# The Look-Ahead Lanczos Process for Nonsymmetric Matrices and its Applications

Roland W. Freund*

**Abstract**

The nonsymmetric Lanczos process can be used to compute approximate eigenvalues of large non-Hermitian matrices and to obtain approximate solutions of large non-Hermitian linear systems. However, the Lanczos algorithm in its original form is susceptible to possible breakdowns and potential instabilities. We describe a look-ahead variant of the Lanczos process that remedies the problems of the original algorithm, and we discuss some of its applications. We also describe related algorithms for constructing formally orthogonal polynomials and for computing inverse triangular factorizations of Hankel matrices.

## 1 Introduction

In 1950, Cornelius Lanczos [22] proposed an iterative procedure for the successive reduction of a given, in general non-Hermitian, square matrix $A$ to a sequence of tridiagonal matrices. In contrast to direct methods that generate a tridiagonalization by eliminating elements of the matrix $A$, the Lanczos process does not modify $A$. In fact, the matrix is only used in the form of matrix-vector products with $A$ and its transpose $A^T$. Because of this feature, the Lanczos process is well suited for large sparse matrices or large structured dense matrices for which matrix-vector products can be computed cheaply. The Lanczos process can be used to compute approximate eigenvalues of $A$ and to obtain approximate solutions of systems of linear equations with $A$ as coefficient matrix.

In the original Lanczos algorithm, breakdowns—triggered by division by 0—may occur in the course of the process. In finite-precision arithmetic, such exact breakdowns are very unlikely; however, near-breakdowns—triggered by division by a number close to 0—may occur, which can then cause numerical instabilities in subsequent iterations. The possibility of breakdowns and near-breakdowns has brought the nonsymmetric Lanczos process into discredit, and it has certainly prevented many people from using the algorithm on non-Hermitian matrices. Nonetheless, it has also been demonstrated (see, e.g., [5]) that the Lanczos process—even in its original form—can be a powerful tool for large sparse matrix computations. We remark that, for the special case of Hermitian matrices $A$, breakdowns and near-breakdowns can be excluded. Unlike the general Lanczos process, the Hermitian Lanczos method has found widespread use and has been studied extensively since the 1970s.

On the other hand, it is possible to modify the Lanczos process so that it skips over those iterations in which exact or near-breakdown would occur. Such a modified Lanczos method, called the *look-ahead Lanczos algorithm*, was first developed by Taylor [26] and Parlett, Taylor, and Liu [25] in the early 1980s. However, their procedure is fairly involved, and in fact, details were only given for the simplest case of look-ahead steps of length 2.

*AT&T Bell Laboratories, Room 2C–420, 600 Mountain Avenue, Murray Hill, New Jersey 07974–0636.

This may be the reason why the look-ahead idea did not receive much attention until 1989. Since then, there has been a true revival of the nonsymmetric Lanczos process and its look-ahead variants. It is due to these recent research efforts that the possible breakdowns and near-breakdowns in the classical algorithm are now well understood. Moreover, look-ahead Lanczos algorithms were developed that can handle look-ahead steps of any length, and that are not restricted to steps of length 2. The renewed interest in the Lanczos process is reflected in the large number of papers that were written on the subject since 1989; see [2, 3, 9, 11, 12, 14, 15, 19, 20, 21, 24] and the references given there.

The Lanczos process has intimate connections with many other areas of mathematics, such as formally orthogonal polynomials (FOPs), Padé approximation, factorization of Hankel matrices, control theory, and coding theory. The problem of breakdowns has a corresponding formulation in all of these areas, and remedies for exact breakdowns in these different settings have been known for quite some time. For example, the breakdown in the Lanczos process is equivalent to a breakdown of the generic three-term recurrence relation for FOPs, and it is well known how to overcome such breakdowns by modifying the recursions for FOPs; see [6, 15, 19] and the references given there. In the context of the partial realization problem in control theory, remedies for breakdowns were given in [18]. The Lanczos process is also closely related to fast algorithms for the factorization of Hankel matrices, and again, it was known how to overcome possible breakdowns of these algorithms; see, e.g., [15]. In view of all these results, it is surprising that numerical analysts did not study the look-ahead Lanczos process earlier.

In this paper, we describe the look-ahead Lanczos process, sketch an actual implementation, and discuss some of its applications in large-scale matrix computations. Among the many possible approaches to the look-ahead Lanczos process, we will use the one based on the connection with FOPs.

Throughout the paper, all vectors and matrices are allowed to have real or complex entries. As usual, $M^T = [\, m_{kj} \,]$ and $M^H = [\, \overline{m_{kj}} \,]$ denote the transpose and the conjugate transpose, respectively, of the matrix $M = [\, m_{jk} \,]$. The vector norm $\|x\| := \sqrt{x^H x}$ is always the Euclidean norm. We use the notation $K_n(c, B) := \mathrm{span}\{c, Bc, \ldots, B^{n-1}c\}$ for the $n$th *Krylov subspace* of $\mathbb{C}^N$ generated by $c \in \mathbb{C}^N$ and $B \in \mathbb{C}^{N\times N}$. We denote by $\mathcal{P}_n$ the set of all polynomials (with coefficients in $\mathbb{C}$) of degree at most $n$, and $\mathcal{P}$ is the set of all complex polynomials of arbitrary degree. The symbol 0 will be used for the number zero, the zero vector, and the zero matrix; its actual meaning and, in the case of the zero vector or the zero matrix, its dimension will always be apparent from the context.

## 2 The Classical Lanczos Process

In this section, we briefly review the classical Lanczos process.

### 2.1 The Algorithm and Some of its Properties

Let $A \in \mathbb{C}^{N\times N}$, and let $v_0, w_0 \in \mathbb{C}^N$, $v_0, w_0 \neq 0$, be two arbitrary nonzero vectors. Starting with $v_0$ and $w_0$, the classical Lanczos algorithm [22] generates two sequences of vectors

$$(1) \qquad \{v_j\}_{j=0}^n \quad \text{and} \quad \{w_j\}_{j=0}^n, \quad n = 0, 1, \ldots,$$

such that, for all $n$,

$$(2) \qquad \mathrm{span}\{v_0, v_1, \ldots, v_n\} = K_{n+1}(v_0, A), \qquad \mathrm{span}\{w_0, w_1, \ldots, w_n\} = K_{n+1}(w_0, A^T),$$

and the two sequences are biorthogonal:

$$w_j^T v_k = \begin{cases} 0, & \text{if } j \neq k, \\ \delta_k \neq 0, & \text{if } j = k, \end{cases} \qquad \text{for all} \quad j, k = 0, 1, \ldots, n. \tag{3}$$

It turns out that vectors (1) satisfying (2) and (3) can be constructed by means of simple three-term vector recurrences. This procedure is just the classical *Lanczos algorithm*.

ALGORITHM 2.1 (Classical Lanczos algorithm [22]).

0) Choose $v_0, w_0 \in \mathbb{C}^N$ with $v_0, w_0 \neq 0$. Set $v_{-1} = w_{-1} = 0 \in \mathbb{C}^N$ and $\delta_{-1} = 1$.

For $n = 0, 1, \ldots$, do:

1) Compute $\delta_n = w_n^T v_n$. If $\delta_n = 0$, then stop.

2) Set

$$v_{n+1} = Av_n - v_n\alpha_n - v_{n-1}\beta_n, \quad w_{n+1} = A^T w_n - w_n\alpha_n - w_{n-1}\beta_n, \tag{4}$$

$$\text{where} \quad \alpha_n = \frac{w_n^T A v_n}{\delta_n} \quad \text{and} \quad \beta_n = \frac{\delta_n}{\delta_{n-1}}. \tag{5}$$

We remark that, in practice, one uses a scaled version of the recurrences (4), in order to avoid over– or underflow. A proven choice is to scale the Lanczos vectors so that $\|v_n\| = \|w_n\| = 1$ for all $n$. In this paper, we will always use the unscaled recurrences (4), since they simplify the exposition of the algorithms.

Next, we list a few properties of Algorithm 2.1. On setting $V_n := [\, v_0 \quad v_1 \quad \cdots \quad v_n \,]$, $W_n := [\, w_0 \quad w_1 \quad \cdots \quad w_n \,]$, and

$$T_n := \begin{bmatrix} \alpha_0 & \beta_1 & 0 & \cdots & 0 \\ 1 & \alpha_1 & \beta_2 & \ddots & \vdots \\ 0 & 1 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \beta_n \\ 0 & \cdots & 0 & 1 & \alpha_n \end{bmatrix}, \tag{6}$$

the recurrences (4) for the vectors $\{v_j\}_{j=0}^{n+1}$ and $\{w_j\}_{j=0}^{n+1}$ can be summarized as follows:

$$AV_n = V_n T_n + [\, 0 \quad \cdots \quad 0 \quad v_{n+1} \,] \quad \text{and} \quad A^T W_n = W_n T_n + [\, 0 \quad \cdots \quad 0 \quad w_{n+1} \,]. \tag{7}$$

The relations (7) suggest to use the eigenvalues of the tridiagonal matrix $T_n$ as approximate eigenvalues of $A$. Indeed, if $v_{n+1} = 0$ or $w_{n+1} = 0$ in (7), then each eigenvalue of $T_n$ would also be an eigenvalue of $A$. Similarly, an approximate solution of $Ax = b$ can be obtained via solving a suitable linear system with $T_n$ as coefficient matrix, or a weighted least-squares problem with an extended version of $T_n$ as coefficient matrix. This will be discussed in more detail in Section 5.

The termination check $w_n^T v_n = 0$ in step 1) of Algorithm 2.1 is indispensable to avoid division by 0 in (5). In exact arithmetic, this check will be satisfied after at most $N$ iterations. Normally, the algorithm stops because $w_n^T v_n = 0$ with $v_n = 0$ or $w_n = 0$. It is easy to verify that the case $v_n = 0$ occurs if, and only if, $K_n(v_0, A)$ is an $A$-invariant subspace of $\mathbb{C}^N$. By (2), the vectors $\{v_j\}_{j=0}^{n-1}$ then build a basis for the $A$-invariant subspace $K_n(v_0, A)$. Similarly, the vectors $\{w_j\}_{j=0}^{n-1}$ span the $A^T$-invariant subspace $K_n(w_0, A^T)$ of $\mathbb{C}^N$ if $w_n = 0$. The event $v_n = 0$ or $w_n = 0$ is called *regular termination*.

Unfortunately, it can also happen that Algorithm 2.1 stops because

$$w_n^T v_n = 0, \quad \text{but} \quad v_n \neq 0 \quad \text{and} \quad w_n \neq 0. \tag{8}$$

In this case, the Lanczos process terminates prematurely, before it has constructed a basis for an $A$-invariant or an $A^T$-invariant subspace. Therefore, termination due to (8) constitutes a *breakdown* of Algorithm 2.1. In finite-precision arithmetic, exact breakdowns (8) are unlikely; however, *near-breakdowns*, *i.e.*,

$$w_n^T v_n \approx 0, \quad \text{but} \quad v_n \not\approx 0 \quad \text{and} \quad w_n \not\approx 0, \tag{9}$$

may occur, which can then cause numerical instabilities in subsequent iterations. Look-ahead Lanczos methods are modifications of Algorithm 2.1 that allow to continue the Lanczos process even in the case of an exact breakdown (8) or a near-breakdown (9).

For the special case of Hermitian matrices, the possibility of breakdowns can be excluded. In fact, if $A = A^H$, then one choose $w_0 = \overline{v_0}$ as the complex conjugate of $v_0$, and (4) and (5) imply that $w_n = \overline{v_n}$ for all $n$. Hence, we have $w_n^T v_n = v_n^H v_n = \|v_n\|^2 > 0$ if $v_n \neq 0$, and thus breakdowns cannot occur.

## 2.2 Polynomial Formulation

There is an equivalent formulation of Algorithm 2.1 in terms of polynomials, instead of vectors. By (4), each pair of Lanczos vectors $v_n$ and $w_n$ can be represented in the form

$$v_n = \varphi_n(A) v_0 \quad \text{and} \quad w_n = \varphi_n(A^T) w_0, \tag{10}$$

where $\varphi_n(\lambda) \equiv \lambda^n + \cdots \in \mathcal{P}_n$ is a *monic* polynomial of degree $n$. Note that $\varphi_n$ is uniquely determined by (10). We refer to the $\varphi_n$'s as the *Lanczos polynomials*. Using (10), we now rewrite the Lanczos process in terms of the Lanczos polynomials.

To this end, we first define a bilinear form $\langle \cdot, \cdot \rangle$ on $\mathcal{P} \times \mathcal{P}$ by setting

$$\langle \psi, \varphi \rangle := w_0^T \psi(A) \varphi(A) v_0 \quad \text{for all} \quad \varphi, \psi \in \mathcal{P}. \tag{11}$$

From (10) and (11), we have

$$w_j^T v_k = \left(\varphi_j(A^T) w_0\right)^T \varphi_k(A) v_0 = w_0^T \varphi_j(A) \varphi_k(A) v_0 = \langle \varphi_j, \varphi_k \rangle \quad \text{for all} \quad j, k,$$

and thus the biorthogonality condition (3) of the Lanczos vectors can be rewritten as follows:

$$\langle \varphi_j, \varphi_k \rangle = \begin{cases} 0, & \text{if } j \neq k, \\ \langle \varphi_k, \varphi_k \rangle \neq 0, & \text{if } j = k, \end{cases} \qquad \text{for all} \quad j, k = 0, 1, \ldots, n. \tag{12}$$

In view of (12), the Lanczos polynomials are orthogonal with respect to the bilinear form (11). Furthermore, the recursions (4) for the Lanczos vectors are just a reformulation of the usual three-term polynomial recurrence

$$\varphi_{n+1} = \lambda \varphi_n - \alpha_n \varphi_n - \beta_n \varphi_{n-1} \tag{13}$$

for orthogonal polynomials (see, e.g., [4]). Note that, by (5), (10), and (11), the coefficients in (13) are given by

$$\alpha_n = \frac{\langle \varphi_n, \lambda \varphi_n \rangle}{\langle \varphi_n, \varphi_n \rangle} \quad \text{and} \quad \beta_n = \frac{\langle \varphi_n, \varphi_n \rangle}{\langle \varphi_{n-1}, \varphi_{n-1} \rangle}. \tag{14}$$

Finally, the conditions (8) and (9) for an exact breakdown or a near-breakdown of Algorithm 2.1 translate into the conditions

$$\langle \varphi_n, \varphi_n \rangle = 0 \quad \text{or} \quad \langle \varphi_n, \varphi_n \rangle \approx 0, \tag{15}$$

which signal an exact breakdown or a near-breakdown in the recursions (13) and (14). In Section 3.2, we present modified recurrences that allow to continue the construction of Lanczos polynomials even if (15) occurs. These modified recurrences are the basis for the look-ahead Lanczos process.

## 3 Hankel Bilinear Forms and the Construction of FOPs

In this section, we consider general Hankel bilinear forms, and we sketch a look-ahead procedure for constructing the associated formally orthogonal polynomials.

### 3.1 Hankel Bilinear Forms

Let $\{h_j\}_{j=0}^{\infty}$ be an arbitrary given sequence of real or complex numbers, and set

$$H_n := [\, h_{j+k} \,]_{j,k=0,\ldots,n} = \begin{bmatrix} h_0 & h_1 & \cdots & h_n \\ h_1 & \cdot^{\cdot^{\cdot}} & \cdot^{\cdot^{\cdot}} & \vdots \\ \vdots & \cdot^{\cdot^{\cdot}} & \cdot^{\cdot^{\cdot}} & h_{2n-1} \\ h_n & \cdots & h_{2n-1} & h_{2n} \end{bmatrix}, \quad n = 0, 1, \ldots . \tag{16}$$

The family of matrices (16) induces a *bilinear form* $\langle \cdot, \cdot \rangle : \mathcal{P} \times \mathcal{P} \longmapsto \mathbb{C}$ by setting

$$\langle \psi, \varphi \rangle := v^T H_n u \tag{17}$$

for all $n = 0, 1, \ldots$ and any two polynomials

$$\varphi(\lambda) \equiv [\, 1 \quad \lambda \quad \cdots \quad \lambda^n \,]\, u, \quad \psi(\lambda) \equiv [\, 1 \quad \lambda \quad \cdots \quad \lambda^n \,]\, v, \quad u, v \in \mathbb{C}^{n+1}, \tag{18}$$

of degree at most $n$. Matrices of the form (16) are called *Hankel matrices*, and we refer to $\langle \cdot, \cdot \rangle$ as a *Hankel* bilinear form. Note that $\langle \cdot, \cdot \rangle$ is *symmetric*, *i.e.*, $\langle \psi, \varphi \rangle = \langle \varphi, \psi \rangle$ for all $\varphi, \psi \in \mathcal{P}$. Furthermore, we have $h_j = \langle \lambda^j, 1 \rangle$, $j = 0, 1, \ldots$, and hence $h_j$ is just the $j$th *moment* associated with $\langle \cdot, \cdot \rangle$. We remark that the Lanczos bilinear form (11) is a special case of a Hankel bilinear with moments given by

$$h_j = w_0^T A^j v_0, \quad j = 0, 1, \ldots . \tag{19}$$

We stress that, in general, the Hankel bilinear form $\langle \cdot, \cdot \rangle$ is not an inner product. Indeed, it is possible that a nonzero polynomial $\varphi$ has "norm" $\langle \varphi, \varphi \rangle = 0$ or $\langle \varphi, \varphi \rangle < 0$. Nevertheless, it turns out to be useful to study polynomials that are orthogonal with respect to $\langle \cdot, \cdot \rangle$. We now give a definition of these so-called *formally orthogonal polynomials* (FOPs).

A monic polynomial $\varphi_n(\lambda) \equiv \lambda^n + \cdots \in \mathcal{P}_n$ is called a FOP of degree $n$ if

$$\langle \psi, \varphi_n \rangle = 0 \quad \text{for all} \quad \psi \in \mathcal{P}_{n-1}. \tag{20}$$

A FOP $\varphi_n$ is said to be *regular* if it is uniquely determined by (20). Note that, for $n = 0$, the condition (20) is void, and thus $\varphi_0(\lambda) \equiv 1$ is always a regular FOP of degree 0.

In contrast to orthogonal polynomials associated with an inner product, FOPs $\varphi_n$ need not exist or need not be unique for every degree $n$. In fact, using (17) and (20), one readily verifies the following condition for the existence of regular FOPs.

LEMMA 3.1. *A regular* FOP $\varphi_n$ *of degree* $n$ *exists if, and only if, the matrix* $H_{n-1}$ *is nonsingular.*

By Lemma 3.1, no regular FOP of degree $n$ exists if $H_{n-1}$ is singular. Moreover, in finite-precision arithmetic, it is not advisable to compute regular FOPs if $H_{n-1}$ is nonsingular, but ill-conditioned. For singular or ill-conditioned matrices $H_{n-1}$, we will construct quasi-FOPs that only satisfy relaxed versions of the orthogonality conditions (20).

A monic polynomial $\varphi_n(\lambda) \equiv \lambda^n + \cdots \in \mathcal{P}_n$ is called an $m$-*quasi*-FOP of degree $n$ if $m < n$ and

$$\langle \psi, \varphi_n \rangle = 0 \quad \text{for all} \quad \psi \in \mathcal{P}_{m-1}. \tag{21}$$

From (17) and (21), we obtain the following result on the existence of $m$-quasi-FOPs.

LEMMA 3.2. *Let $1 \le m < n$. If $H_{m-1}$ is nonsingular, then an $m$-quasi-*FOP *$\varphi_n$ of degree $n$ exists.*

## 3.2 Constructing FOPs Associated with Hankel Bilinear Forms

In [15], Freund and Zha derived a look-ahead procedure for constructing FOPs associated with general Hankel bilinear forms $\langle \cdot, \cdot \rangle$. We now present a sketch of this algorithm.

The look-ahead procedure generates a sequence of regular FOPs and quasi-FOPs

$$\{\varphi_n\}_{n=0}^{\infty}, \tag{22}$$

where each polynomial $\varphi_n$ is monic and of exact degree $n$. We use the indices

$$0 =: n_0 < n_1 < \cdots < n_k < \cdots \tag{23}$$

to mark those polynomials in the sequence (22) that are generated as regular FOPs $\varphi_{n_k}$, $k = 0, 1, \ldots$; we refer to the numbers $n_k$ in (23) as *regular indices*. By Lemma 3.1, $n_k > 0$ can be chosen as a regular index only if the corresponding $n_k \times n_k$ Hankel matrix

$$H_{n_k-1} \quad \text{is nonsingular.} \tag{24}$$

The remaining polynomials $\varphi_n$, $n_k < n < n_{k+1}$, $k = 0, 1, \ldots$, in the sequence (22) are chosen as $n_k$-quasi-FOPs of degree $n$. By (24) and Lemma 3.2, this is always possible.

For the formulation of the algorithm, we will need some further notation. For each fixed $n$, we define $l = l(n)$ as the unique integer with $n_l \le n < n_{l+1}$. Note that $l = l(n)$ is just the number of the last regular FOP with degree $\le n$. The first $n+1$ polynomials $\varphi_0, \varphi_1, \ldots, \varphi_n$ of the sequence (22) can be grouped into $l+1$ blocks

$$\Phi^{(k)} := \begin{cases} [\,\varphi_{n_k} \quad \varphi_{n_k+1} \quad \cdots \quad \varphi_{n_{k+1}-1}\,], & \text{if } 0 \le k < l, \\ [\,\varphi_{n_l} \quad \varphi_{n_l+1} \quad \cdots \quad \varphi_n\,], & \text{if } k = l. \end{cases} \tag{25}$$

The first polynomial $\varphi_{n_k}$ in each block is a regular FOP, while the remaining polynomials are $n_k$-quasi FOPs. We remark that the $l$th block is *complete* if $n = n_{l+1} - 1$; in this case, the next polynomial $\varphi_{n+1}$ is constructed as a regular FOP. Otherwise, if $n < n_{l+1} - 1$, the $l$th block is still *incomplete*, and $\varphi_{n+1}$ is added to the $l$th block as an $n_l$-quasi-FOP. Using the orthogonality conditions (20) for FOPs and the relations (21) for $n_k$-quasi FOPs, one readily verifies that the blocks (25) satisfy the *block-orthogonality* conditions

$$\langle \Phi^{(j)}, \Phi^{(k)} \rangle = 0 \quad \text{for all} \quad j \ne k, \ j, k = 0, 1, \ldots, l. \tag{26}$$

Here and in the sequel, we use the notation

$$\langle \Psi, \Phi \rangle := \begin{bmatrix} \langle \psi_0, \phi_0 \rangle & \cdots & \langle \psi_0, \phi_m \rangle \\ \vdots & & \vdots \\ \langle \psi_i, \phi_0 \rangle & \cdots & \langle \psi_i, \phi_m \rangle \end{bmatrix} \in \mathbb{C}^{(i+1)\times(m+1)}$$

for any pair $\Phi = [\,\phi_0 \quad \phi_1 \quad \cdots \quad \phi_m\,]$ and $\Psi = [\,\psi_0 \quad \psi_1 \quad \cdots \quad \psi_i\,]$ of row vectors of polynomials in $\mathcal{P}$. Furthermore, we set

$$\delta^{(k)} := \langle \Phi^{(k)}, \Phi^{(k)} \rangle, \quad k = 0, 1, \ldots, l, \quad \text{and} \quad D_n := \operatorname{diag}\left(\delta^{(0)}, \delta^{(1)}, \cdots, \delta^{(l)}\right). \tag{27}$$

It can be shown [9, Theorem 3.5] that

$$\det H_{n_l - 1} = \prod_{k=0}^{l-1} \det \delta^{(k)} \quad \text{and} \quad \det H_n = (\det H_{n_l-1})(\det \delta^{(l)}). \tag{28}$$

Thus, by (24) and (28), for constructing $\varphi_{n+1}$ as a regular FOP it is necessary that

$$\delta^{(l)} \quad \text{is nonsingular}. \tag{29}$$

With these preparations, the look-ahead procedure for generating the polynomials (22) can be sketched as follows.

ALGORITHM 3.3 (Look-ahead process for constructing FOPs [15]).

0) Set $\Phi^{(0)} = \varphi_0 = 1$, $\delta^{(0)} = \langle \Phi^{(0)}, \Phi^{(0)} \rangle$, $n_0 = l = 0$, $\Phi^{(-1)} = 0$, and $\delta^{(-1)} = 1$.

For $n = 0, 1, \ldots,$ do:

1) Decide whether to construct $\varphi_{n+1}$ as a regular FOP or as an $n_l$-quasi-FOP and go to 2) or 3), respectively.

2) (*Constructing a regular* FOP) Set

$$\alpha_n = (\delta^{(l)})^{-1} \langle \Phi^{(l)}, \lambda\varphi_n \rangle, \quad \beta_n = (\delta^{(l-1)})^{-1} \langle \Phi^{(l-1)}, \lambda\varphi_n \rangle, \tag{30}$$

$$\text{and} \quad \varphi_{n+1} = \lambda\varphi_n - \Phi^{(l)}\alpha_n - \Phi^{(l-1)}\beta_n. \tag{31}$$

Set $n_{l+1} = n + 1$, $l = l + 1$, $\Phi^{(l)} = \emptyset$, and go to 4).

3) (*Constructing an $n_l$-quasi*-FOP) Set

$$\beta_n = (\delta^{(l-1)})^{-1} \langle \Phi^{(l-1)}, \lambda\varphi_n \rangle \quad \text{and} \quad \varphi_{n+1} = \lambda\varphi_n - \Phi^{(l-1)}\beta_n - \sum_{i=n_l}^{n} \xi_i^{(n)} \varphi_i. \tag{32}$$

4) Set $\Phi^{(l)} = [\,\Phi^{(l)} \quad \varphi_{n+1}\,]$, $\delta^{(l)} = \langle \Phi^{(l)}, \Phi^{(l)} \rangle$.

We remark that, in (32), $\xi_i^{(n)} \in \mathbb{C}$ are arbitrary recurrence coefficients.

The strategy for deciding in step 1) of Algorithm 3.3 whether to build the next polynomial as a regular FOP or as a quasi-FOP is called the *look-ahead strategy*. In Section 4, we will briefly discuss the look-ahead strategy for one particular application of Algorithm 3.3. Recall that condition (29) is necessary for constructing $\varphi_{n+1}$ as a regular FOP. In particular, if the look-ahead strategy were based only on (29), then Algorithm 3.3 constructs all existing regular FOPs. Note that the condition (29) guarantees that the inverse matrices in (30), respectively (32), always exist.

Of course, it needs to be verified that the polynomials generated by Algorithm 3.3 indeed satisfy the required block-orthogonality relations (26). This can be proved by simple induction on $n$; we refer the reader to [15].

Algorithm 3.3 shows that regular FOPs associated with general Hankel bilinear forms can always be constructed by means of short recurrences (31), respectively (32), which involve only polynomials from the last two blocks $\Phi^{(l)}$ and $\Phi^{(l-1)}$. Finally, we remark that the update (31) just reduces to a simple three-term recurrence of the form (13) if all polynomials are constructed as regular FOPs.

## 4 A Look-Ahead Lanczos Algorithm for Nonsymmetric Matrices

We now return to the Lanczos process for general nonsymmetric matrices, and as a first application of Algorithm 3.3, we obtain a look-ahead Lanczos algorithm.

As in Section 2.1, let $A \in \mathbb{C}^{N \times N}$, and let $v_0, w_0 \in \mathbb{C}^N$, $v_0, w_0 \neq 0$, be two arbitrary nonzero vectors. Let $\langle \cdot, \cdot \rangle$ be the Lanczos bilinear form with moments defined by (19), and let $\{\varphi_n\}_{n=0}^{\infty}$ be the associated sequence (22) of FOPs and $n_k$-quasi-FOPs generated by Algorithm 3.3. Recall from (23) that the indices $n_k$ mark the regular FOPs.

In analogy to (10), we use the polynomials $\varphi_n$ to define Lanczos vectors by setting

$$v_n := \varphi_n(A) v_0 \quad \text{and} \quad w_n := \varphi_n(A^T) w_0. \tag{33}$$

If $n = n_k$ and thus $\varphi_{n_k}$ is a regular FOP, then the vectors $v_{n_k}$ and $w_{n_k}$ are called *regular vectors*. If $\varphi_n$ is a quasi-FOP, then we refer to $v_n$ and $w_n$ as *inner vectors*. As in Section 3.2, we denote by $l = l(n)$ the integer with $n_l \leq n < n_{l+1}$. Thus $n_l$ is the index of the last pair of regular vectors $v_{n_l}$ and $w_{n_l}$ with index $\leq n$. In view of the transition rules (33), the blocks $\Phi^{(k)}$, $k = 0, 1, \ldots, l$, in (25) now correspond to blocks of Lanczos vectors defined by

$$V^{(k)} := \begin{cases} [\, v_{n_k} \quad v_{n_k+1} \quad \cdots \quad v_{n_{k+1}-1} \,], & \text{if } 0 \leq k < l, \\ [\, v_{n_l} \quad v_{n_l+1} \quad \cdots \quad v_n \,], & \text{if } k = l, \end{cases} \tag{34}$$

and

$$W^{(k)} := \begin{cases} [\, w_{n_k} \quad w_{n_k+1} \quad \cdots \quad w_{n_{k+1}-1} \,], & \text{if } 0 \leq k < l, \\ [\, w_{n_l} \quad w_{n_l+1} \quad \cdots \quad w_n \,], & \text{if } k = l. \end{cases} \tag{35}$$

Furthermore, for the blocks $\delta^{(k)}$ defined in (27), we obtain the following representations:

$$\delta^{(k)} = (W^{(k)})^T V^{(k)} \quad \text{for all} \quad k = 0, 1, \ldots, l. \tag{36}$$

Note that $V^{(k)}$ and $W^{(k)}$ are $N \times \rho_k$ matrices, where $\rho_k := n_{k+1} - n_k$ if $0 \leq k < l$, and $\rho_l := n + 1 - n_l$. The first vectors $v_{n_k}$ and $w_{n_k}$ in each of the blocks (34) and (35), respectively, are always regular vectors, while the remaining ones are inner vectors. The number $\rho_k$ is called the *length* of the $k$th look-ahead step. If $\rho_k = 1$, then $V^{(k)} = v_{n_k}$ and $W^{(k)} = w_{n_k}$ just consist of a single regular vector. We refer to steps of length $\rho_k > 1$ as *true* look-ahead steps. We remark that the matrices (34) and (35) define a partitioning of the first $n+1$ Lanczos vectors

$$\{v_j\}_{j=0}^{n} \quad \text{and} \quad \{w_j\}_{j=0}^{n} \tag{37}$$

into blocks, according to the look-ahead steps taken:

$$\begin{aligned} V_n &:= [\, v_0 \quad v_1 \quad \cdots \quad v_n \,] = [\, V^{(0)} \quad V^{(1)} \quad \cdots \quad V^{(l)} \,], \\ W_n &:= [\, w_0 \quad w_1 \quad \cdots \quad w_n \,] = [\, W^{(0)} \quad W^{(1)} \quad \cdots \quad W^{(l)} \,]. \end{aligned} \tag{38}$$

By means of (33)–(36), we can now rewrite Algorithm 3.3 in terms of the Lanczos vectors (33), instead of the polynomials (22). The resulting procedure is just the look-ahead Lanczos algorithm proposed by Freund, Gutknecht, and Nachtigal [11]. An outline of this algorithm is as follows.

ALGORITHM 4.1 (Sketch of the look-ahead Lanczos algorithm [11]).

0) Choose $v_0, w_0 \in \mathbb{C}^N$ with $v_0, w_0 \neq 0$. Set $V^{(0)} = v_0$, $W^{(0)} = w_0$, $\delta^{(0)} = (W^{(0)})^T V^{(0)}$,

$n_0 = l = 0$, $v_{-1} = w_{-1} = 0 \in \mathbb{C}^N$, $V^{(-1)} = v_{-1}$, $W^{(-1)} = w_{-1}$, and $\delta^{(-1)} = 1$.

For $n = 0, 1, \ldots$, do:

1) Decide whether to construct $v_{n+1}$ and $w_{n+1}$ as regular or inner vectors and go to 2) or 3), respectively.

2) (*Regular step*) Compute

$$\alpha_n = (\delta^{(l)})^{-1}(W^{(l)})^T A v_n, \quad \beta_n = (\delta^{(l-1)})^{-1}(W^{(l-1)})^T A v_n, \tag{39}$$

$$v_{n+1} = A v_n - V^{(l)}\alpha_n - V^{(l-1)}\beta_n, \tag{40}$$

$$w_{n+1} = A^T w_n - W^{(l)}\alpha_n - W^{(l-1)}\beta_n, \tag{41}$$

set $n_{l+1} = n + 1$, $l = l + 1$, $V^{(l)} = W^{(l)} = \emptyset$, and go to 4).

3) (*Inner step*) Compute

$$\beta_n = (\delta^{(l-1)})^{-1}(W^{(l-1)})^T A v_n,$$

$$v_{n+1} = A v_n - v_n\zeta_n - v_{n-1}\eta_n - V^{(l-1)}\beta_n, \tag{42}$$

$$w_{n+1} = A^T w_n - w_n\zeta_n - w_{n-1}\eta_n - W^{(l-1)}\beta_n. \tag{43}$$

4) If $v_{n+1} = 0$ or $w_{n+1} = 0$, then stop. Otherwise, set

$$V^{(l)} = [\, V^{(l)} \quad v_{n+1} \,], \quad W^{(l)} = [\, W^{(l)} \quad w_{n+1} \,], \quad \delta^{(l)} = (W^{(l)})^T V^{(l)}.$$

It remains to specify the look-ahead strategy that is used for the decision in step 1) of Algorithm 4.1. In [11], we developed a look-ahead strategy based on three different checks. If all these checks are satisfied, then the vectors $v_{n+1}$ and $w_{n+1}$ are constructed as regular vectors; otherwise, they are built as inner vectors. First, it is checked whether $\delta^{(l)}$ is nonsingular; recall from (29) that this is a necessary condition for performing a regular step. The other two checks compare the norms of $\alpha_n$ and $\beta_n$ in (39) to some suitable estimate $n(A)$ of the norm of $A$. The vectors $v_{n+1}$ and $w_{n+1}$ are built as regular vectors only if the norms of $\alpha_n$ and $\beta_n$ are not substantially bigger than $n(A)$. Roughly speaking, the motivation for these two checks comes from the recurrences (40) and (41), which show that the components $Av_n \in K_{n+1}(v_0, A)$ and $A^T w_n \in K_{n+1}(w_0, A^T)$ of the new Lanczos vectors $v_{n+1}$ and $w_{n+1}$ would be dominated by components in the old Krylov subspaces $K_n(v_0, A)$ and $K_n(w_0, A^T)$ if the norms of the coefficient vectors $\alpha_n$ and $\beta_n$ are large compared to the norm of $A$. As a result, $v_{n+1}$ and $w_{n+1}$ would be poor basis vectors for $K_{n+1}(v_0, A)$ and $K_{n+1}(w_0, A^T)$ if they were built as regular vectors. For a more detailed description of this look-ahead strategy, we refer the reader to [11, 14]. Here, we only note that the outlined look-ahead strategy builds regular vectors in preference to inner vectors, and it takes as few true look-ahead steps as possible.

We remark that, in (42) and (43), $\zeta_n \in \mathbb{C}$ and $\eta_n \in \mathbb{C}$, $n = 0, 1, \ldots$, are arbitrary recurrence coefficients with $\eta_{n_k} = 0$, $k = 1, 2, \ldots$ . The choice of these coefficients is not overly important, since Algorithm 4.1 usually performs only few inner steps anyway. We stress that, as in the case of the classical Lanczos algorithm, in practice, the Lanczos vectors in Algorithm 4.1 should be scaled to have unit length. A FORTRAN 77 implementation of Algorithm 4.1 is available from netlib as part of QMRPACK[1].

[1] QMRPACK can be obtained by using xnetlib or by anonymous ftp. In the latter case, ftp to "netlib.att.com" and then get the file "qmrpack.tar.Z" from the directory "netlib/linalg".

Next, we list a few properties of the look-ahead Lanczos Algorithm 4.1. Recall that the vectors generated by the classical Lanczos Algorithm 2.1 are characterized by the properties (2) and (3). The Lanczos vectors (37) constructed by Algorithm 4.1 still satisfy (2). However, instead of (3), the vectors (37) are now only block-biorthogonal:

$$(W^{(j)})^T V^{(k)} = \begin{cases} 0, & \text{if } j \neq k, \\ \delta^{(k)}, & \text{if } j = k, \end{cases} \qquad \text{for all} \quad j \neq k, \quad j,k = 0,1,\ldots,l. \tag{44}$$

We note that (44) is just a reformulation of the block-orthogonality condition (26) for polynomials. Using the matrices $V_n$ and $W_n$ defined in (38), the recurrences (40)–(43) for the look-ahead Lanczos vectors $\{v_j\}_{j=0}^{n+1}$ and $\{w_j\}_{j=0}^{n+1}$ can again be summarized in relations of the form (7), where $T_n$ is now an $(n+1)\times(n+1)$ block-tridiagonal matrix of the form

$$T_n := \begin{bmatrix} \mu_0 & \nu_1 & 0 & \cdots & 0 \\ \iota_1 & \mu_1 & \ddots & \ddots & \vdots \\ 0 & \iota_2 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \nu_l \\ 0 & \cdots & 0 & \iota_l & \mu_l \end{bmatrix}. \tag{45}$$

In (45), the diagonal blocks $\mu_k$ are $\rho_k \times \rho_k$ matrices, and the subdiagonal blocks $\iota_k$ are $\rho_k \times \rho_{k-1}$ matrices whose elements are all zero, except for an element 1 in their upper right corner. Recall that $\rho_k$ is the length of the $k$th look-ahead step. If no true look-ahead steps are performed, then Algorithm 4.1 reduces to the classical Lanczos Algorithm 2.1, and the matrix (45) is just the scalar tridiagonal matrix (6) associated with Algorithm 2.1.

We remark that the diagonal blocks $\mu_k$ in (45) are upper Hessenberg matrices with all 1's on their subdiagonals. Thus, in addition to being block tridiagonal, the matrix $T_n$ is also upper Hessenberg with all 1's on its subdiagonal. This guarantees that the extended $(n+2)\times(n+1)$ matrix

$$T_n^{(e)} := \begin{bmatrix} T_n \\ e_n^T \end{bmatrix}, \quad \text{where} \quad e_n := [0 \quad \cdots \quad 0 \quad 1]^T \in \mathbb{R}^{n+1}, \tag{46}$$

has always full column rank:

$$\operatorname{rank} T_n^{(e)} = n+1 \tag{47}$$

## 5 Applications in Matrix Computations

In this section, we discuss applications of Algorithm 4.1 in matrix computations.

### 5.1 Approximate Eigenvalues

As we already mentioned in Section 2.1, relations of the type (7) suggest to use the eigenvalues of $T_n$ as approximations to the eigenvalues of $A$. The resulting procedure thus consists of two steps. First, we run the look-ahead Lanczos Algorithm 4.1 for $n$ iterations to obtain the block-tridiagonal Lanczos matrix (45), $T_n$. Second, we compute some or all of the eigenvalues of $T_n$. Note that, unless we are also interested in computing approximate right or left eigenvectors of $T_n$, it is not necessary to store the Lanczos vectors. For an actual implementation of this procedure, a number of important issues need to be addressed, such as the solution of the eigenvalue problem for $T_n$, convergence tests, and the detection of spurious eigenvalues caused by loss of biorthogonality. For a discussion of these practical aspects, we refer the reader to the paper by Cullum and Willoughby [5], where the classical Lanczos algorithm without look-ahead is considered. Numerical results of eigenvalue computations based on Algorithm 4.1 can be found in [11].

### 5.2 Iterative Solution of Linear Systems

Next, we turn to systems of linear equations,

$$Ax = b, \tag{48}$$

with $A \in \mathbb{C}^{N\times N}$ as coefficient matrix. Let $x_0 \in \mathbb{C}^N$ be an arbitrary initial guess for (48), and set $r_0 := b - Ax_0$. Our goal is to construct approximate solutions of (48) of the form

$$x_{n+1} \in x_0 + K_{n+1}(r_0, A), \quad n = 0, 1, \ldots . \tag{49}$$

We now choose $v_0 = r_0$ and any $w_0 \in \mathbb{C}^N$, $w_0 \neq 0$, as the starting vectors for Algorithm 4.1. Then, by (2), the Lanczos vectors $\{v_j\}_{j=0}^n$ span the Krylov subspace $K_{n+1}(r_0, A)$ in (49). Therefore, any iterate (49) can be represented in the form

$$x_{n+1} = x_0 + V_n z_n, \quad \text{where} \quad z_n \in \mathbb{C}^{n+1}, \tag{50}$$

where $V_n$ is the matrix defined in (38). It remains to select the free parameter vector $z_n$ in (50). Based on the Lanczos matrix (45), $T_n$, and its extended version (46), $T_n^{(e)}$, there are two natural approaches. The first one is to choose $z_n$ as the solution of the linear system

$$T_n z_n = e_1, \quad \text{where} \quad e_1 := [\,1 \quad 0 \quad \cdots \quad 0\,]^T \in \mathbb{R}^{n+1}. \tag{51}$$

To motivate (51), we remark that, by the first relation in (7), the residual vector $r_{n+1} := b - Ax_{n+1}$ corresponding to any iterate (50) satisfies

$$r_{n+1} = r_0 - AV_n z_n = V_n(e_1 - T_n z_n) - v_{n+1}(z_n)_n. \tag{52}$$

Here, $e_1$ is the first unit vector defined in (51), and $(z_n)_n$ denotes the last component of the vector $z_n$. Hence the choice (51) of $z_n$ just eliminates all components along $V_n$ in the representation (52) of $r_{n+1}$. It can be shown that the resulting iterative procedure is mathematically equivalent to the classical *biconjugate gradient algorithm* (BCG) [23], provided that no look-ahead steps are performed in the underlying Lanczos Algorithm 4.1. The problem with this first approach is that the coefficient matrix $T_n$ in (51) may be singular or ill-conditioned. In fact, this is one reason for the typical erratic convergence behavior of BCG with wild oscillations in the residual norm.

In order to obtain a method with better numerical and theoretical properties than BCG, Freund and Nachtigal [12] suggested to choose $z_n$ as the solution of the least-squares problem

$$\|f_n - \Omega_n T_n^{(e)} z_n\| = \min_{z\in\mathbb{C}^{n+1}} \|f_n - \Omega_n T_n^{(e)} z\|. \tag{53}$$

Here, $f_n := [\,\|r_0\| \quad 0 \quad \cdots \quad 0\,]^T \in \mathbb{R}^{n+2}$, and $\Omega_n$ is a diagonal scaling matrix given by $\Omega_n := \operatorname{diag}(\|v_0\|, \|v_1\|, \ldots, \|v_{n+1}\|)$. Note that, in view of (47), the coefficient matrix $\Omega_n T_n^{(e)}$ of (53) has full column rank $n+1$, and thus (53) always has a unique solution $z_n$. The motivation for this choice of $z_n$ is as follows. By inserting the scaling matrix $\Omega_n$, we can rewrite the formula (52) for the residual vector of any iterate (50) as follows:

$$r_{n+1} = V_{n+1}(\Omega_n)^{-1}(f_n - \Omega_n T_n^{(e)} z_n). \tag{54}$$

Hence, in view of (53), the iterate $x_{n+1}$ is characterized by a minimization of the second factor in the representation (54) of its residual vector $r_{n+1}$. This is called a *quasi-minimal residual* (QMR) property, and the resulting iterative scheme for solving linear systems (48) is the QMR method. We refer the reader to [12, 13, 14] for a detailed description of two different implementations of the QMR method. Theoretical properties of the QMR method, such as error bounds, are derived in [8, 12].

### 5.3 Padé Approximation of Transfer Functions

Let $A \in \mathbb{C}^{N\times N}$ and $b, c \in \mathbb{C}^N$ with $b, c \neq 0$ be given, and let $H$ be the rational function

$$H(\sigma) \equiv c^T (I - \sigma A)^{-1} b \equiv \sum_{j=0}^{\infty} c^T A^j b \, \sigma^j. \tag{55}$$

Functions of the type (55) arise as transfer functions, for instance, in circuit simulation where $H$ describes the Laplace-domain impulse response of electronic circuits; see, e.g., [7]. Often, $N$ is so large that the evaluation of $H$ itself becomes prohibitive, and instead, one approximates $H$ by a *Padé approximant* $H_n$ for some $n \ll N$. Here, $H_n = p_n/q_{n+1}$, $p_n \in \mathcal{P}_n$, $q_{n+1} \in \mathcal{P}_{n+1}$, is the rational function whose Taylor expansion about $\sigma = 0$ agrees with the Taylor expansion of $H$ in at least the first $2(n+1)$ terms, *i.e.*, $H_n(\sigma) = H(\sigma) + \mathcal{O}(\sigma^{2(n+1)})$. It is well known [17, 18, 19] that $H_n$ can be obtained from the Lanczos process.

Indeed, let $T_n$ be the block-tridiagonal matrix (45) and $D_n$ be the block-diagonal matrix (27) (with blocks $\delta^{(k)}$ given by (36)) that are generated by running $n$ steps of the look-ahead Lanczos Algorithm 4.1 (with starting vectors $v_0 = b$ and $w_0 = c$). Using (7) and (44), one readily verifies that

$$c^T A^j b = \frac{c^T b}{e_1^T D_n e_1} \, e_1^T D_n T_n^j e_1, \quad j = 0, 1, \ldots, 2n+1,$$

where $e_1$ is the first unit vector in $\mathbb{R}^{n+1}$, and together with (55), it follows that

$$H_n(\sigma) \equiv \frac{c^T b}{e_1^T D_n e_1} \, e_1^T D_n (I - \sigma T_n)^{-1} e_1 \equiv \sum_{j=0}^{2n+1} c^T A^j b \, \sigma^j + \mathcal{O}(\sigma^{2(n+1)})$$

is the desired Padé approximant. The resulting PVL algorithm for computing **P**adé approximants **V**ia the look-ahead **L**anczos process, and the application of PVL in circuit simulation is described in detail in [7]. We illustrate the behavior of the PVL algorithm with one typical example taken from [7]. In Figure 1, we show the exact impulse response (solid line) and the approximation obtained after 60 Lanczos steps (dash-dotted line) for the PEEC circuit, which consists of 2100 capacitors, 172 inductors, and 6990 inductive couplings. Notice the excellent agreement of the two curves over a wide frequency range.

## 6 Inverse Block-Factorization of Hankel Matrices

In this section, as a second application of Algorithm 3.3, we sketch a look-ahead procedure for inverse Hankel factorization.

Let $\{H_n\}_{n=0}^{\infty}$ be a sequence of Hankel matrices (16), and let $\{\varphi_n\}_{n=0}^{\infty}$ be the corresponding sequence (22) of FOPs and quasi-FOPs generated by Algorithm 3.3. Recall that each

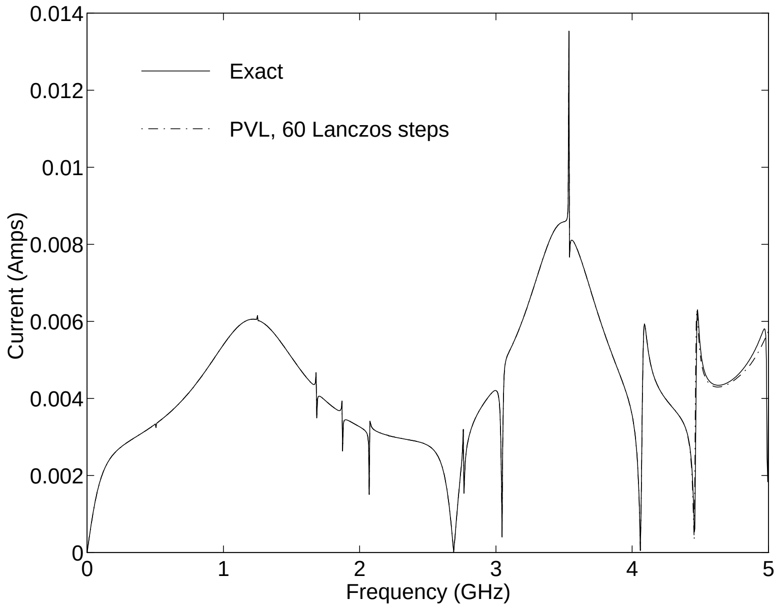


FIG. 1. Simulation of the PEEC circuit, 60 Lanczos steps

polynomial $\varphi_n$ is monic, and hence it can be written as

$$\varphi_n(\lambda) \equiv [\,1 \quad \lambda \quad \cdots \quad \lambda^n\,]\, u_n, \quad \text{where} \quad u_n := \begin{bmatrix} u_{0n} \\ \vdots \\ u_{n-1,n} \\ 1 \end{bmatrix} \in \mathbb{C}^{n+1}. \tag{56}$$

Moreover, we denote by $U_n := [\,u_{ij}\,]_{i,j=0,\ldots,n}$ the unit upper triangular matrix whose entries are just the coefficients of the polynomials $\{\varphi_j\}_{j=0}^n$. Then, by means of (17), (18), and (56), we can rewrite the block-orthogonality conditions (26) in terms of $H_n$ and $U_n$. The result is an *inverse block-factorization* of $H_n$ of the form

$$U_n^T H_n U_n = D_n, \tag{57}$$

where $D_n$ is the symmetric block-diagonal matrix defined in (27). Furthermore, setting

$$U^{(k)} := \begin{cases} [\,u_{ij}\,]_{i=0,\cdots,n_{k+1}-1;j=n_k,\cdots,n_{k+1}-1}\,, & \text{if } 0 \le k < l, \\ [\,u_{ij}\,]_{i=0,\cdots,n;j=n_l,\cdots,n}\,, & \text{if } k = l, \end{cases}$$

the blocks $\Phi^{(k)}$ in (25) can be represented as follows:

$$\Phi^{(k)}(\lambda) \equiv [\,1 \quad \lambda \quad \cdots \quad \lambda^n\,]\, U^{(k)}, \quad k = 0, 1, \ldots, l. \tag{58}$$

Adopting the notation introduced in Section 4, we call the vector $u_n$ in (56) *regular* if it corresponds to a regular FOP $\varphi_n$, and $u_n$ is said to be an *inner* vector if $\varphi_n$ is a quasi-FOP.

Finally, using (56) and (58) and setting $\xi_i^{(n)} = 0$ for all $i, n$ in (32), we can rewrite Algorithm 3.3 in terms of the matrices $U_n$ and $D_n$ defining the inverse block-factorization (57) of $H_n$. This leads to the following algorithm, which was first proposed by Freund and Zha [15]. An outline of this algorithm is as follows.

ALGORITHM 6.1 (Sketch of a look-ahead algorithm for inverse Hankel factorization [15]).

0) Set $u_0 = 1$, $U^{(0)} = u_0$, $\delta^{(0)} = (U^{(0)})^T H_0 U^{(0)}$, $n_0 = l = 0$, $U^{(-1)} = 0$, and $\delta^{(-1)} - 1$.

For $n = 0, 1, \ldots,$ do:

1) Decide whether to construct $u_{n+1}$ as a regular or as an inner vector and go to 2) or 3), respectively.

2) (*Regular step*) Compute $\alpha_n$ and $\beta_n$ by solving

$$\delta^{(l)}\alpha_n = \begin{bmatrix} U^{(l)} \\ 0 \end{bmatrix}^T H_{n+1} \begin{bmatrix} 0 \\ u_n \end{bmatrix} \quad \text{and} \quad \delta^{(l-1)}\beta_n = \begin{bmatrix} U^{(l-1)} \\ 0 \end{bmatrix}^T H_{n+1} \begin{bmatrix} 0 \\ u_n \end{bmatrix},$$

respectively, and set

$$u_{n+1} = \begin{bmatrix} 0 \\ u_n \end{bmatrix} - \begin{bmatrix} U^{(l)} \\ 0 \end{bmatrix} \alpha_n - \begin{bmatrix} U^{(l-1)} \\ 0 \end{bmatrix} \beta_n.$$

Set $n_{l+1} = n + 1$, $l = l + 1$, $U^{(l)} = \emptyset$, and go to 4).

3) (*Inner step*) Compute $\beta_n$ by solving

$$\delta^{(l-1)}\beta_n = \begin{bmatrix} U^{(l-1)} \\ 0 \end{bmatrix}^T H_{n+1} \begin{bmatrix} 0 \\ u_n \end{bmatrix} \quad \text{and set} \quad u_{n+1} = \begin{bmatrix} 0 \\ u_n \end{bmatrix} - \begin{bmatrix} U^{(l-1)} \\ 0 \end{bmatrix} \beta_n.$$

4) Set

$$U^{(l)} = \left[\begin{array}{c|c} U^{(l)} & \\ 0 & u_{n+1} \end{array}\right] \quad \text{and} \quad \delta^{(l)} = (U^{(l)})^T H_{n+1} U^{(l)}.$$

For a discussion of the look-ahead strategy used in step 1) and other implementation details of Algorithm 6.1, we refer the reader to [15]. We remark that Algorithm 6.1 can be viewed as an extension of a classical procedure due to Trench [27] for computing inverse factorizations of a sequence of nonsingular Hankel matrices.

## 7 Concluding Remarks

We presented a look-ahead variant of the nonsymmetric Lanczos process that remedies the problem of possible breakdowns and near-breakdowns in the original algorithm proposed by Lanczos. Among the many possible approaches to the look-ahead Lanczos process, we have used the one based on the connection with formally orthogonal polynomials (FOPs) associated with Hankel bilinear forms. The reason for this is twofold. First, this approach appears to be the most elementary. Second, as a by-product, we immediately obtained a look-ahead procedure for computing inverse block-factorizations of Hankel matrices.

We remark that Freund and Zha in [16] used techniques similar to the ones described in this paper to devise a look-ahead procedure for constructing formally biorthogonal polynomials associated with general Toeplitz bilinear forms. The resulting procedure can be used to derive fast algorithms for the factorization of general Toeplitz matrices; see [10, 16].

Finally, we note that Aliaga, Boley, Freund, and Hernandez [1] recently developed a look-ahead Lanczos procedure for generating biorthogonal bases of block Krylov subspaces.

## References

[1] J. I. Aliaga, D. L. Boley, R. W. Freund, and V. Hernandez, *A look-ahead Lanczos algorithm for multiple starting vectors*, in preparation.

[2] D. L. Boley and G. H. Golub, *The nonsymmetric Lanczos algorithm and controllability*, Systems Control Lett., 16 (1991), pp. 97–105.

[3] C. Brezinski, M. Redivo Zaglia, and H. Sadok, *Avoiding breakdown and near-breakdown in Lanczos type algorithms*, Numer. Algorithms, 1 (1991), pp. 261–284.
[4] T. S. Chihara, *An Introduction to Orthogonal Polynomials*, Gordon and Breach, New York, 1978.
[5] J. Cullum and R. A. Willoughby, *A practical procedure for computing eigenvalues of large sparse nonsymmetric matrices*, in Large Scale Eigenvalue Problems, North-Holland, Amsterdam, Holland, 1986.
[6] A. Draux, *Polynômes Orthogonaux Formels – Applications*, Lecture Notes in Mathematics, Vol. 974, Springer-Verlag, Berlin, 1983.
[7] P. Feldmann and R. W. Freund, *Efficient linear circuit analysis by Padé approximation via the Lanczos process*, Technical Report No. 11274-940217-02TM, AT&T Bell Laboratories, Murray Hill, NJ, 1994.
[8] R. W. Freund, *Quasi-kernel polynomials and convergence results for quasi-minimal residual iterations*, in Numerical Methods of Approximation Theory, Birkhäuser, Basel, 1992.
[9] ——, *The look-ahead Lanczos process for large nonsymmetric matrices and related algorithms*, in Linear Algebra for Large Scale and Real-Time Applications, Kluwer Academic Publishers, Dordrecht, Holland, 1993.
[10] ——, *A look-ahead Bareiss algorithm for general Toeplitz matrices*, Numer. Math., to appear (1994).
[11] 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 (1993), pp. 137–158.
[12] R. W. Freund and N. M. Nachtigal, QMR: *a quasi-minimal residual method for non-Hermitian linear systems*, Numer. Math., 60 (1991), pp. 315–339.
[13] ——, *Implementation details of the coupled* QMR *algorithm*, in Numerical Linear Algebra, W. de Gruyter, Berlin, 1993.
[14] ——, *An implementation of the* QMR *method based on coupled two-term recurrences*, SIAM J. Sci. Comput., 15 (1994), pp. 313–337.
[15] R. W. Freund and H. Zha, *A look-ahead algorithm for the solution of general Hankel systems*, Numer. Math., 64 (1993), pp. 295–321.
[16] ——, *Formally biorthogonal polynomials and a look-ahead Levinson algorithm for general Toeplitz systems*, Linear Algebra Appl., 188/189 (1993), pp. 255–303.
[17] W. B. Gragg, *Matrix interpretations and applications of the continued fraction algorithm*, Rocky Mountain J. Math., 4 (1974), pp. 213–225.
[18] W. B. Gragg and A. Lindquist, *On the partial realization problem*, Linear Algebra Appl., 50 (1983), pp. 277–319.
[19] M. H. Gutknecht, *A completed theory of the unsymmetric Lanczos process and related algorithms, Part I*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 594–639.
[20] ——, *A completed theory of the unsymmetric Lanczos process and related algorithms, Part II*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 15–58.
[21] W. D. Joubert, *Lanczos methods for the solution of nonsymmetric systems of linear equations*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 926–943.
[22] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Res. Nat. Bur. Standards, 45 (1950), pp. 255–282.
[23] ——, *Solution of systems of linear equations by minimized iterations*, J. Res. Nat. Bur. Standards, 49 (1952), pp. 33–53.
[24] B. N. Parlett, *Reduction to tridiagonal form and minimal realizations*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 567–593.
[25] B. N. Parlett, D. R. Taylor, and Z. A. Liu, *A look-ahead Lanczos algorithm for unsymmetric matrices*, Math. Comp., 44 (1985), pp. 105–124.
[26] D. R. Taylor, *Analysis of the look ahead Lanczos algorithm*, Ph.D. thesis, University of California, Berkeley, California, 1982.
[27] W. F. Trench, *An algorithm for the inversion of finite Hankel matrices*, J. Soc. Indust. Appl. Math., 13 (1965), pp. 1102–1107.