [6] for example, the problem is also strongly connected with the theory of formal orthogonal polynomials. Such a connection was, in fact, known from the very beginning [38] but, although several papers emphasize the important role of orthogonal polynomials in the theory (see [36] for a survey), it was mostly forgotten and replaced by pure linear algebra.

It is not our purpose, in this paper, to give a review of the existing literature on the subject but to stress again the theory of formal orthogonal polynomials and to show how to use it for deriving all the known algorithms (such as Lanczos/Orthodir, Lanczos/Orthores, Lanczos/Orthomin, BCG, and BIODIR) from a unified framework, how to obtain much more easily the classical theoretical results about Lanczos method, and finally to derive new results and algorithms, or new ways for computing the coefficients appearing in the recurrence relations of the various algorithms. We shall show that all these algorithms consist, in fact, in using different recurrence relationships for computing the orthogonal polynomials involved in Lanczos method. Moreover, new algorithms will automatically follow from the theory. Of course, these algorithms manifest themselves readily, once the idea of the underlying theory of orthogonal polynomials and their recurrence relations is understood. However, surprisingly, they were not previously studied. New formulae and a new interpretation of the conjugate gradient squared (CGS) algorithm, based on some extrapolation methods for vector sequences, will be given. Our aim is to show that the CGS is a barycentric combination of known vector sequence transformations. A new formula for the second topological ε-algorithm is also obtained. Let us also mention that the ε-algorithm provides a generalization of Lanczos method to systems of nonlinear equations which is derivative-free and has quadratic convergence under the usual assumptions [40]. Other generalizations of Lanczos method to nonlinear systems are given in [15].

The coefficients of the various recurrence relationships for orthogonal polynomials are given as ratios of scalar products. When a scalar product in a denominator vanishes, then a breakdown occurs in the algorithm. This is due to the non-existence of some orthogonal polynomial. We shall show that, in such a case, it is possible to jump over such non-existing polynomials, and breakdown-free algorithms are thus obtained. The breakdown can be avoided similarly in the conjugate gradient squared algorithm. The case of near-breakdown can be treated in a similar way.

Let us mention that formal orthogonal polynomials have almost all the algebraic properties of the usual orthogonal polynomials, except those for their zeros. Thus, our approach is a quite simple one and it could be understood without any prerequisite or effort. Lanczos method has been successfully taught that way to graduate students in numerical analysis at our university for several years.

Although several other papers, such as [28], deal with the same technique, we think that our approach is simpler. Indeed, only orthogonal polynomials are needed in our context, that is the denominators of the Padé approximants after reversing the numbering of the coefficients, and since their numerators are of no use in the theory, it seems to us that introducing Padé approximants or continued fractions [29] is not necessary for our purpose although they provide some new insight into the theory. As mentioned in [42], the use of orthogonal polynomials is *useful and simplifies the statements.*

It must be also mentioned that the starting point of [28] is Lanczos biorthogonalization

algorithm and not Lanczos method for solving a system of linear equations, which explains some of the differences between both approaches.

We must also point out that this paper only deals with the theory and the derivation of old and new algorithms for implementing Lanczos method but that we shall not discuss here their practical implementation nor their numerical stability. These questions are under consideration [2].

Let us now begin by some definitions and show how the theory of formal orthogonal polynomials provides a very natural basis for the study of Lanczos method.

We consider a system of n linear equations in n unknowns,

$$Ax = b, \tag{1.1}$$

where $A \in \mathbb{C}^{n\times n}$, $b \in \mathbb{C}^n$, and $x \in \mathbb{C}^n$.

Let x_0 and y be two nonzero arbitrary vectors in $\mathbb{C}^n$ and let (x_k) be the sequence of vectors defined by

$$x_k - x_0 \in K_k(A, r_0) \tag{1.2}$$

and

$$r_k = b - Ax_k \perp K_k(A^*, y), \tag{1.3}$$

where $K_k(A, r) = \operatorname{span}(r, Ar, \ldots, A^{k-1}r)$ and where A^* denotes the conjugate transpose of A.

From (1.2), $x_k - x_0$ can be written as

$$x_k - x_0 = -\alpha_1 r_0 - \cdots - \alpha_k A^{k-1} r_0, \tag{1.4}$$

and thus, multiplying both sides by A, adding and subtracting b, we have

$$r_k = r_0 + \alpha_1 A r_0 + \cdots + \alpha_k A^k r_0. \tag{1.5}$$

The orthogonality condition (1.3) can be written as

$$\left(A^{*i} y, r_k\right) = 0 \quad \text{for } i = 0, \ldots, k-1 \tag{1.6}$$

and, by (1.5), we obtain the system of linear equations

$$\begin{aligned} &\alpha_1(y, Ar_0) + \cdots + \alpha_k(y, A^k r_0) = -(y, r_0), \\ &\vdots \\ &\alpha_1\left(A^{*k-1} y, Ar_0\right) + \cdots + \alpha_k\left(A^{*k-1} y, A^k r_0\right) = -\left(A^{*k-1} y, r_0\right). \end{aligned} \tag{1.7}$$

The scalar product is defined as usual with the first argument conjugated.

If the determinant of this system is different from zero, then x_k exists and formulae (1.4) and (1.5) allow to obtain x_k and r_k. Of course, solving (1.7) directly for every k is not interesting and we shall see now how to solve it recursively for increasing values of k.

If we set

$$P_k(\xi) = 1 + \alpha_1 \xi + \cdots + \alpha_k \xi^k, \tag{1.8}$$

then (1.5) implies

$$r_k = P_k(A)r_0. \tag{1.9}$$

The polynomials P_k are commonly known as the residual polynomials. Another interpretation of the P_k's can be found in [16]. Moreover if we set

$$c_i = \left(A^{*i}y,\ r_0\right) = \left(y,\ A^i r_0\right), \quad i = 0, 1, \ldots, \tag{1.10}$$

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

$$c\left(\xi^i\right) = c_i, \quad i = 0, 1, \ldots, \tag{1.11}$$

then system (1.7) can be written as

$$c\left(\xi^i P_k(\xi)\right) = 0 \quad \text{for } i = 0, \ldots, k-1. \tag{1.12}$$

These conditions show that P_k is the polynomial of degree at most k belonging to the family of formal orthogonal polynomials with respect to the linear functional c (see [6]). These polynomials are normalized by the condition $P_k(0) = 1$, and P_k exists and is unique if and only if the Hankel determinant

$$H_k^{(1)} = \begin{vmatrix} c_1 & c_2 & \ldots & c_k \\ c_2 & c_3 & \ldots & c_{k+1} \\ \vdots & \vdots & & \vdots \\ c_k & c_{k+1} & \ldots & c_{2k-1} \end{vmatrix} \tag{1.13}$$

is different from zero.

We shall now assume, unless the contrary is specified, that this condition holds for all $k \leqslant n$.

Knowing P_k, r_k can be computed by (1.9) and x_k is obtained by

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

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

This is Lanczos method.

Orthogonal polynomials are known to satisfy recurrence relationships. Using such relations will give various procedures for computing recursively the polynomials P_k. Thus we shall obtain various algorithms for computing recursively the vectors r_k and x_k. Such methods are called Lanczos-type algorithms. As we shall see in Section 3, they differ by the recurrence relationship used for obtaining the polynomials P_k. Thus we shall study, in Section 2, how to compute recursively these orthogonal polynomials. However let us first prove an important result about Lanczos method.

Theorem 1.1. *If the vectors* $y, A^*y, \ldots, A^{*n-1}y$ *are linearly independent, then there exists* $k \leqslant n$ *such that* $r_k = 0$, *that is* $x_k = x$.

Proof. By the orthogonality conditions (1.12) and definitions (1.10) and (1.11) of the linear functional c we have

$$c\left(\xi^i P_k\right) = \left(A^{*i}y,\ r_k\right) = 0 \quad \text{for } i = 0, \ldots, k-1.$$

Thus, since r_k is orthogonal to $A^{*i}y$ for $i=0,\dots,k-1$ and since these vectors are assumed to be linearly independent, the vectors $y, A^*y,\dots,A^{*k-1}y$, and r_k are also linearly independent. In a vector space of dimension n, we cannot have more than n linearly independent vectors which proves the result. □

Of course this result is a classical one but usually its proof is quite long and tedious (see, for example, [31]). In giving it here, our aim is to show the simplification brought by the theory of formal orthogonal polynomials which is due to the orthogonality property of the polynomials. For a discussion of this value of k, which is in fact related to the degree of the minimal polynomial of A for y (which emphasizes the importance of choice of y) see the theory developed by Faber and Manteuffel [20]. If $A=A^*$ and $y=r_0$, then $c(P_iP_k)=(r_i, r_k)=0$ for $i\neq k$ and we recover the classical orthogonality property of the conjugate gradient method.

By using the determinantal formula for P_k (see (2.1) below) and $r_k=P_k(A)r_0=b-Ax_k$, we immediately obtain the following result which generalizes the result given in [6] (see also [13]).

Theorem 1.2.

$$x_k=\frac{\begin{vmatrix} x_0 & -r_0 & -Ar_0 & \dots & -A^{k-1}r_0 \\ c_0 & c_1 & c_2 & \dots & c_k \\ \vdots & \vdots & \vdots & & \vdots \\ c_{k-1} & c_k & c_{k+1} & \dots & c_{2k-1} \end{vmatrix}}{\begin{vmatrix} c_1 & \dots & c_k \\ \vdots & & \vdots \\ c_k & \dots & c_{2k-1} \end{vmatrix}}.$$

The determinant in the numerator of this formula denotes the vector obtained by expanding it with respect to its first row by the classical rules for expanding a determinant.

Now, solving (1.7) and substituting in (1.4) or using a generalization of Schur's formula proved in [7] and denoting by $(r_0, Ar_0,\dots,A^{k-1}r_0)$ the matrix whose columns are $r_0,\dots,A^{k-1}r_0$ we obtain:

Theorem 1.3.

$$x_k=x_0-(r_0, Ar_0,\dots,A^{k-1}r_0)\begin{pmatrix} c_1 & \dots & c_k \\ \vdots & & \vdots \\ c_k & \dots & c_{2k-1} \end{pmatrix}^{-1}\begin{pmatrix} c_0 \\ \vdots \\ c_{k-1} \end{pmatrix}.$$

The formulae given in these two theorems have applications in the theory (see [5–7,40] for example).

2. Formal orthogonal polynomials

We shall use the standard notations of the theory of orthogonal polynomials and recall the main results that will be useful. For a complete exposition, see [6]. We have

$$P_k(\xi) = \frac{\begin{vmatrix} 1 & \xi & \dots & \xi^k \\ c_0 & c_1 & \dots & c_k \\ \vdots & \vdots & & \vdots \\ c_{k-1} & c_k & \dots & c_{2k-1} \end{vmatrix}}{\begin{vmatrix} c_1 & \dots & c_k \\ \vdots & & \vdots \\ c_k & \dots & c_{2k-1} \end{vmatrix}}. \tag{2.1}$$

Thus, obviously, conditions (1.12) hold and $P_k(0) = 1$. Moreover the denominator of P_k is $H_k^{(1)}$, the determinant of the system (1.7). We shall now assume that $\forall k$, $H_k^{(1)} \neq 0$, which implies that all the polynomials P_k (and $P_k^{(1)}$ below) exist for all k. If, for some k, $H_k^{(1)} = 0$, then P_k does not exist and a breakdown occurs in the algorithm. It is not our purpose here to study this case in detail but we shall say a few words on it in Section 5.

Let us now consider the monic polynomials $P_k^{(1)}$ defined by

$$P_k^{(1)}(\xi) = \frac{\begin{vmatrix} c_1 & c_2 & \dots & c_{k+1} \\ \vdots & \vdots & & \vdots \\ c_k & c_{k+1} & \dots & c_{2k} \\ 1 & \xi & \dots & \xi^k \end{vmatrix}}{\begin{vmatrix} c_1 & \dots & c_k \\ \vdots & & \vdots \\ c_k & \dots & c_{2k-1} \end{vmatrix}}. \tag{2.2}$$

$P_k^{(1)}$ exists under the same condition that $H_k^{(1)} \neq 0$. Moreover it satisfies

$$c\left(\xi^{i+1} P_k^{(1)}\right) = 0 \quad \text{for } i = 0, \dots, k-1, \tag{2.3}$$

since multiplying the last row in the numerator of (2.2) by ξ^{i+1} and applying c yields two identical rows. If we define the linear functional $c^{(1)}$ by

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

then the relations (2.3) become

$$c^{(1)}\left(\xi^i P_k^{(1)}\right) = 0 \quad \text{for } i = 0, \dots, k-1, \tag{2.4}$$

which shows that the polynomials $\{P_k^{(1)}\}$ form a family of formal orthogonal polynomials with respect to $c^{(1)}$. $\{P_k\}$ and $\{P_k^{(1)}\}$ are said to be adjacent families of formal orthogonal polynomials. Notice that the powers of ξ have been put as the first row in the numerator of (2.1) and as the last row in (2.2) for a sign reason.

Since the polynomials of both families are uniquely determined, it is easy to check by imposing (1.12) and (2.4) that it holds [13] that

$$P_{k+1}(\xi) = P_k(\xi) - \lambda_k \xi P_k^{(1)}(\xi) \tag{2.5}$$

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

Let us now show how to obtain an expression for λ_k. Let $\{U_i\}$ be an arbitrary family of polynomials such that $\forall i$, U_i has exactly the degree i. Multiplying both sides of (2.5) by U_k, applying the functional c and making use of (1.12) and (2.4) we immediately get

$$\lambda_k = c(U_k P_k)/c\big(\xi U_k P_k^{(1)}\big). \tag{2.6}$$

Let us recall that, with the assumptions $H_k^{(1)} \neq 0$, $P_k^{(1)}$ has exactly degree k. Thus P_{k+1} has exactly degree $k+1$ if and only if $\lambda_k \neq 0$. When $\lambda_k = 0$, then P_k and P_{k+1} are identical. Moreover, with this assumption, the denominator in (2.6) does not vanish.

It is well known that a family of monic orthogonal polynomials, where $P_k^{(1)}$ has the degree k (which is true since $H_k^{(1)} \neq 0$), satisfies a three-term recurrence relationship. Thus we have

$$P_k^{(1)}(\xi) = (\xi - a_k) P_{k-1}^{(1)}(\xi) - b_k P_{k-2}^{(1)}(\xi) \tag{2.7}$$

with $P_0^{(1)}(\xi) = 1$ and $P_{-1}^{(1)}(\xi) = 0$.

As proved in [9], using again an auxiliary family $\{U_i\}$ we have, by a similar argument as above,

$$b_k = c\big(\xi^2 U_{k-2} P_{k-1}^{(1)}\big)/c\big(\xi U_{k-2} P_{k-2}^{(1)}\big) \tag{2.8}$$

and

$$a_k = \big[c\big(\xi^2 U_{k-1} P_{k-1}^{(1)}\big) - b_k c\big(\xi U_{k-1} P_{k-2}^{(1)}\big)\big]/c\big(\xi U_{k-1} P_{k-1}^{(1)}\big). \tag{2.9}$$

For example, if we choose $U_k = P_k^{(1)}$, we recover the usual formula

$$a_k = c\big(\xi^2 P_{k-1}^{(1)^2}\big)/c\big(\xi P_{k-1}^{(1)^2}\big). \tag{2.10}$$

We shall now study the particular case where P_k has exactly degree k, that is, by (2.1), when the Hankel determinant

$$H_k^{(0)} = \begin{vmatrix} c_0 & \cdots & c_{k-1} \\ \vdots & & \vdots \\ c_{k-1} & \cdots & c_{2k-2} \end{vmatrix}$$

is different from zero for all k. Moreover, we still assume that $H_k^{(1)} \neq 0$.

In that case, the family $\{P_k\}$ also satisfies a three-term recurrence relationship which can be written as

$$P_{k+1}(\xi) = -\eta_k\big[(\xi - \gamma_k) P_k(\xi) - \delta_k P_{k-1}(\xi)\big] \tag{2.11}$$

with $P_0(\xi) = 1$, $P_{-1}(\xi) = 0$, and $\delta_0 = 0$.

Since $P_k(0) = 1$, we must have

$$\eta_k = 1/(\gamma_k + \delta_k). \tag{2.12}$$

Using again an auxiliary family $\{U_i\}$, as above, we have

$$c(U_{k-1} P_{k+1}) = 0 = -\eta_k\big[c(\xi U_{k-1} P_k) - \delta_k c(U_{k-1} P_{k-1})\big]$$

and thus

$$\delta_k = c(\xi U_{k-1} P_k)/c(U_{k-1}P_{k-1}). \tag{2.13}$$

Similarly

$$c(U_k P_{k+1}) = 0 = -\eta_k\left[c(\xi U_k P_k) - \gamma_k c(U_k P_k) - \delta_k c(U_k P_{k-1})\right]$$

and thus we obtain

$$\gamma_k = \left[c(\xi U_k P_k) - \delta_k c(U_k P_{k-1})\right]/c(U_k P_k). \tag{2.14}$$

Again, for the particular choice $U_k = P_k$, we recover the usual formulae [6, p. 46].

Let us set

$$Q_k(\xi) = (-1)^k H_k^{(0)} P_k^{(1)}(\xi)/H_k^{(1)}. \tag{2.15}$$

Since P_k and $P_k^{(1)}$ have both exactly degree k, the coefficients of ξ^k in P_k and Q_k are the same and Q_k is proportional to $P_k^{(1)}$. Thus Q_k has the form

$$Q_k(\xi) = \frac{\begin{vmatrix} 1 & \xi & \dots & \xi^k \\ c_1 & c_2 & \dots & c_{k+1} \\ \vdots & \vdots & & \vdots \\ c_k & c_{k+1} & \dots & c_{2k} \end{vmatrix}}{\begin{vmatrix} c_1 & \dots & c_k \\ \vdots & & \vdots \\ c_k & \dots & c_{2k-1} \end{vmatrix}} \frac{H_k^{(0)}}{H_k^{(1)}}. \tag{2.16}$$

Moreover

$$c(\xi^{i+1} Q_k) = 0 \quad \text{for } i = 0, \dots, k-1. \tag{2.17}$$

By using the same uniqueness argument as above we can write

$$Q_k(\xi) = P_k(\xi) + \alpha_k Q_{k-1}(\xi) \tag{2.18}$$

and

$$P_{k+1}(\xi) = P_k(\xi) - \beta_k \xi Q_k(\xi) \tag{2.19}$$

with

$$\alpha_k = -c(\xi U_{k-1} P_k)/c(\xi U_{k-1} Q_{k-1}) \tag{2.20}$$

and

$$\beta_k = c(U_k P_k)/c(\xi U_k Q_k). \tag{2.21}$$

Let us remark that (2.19) can be obtained from (2.5) by replacing $P_k^{(1)}$ by Q_k and with $\beta_k = (-1)^k \lambda_k H_k^{(1)}/H_k^{(0)}$.

3. Lanczos-type algorithms

We shall now examine the various possibilities for computing recursively the vectors r_k defined by $r_k = b - Ax_k = P_k(A)r_0$. They give rise to the different algorithms which are known, and some new ones can also be obtained as we shall see below.

It is not our purpose here to compare these algorithms from the point of view of their efficiency, their stability, ...; all these questions are now under consideration [2]. Our purpose is to show clearly how to obtain the various known algorithms as a by-product of the theory of orthogonal polynomials, to obtain variants of them by choosing the arbitrary family of polynomials $\{U_k\}$, and to derive new algorithms by using other recurrence relations for computing orthogonal polynomials.

3.1. The Lanczos / Orthodir algorithm

Let us set

$$z_k = P_k^{(1)}(A)r_0. \tag{3.1}$$

From (2.5) we immediately obtain

$$r_{k+1} = r_k - \lambda_k A z_k. \tag{3.2}$$

Using $r_k = b - Ax_k$, this relation gives

$$x_{k+1} = x_k + \lambda_k z_k. \tag{3.3}$$

From (2.7) we have

$$z_k = Az_{k-1} - a_k z_{k-1} - b_k z_{k-2} \tag{3.4}$$

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

The method called Lanczos/Orthodir is defined by the formulae (3.2), (3.3), and (3.4) (see [33,49]).

Let us now see how to compute the coefficients λ_k, a_k, and b_k appearing in these formulae or, in other words, how to choose the auxiliary polynomials $\{U_i\}$.

The simplest choice consists in taking

$$U_k(\xi) = \xi^k.$$

From (2.6) we have

$$\lambda_k = c(\xi^k P_k)/c(\xi^{k+1}P_k^{(1)}). \tag{3.5}$$

But

$$c(\xi^k P_k) = \left(A^{*k}y,\ P_k(A)r_0\right)$$

and

$$c(\xi^{k+1}P_k^{(1)}) = \left(A^{*k+1}y,\ P_k^{(1)}(A)r_0\right),$$

and we finally obtain

$$\lambda_k = \left(A^{*^k} y,\ r_k\right) / \left(A^{*^{k+1}} y,\ z_k\right). \tag{3.6}$$

Similarly, (2.8) and (2.9) give

$$b_k = \left(A^{*^k} y,\ z_{k-1}\right) / \left(A^{*^{k-1}} y,\ z_{k-2}\right) \tag{3.7}$$

and

$$a_k = \left[\left(A^{*^k} y,\ Az_{k-1}\right) - b_k\left(A^{*^{k-1}} y,\ Az_{k-2}\right)\right] / \left(A^{*^k} y,\ z_{k-1}\right). \tag{3.8}$$

Gathering together all these formulae, we finally obtain the following new algorithm:

Step 1. Choose x_0 and y. Set

$$r_0 = b - Ax_0, \qquad z_0 = r_0, \qquad y_0 = y.$$

Step 2. For $k = 0, 1, \ldots$ compute

$$\lambda_k = (y_k, r_k)/(y_k, Az_k),$$
$$x_{k+1} = x_k + \lambda_k z_k,$$
$$r_{k+1} = r_k - \lambda_k Az_k.$$

Step 3. If $r_{k+1} \neq 0$, then compute

$$y_{k+1} = A^* y_k,$$
$$b_{k+1} = (y_k, Az_k)/(y_k, Az_{k-1}), \quad \text{if } k > 0 \text{ and } b_1 = 0,$$
$$a_{k+1} = [(y_{k+1}, Az_k) - b_{k+1}(y_{k+1}, z_{k-1})]/(y_k, Az_k),$$
$$z_{k+1} = Az_k - a_{k+1} z_k - b_{k+1} z_{k-1}.$$

Obviously, (y_k, Az_k) in the denominator of a_{k+1} could be replaced by (y_{k+1}, z_k) but the algorithm will require one more inner product.

Now let us make the choice

$$U_k(\xi) = P_k^{(1)}(\xi).$$

From (2.6) we have

$$\lambda_k = c\left(P_k^{(1)} P_k\right) / c\left(\xi P_k^{(1)^2}\right).$$

Thus if we set

$$\tilde{z}_k = \bar{P}_k^{(1)}(A^*) y = P_k^{(1)}(A)^* y, \tag{3.9}$$

we obtain

$$\lambda_k = (\tilde{z}_k, r_k)/(\tilde{z}_k, Az_k). \tag{3.10}$$

From (2.10), we have

$$a_k = \frac{c\left(\xi^2 P_{k-1}^{(1)^2}\right)}{c\left(\xi P_{k-1}^{(1)^2}\right)} = \frac{(A^* \tilde{z}_{k-1},\ Az_{k-1})}{(\tilde{z}_{k-1},\ Az_{k-1})} \tag{3.11}$$

and then

$$b_k = \frac{c\left(\xi^2 P_{k-2}^{(1)} P_{k-1}^{(1)}\right)}{c\left(\xi P_{k-2}^{(1)^2}\right)} = \frac{\left(A^* \tilde{z}_{k-2},\, Az_{k-1}\right)}{\left(\tilde{z}_{k-2},\, Az_{k-2}\right)}. \tag{3.12}$$

But $c(\xi^2 P_{k-2}^{(1)} P_{k-1}^{(1)}) = c(\xi P_{k-1}^{(1)^2})$ by the orthogonality of $P_{k-1}^{(1)}$ with respect to $c^{(1)}$ to any polynomial of degree strictly less than $k-1$, and thus

$$b_k = (\tilde{z}_{k-1},\, Az_{k-1})/(\tilde{z}_{k-2},\, Az_{k-2}). \tag{3.13}$$

Now, from (3.9) and (2.7), we obtain

$$\tilde{z}_k = A^* \tilde{z}_{k-1} - \bar{a}_k \tilde{z}_{k-1} - \bar{b}_k \tilde{z}_{k-2}. \tag{3.14}$$

If we set

$$\tilde{r}_k = P_k(A)^* r_0, \tag{3.15}$$

then we have, from (2.5),

$$\tilde{r}_{k+1} = \tilde{r}_k - \bar{\lambda}_k A^* \tilde{z}_k. \tag{3.16}$$

If we gather all these formulae together, we obtain the following algorithm known under the names of Lanczos/Orthodir [49] and of BIODIR [29].

Step 1. Choose x_0 and y. Set

$$r_0 = b - Ax_0, \qquad z_0 = r_0, \qquad \tilde{r}_0 = \tilde{z}_0 = y.$$

Step 2. For $k = 0, 1, \ldots$ compute

$$\lambda_k = (\tilde{z}_k,\, r_k)/(\tilde{z}_k,\, Az_k),$$
$$x_{k+1} = x_k + \lambda_k z_k,$$
$$r_{k+1} = r_k - \lambda_k Az_k.$$

Step 3. If $r_{k+1} \neq 0$, then compute

$$b_{k+1} = (\tilde{z}_k,\, Az_k)/(\tilde{z}_{k-1},\, Az_{k-1}), \quad \text{if } k > 0 \text{ and } b_1 = 0,$$
$$a_{k+1} = (A^* \tilde{z}_k,\, Az_k)/(\tilde{z}_k,\, Az_k),$$
$$z_{k+1} = Az_k - a_{k+1} z_k - b_{k+1} z_{k-1},$$
$$\tilde{z}_{k+1} = A^* \tilde{z}_k - \bar{a}_{k+1} \tilde{z}_k - \bar{b}_{k+1} \tilde{z}_{k-1}.$$

The terminology for these methods follows [19,29,33,49].

Remark 3.1. The vectors $\tilde{r}_k$ are not used in the algorithm and thus their computation was skipped. However, let us remark that, if $y = r_0$, then $c^{(1)}(P_i^{(1)} P_k^{(1)}) = (\tilde{z}_i,\, Az_k) = 0$ for $i \neq k$, which is the usual conjugacy property whose proof is now obvious via the theory of orthogonal polynomials.

Remark 3.2. This algorithm makes use of two auxiliary vectors and needs one more multiplication of a matrix by a vector and one more scalar product by iteration than the preceding algorithm. It does not seem possible to avoid using A^* in this algorithm.

Remark 3.3. This algorithm is defined (that is, x_{k+1} exists) only if $P_k^{(1)}$ has exactly degree k or, in other words, if $\forall k$, $H_k^{(1)} \neq 0$. If this condition is not satisfied, it is possible to jump over the non-existing polynomials $P_k^{(1)}$ and to use only those which exist. Thus we obtain a generalization of BIODIR without breakdown. This algorithm, called the MRZ (Method of Recursive Zoom), was given in [13]. Related algorithms can also be found in [28]. They will be briefly described in Section 5.

3.2. The Lanczos / Orthores algorithm

Let us now assume that P_k has exactly degree k. Thus the three-term recurrence relationship (2.11) holds and by (1.9) we immediately obtain

$$r_{k+1} = -\eta_k(Ar_k - \gamma_k r_k - \delta_k r_{k-1}). \tag{3.17}$$

Using $r_k = b - Ax_k$, (2.12), and (3.17), we obtain

$$x_{k+1} = \eta_k(r_k + \gamma_k x_k + \delta_k x_{k-1}). \tag{3.18}$$

Formulae (3.17) and (3.18) define the method known under the name of Lanczos/Orthores (see [33]).

Let us now see how to compute the coefficients η_k, γ_k, and δ_k appearing in these formulae. There are several possibilities according to the choice of the auxiliary polynomials $\{U_i\}$.

Let us start again by taking

$$U_k(\xi) = \xi^k.$$

We have

$$\delta_k = c(\xi^k P_k)/c(\xi^{k-1} P_{k-1})$$

and by (1.9) and (1.10) it holds that

$$\delta_k = \left(A^{*^k} y,\, r_k\right)/\left(A^{*^{k-1}} y,\, r_{k-1}\right). \tag{3.19}$$

Similarly, from (2.14), we have

$$\gamma_k = \left[c(\xi^{k+1} P_k) - \delta_k c(\xi^k P_{k-1})\right]/c(\xi^k P_k),$$

that is

$$\gamma_k = \left[\left(A^{*^k} y,\, Ar_k\right) - \delta_k\left(A^{*^{k-1}} y,\, Ar_{k-1}\right)\right]/\left(A^{*^k} y,\, r_k\right). \tag{3.20}$$

Gathering all these formulae together, we have the following new algorithm:

Step 1. Choose x_0 and y. Set

$$r_0 = b - Ax_0, \qquad y_0 = y.$$

Step 2. For $k = 0, 1, \ldots$ compute

$$\delta_k = (y_k, r_k)/(y_{k-1}, r_{k-1}), \quad \text{if } k > 0 \text{ and } \delta_0 = 0,$$
$$\gamma_k = [(y_k, Ar_k) - \delta_k(y_{k-1}, Ar_{k-1})]/(y_k, r_k),$$
$$\eta_k = 1/(\gamma_k + \delta_k),$$
$$x_{k+1} = \eta_k(r_k + \gamma_k x_k + \delta_k x_{k-1}),$$
$$r_{k+1} = -\eta_k(Ar_k - \gamma_k r_k - \delta_k r_{k-1}).$$

Step 3. If $r_{k+1} \neq 0$, then compute

$$y_{k+1} = A^* y_k.$$

Since P_k has exactly degree k, let us now make the choice

$$U_k(\xi) = P_k(\xi).$$

In that case

$$\delta_k = c(\xi P_{k-1} P_k)/c(P_{k-1}^2).$$

But, multiplying both sides of (2.11) by P_{k+1} and applying c, we obtain

$$c(P_{k+1}^2) = -\eta_k c(\xi P_k P_{k+1}),$$

and thus

$$\delta_k = -\frac{1}{\eta_{k-1}} \frac{c(P_k^2)}{c(P_{k-1}^2)}.$$

If we set

$$\tilde{\rho}_k = \bar{P}_k(A^*)y, \tag{3.21}$$

then

$$\delta_k = -\frac{1}{\eta_{k-1}} \frac{(\tilde{\rho}_k, r_k)}{(\tilde{\rho}_{k-1}, r_{k-1})}. \tag{3.22}$$

We also have, since $c(P_{k-1}P_k) = 0$,

$$\gamma_k = c(\xi P_k^2)/c(P_k^2),$$

that is

$$\gamma_k = (\tilde{\rho}_k, Ar_k)/(\tilde{\rho}_k, r_k). \tag{3.23}$$

Finally (2.11) gives us

$$\tilde{\rho}_{k+1} = -\bar{\eta}_k\left(A^*\tilde{\rho}_k - \bar{\gamma}_k\tilde{\rho}_k - \bar{\delta}_k\tilde{\rho}_{k-1}\right). \tag{3.24}$$

Formulae (3.17) and (3.18) together with (3.22), (3.23), and (3.24) define the algorithm known under the names of Lanczos/Orthores [49] and of BIORES [29] which is as follows:

Step 1. Choose x_0 and y. Set

$$r_0 = b - Ax_0, \qquad \tilde{\rho}_0 = y.$$

Step 2. For $k = 0, 1, \ldots$ compute

$$\gamma_k = (\tilde{\rho}_k, Ar_k)/(\tilde{\rho}_k, r_k),$$

$$\delta_k = -\frac{1}{\eta_{k-1}} \frac{(\tilde{\rho}_k, r_k)}{(\tilde{\rho}_{k-1}, r_{k-1})}, \quad \text{if } k > 0 \text{ and } \eta_0 = 0,$$

$$\eta_k = 1/(\gamma_k + \delta_k),$$

$$x_{k+1} = \eta_k(r_k + \gamma_k x_k + \delta_k x_{k-1}),$$

$$r_{k+1} = -\eta_k(Ar_k - \gamma_k r_k - \delta_k r_{k-1}),$$

$$\tilde{\rho}_{k+1} = -\bar{\eta}_k\left(A^*\tilde{\rho}_k - \bar{\gamma}_k\tilde{\rho}_k - \bar{\delta}_k\tilde{\rho}_{k-1}\right).$$

Step 3. If $r_{k+1} \neq 0$, then continue.

Remark 3.4. The choice $U_k(\xi) = \xi^k$ uses less storage and less vector operations than the BIORES algorithm.

Remark 3.5. The two algorithms described in this subsection need the supplementary assumption that P_k has exactly degree k. Thus, for all k, $H_k^{(0)}$ and $H_k^{(1)}$ have both to be different from zero. The supplementary assumption $H_k^{(0)} \neq 0$ is needed by the procedure used in the recursive computation but it is, in fact, an unnecessary assumption in the theory of the Lanczos method and it can be a supplementary reason for a breakdown. The remedy will be indicated in Section 5.

3.3. The Lanczos / Orthomin algorithm

Instead of using the polynomials $P_k^{(1)}$ as in the Lanczos/Orthores method, we shall make use of the polynomials Q_k defined by (2.16). Thus P_k is again assumed to have exactly degree k that is $H_k^{(0)} \neq 0$. We set

$$p_k = Q_k(A)r_0. \tag{3.25}$$

From (2.18) and (2.19) we have

$$p_k = r_k + \alpha_k p_{k-1} \tag{3.26}$$

and

$$r_{k+1} = r_k - \beta_k A p_k. \tag{3.27}$$

Using the definition of r_k, (3.27) gives

$$x_{k+1} = x_k + \beta_k p_k. \tag{3.28}$$

Formulae (3.26), (3.27), and (3.28) define the method known under the name of Lanczos/Orthomin (see [49]). This method is due to Vinsome [47].

Let us now see how to compute the coefficients α_k and β_k appearing in these formulae. (Let us notice that the roles of these two coefficients are usually reversed.)

We begin by the choice

$$U_k(\xi) = \xi^k.$$

Thus, from (2.20) and (2.21), we have

$$\alpha_k = -c(\xi^k P_k)/c(\xi^k Q_{k-1}),$$
$$\beta_k = c(\xi^k P_k)/c(\xi^{k+1} Q_k),$$

and it follows that

$$\alpha_k = -\left(A^{*^k} y,\, r_k\right)/\left(A^{*^k} y,\, p_{k-1}\right) \tag{3.29}$$

and

$$\beta_k = \left(A^{*^k} y,\, r_k\right)/\left(A^{*^{k+1}} y,\, p_k\right). \tag{3.30}$$

We finally obtain the following algorithm:

Step 1. Choose x_0 and y. Set

$$r_0 = b - Ax_0 = p_0, \qquad y_0 = y.$$

Step 2. For $k = 0, 1, \ldots$ compute

$$\beta_k = (y_k, r_k)/(y_k, Ap_k),$$
$$x_{k+1} = x_k + \beta_k p_k,$$
$$r_{k+1} = r_k - \beta_k A p_k.$$

Step 3. If $r_{k+1} \neq 0$, then compute

$$y_{k+1} = A^* y_k,$$
$$\alpha_{k+1} = -(y_{k+1}, r_{k+1})/(y_k, Ap_k),$$
$$p_{k+1} = r_{k+1} + \alpha_{k+1} p_k.$$

Let us now choose

$$U_k(\xi) = P_k(\xi).$$

In that case

$$\beta_k = c(P_k^2)/c(\xi P_k Q_k).$$

But, since the coefficients of ξ^k in P_k and in Q_k are the same we have

$$P_k(\xi) = Q_k(\xi) + p(\xi),$$

where p is a polynomial of degree $k-1$ at most. Thus

$$c(\xi P_k Q_k) = c(\xi Q_k^2) + c(\xi p Q_k).$$

But, since Q_k is proportional to $P_k^{(1)}$, we have, by the orthogonality property of $P_k^{(1)}$, $c(\xi p Q_k) = 0$ and thus

$$\beta_k = c(P_k^2)/c(\xi Q_k^2).$$

Setting

$$\tilde{r}_k = P_k(A)^* y, \tag{3.31}$$

we obtain

$$\beta_k = (\tilde{r}_k, r_k)/(\tilde{p}_k, Ap_k). \tag{3.32}$$

From (2.20) we have

$$\alpha_k = -c(\xi P_{k-1} P_k)/c(\xi P_{k-1} Q_{k-1}).$$

But $\beta_{k-1} = c(P_{k-1}^2)/c(\xi P_{k-1} Q_{k-1})$ and thus

$$\alpha_k = -\beta_{k-1} c(\xi P_{k-1} P_k)/c(P_{k-1}^2). \tag{3.33}$$

On the other hand, writing (2.19) for the index k, multiplying it by P_k, and applying c gives

$$c(P_k^2) = -\beta_{k-1} c(\xi P_k Q_{k-1}).$$

Moreover, from (2.18),

$$c(\xi P_k Q_{k-1}) = c(\xi P_k P_{k-1}) + \alpha_{k-1} c(\xi P_k Q_{k-2}).$$

Since P_k is orthogonal to ξQ_{k-2}, which is a polynomial of degree $k-1$, it follows that

$$c(P_k^2) = -\beta_{k-1} c(\xi P_k P_{k-1}).$$

Thus, by (3.33), we finally obtain

$$\alpha_k = c(P_k^2)/c(P_{k-1}^2), \tag{3.34}$$

that is

$$\alpha_k = (\tilde{r}_k, r_k)/(\tilde{r}_{k-1}, r_{k-1}). \tag{3.35}$$

If we set

$$\tilde{p}_k = Q_k(A)^* y, \tag{3.36}$$

we obtain from (2.19)

$$\tilde{r}_{k+1} = \tilde{r}_k - \bar{\beta}_k A^* \tilde{p}_k, \tag{3.37}$$

and from (2.18)

$$\tilde{p}_k = \tilde{r}_k + \bar{\alpha}_k \tilde{p}_{k-1}. \tag{3.38}$$

These formulae define the so-called biconjugate gradient method (BCG) due to Lanczos [38,39] but popularized by Fletcher [21]. This algorithm, also known under the names of Lanczos/Orthomin [49] and of BIOMIN [29], is as follows:

Step 1. Choose x_0 and y. Set

$$r_0 = b - Ax_0 = p_0, \qquad \tilde{r}_0 = \tilde{p}_0 = y.$$

Step 2. For $k = 0, 1, \ldots$ compute

$$\beta_k = (\tilde{r}_k, r_k)/(\tilde{p}_k, Ap_k),$$
$$x_{k+1} = x_k + \beta_k p_k,$$
$$r_{k+1} = r_k - \beta_k Ap_k,$$
$$\tilde{r}_{k+1} = \tilde{r}_k - \bar{\beta}_k A^* \tilde{p}_k.$$

Step 3. If $r_{k+1} \neq 0$, then compute

$$\alpha_{k+1} = (\tilde{r}_{k+1}, r_{k+1})/(\tilde{r}_k, r_k),$$
$$p_{k+1} = r_{k+1} + \alpha_{k+1} p_k,$$
$$\tilde{p}_{k+1} = \tilde{r}_{k+1} + \bar{\alpha}_{k+1} \tilde{p}_k.$$

For this algorithm, Remarks 3.4 and 3.5 above are still valid. Moreover if $A^* = A$ and $y = r_0$, it reduces to the conjugate gradient method of Hestenes and Stiefel [32].

For the vectors p_k defined by (3.25), $\tilde{p}_k$ defined by (3.36), r_k defined by (1.9), and $\tilde{r}_k$ defined by (3.31), the following theorem holds:

Theorem 3.1.

$$\forall i \neq k \quad (\tilde{r}_k, r_i) = 0,$$
$$\forall i \neq k \quad (\tilde{p}_k, Ap_i) = 0,$$
$$\forall i < k \quad (\tilde{p}_i, r_k) = (\tilde{r}_k, p_i) = 0,$$
$$\forall i < k \quad (\tilde{r}_i, Ap_k) = (\tilde{p}_k, Ar_i) = 0.$$

Proof. By the orthogonality of the families $\{P_k\}$ and $\{P_k^{(1)}\}$ we have for $i \neq k$

$$c(P_k P_i) = (y, P_k(A)P_i(A)r_0) = 0$$

and

$$c^{(1)}(P_k^{(1)}P_i^{(1)}) = (y, AP_k^{(1)}(A)P_i^{(1)}(A)r_0) = 0,$$

which proves the first two results since Q_k is proportional to $P_k^{(1)}$ by (2.15).

We also have for $i < k$

$$c(P_i^{(1)}P_k) = (y, P_i^{(1)}(A)P_k(A)r_0) = 0$$

and

$$c^{(1)}(P_i P_k^{(1)}) = (y, AP_i(A)P_k^{(1)}(A)r_0) = 0$$

which proves the two other results. □

As we see, the proofs of these results are greatly simplified by the theory of formal orthogonal polynomials. The first result is similar to relation (5.2) of Fletcher [21] and shows the biorthogonality of the families $\{r_k\}$ and $\{\tilde{r}_k\}$. The second result is similar to relation (5.3) of Fletcher [21] and shows the biconjugacy of $\{p_k\}$ and $\{\tilde{p}_k\}$. The third result is relation (5.7) of Fletcher [21] while the fourth result seems to be new. For another proof, see [1].

Obviously these results also hold for the other algorithms since they are independent of the recurrence relations used for computing the sequences of vectors $\{r_k = P_k(A)r_0\}$, $\{z_k = P_k^{(1)}(A)r_0\}$, $\{\tilde{r}_k = P_k(A)^* y\}$, and $\{\tilde{z}_k = P_k^{(1)}(A)^* y\}$. Thus, in the preceding theorem, p_k can be replaced by z_k and $\tilde{p}_k$ by $\tilde{z}_k$. They are also valid if the corresponding vectors are computed by one of the schemes given in the next section.

3.4. Other methods

In the preceding algorithms, two recurrence relationships were needed: one for P_{k+1} and one for $P_{k+1}^{(1)}$. Using the notation $z \leftarrow u,v$ for indicating that z is computed from u and v, we have for

- Lanczos/Orthodir:

$$P_{k+1} \leftarrow P_k,\ P_k^{(1)},$$

$$P_{k+1}^{(1)} \leftarrow P_k^{(1)},\ P_{k-1}^{(1)};$$

- Lanczos/Orthores:

$$P_{k+1} \leftarrow P_k,\ P_{k-1};$$

- Lanczos/Orthomin:

$$P_{k+1} \leftarrow P_k,\ Q_k,$$

$$Q_{k+1} \leftarrow P_{k+1},\ Q_k \quad \text{with } Q_k \text{ proportional to } P_k^{(1)}.$$

Thus all these algorithms for implementing Lanczos method are characterized by the choice of one or two recurrence relationships. We shall now see that other choices are also possible, thus leading to other (new) algorithms. We have Table 1 (the polynomials Q_k are absent since they are proportional to the polynomials $P_k^{(1)}$). Thus, from the first row of Table 1, we see that P_{k+1} can be computed from P_{k-1} and $P_{k-1}^{(1)}$, a formula called (A1), and that $P_{k+1}^{(1)}$ can be computed from P_{k-1} and $P_{k-1}^{(1)}$, a formula called (B1). And so on for the other rows of the table.

We shall combine these formulae to obtain Lanczos-type algorithms denoted by (A_i/B_j) (or by (B_j/A_i) since for some formulae $P_{k+1}^{(1)}$ has to be computed before P_{k+1}). Thus (A4) is Lanczos/Orthores, (A8/B6) is Lanczos/Orthodir, and (A8/B10) is Lanczos/Orthomin.

Among the previous list some formulae do not exist, for example (A9) and (A10). This is due to the fact that the orthogonality relations of both families cannot be satisfied by writing the polynomial to be computed under the form proposed and thus the corresponding formula cannot hold.

Table 1
Recurrence relationships

Computation of P_{k+1} from			Computation of $P_{k+1}^{(1)}$ from		
P_{k-1}	$P_{k-1}^{(1)}$	(A1)	P_{k-1}	$P_{k-1}^{(1)}$	(B1)
P_{k-1}	$P_k^{(1)}$	(A2)	P_{k-1}	$P_k^{(1)}$	(B2)
P_{k-1}	$P_{k+1}^{(1)}$	(A3)	P_{k-1}	P_{k+1}	(B3)
P_{k-1}	P_k	(A4)	P_{k-1}	P_k	(B4)
$P_{k-1}^{(1)}$	P_k	(A5)	$P_{k-1}^{(1)}$	P_k	(B5)
$P_{k-1}^{(1)}$	$P_k^{(1)}$	(A6)	$P_{k-1}^{(1)}$	$P_k^{(1)}$	(B6)
$P_{k-1}^{(1)}$	$P_{k+1}^{(1)}$	(A7)	$P_{k-1}^{(1)}$	P_{k+1}	(B7)
P_k	$P_k^{(1)}$	(A8)	P_k	$P_k^{(1)}$	(B8)
P_k	$P_{k+1}^{(1)}$	(A9)	P_k	P_{k+1}	(B9)
$P_k^{(1)}$	$P_{k+1}^{(1)}$	(A10)	$P_k^{(1)}$	P_{k+1}	(B10)

Some combinations (A_i/B_j) are impossible. They are (A3), (A7), (A9), and (A10) with (B3), (B7), (B9), and (B10) and conversely. This is due to the fact that, for example in (A3/B3), $P_{k+1}^{(1)}$ is needed in the computation of P_{k+1} and P_{k+1} is necessary for computing $P_{k+1}^{(1)}$ which is impossible.

It is not our purpose here to study all these formulae and all the possible combinations (this is the purpose of [2]) but we shall only give an example of a new method. It corresponds to (A8/B8).

Since $P_{k-1}^{(1)}$ and $P_k^{(1)}$ are monic polynomials, (B8) has the form

$$P_k^{(1)}(\xi) = (\xi + \delta_k)P_{k-1}^{(1)}(\xi) + \gamma_k P_{k-1}(\xi) \tag{3.39}$$

with

$$\gamma_k = -c^{(1)}\big(\xi^{k-1}P_{k-1}^{(1)}\big)/c\big(\xi^{k-1}P_{k-1}\big), \qquad \gamma_1 = -c^{(1)}\big(\xi P_0^{(1)}\big)/c(\xi P_0)$$

and

$$\delta_k = \frac{c\big(\xi^k P_{k-1}\big)}{c\big(\xi^{k-1}P_{k-1}\big)} - \frac{c^{(1)}\big(\xi^k P_{k-1}^{(1)}\big)}{c^{(1)}\big(\xi^{k-1}P_{k-1}^{(1)}\big)}, \qquad \delta_1 = 0.$$

Setting

$$z_k = P_k^{(1)}(A)r_0$$

and using (2.5), which has the form (A8), we obtain the following algorithm:

Step 1. Choose x_0 and y. Set

$$r_0 = b - Ax_0, \qquad y_0 = z_0 = y.$$

Step 2. For $k = 0, 1, \ldots$ compute

$$\lambda_k = (y_k, r_k)/(y_k, Az_k),$$
$$x_{k+1} = x_k + \lambda_k z_k,$$
$$r_{k+1} = r_k - \lambda_k Az_k.$$

Step 3. If $r_{k+1} \neq 0$, then compute

$$y_{k+1} = A^* y_k,$$
$$\gamma_{k+1} = -(y_{k+1}, z_k)/(y_k, r_k) = -(y_k, Az_k)/(y_k, r_k),$$
$$\delta_{k+1} = (y_{k+1}, r_k)/(y_k, r_k) - (y_{k+1}, Az_k)/(y_k, Az_k),$$
$$z_{k+1} = (A + \delta_{k+1})z_k + \gamma_{k+1}r_k.$$

As it was the case for the other algorithms, the coefficients λ_k and γ_k can be computed in a different way. Setting

$$\tilde{z}_k = P_k^{(1)}(A)^* y, \tag{3.40}$$
$$\tilde{r}_k = P_k(A)^* y, \tag{3.41}$$

we already saw that λ_k can be obtained by (3.10). Similarly we have

$$\gamma_k = -c\big(\xi^2 P_{k-1}^{(1)} P_{k-2}^{(1)}\big)/c\big(\xi P_{k-1} P_{k-2}^{(1)}\big) = -(A^*\tilde{z}_{k-2}, Az_{k-1})/(A^*\tilde{z}_{k-2}, r_{k-1}),$$
$$\gamma_1 = -(A^*\tilde{z}_0, Az_0)/(A^*z_0, r_0)$$

and

$$\delta_k = \big[c\big(\xi^2 P_{k-1}^{(1)^2}\big) + \gamma_k c\big(\xi P_{k-1} P_{k-1}^{(1)}\big)\big]/c\big(\xi P_{k-1}^{(1)^2}\big)$$
$$= \big[(A^*\tilde{z}_{k-1}, Az_{k-1}) + \gamma_k (A^*z_{k-1}, r_{k-1})\big]/(A^*\tilde{z}_{k-1}, z_{k-1}).$$

Thus we have the following algorithm:

Step 1. Choose x_0 and y. Set

$$r_0 = b - Ax_0, \qquad \tilde{r}_0 = \tilde{z}_0 = z_0 = y.$$

Step 2. For $k = 0, 1, \ldots$ compute

$$\lambda_k = (\tilde{z}_k, r_k)/(\tilde{z}_k, Az_k),$$
$$x_{k+1} = x_k + \lambda_k z_k,$$
$$r_{k+1} = r_k - \lambda_k Az_k,$$
$$\tilde{r}_{k+1} = \tilde{r}_k - \bar{\lambda}_k A^*\tilde{z}_k.$$

Step 3. If $r_{k+1} \neq 0$, then compute

$$\gamma_{k+1} = -(A^*\tilde{z}_{k-1}, Az_k)/(A^*\tilde{z}_{k-1}, r_k) \quad \text{or} \quad \gamma_{k+1} = -(\tilde{z}_k, Az_k)/(\tilde{z}_k, r_k),$$
$$\gamma_1 = -(A^*\tilde{z}_0, Az_0/(A^*z_0, r_0),$$
$$\delta_{k+1} = \big[(A^*\tilde{z}_k, Az_k) + \gamma_{k+1}(A^*z_k, r_k)\big]/(A^*\tilde{z}_k, z_k), \qquad \delta_1 = 0,$$
$$z_{k+1} = (A + \delta_{k+1})z_k + \gamma_{k+1} r_k,$$
$$\tilde{z}_{k+1} = \big(A^* + \bar{\delta}_{k+1}\big)\tilde{z}_k + \bar{\gamma}_{k+1}\tilde{r}_k.$$

For this algorithm, Remarks 3.4 and 3.5 above are still valid. In particular, it can only be used if $c(\xi^{k-1}P_{k-1}) \neq 0$, that is $H_k^{(0)} \neq 0$ or, in other words, if P_k has exactly degree k. Moreover, if this algorithm is as stable as the BCG, then it will be interesting to construct an algorithm using these formulae in the case where P_k has exactly degree k and the formulae of BIODIR otherwise, as was done in [35].

Remark 3.6. Since $\gamma_{k+1} = -1/\lambda_k$, we have

$$r_{k+1} = r_k + \frac{1}{\gamma_{k+1}} Az_k,$$
$$\delta_{k+1} = -(A^*z_k, Az_k + \gamma_{k+1} r_k)/(\tilde{z}_k, Az_k) = (A^*z_k, r_{k+1})/(\tilde{z}_k, r_k),$$

and finally

$$z_{k+1} = -r_{k+1}/\lambda_k + \delta_{k+1} z_k, \qquad \tilde{z}_{k+1} = -\tilde{r}_{k+1}/\bar{\lambda}_k + \bar{\delta}_{k+1}\tilde{z}_k,$$

which is (B10). Thus this method is (A8/B10) which is similar to the biconjugate gradient

method since it uses the vectors z_k instead of the vectors p_k. When $\lambda_k = 0$, the formulae of Orthodir can be used thus avoiding the so-called soft-breakdown.

In this section we only considered two algorithmic variants, one based on $U_k(\xi) = \xi^k$ and another based on $U_k(\xi) = P_k(\xi)$ or $P_k^{(1)}(\xi)$. Obviously, from the theoretical point of view, any choice can be made. However, in practice, the computation of quantities as $c(U_k P_k)$ for example must be easily performed and thus it restricts our choice to polynomials U_k satisfying a recurrence relation (such as $\xi^{k+1} = \xi\xi^k$ or any family of orthogonal polynomials such as Chebyshev's for example) or to polynomials already involved in the algorithm such as P_k, $P_k^{(1)}$ or Q_k. This question is under consideration.

4. The topological ε-algorithm

It seems, a priori, that this section has no connection with the previous ones since Shanks' transformation [44], which is usually implemented via the ε-algorithm of Wynn [48], is an extrapolation method used to accelerate the convergence of scalar sequences [10]. But, as we shall see below, such a connection exists and it was extensively studied in [45] for example and the papers quoted herein. Let us begin by recalling some generalizations of Shanks' transformation and the ε-algorithm which can be found in [5].

Let (S_m) be a sequence of vectors of $\mathbb{C}^n$. We consider the following ratios of determinants which generalize Shanks' transformation

$$ {}_ie_k^{(m)} = \frac{\begin{vmatrix} S_{m+i} & \dots & S_{m+i+k} \\ (y, \Delta S_m) & \dots & (y, \Delta S_{m+k}) \\ \vdots & & \vdots \\ (y, \Delta S_{m+k-1}) & \dots & (y, \Delta S_{m+2k-1}) \end{vmatrix}}{\begin{vmatrix} 1 & \dots & 1 \\ (y, \Delta S_m) & \dots & (y, \Delta S_{m+k}) \\ \vdots & & \vdots \\ (y, \Delta S_{m+k-1}) & \dots & (y, \Delta S_{m+2k-1}) \end{vmatrix}}, \tag{4.1} $$

where $\Delta S_j = S_{j+1} - S_j$.

If $i = 0$, the vectors ${}_0e_k^{(m)}$ can be recursively computed by the so-called first topological ε-algorithm (denoted by TEA1 in the sequel) whose rules are as follows

$$ \begin{aligned} &\varepsilon_{-1}^{(m)} = 0, \qquad \varepsilon_0^{(m)} = S_m, && m = 0, 1, \dots, \\ &\varepsilon_{2k+1}^{(m)} = \varepsilon_{2k-1}^{(m+1)} + y/\big(y, \Delta\varepsilon_{2k}^{(m)}\big), && k, m = 0, 1, \dots, \\ &\varepsilon_{2k+2}^{(m)} = \varepsilon_{2k}^{(m+1)} + \Delta\varepsilon_{2k}^{(m)}/\big(\Delta\varepsilon_{2k+1}^{(m)}, \Delta\varepsilon_{2k}^{(m)}\big), && k, m = 0, 1, \dots, \end{aligned} \tag{4.2} $$

where $\Delta\varepsilon_k^{(m)} = \varepsilon_k^{(m+1)} - \varepsilon_k^{(m)}$.

We have, for all k and m,

$$ \varepsilon_{2k}^{(m)} = {}_0e_k^{(m)}. $$

If $i=k$, the vectors ${}_ke_k^{(m)}$ can be recursively computed by the so-called second topological ε-algorithm (denoted by TEA2 in the sequel) whose rules are

$$\begin{aligned}
&\varepsilon_{-1}^{(m)}=0, \qquad \varepsilon_0^{(m)}=S_m, && m=0,1,\ldots,\\
&\varepsilon_{2k+1}^{(m)}=\varepsilon_{2k-1}^{(m+1)}+y/\big(y,\Delta\varepsilon_{2k}^{(m)}\big), && k,m=0,1,\ldots,\\
&\varepsilon_{2k+2}^{(m)}=\varepsilon_{2k}^{(m+1)}+\Delta\varepsilon_{2k}^{(m+1)}/\big(\Delta\varepsilon_{2k+1}^{(m)},\Delta\varepsilon_{2k}^{(m+1)}\big), && k,m=0,1,\ldots.
\end{aligned} \tag{4.3}$$

We have, for all k and m,

$$\varepsilon_{2k}^{(m)}={}_ke_k^{(m)}.$$

Let us now study the link between these two algorithms and Lanczos method. We assume that the sequence (S_m) is generated by

$$S_{m+1}=(I+A)S_m-b, \quad m=0,1,\ldots, \tag{4.4}$$

where S_0 is an arbitrary vector.

Thus $\Delta S_{m+1}=(I+A)\Delta S_m$ and $\Delta^kS_m=A\Delta^{k-1}S_m$ since $\Delta S_m=AS_m-b$. It follows that

$$\Delta^kS_m=A^{k-1}\Delta S_m, \tag{4.5}$$

$$\Delta S_m=(I+A)^m\Delta S_0. \tag{4.6}$$

Let us take $m=0$ and set $x_k={}_0e_k^{(0)}$. Thus $x_0=S_0$ and $\Delta S_0=AS_0-b=-r_0$. Replacing, in (4.1), each column by its difference with the preceding one, we obtain

$$x_k=\frac{\begin{vmatrix} x_0 & \Delta S_0 & \ldots & \Delta^kS_0\\ (y,\Delta S_0) & (y,\Delta^2S_0) & \ldots & (y,\Delta^{k+1}S_0)\\ \vdots & \vdots & & \vdots\\ (y,\Delta^kS_0) & (y,\Delta^{k+1}S_0) & \ldots & (y,\Delta^{2k}S_0)\end{vmatrix}}{\begin{vmatrix} 1 & 0 & \ldots & 0\\ (y,\Delta S_0) & (y,\Delta^2S_0) & \ldots & (y,\Delta^{k+1}S_0)\\ \vdots & \vdots & & \vdots\\ (y,\Delta^kS_0) & (y,\Delta^{k+1}S_0) & \ldots & (y,\Delta^{2k}S_0)\end{vmatrix}}. \tag{4.7}$$

From (4.4), $\Delta^kS_0=A^{k-1}\Delta S_0=-A^{k-1}r_0$ and thus

$$x_k=\frac{\begin{vmatrix} x_0 & -r_0 & \ldots & -A^{k-1}r_0\\ (y,r_0) & (y,Ar_0) & \ldots & (y,A^kr_0)\\ \vdots & \vdots & & \vdots\\ (y,A^{k-1}r_0) & (y,A^kr_0) & \ldots & (y,A^{2k-1}r_0)\end{vmatrix}}{\begin{vmatrix} (y,Ar_0) & \ldots & (y,A^kr_0)\\ \vdots & & \vdots\\ (y,A^kr_0) & \ldots & (y,A^{2k-1}r_0)\end{vmatrix}}. \tag{4.8}$$

Comparing with the formula of Theorem 1.2, we see that the vectors ${}_0e_k^{(0)}$ are identical to the vectors obtained by the Lanczos method and that they can be recursively computed by the TEA1. This result was first proved in [6, pp. 84–91, 184–189]. A numerical comparison of Lanczos method, the BCG, and the TEA1 can be found in [26].

We shall now give a new recursive method for computing the vectors ${}_ke_k^{(m)}$ for a fixed value of m.

Let us now set

$$x_k = {}_ke_k^{(0)}, \qquad r_k = b - Ax_k.$$

From (4.1) we have

$$x_k = \frac{\begin{vmatrix} S_k & \Delta S_k & \dots & \Delta^k S_k \\ (y, \Delta S_0) & (y, \Delta^2 S_0) & \dots & (y, \Delta^{k+1} S_0) \\ \vdots & \vdots & & \vdots \\ (y, \Delta^k S_0) & (y, \Delta^{k+1} S_0) & \dots & (y, \Delta^{2k} S_0) \end{vmatrix}}{\begin{vmatrix} (y, \Delta^2 S_0) & \dots & (y, \Delta^{k+1} S_0) \\ \vdots & & \vdots \\ (y, \Delta^{k+1} S_0) & \dots & (y, \Delta^{2k} S_0) \end{vmatrix}}. \tag{4.9}$$

Using (4.5) and (4.6)

$$\Delta^j S_k = (I+A)^k A^{j-1} \Delta S_0 = -(I+A)^k A^{j-1} r_0.$$

Moreover $b - AS_k = -\Delta S_k = (I+A)^k r_0$ and thus we finally obtain

$$r_k = \frac{\begin{vmatrix} (I+A)^k r_0 & (I+A)^k A r_0 & \dots & (I+A)^k A^k r_0 \\ (y, r_0) & (y, Ar_0) & \dots & (y, A^k r_0) \\ \vdots & \vdots & & \vdots \\ (y, A^{k-1} r_0) & (y, A^k r_0) & \dots & (y, A^{2k-1} r_0) \end{vmatrix}}{\begin{vmatrix} (y, Ar_0) & \dots & (y, A^k r_0) \\ \vdots & & \vdots \\ (y, A^k r_0) & \dots & (y, A^{2k-1} r_0) \end{vmatrix}}, \tag{4.10}$$

that is

$$r_k = (I+A)^k P_k(A) r_0. \tag{4.11}$$

Thus, as in Lanczos-type algorithms, the polynomials P_k can be recursively computed which gives rise to recurrence relationships for the vectors r_k defined by (4.11). For illustrating such a possibility we shall give a new algorithm which will be called TEA2/Orthomin.

We set

$$\varphi_k(\xi) = (1+\xi)^k P_k(\xi), \tag{4.12}$$

$$\psi_k(\xi) = (1+\xi)^k Q_k(\xi). \tag{4.13}$$

Thus, from (2.19), we have

$$\varphi_{k+1}(\xi) = (1+\xi)[\varphi_k(\xi) - \beta_k \xi \psi_k(\xi)] \tag{4.14}$$

and, from (2.18), we obtain

$$\psi_k(\xi) = \varphi_k(\xi) + \alpha_k(1+\xi)\psi_{k-1}(\xi). \tag{4.15}$$

Setting

$$q_k = \psi_k(A) r_0 \tag{4.16}$$

(4.14) gives

$$r_{k+1} = (I+A)(r_k - \beta_k A q_k), \tag{4.17}$$

and (4.15)

$$q_k = r_k + \alpha_k(I+A)q_{k-1}. \tag{4.18}$$

Using the definition of r_k in (4.14), we obtain

$$x_{k+1} = x_k - r_k - \beta_k(I+A)q_k. \tag{4.19}$$

Let us now see how to compute the coefficients α_k and β_k. From (2.20) we have, for the choice $U_k(\xi) = \xi^k$ and thanks to the orthogonality of P_k to every polynomial of degree strictly less than k

$$\alpha_k = -c(\xi^k P_k)/c(\xi^k Q_{k-1}) = -c\big((1+\xi)^k P_k\big)/c\big(\xi(1+\xi)^{k-1} Q_{k-1}\big),$$

that is

$$\alpha_k = -(y, r_k)/(y, Aq_{k-1}). \tag{4.20}$$

Similarly we obtain from (2.21)

$$\beta_k = c(\xi^k P_k)/c(\xi^{k+1} Q_k) = c\big((1+\xi)^k P_k\big)/c\big(\xi(1+\xi)^k Q_k\big),$$

that is

$$\beta_k = (y, r_k)/(y, Aq_k). \tag{4.21}$$

Gathering all these formulae together we finally obtain the following algorithm:

Step 1. Choose x_0 and y. Set

$$r_0 = b - Ax_0, \qquad q_0 = y.$$

Step 2. For $k = 0, 1, \ldots$ compute

$$\beta_k = (y, r_k)/(y, Aq_k),$$

$$x_{k+1} = x_k - r_k - \beta_k(I+A)q_k,$$

$$r_{k+1} = (I+A)(r_k - \beta_k A q_k).$$

Step 3. If $r_{k+1} \neq 0$, then compute

$$\alpha_{k+1} = -(y, r_{k+1})/(y, Aq_k),$$
$$q_{k+1} = r_{k+1} + \alpha_{k+1}(I+A)q_k.$$

Remark 4.1. This algorithm only needs two vector operations by iteration, the same number as in the BCG, but they do not involve the matrix A^*.

Remark 4.2 Since r_k is given by (4.11), this method is better than Lanczos method if $\| I+A \| < 1$.

Remark 4.3. Using the same technique, it is easy to derive new algorithms of the Orthodir or Orthores type.

Let us now examine the relation between the conjugate gradient squared method (CGS) of Sonneveld [46] and the previous generalizations of Shanks' transformation. The CGS is defined by

$$r_k = P_k(A)^2 r_0. \tag{4.22}$$

The recursive computation of r_k was made possible by squaring the relations used in the BCG for the polynomials P_k. For this algorithm, no algebraic results are known and, in particular, no closed formula for the vectors x_k exists. We shall now give such a formula and prove that x_k is a barycentric combination of the vectors ${}_0e_k^{(0)}, \ldots, {}_ke_k^{(0)}$.

Let us first remark that

$$S_{k+1} = (I+A)S_k - b, \qquad \Delta S_{k+1} = (I+A)\,\Delta S_k.$$

Thus

$${}_{i+1}e_k^{(0)} = (I+A)\,{}_ie_k^{(0)} - b. \tag{4.23}$$

Let

$$\tilde{r}_k = P_k(A)r_0, \qquad \tilde{x}_k = x_0 - R_{k-1}(A)r_0$$

be the vectors obtained by Lanczos method where, as in Section 1, $P_k(\xi) = 1 + \xi R_{k-1}(\xi)$. Then, using (1.14), we obtain a determinantal formula for the vectors r_k given by (4.22) and it follows, by $r_k = b - Ax_k$, that

$$x_k = \frac{\begin{vmatrix} \tilde{x}_k & -\tilde{r}_k & \ldots & -A^{k-1}\tilde{r}_k \\ (y, r_0) & (y, Ar_0) & \ldots & (y, A^k r_0) \\ \vdots & \vdots & & \vdots \\ (y, A^{k-1}r_0) & (y, A^k r_0) & \ldots & (y, A^{2k-1}r_0) \end{vmatrix}}{\begin{vmatrix} (y, Ar_0) & \ldots & (y, A^k r_0) \\ \vdots & & \vdots \\ (y, A^k r_0) & \ldots & (y, A^{2k-1}r_0) \end{vmatrix}}.$$

Thus we proved:

Theorem 4.1. *Let $(\tilde{x}_k)$ be the vectors obtained by Lanczos method with $\tilde{x}_0 = x_0$ and let us set $\tilde{r}_k = b - A\tilde{x}_k$. Then the vectors (x_k) obtained by the CGS are given by*

$$x_k = \frac{\begin{vmatrix} \tilde{x}_k & -\tilde{r}_k & \dots & -A^{k-1}\tilde{r}_k \\ c_0 & c_1 & \dots & c_k \\ \vdots & \vdots & & \vdots \\ c_{k-1} & c_k & \dots & c_{2k-1} \end{vmatrix}}{\begin{vmatrix} c_1 & \dots & c_k \\ \vdots & & \vdots \\ c_k & \dots & c_{2k-1} \end{vmatrix}}$$

and by

$$x_k = \frac{\begin{vmatrix} {}_0e_k^{(0)} & {}_1e_k^{(0)} & \dots & {}_ke_k^{(0)} \\ (y, \Delta S_0) & (y, \Delta S_1) & \dots & (y, \Delta S_k) \\ \vdots & \vdots & & \vdots \\ (y, \Delta S_{k-1}) & (y, \Delta S_k) & \dots & (y, \Delta S_{2k-1}) \end{vmatrix}}{\begin{vmatrix} 1 & \dots & 1 \\ (y, \Delta S_0) & \dots & (y, \Delta S_k) \\ \vdots & & \vdots \\ (y, \Delta S_{k-1}) & \dots & (y, \Delta S_{2k-1}) \end{vmatrix}}.$$

The proof of this second identity is similar to that of (4.7) and (4.8) because (4.23) is similar to (4.4).

Remark 4.4. If, instead of (4.4), the sequence (S_m) is generated by

$$S_{m+1} = (I - A)S_m + b,$$

then similar results hold after replacing, where is appears, $I + A$ by $I - A$.

Remark 4.5. We can similarly treat methods in which $r_k = V_k(A)P_k(A)r_0$, where V_k is an arbitrary polynomial (not necessarily of degree k) such that $V_k(0) = 1$.

Remark 4.6. The ε-algorithms can be used for solving the system of nonlinear equations $x = F(x)$ where F maps $\mathbb{C}^n$ into itself. We first choose x_0 and y. Then one iteration is as follows for $k = 0, 1, \dots$ [4]:

- Set $u_0 = x_k$ and compute $u_{i+1} = F(u_i)$ for $i = 0, \dots, 2n - 1$.
- Apply the ε-algorithm to the vectors $u_0, \dots, u_{2n}$ and take $x_{k+1} = \varepsilon_{2n}^{(0)}$.

It can be proved that, under some conditions, the sequence (x_k) converges quadratically to x [40].

5. Treatment of breakdown and near-breakdown

Let us now assume that some of the polynomials P_k, and thus the corresponding polynomials $P_k^{(1)}$, do not exist. Then a *breakdown* will occur in the Lanczos-type algorithms described in Section 3 due to a division by zero. It is possible to avoid such a breakdown by jumping over the non-existing polynomials and considering only the existing ones which are usually called regular. The problem of avoiding the breakdown in Lanczos algorithms has been treated by several authors. The first attempt is the so-called look-ahead Lanczos algorithm by Parlett, Taylor and Liu [41] where a 2×2 pivot in triangular factorization is used. An improvement of this algorithm is given in [37].

An approach very similar, although more complicated, to ours is due to Gutknecht [28]. It leads to algorithms whose implementation is discussed in [25]. Other procedures for avoiding the breakdown are also proposed in [3,30].

It is not our purpose here to compare the advantages of these various look-ahead algorithms both from the theoretical or the practical point of view but to pursue our approach by orthogonal polynomials and to show that it leads to a very simple solution of this problem. The case of the near-breakdown will also be treated similarly. The breakdown and near-breakdown problems in the CGS could also be cured in the same way.

We shall now change our notations a little and call P_k and P_{k+1} two successive regular orthogonal polynomials with respect to c, of degrees at most n_k and $n_{k+1} = n_k + m_k$ respectively. Similarly $P_k^{(1)}$ and $P_{k+1}^{(1)}$ will denote two successive regular monic orthogonal polynomials with respect to $c^{(1)}$, of degrees n_k and n_{k+1} respectively. This means that, in fact, we shall write P_k instead of P_{n_k} and $P_1^{(k)}$ instead of $P_{n_k}^{(1)}$. No confusion is possible since the polynomials of degree $n_k + 1, \ldots, n_{k+1} - 1$ do not exist.

It was proved by Draux [17] that $P_k^{(1)}$ satisfies the conditions

$$\begin{aligned} &c^{(1)}\big(\xi^i P_k^{(1)}\big) = c\big(\xi^{i+1} P_k^{(1)}\big) = 0 \quad \text{for } i = 0, \ldots, n_k + m_k - 2, \\ &c^{(1)}\big(\xi^{n_k+m_k-1} P_k^{(1)}\big) = c\big(\xi^{n_k+m_k} P_k^{(1)}\big) \neq 0. \end{aligned} \tag{5.1}$$

These conditions determine the length m_k of the *jump*. Moreover the polynomials $P_k^{(1)}$ satisfy the recurrence relationship

$$P_{k+1}^{(1)}(\xi) = q_k(\xi) P_k^{(1)}(\xi) - C_{k+1} P_{k-1}^{(1)}(\xi), \quad k = 0, 1, \ldots \tag{5.2}$$

with $P_{-1}^{(1)}(\xi) = 0$, $P_0^{(1)}(\xi) = 1$, and q_k a monic polynomial of degree m_k. If we set

$$q_k(\xi) = \alpha_0 + \cdots + \alpha_{m_k-1}\xi^{m_k-1} + \xi^{m_k}, \tag{5.3}$$

then, by (5.1), we have

$$\begin{aligned} &c\big(\xi^{n_k+m_k} P_k^{(1)}\big) = C_{k+1} c\big(\xi^{n_k} P_{k-1}^{(1)}\big), \\ &\alpha_{m_k-1} c\big(\xi^{n_k+m_k} P_k^{(1)}\big) + c\big(\xi^{n_k+m_k+1} P_k^{(1)}\big) = C_{k+1} c\big(\xi^{n_k+1} P_{k-1}^{(1)}\big), \\ &\quad \vdots \\ &\alpha_0 c\big(\xi^{n_k+m_k} P_k^{(1)}\big) + \cdots + \alpha_{m_k-1} c\big(\xi^{n_k+2m_k-1} P_k^{(1)}\big) + c\big(\xi^{n_k+2m_k} P_k^{(1)}\big) \\ &\quad = C_{k+1} c\big(\xi^{n_k+m_k} P_{k-1}^{(1)}\big). \end{aligned} \tag{5.4}$$

Now, by (5.1) and using the so-called block bordering method [10] for solving recursively the system of linear equations giving the coefficients of P_k, it was proved in [13] that

$$P_{k+1}(\xi) = P_k(\xi) - \xi w_k(\xi) P_k^{(1)}(\xi) \tag{5.5}$$

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

If we set

$$w_k(\xi) = \beta_0 + \cdots + \beta_{m_k-1}\xi^{m_k-1}, \tag{5.6}$$

then it holds that

$$\begin{aligned} &\beta_{m_k-1} c\left(\xi^{n_k+m_k} P_k^{(1)}\right) = c\left(\xi^{n_k} P_k\right), \\ &\quad\vdots \\ &\beta_0 c\left(\xi^{n_k+m_k} P_k^{(1)}\right) + \cdots + \beta_{m_k-1} c\left(\xi^{n_k+2m_k-1} P_k^{(1)}\right) = c\left(\xi^{n_k+m_k-1} P_k\right). \end{aligned} \tag{5.7}$$

Gathering all these formulae together we have the following algorithm called the MRZ (Method of Recursive Zoom) [13]:

Step 1. Choose x_0 and y. Set

$$r_0 = z_0 = b - Ax_0, \qquad z_{-1} = 0, \qquad n_0 = 0.$$

Step 2. For $k = 0, 1, \ldots$ compute m_k and then if $n_k + m_k \leqslant n$

$$x_{k+1} = x_k + w_k(A) z_k,$$

$$r_{k+1} = r_k - A w_k(A) z_k.$$

Step 3. If $r_{k+1} \neq 0$, then compute

$$n_{k+1} = n_k + m_k,$$

$$z_{k+1} = q_k(A) z_k - C_{k+1} z_{k-1}.$$

Clearly this algorithm is a generalization of the Lanczos/Orthodir and BIODIR algorithms. It cannot suffer from breakdown except the incurable hard one which occurs if $c(\xi^n P_k^{(1)}) = 0$. The corresponding subroutine is given in [12].

If we make the supplementary assumption that $c(\xi^{n_k} P_k) \neq 0$, then it is possible to generalize some of the other algorithms given in Section 3. For example, in [12], a generalization (the SMRZ) of (A8/B8) is given and also a generalization (the BMRZ) of (A8/B10). Related algorithms can be found in [28].

From the above formulae it is obviously possible to avoid also the breakdown in the CGS algorithm. This is done in [14].

Now, if

$$\begin{aligned} &|c\left(\xi^{i+1} P_k^{(1)}\right)| \leqslant \varepsilon \quad \text{for } i = 0, \ldots, n_k + m_k - 2, \\ &|c\left(\xi^{n_k+m_k} P_k^{(1)}\right)| > \varepsilon, \end{aligned} \tag{5.8}$$

then rounding errors could affect the algorithms described in Section 3—a situation known under the name of *near-breakdown*. It is again possible to jump over the polynomials affected by such errors and to compute only those satisfying the above conditions (5.8). Such algorithms were given in [12]. They generalize the methods (A8/B6), (A8/B8), and (A8/B10). Near-breakdown can be similarly avoided in the CGS algorithm (see [11]). As explained above, breakdown is due to the non-existence of some polynomials P_k and the remedy is to jump over these non-existing polynomials. In the methods Lanczos/Orthores and Lanczos/Orthomin, we made the supplementary assumption that P_k has exactly degree k. In fact this assumption is totally unnecessary in the theory of the Lanczos method but it was required by the form of the recurrence relation used, thus being a supplementary (and unnecessary) cause for breakdown. For a quite complete exposition of the breakdown and its various sources, see [34].

It seems to us that many results and algorithms are much more easily derived using the theory of formal orthogonal polynomial than by purely linear algebra techniques. In particular the transpose-free algorithm for treating the near-breakdown in the CGS given in [11] could hardly have been obtained without the use of orthogonal polynomials.

References

[1] S.F. Ashby, T.A. Manteuffel and P.E. Saylor, A taxonomy for conjugate gradient methods, *SIAM J. Numer. Anal.* 27 (1990) 1542–1568.

[2] C. Baheux, Algorithme d'implémentation de la méthode de Lanczos, Thèse, Université de Lille, Lille, France (to appear).

[3] D.L. Boley, S. Elhay, G.H. Golub and M.H. Gutknecht, Nonsymmetric Lanczos and finding orthogonal polynomials associated with indefinite weights, *Numer. Algorithms* 1 (1991) 21–43.

[4] C. Brezinski, Application de l'ε-algorithme à la résolution des systèmes non linéaires, *C.R. Acad. Sci. Paris* 271 A (1970) 1174–1177.

[5] C. Brezinski, Généralisation de la transformation de Shanks, de la table de Padé et de l'ε-algorithme, *Calcolo* 12 (1975) 317–360.

[6] C. Brezinski, *Padé-Type Approximation and General Orthogonal Polynomials*, International Series of Numerical Mathematics 50 (Birkhäuser, Basel, 1980).

[7] C. Brezinski, Other manifestations of the Schur complement, *Linear Algebra Appl.* 111 (1988) 231–247.

[8] C. Brezinski, The methods of Vorobyev and Lanczos (submitted).

[9] C. Brezinski and M. Redivo Zaglia, A new presentation of orthogonal polynomials with applications to their computation, *Numer. Algorithms* 1 (1991) 207–221.

[10] C. Brezinski and M. Redivo Zaglia, *Extrapolation Methods, Theory and Practice* (North-Holland, Amsterdam, 1991).

[11] C. Brezinski and M. Redivo Zaglia, Treatment of near-breakdown in the CGS algorithm (submitted).

[12] C. Brezinski, M. Redivo Zaglia and H. Sadok, Avoiding breakdown and near-breakdown in Lanczos type algorithms, *Numer. Algorithms* 1 (1991) 261–284.

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

[14] C. Brezinski and H. Sadok, Avoiding breakdown in the CGS algorithm, *Numer. Algorithms* 1 (1991) 199–206.

[15] P.N. Brown and Y. Saad, Hybrid Krylov methods for nonlinear systems of equations, *SIAM J. Sci. Statist. Comput.* 11 (1990) 450–481.

[16] G. Cybenko, An explicit formula for Lanczos polynomials, *Linear Algebra Appl.* 88/89 (1987) 99–115.

[17] A. Draux, *Polynômes Orthogonaux Formels: Applications*, Lecture Notes in Mathematics 974 (Springer, Berlin, 1983).

[18] S.C. Eisenstat, H.C. Elman and M.H. Schultz, Variational iterative methods for nonsymmetric systems of linear equations, *SIAM J. Numer. Anal.* 20 (1983) 345–361.
[19] H.C. Elman, Iterative methods for large, sparse, nonsymmetric systems of linear equations, Ph.D. Thesis, Yale University, New Haven, CT (1982).
[20] V. Faber and T.A. Manteuffel, Necessary and sufficient conditions for the existence of a conjugate gradient method, *SIAM J. Numer. Anal.* 21 (1984) 352–362.
[21] R. Fletcher, Conjugate gradient methods for indefinite systems, in: G.A. Watson, ed., *Numerical Analysis*, Lecture Notes in Mathematics 506 (Springer, Berlin, 1976) 73–89.
[22] R. Freund, On conjugate gradient type methods and polynomial preconditioners for a class of complex non-Hermitian matrices, *Numer. Math.* 57 (1990) 285–312.
[23] R. Freund, Conjugate gradient-type methods for linear systems with complex symmetric coefficient matrices, *SIAM J. Sci. Statist. Comput.* 13 (1992) 425–448.
[24] R.W. Freund, G.H. Golub and N.M. Nachtigal, Iterative solution of linear systems, *Acta Numer.* 1 (1992) 57–100.
[25] 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. Statist. Comput.* 14 (1993) 137–158.
[26] W. Gander, G.H. Golub and D. Gruntz, Solving linear equations by extrapolation, Manuscript NA-89-11, Computer Science Department, Stanford University, Stanford, CA (1989).
[27] G.H. Golub and D.P. O'Leary, Some history of the conjugate and Lanczos algorithms, *SIAM Rev.* 31 (1989) 50–102.
[28] M.H. Gutknecht, A completed theory of the unsymmetric Lanczos process and related algorithms, Part I, *SIAM J. Matrix Anal. Appl.* 13 (1992) 594–639.
[29] M.H. Gutknecht, The unsymmetric Lanczos algorithms and their relations to Padé approximation, continued fractions, and the QD algorithm, in: *Proceedings Copper Mountain Conference on Iterative Methods* (1990).
[30] C.J. Hegedüs, Generating conjugate directions for arbitrary matrices by matrix equations, I and II, *Comput. Math. Appl.* 21 (1991) 71–85 and 87–94.
[31] M.R. Hestenes, *Conjugate Direction Methods in Optimization* (Springer, Berlin, 1980).
[32] M.R. Hestenes and E. Stiefel, Methods of conjugate gradients for solving linear systems, *J. Res. Nat. Bur. Stand.* 49 (1952) 409–436.
[33] K.C. Jea and D.M. Young, On the simplification of generalized conjugate-gradient methods of nonsymmetrizable linear systems, *Linear Algebra Appl.* 52/53 (1983) 399–417.
[34] W. Joubert, Generalized conjugate gradient and Lanczos methods for the solution of nonsymmetric systems of linear equations, Ph.D. Thesis, The University of Texas at Austin, Austin, TX, (1990).
[35] W. Joubert, Lanczos methods for the solution of nonsymmetric systems of linear equations, *SIAM J. Matrix Anal. Appl.* 13 (1992) 926–943.
[36] W.D. Joubert and T.A. Manteuffel, Iterative methods for nonsymmetric linear systems, in: D.R. Kincaid and L.J. Hayes, eds., *Iterative Methods for Large Linear Systems* (Academic Press, New York, 1990) 149–171.
[37] M. Khelifi, Lanczos maximal algorithm for unsymmetric eigenvalues problems, *Appl. Numer. Math.* 7 (1991) 179–193.
[38] C. Lanczos, An iteration method for the solution of the eigenvalue problem of linear differential and integral operators, *J. Res. Nat. Bur. Stand.* 45 (1950) 255–282.
[39] C. Lanczos, Solution of systems of linear equations by minimized iterations, *J. Res. Nat. Bur. Stand.* 49 (1952) 33–53.
[40] H. Le Ferrand, Convergence of the topological ε-algorithm for solving systems of nonlinear equations, *Numer. Algorithms* 3 (1992) 273–284.
[41] B.N. Parlett, D.R. Taylor and Z.A. Liu, A look-ahead Lanczos algorithm for unsymmetric matrices, *Math. Comp.* 44 (1985) 105–124.
[42] Y. Saad, The Lanczos biorthogonalization algorithm and other oblique projection methods for solving large unsymmetric systems, *SIAM J. Numer. Anal.* 19 (1982) 485–506.
[43] Y. Saad and M.H. Schultz, Conjugate gradient-like algorithms for solving nonsymmetric linear systems, *Math. Comp.* 44 (1985) 417–424.
[44] D. Shanks, Non linear transformations of divergent and slowly convergent sequences, *J. Math. Phys.* 34 (1955) 1–42.

[45] A. Sidi, Extrapolation vs. projection methods for linear systems of equations, *J. Comput. Appl. Math.* 22 (1988) 71–88.
[46] P. Sonneveld, A fast Lanczos-type solver for nonsymmetric linear systems, *SIAM J. Sci. Statist. Comput.* 10 (1989) 35–52.
[47] P.K.W. Vinsome, Orthomin, an iterative method for solving sparse sets of simultaneous linear equations, in: *Proceedings 4th Symposium on Reservoir Simulation* (Society of Petroleum Engineers of AIME, 1976) 149–159.
[48] P. Wynn, On a device for computing the $e_m(S_n)$ transformation, *MTAC* 10 (1956) 91–96.
[49] D.M. Young and K.C. Jea, Generalized conjugate-gradient acceleration of nonsymmetrizable iterative methods, *Linear Algebra Appl.* 34 (1980) 159–194.