SIAM J. MATRIX ANAL. APPL.
Vol. 21, No. 4, pp. 1051–1078

LOOK-AHEAD PROCEDURES FOR LANCZOS-TYPE PRODUCT METHODS BASED ON THREE-TERM LANCZOS RECURRENCES*

MARTIN H. GUTKNECHT† AND KLAUS J. RESSEL‡

Abstract. Lanczos-type product methods for the solution of large sparse non-Hermitian linear systems either square the Lanczos process or combine it with a local minimization of the residual. They inherit from the underlying Lanczos process the danger of breakdown. For various Lanczos-type product methods that are based on the Lanczos three-term recurrence, look-ahead versions are presented which avoid such breakdowns or near-breakdowns at the cost of a small computational overhead. Different look-ahead strategies are discussed and their efficiency is demonstrated by several numerical examples.

Key words. Lanczos-type product methods, look-ahead, iterative methods, non-Hermitian matrices, sparse linear systems

AMS subject classifications. 65F15, 65F10

PII. S0895479896306744

1. Introduction. Lanczos-type product methods (LTPMs) such as (Bi)CGS [42], BiCGStab [44], BiCGStab2 [22], and BiCGStab(ℓ) [38], [41] are among the most efficient methods for solving large systems of linear equations $Ax = b$ with nonsymmetric sparse system matrix $A \in \mathbb{C}^{N\times N}$. Compared to the biconjugate gradient (BiCG) method, they have the advantage of converging roughly twice as fast and of not requiring a routine for applying the adjoint system matrix A^H to a vector. Nevertheless, they inherit from BiCG the short recurrence formulas for generating the approximations x_k and the corresponding residuals $r_k := b - Ax_k$.

As for BiCG, where the convergence can be smoothed by applying the quasi-minimal residual (QMR) method [16] or the local minimum residual process (MR smoothing) [37], [47], product methods can be combined with the same techniques [14], [47] to avoid the likely "erratic" convergence behavior [11], [36]; the benefit of these smoothing techniques is disputed, however.

A well-known problem of all methods that make implicit use of the Lanczos polynomials generated by a non-Hermitian matrix is the danger of breakdown. Although exact breakdowns are very rare in practice, it has been observed that near-breakdowns can slow down or even prevent convergence [15]. Look-ahead techniques for the Lanczos process [15], [21], [23], [34], [35], [43] allow us to avoid this problem when we use variants of the BiCG method with or without the mentioned smoothing techniques. However, the general look-ahead procedures that have so far been proposed for (Bi)CGS or other LTPMs are either limited to exact breakdowns [8], [9] or are based on different look-ahead recursions. In fact, the look-ahead steps proposed by Brezinski and Redivo Zaglia [5] to avoid near-breakdowns in the conjugate gradient squared (CGS) method and those in [6], which are applicable to all LTPMs, are based on the so-called BSMRZ algorithm, which is itself based on a generalization of cou-

*Received by the editors July 17, 1996; accepted for publication (in revised form) by Ž. Strakoš June 4, 1999; published electronically March 21, 2000.

http://www.siam.org/journals/simax/21-4/30674.html

†Seminar for Applied Mathematics, ETH Zürich, ETH-Zentrum HG, CH-8092 Zürich, Switzerland (mhg@sam.math.ethz.ch). Formerly at the Swiss Center for Scientific Computing (CSCS/SCSC).

‡German Aerospace Research Establishment (DLR), German Remote Sensing Data Center (DFD), D-82234 Oberpfaffenhofen, Germany (kjr@dfd.dlr.de). Formerly at CSCS/SCSC.

pled recurrences (different from those of the standard BiOMin and BiODir versions of the BiCG method) for implementing the Lanczos method. For a theoretical comparison of the two approaches we refer to [25, section 19]. Recently, a number of further "look-ahead-like" algorithms for Lanczos-type solvers have been proposed by Ayachour [1], Brezinski, Redivo Zaglia, and Sadok [7], Graves-Morris [19], and Ziegler [48]. Their discussion is beyond the scope of this paper, but it seems that [1], [7], [48] are restricted to exact breakdowns.

In this paper, we start from the approach in [15] and [23, section 9] and derive an alternative look-ahead procedure for LTPM algorithms that make use of the Lanczos three-term recurrences. Compared to the standard coupled two-term recurrences they have the advantage of being simpler to handle with regard to look-ahead, since they are only affected by one type of breakdown. In contrast to the first version of this work [26], we also capitalize upon an enhancement for the look-ahead Lanczos algorithm pointed out by Hochbruck recently (see [28], [25]), which is here adapted to LTPMs. Other improvements help to further reduce the overhead and stabilize the process.

Starting with an initial approximation x_0 and a corresponding initial residual $y_0 := b - Ax_0$, LTPMs generate in n steps a basis of the $2n$-dimensional Krylov space $\mathcal{K}_{2n} := \mathcal{K}_{2n}(A, y_0)$ in such a way that the even indexed basis vectors are of the form

$$\tau_n(A)\rho_n(A)y_0, \tag{1.1}$$

where ρ_n is the nth Lanczos polynomial (see below) and τ_n is another suitably chosen polynomial of exact degree n. In the algorithms we discuss, these vectors will be either the residual of the nth approximation x_n or a scalar multiple of it. By allowing them to be multiples of the residuals, that is, by considering so-called unnormalized [20] or inconsistent [25] Krylov space solvers, we avoid the occurrence of pivot (or ghost) breakdowns. (For the various types of breakdowns and their connection to the block structure of the Padé table see [20], [24], [30], [31]. In the setting of [5], [6], they have been addressed particularly in [4].)

More generally, we define a doubly indexed sequence of *product vectors* w_n^l by

$$w_n^l := \tau_l(A)y_n := \tau_l(A)\rho_n(A)y_0. \tag{1.2}$$

The aim is to find in the $(n+1)$th step of an LTPM an improved approximation x_{n+1} by computing a new product vector w_{n+1}^{n+1} from previously determined ones in a stable way. To visualize the progression of an algorithm and the recurrences it uses we arrange the product vectors w_n^l in a w-table.[1] Its n-axis points downward and its l-axis to the right. We describe then how an algorithm moves in this table from the upper left corner downward to the right.

This paper is organized as follows. In section 2 we review the look-ahead Lanczos process. Versions based on the Lanczos three-term recurrences of various LTPMs are introduced in section 3. In section 4 we present look-ahead procedures for these LTPMs and analyze their computational overhead, and in section 5 several look-ahead strategies are discussed. Our preferred way of applying the look-ahead procedures to obtain the solution of a linear system is presented in section 6. In section 7 the efficiency of the proposed algorithms is demonstrated by numerical examples, and in section 8 we draw some conclusions.

[1] The w-table is different from the scheme introduced by Sleijpen and Fokkema [38] for BiCGStab(ℓ).

2. The look-ahead Lanczos process. The primary aim of the Lanczos process [32] is the construction of a pair of biorthogonal bases for two nested sequences of Krylov spaces. Given $A \in \mathbb{C}^{N\times N}$ and a pair of starting vectors, $(\widetilde{y}_0, y_0)$, the Lanczos biorthogonalization (BiO) algorithm generates a pair of finite sequences, $\{\widetilde{y}_n\}_{n=0}^{\nu}$, $\{y_n\}_{n=0}^{\nu}$ of *left* and *right Lanczos vectors* such that

$$
\begin{aligned}
y_n \in \mathcal{K}_{n+1} &:= \operatorname{span}\{y_0, Ay_0, \ldots, A^n y_0\}, \\
\widetilde{y}_n \in \widetilde{\mathcal{K}}_{n+1} &:= \operatorname{span}\{\widetilde{y}_0, A^H\widetilde{y}_0, \ldots, (A^H)^n\widetilde{y}_0\},
\end{aligned} \tag{2.1}
$$

and

$$
\langle \widetilde{y}_m, y_n\rangle = \begin{cases} 0 & \text{if} \quad m \neq n \ \text{ or } \ m = n = \nu, \\ \delta_n \neq 0 & \text{if} \quad m = n < \nu, \end{cases} \tag{2.2}
$$

or, equivalently,

$$
\widetilde{y}_n \perp \mathcal{K}_n, \quad \widetilde{\mathcal{K}}_n \perp y_n. \tag{2.3}
$$

Here, $\langle\cdot,\cdot\rangle$ denotes the inner product in $\mathbb{C}^N$, which we choose to be linear in the second argument, and $\perp$ indicates the corresponding orthogonality.

The sequence of pairs of Lanczos vectors can be constructed by the three-term recursions

$$
\begin{aligned}
y_{n+1} &= (Ay_n - y_n\alpha_n - y_{n-1}\beta_n)/\gamma_n, \\
\widetilde{y}_{n+1} &= (A^H\widetilde{y}_n - \widetilde{y}_n\overline{\alpha}_n - \widetilde{y}_{n-1}\overline{\beta}_n)/\overline{\gamma}_n,
\end{aligned} \tag{2.4}
$$

with coefficients α_n and β_n that are determined from the orthogonality condition (2.2) and nonvanishing scale factors γ_n that can be chosen arbitrarily. Choosing $\gamma_n = -\alpha_n - \beta_n$ would allow us to consider the right Lanczos vectors y_n as residuals and to update the iterates x_n particularly simply. But this choice also introduces the possibility of a breakdown due to $\alpha_n + \beta_n = 0$. Therefore, we suggest in section 6 a different way of defining iterates.

From (2.4) it follows directly that the Lanczos vectors can be written in the form

$$
y_n = \rho_n(A)y_0, \tag{2.5}
$$

$$
\widetilde{y}_n = \overline{\rho}_n(A^H)\widetilde{y}_0, \tag{2.6}
$$

where ρ_n denotes the nth Lanczos polynomial. Since we aim here at LTPMs, we consider for the Krylov spaces $\widetilde{\mathcal{K}}_n$ more general basis vectors of the form

$$
\widetilde{z}_n = \overline{\tau}_n(A^H)\widetilde{z}_0 \tag{2.7}
$$

with arbitrary but suitably chosen polynomials τ_n of exact degree n and $\widetilde{z}_0 := \widetilde{y}_0$. In general, $\widetilde{z}_{n+1} \perp \mathcal{K}_{n+1}$ will no longer hold, but $\widetilde{\mathcal{K}}_{n+1} \perp y_{n+1}$ can still be attained by enforcing $\widetilde{z}_{n-1} \perp y_{n+1}$ and $\widetilde{z}_n \perp y_{n+1}$, which means choosing the coefficients α_n and β_n in (2.4) in the following way: if $n > 0$ we need

$$
0 = \langle \widetilde{z}_{n-1}, y_{n+1}\rangle = \left(\langle \widetilde{z}_{n-1}, Ay_n\rangle - \langle \widetilde{z}_{n-1}, y_n\rangle\alpha_n - \langle \widetilde{z}_{n-1}, y_{n-1}\rangle\beta_n\right)/\gamma_n,
$$

but since $\langle \widetilde{z}_{n-1}, y_n\rangle = 0$, we have, with $\delta_n^\tau := \langle \widetilde{z}_n, y_n\rangle$,

$$
\beta_n = \frac{\langle \widetilde{z}_{n-1}, Ay_n\rangle}{\langle \widetilde{z}_{n-1}, y_{n-1}\rangle} = \frac{\langle \widetilde{z}_{n-1}, Ay_n\rangle}{\delta_{n-1}^\tau}. \tag{2.8}
$$

When $n = 0$, we set $\beta_0 = 0$. Similarly, from the orthogonality condition $\langle \widetilde{z}_n, y_{n+1}\rangle = 0$ we obtain

$$\alpha_n = \frac{\langle \widetilde{z}_n, Ay_n - y_{n-1}\beta_n\rangle}{\langle \widetilde{z}_n, y_n\rangle} = \frac{\langle \widetilde{z}_n, Ay_n - y_{n-1}\beta_n\rangle}{\delta_n^\tau}. \tag{2.9}$$

Equations (2.7)–(2.9) and the first recursion of (2.4) specify what one might call the *one-sided* Lanczos process, formulated in [36] to derive LTPMs.

Clearly, this recursive process terminates with $y_\nu = 0$ or $\widetilde{z}_\nu = 0$, or it breaks down with $\delta_\nu^\tau = 0$ and $y_\nu \neq 0$, $\widetilde{z}_\nu \neq 0$. The *look-ahead* Lanczos process [23, section 9], [15] overcomes such a breakdown if curable, i.e., if for some k, $\langle \widetilde{z}_\nu, A^k y_\nu\rangle \neq 0$. However, its role is not restricted to treating such exact breakdowns with $\delta_n^\tau = 0$: it allows us to continue the BiO process whenever for stability reasons we choose to enforce the orthogonality condition (2.2) only partially for a couple of steps. We will come back later to the conditions that make us start such a look-ahead phase. In the look-ahead Lanczos process the price we have to pay is that the Gramian matrix $D := (\langle \widetilde{z}_m, y_n\rangle)$ becomes block lower triangular instead of triangular (as in the generic one-sided Lanczos algorithm) or diagonal (as in the generic two-sided Lanczos algorithm). In other words, we replace the conditions $\widetilde{\mathcal{K}}_n \perp y_n$ and $\delta_n^\tau \neq 0$ by

$$\widetilde{\mathcal{K}}_{n_j} \perp y_n \quad \text{if } n \geq n_j \ (j = 0, \ldots, J) \tag{2.10}$$

and

$$D_j := (\langle \widetilde{z}_k, y_l\rangle)_{k,l=n_j}^{n_{j+1}-1} \quad \text{nonsingular if} \quad j < J, \tag{2.11}$$

where $0 = n_0 < n_1 < \cdots < n_J = \nu \leq N$ is the subsequence of indices of (well-conditioned) *regular* Lanczos vectors y_{n_j}; for the other indices the vectors are called *inner* vectors. Likewise, we refer to n as a *regular index* if $n = n_j$ for some j, while n is called an *inner index* otherwise. Note that there is considerable freedom in choosing the subsequence $\{n_j\}_{j=0}^J$: for example, we might request that the smallest singular value $\sigma_{\min}(D_j)$ be sufficiently large.

The sequence $\{y_n\}$ is determined by the condition $\widetilde{\mathcal{K}}_{n_j} \perp y_n$ if $n_j \leq n < n_{j+1}$ and hence it does not depend directly on the sequence $\{\widetilde{z}_n\}$ but only on the spaces $\widetilde{\mathcal{K}}_{n_j}$ that are spanned by the first n_j elements of $\{\widetilde{z}_n\}$. On the other hand, the smallest singular value of each D_j depends on the basis chosen, a fact that one should take into account. Forming the blocks

$$Y_j := \begin{bmatrix} y_{n_j} & y_{n_j+1} & \cdots & y_{n_{j+1}-1} \end{bmatrix}, \quad \widetilde{Z}_j := \begin{bmatrix} \widetilde{z}_{n_j} & \widetilde{z}_{n_j+1} & \cdots & \widetilde{z}_{n_{j+1}-1} \end{bmatrix},$$

we can express (2.10) as

$$\widetilde{Z}_l^H Y_j = \begin{cases} D_j & \text{if } l = j, \\ 0 & \text{if } l < j. \end{cases} \tag{2.12}$$

The look-ahead step size is in the following denoted by $h_j := n_{j+1} - n_j$.

In the look-ahead case the Lanczos vectors $\{y_n\}$ can be generated by the recursion [23, section 9], [15]:

$$y_{n+1} = (Ay_n - \widehat{Y}_j\alpha_n - Y_{j-1}\beta_n)/\gamma_n \qquad (n_j < n + 1 \leq n_{j+1}), \tag{2.13}$$

where $\widehat{Y}_j := \left[\begin{array}{ccc} y_{n_j} & \dots & y_n \end{array}\right]$ denotes the not yet fully completed j-block if $n+1$ is an inner index, while $\widehat{Y}_j = Y_j$ if $n+1$ is regular, that is, if $n+1 = n_{j+1}$. The coefficient vector β_n is determined by the condition

$$0 = \widetilde{Z}_{j-1}^H y_{n+1} = \widetilde{Z}_{j-1}^H A y_n - \widetilde{Z}_{j-1}^H Y_{j-1}\beta_n,$$

thus,

$$\beta_n = D_{j-1}^{-1}\widetilde{Z}_{j-1}^H A y_n. \tag{2.14}$$

The coefficient vector α_n can be chosen arbitrarily if $n+1$ is an inner index. Obviously, the choice $\alpha_n = 0$ yields the cheapest recursion, but we may gain numerical stability by choosing $\alpha_n \neq 0$. On the other hand, if $n+1$ is a regular index, α_n results from the condition

$$0 = \widetilde{Z}_j^H y_{n+1} = \widetilde{Z}_j^H A y_n - \widetilde{Z}_j^H Y_j \alpha_n - \widetilde{Z}_j^H Y_{j-1}\beta_n,$$

that is,

$$\alpha_n = D_j^{-1}\widetilde{Z}_j^H \left(A y_n - Y_{j-1}\beta_n\right). \tag{2.15}$$

Recently, it was pointed out by Hochbruck [28] that the recursion (2.13) can be simplified since the contribution of the older block Y_{j-1} can be represented by multiples of a single *auxiliary vector* y'_{j-1}. This is due to the fact (noticed in [23]) that the matrix made up of the coefficient vectors $\beta_{n_j}, \dots, \beta_{n_{j+1}-1}$ for block j is of rank one; see also [25, section 19]. This simplification has also been capitalized upon in the look-ahead Hankel solver of Freund and Zha [17], which is closely related to the look-ahead Lanczos algorithm.

In particular, due to (2.10) or (2.12) we have

$$\langle \widetilde{z}_{l+1}, y_n\rangle = 0 \quad \text{and} \quad \langle \widetilde{z}_l, A y_n\rangle = 0 \quad \text{if} \;\; l < n_j - 1 < n. \tag{2.16}$$

Therefore, (2.14) simplifies to

$$\beta_n = D_{j-1}^{-1}\widetilde{Z}_{j-1}^H A y_n = D_{j-1}^{-1}\,\ell\,\widetilde{z}_{n_j-1}^H A y_n =: D_{j-1}^{-1}\ell\,\beta'_n, \tag{2.17}$$

where $\ell := [0, \dots, 0, 1]^T$ and

$$\beta'_n := \langle \widetilde{z}_{n_j-1}, A y_n\rangle. \tag{2.18}$$

Introducing the *auxiliary vector*

$$y'_{j-1} := Y_{j-1}\left(D_{j-1}^{-1}\ell\right), \tag{2.19}$$

we have

$$Y_{j-1}\beta_n = \left(Y_{j-1}D_{j-1}^{-1}\ell\right)\left(\widetilde{z}_{n_j-1}^H A y_n\right) = y'_{j-1}\beta'_n, \tag{2.20}$$

so that the recurrence (2.13) becomes

$$y_{n+1} = \left(A y_n - \widehat{Y}_j\alpha_n - y'_{j-1}\beta'_n\right)/\gamma_n \quad (n_j < n+1 \le n_{j+1}) \tag{2.21}$$

and (2.15) changes to

$$\alpha_n = D_j^{-1}\widetilde{Z}_j^H \left(A y_n - y'_{j-1}\beta'_n\right). \tag{2.22}$$

3. LTPMs. LTPMs are based on two ideas. The first is to derive for a certain sequence of product vectors w_{n+1}^{l+1} recursion formulas that involve only previously computed product vectors, so that there is no need to explicitly compute the vectors $\widetilde{z}_l$ and y_n. By multiplying the three-term recursion (2.4) for the right Lanczos vectors y_n by $\tau_l(A)$ we obtain

$$w_{n+1}^l = \left(Aw_n^l - w_n^l\alpha_n - w_{n-1}^l\beta_n\right)/\gamma_n. \tag{3.1}$$

This recursion can be applied to move forward in the vertical direction in the w-table. To obtain a formula for proceeding in the horizontal direction, the recursion for the chosen polynomials $\tau_l(\zeta)$ is capitalized upon in an analogous way.

The second basic idea is to rewrite the inner products that appear in the Lanczos process in terms of product vectors:

$$\begin{aligned}\langle \widetilde{z}_l, y_n\rangle &= \langle \overline{\tau}_l(A^H)\widetilde{z}_0, y_n\rangle &= \langle \widetilde{z}_0, \tau_l(A)y_n\rangle &= \langle \widetilde{z}_0, w_n^l\rangle,\\ \langle \widetilde{z}_l, Ay_n\rangle &= \langle \overline{\tau}_l(A^H)\widetilde{z}_0, Ay_n\rangle &= \langle \widetilde{z}_0, A\tau_l(A)y_n\rangle &= \langle \widetilde{z}_0, Aw_n^l\rangle.\end{aligned}$$

For the coefficients α_n and β_n of (2.8) and (2.9) this results in

$$\beta_n = \frac{\langle \widetilde{z}_0, Aw_n^{n-1}\rangle}{\langle \widetilde{z}_0, w_{n-1}^{n-1}\rangle}, \quad \alpha_n = \frac{\langle \widetilde{z}_0, Aw_n^n - w_{n-1}^n\beta_n\rangle}{\langle \widetilde{z}_0, w_n^n\rangle}. \tag{3.2}$$

LTPMs still have short recurrence formulas if the polynomials τ_l have a short one. In addition, they have two advantages over BiCG: first, multiplications with the adjoint system matrix A^H are avoided; second, for an appropriate choice of the polynomials $\tau_l(\zeta)$, smaller new residuals $r_l := w_l^l = \tau_l(A)y_l$ can be expected because of a further reduction of y_l by the operator $\tau_l(A)$. Different choices of the polynomials $\tau_l(\zeta)$ lead to different LTPMs. In the following we briefly review some possible choices for these polynomials. We start with the general class where they satisfy a three-term recursion. Since then both the "left" polynomials τ_l and the "right" polynomials ρ_n fulfill a three-term recursion, we say that this is the class of $(3,3)$-*type LTPMs*.

In the following we briefly review some possible choices for these polynomials. For a more detailed discussion and a pseudocode for the resulting algorithms we refer to [36].

3.1. LTPMs based on a three-term recursion for $\{\tau_l\}$: BiOxCheb and BiOxMR2. Assume the polynomials $\tau_l(\zeta)$ are generated by a three-term recursion of the form [22]

$$\tau_{l+1}(\zeta) = (\xi_l + \eta_l\zeta)\,\tau_l(\zeta) + (1-\xi_l)\,\tau_{l-1}(\zeta) \tag{3.3}$$

with $\tau_{-1}(\zeta) \equiv 0$, $\tau_0(\zeta) \equiv 1$, $\xi_0 = 1$, and $\eta_l \neq 0$ for all l. Note that, by induction, τ_l has exact degree l and $\tau_l(0) = 1$ for all l. Hence, the polynomials qualify as residual polynomials of a Krylov space method.

Multiplying y_n by $\tau_{l+1}(A)$ from the left and applying (3.3) yields the horizontal recursion

$$w_n^{l+1} = Aw_n^l\eta_l + w_n^l\xi_l + w_n^{l-1}(1-\xi_l). \tag{3.4}$$

By applying (3.1) and (3.4), Loop 3.1 produces a new product vector w_{n+1}^{n+1} from previously calculated ones, namely w_m^l, $(m,l) := (n,n), (n-1,n), (n,n-1), (n-1,n-1)$, and the product Aw_n^{n-1} also evaluated before. At the end of the loop, besides w_{n+1}^{n+1}, also w_n^{n+1}, w_{n-1}^{n+1}, and Aw_{n+1}^n, which will be needed in the next run through the loop, are available.

Loop 3.1 (general $(3,3)$-type LTPM).

1. *Compute* Aw_n^n *and determine* β_n *and* α_n.
2. *Use* (3.1) *to compute* w_{n+1}^{n-1} *(if* $n > 0$*) and* w_{n+1}^n.
3. *Compute* Aw_{n+1}^n *and determine* ξ_n *and* η_n.
4. *Use* (3.4) *to compute* w_n^{n+1} *and* w_{n+1}^{n+1}.

Next we discuss two ways of choosing the polynomials τ_l, that is, of specifying the recurrence coefficients ξ_l and η_l.

One possibility is to combine the Lanczos process with the Chebyshev method [13], [45] by choosing τ_l as a suitably shifted and scaled Chebyshev polynomial. This combination was suggested in [3], [22], [44]. Let us call the resulting $(3,3)$-type LTPM BiOxCheb. It is well known that these polynomials satisfy a recurrence of the form (3.3). After acquiring some information about the spectrum of the matrix A, for example, by performing a few iterations with another Krylov space method, one can determine the recursion for the scaled and shifted Chebyshev polynomials that correspond to an ellipse surrounding the estimated spectrum [33]. However, the necessity to provide spectral information is often seen as a drawback of this method.

Another idea is to use the coefficients ξ_l and η_l of (3.3) for locally minimizing the norm of the new residual w_{l+1}^{l+1}. This idea is borrowed from the BiCGStab and BiCGStab2 methods [44], [22] reviewed below: ξ_l and η_l are determined by solving the two-dimensional minimization problem

$$\min_{\xi,\eta\in\mathbb{C}} \left\| w_{l+1}^{l-1} + (w_{l+1}^l - w_{l+1}^{l-1})\xi + Aw_{l+1}^l \eta \right\|. \tag{3.5}$$

Introducing the $N \times 2$ matrix $B_{l+1} := \left[\ (w_{l+1}^l - w_{l+1}^{l-1}) \ \middle|\ Aw_{l+1}^l \ \right]$ we can write (3.5) as the least-squares problem

$$\min_{\xi,\eta\in\mathbb{C}} \left\| w_{l+1}^{l-1} + B_{l+1} \begin{bmatrix} \xi \\ \eta \end{bmatrix} \right\|.$$

Therefore, ξ_l and η_l can be computed as the solution of the normal equations

$$\left(B_{l+1}^H B_{l+1}\right) \begin{bmatrix} \xi_l \\ \eta_l \end{bmatrix} = -B_{l+1}^H w_{l+1}^{l-1}. \tag{3.6}$$

In view of the two-dimensional local residual norm minimization performed at every step (except the first one) we call this the BiOxMR2 method.[2] A version based on coupled two-term Lanczos recurrences of this method was introduced by the first author in a talk in Oberwolfach (April 1994). Independently it was as well proposed by Cao [10] and by Zhang [46], whose technical report is dated April 1993. Zhang also considered two-term formulas for τ_l and presented very favorable numerical results.

3.2. LTPMs based on a two-term recursion for $\{\tau_l\}$: BiOStab. In Van der Vorst's BiCGStab [44], the polynomials $\tau_l(\zeta)$ are built up successively as products of polynomials of degree 1:

$$\tau_0(\zeta) \equiv 1, \qquad \tau_{l+1}(\zeta) = (1 - \chi_l\zeta)\tau_l(\zeta) \quad \text{for } l \geq 0. \tag{3.7}$$

[2]The letter x in the name BiOxMR2 reflects the fact that the residual polynomials of this method are the products of the Lanczos polynomials generated by the BiO process with polynomials obtained from a successive two-dimensional minimization of the residual (MR2). Similarly, BiOxCheb means a combination of the BiO process with a Chebyshev process.

Inserting here for ζ the system matrix A and multiplying by y_n from the right yields

$$w_n^{l+1} = w_n^l - Aw_n^l \chi_l. \tag{3.8}$$

The coefficient χ_l is determined by minimizing the norm of $w_{l+1}^{l+1} = w_{l+1}^l - Aw_{l+1}^l \chi_l$, that is, by solving the one-dimensional minimization problem

$$\min_{\chi \in \mathbb{C}} \left\| w_{l+1}^l - Aw_{l+1}^l \chi \right\|, \tag{3.9}$$

which leads to

$$\chi_l = \frac{\langle Aw_{l+1}^l, w_{l+1}^l \rangle}{\langle Aw_{l+1}^l, Aw_{l+1}^l \rangle}. \tag{3.10}$$

Recurrences (3.1) and (3.8) can be used to compute a new product vector w_{n+1}^{n+1} from w_n^n and w_{n-1}^n as described in Loop 3.1 with $\xi_n = 1$ and $\eta_n = -\chi_n$, where χ_n is calculated from (3.10). However, since (3.8) is only a two-term recurrence, there is no need now to compute w_{n+1}^{n-1} in substep 2 of the loop.

We call this algorithm BiOStab since it is a version of BiCGStab that is based on the three-term recurrences of the Lanczos BiO process.[3]

3.3. BiOStab2. BiOStab2 is the version of BiCGStab2 [22] based on the three-term Lanczos recurrences. The polynomials $\tau_l(\zeta)$ satisfy the recursions

$$\begin{array}{rcll} \tau_0(\zeta) & \equiv & 1, & \\ \tau_{l+1}(\zeta) & = & (1 - \chi_l \zeta)\, \tau_l(\zeta) & \text{if } l \text{ is even}, \\ \tau_{l+1}(\zeta) & = & (\xi_l + \eta_l \zeta)\, \tau_l(\zeta) + (1 - \xi_l)\, \tau_{l-1}(\zeta) & \text{if } l \text{ is odd}. \end{array} \tag{3.11}$$

Here χ_l may be obtained by solving the one-dimensional minimization problem (3.9), so χ_l is given by (3.10). However, if $|\chi_l|$ is small, this choice is dangerous since the vector component needed to enlarge the Krylov space becomes negligible [38]. Then some other value of χ_l should be chosen. Except for roundoff, the choice has no effect on later steps, because ξ_l and η_l are determined by solving the two-dimensional minimization problem (3.5).

Multiplying y_n by $\tau_{l+1}(A)$ and applying (3.11) leads to

$$w_n^{l+1} = \begin{cases} w_n^l - Aw_n^l \chi_l & \text{if } l \text{ is even}, \\ Aw_n^l \eta_l + w_n^l \xi_l + w_n^{l-1}(1 - \xi_l) & \text{if } l \text{ is odd}. \end{cases} \tag{3.12}$$

Now, Loop 3.1 applies with $\xi_n = 1$, $\eta_n = -\chi_n$ if n is even, while ξ_n and η_n are chosen as indicated above if n is odd. If n is even, there is no need to compute w_{n+1}^{n-1} in substep 2 of the loop.

3.4. BiO-squared. BiO-squared (BiOS) is obtained by "squaring" the three-term Lanczos process: among the basis vectors generated are those Krylov-space vectors that correspond to the squared Lanczos polynomials. By complementing BiOS with a recursion for Galerkin iterates we will obtain BiOResS, a $(3,3)$-type version of Sonneveld's $(2,2)$-type *CGS method* [42]. The method fits into the framework of LTPMs if we identify

$$\tau_l(\zeta) = \rho_l(\zeta). \tag{3.13}$$

[3]Eijkhout [12] also proposed such a variant of BiCGStab; however, his way of computing the Lanczos coefficients is much too complicated.

The vectors $\widetilde{z}_n = \overline{\tau}_n(A^H)\widetilde{z}_0 = \overline{\rho}_n(A^H)\widetilde{y}_0 = \widetilde{y}_n$ are then exactly the left Lanczos vectors so that now $y_n \perp \widetilde{\mathcal{K}}_n$ as well as $\widetilde{z}_n \perp \mathcal{K}_n$ is fulfilled. Thus, the coefficient α_n in (3.2) simplifies to

$$\alpha_n = \frac{\langle \widetilde{z}_0, Aw_n^n \rangle}{\langle \widetilde{z}_0, w_n^n \rangle}, \tag{3.14}$$

and the w-table becomes symmetric since

$$w_n^l = \rho_l(A)\rho_n(A)y_0 = \rho_n(A)\rho_l(A)y_0 = w_l^n. \tag{3.15}$$

Consequently, we have in analogy to (3.1)

$$w_n^{l+1} = \left(Aw_n^l - w_n^l\alpha_l - w_n^{l-1}\beta_l\right)/\gamma_l. \tag{3.16}$$

These two recursions lead to the following loop.

LOOP 3.2 (BiOS).
1. *Compute* Aw_n^n *and determine* β_n *and* α_n.
2. *Use* (3.1) *to compute* w_{n+1}^{n-1} *(if* $n > 0$*) and* w_{n+1}^n.
3. *Compute* Aw_{n+1}^n.
4. *Use* (3.16) *to compute* w_{n+1}^{n+1}.

Note that we exploit in substep 2 the symmetry of the w-table: the product vector w_{n-1}^n, which is needed for the calculation of w_{n+1}^n by (3.1), is equal to w_n^{n-1} and need not be stored. We also point out that (3.16) is not of the form (3.4); in particular, the coefficients of w_n^{l-1} and w_n^l need not sum up to 1.

4. Look-ahead procedures for LTPMs.

Look-ahead steps in an LTPM serve to stabilize the Lanczos process, the vertical movement in the w-table. Except for BiOS, the recursion formulas for the horizontal movement remain the same. The vertical movement is now in general based on the recurrence formula (2.21) for the Lanczos vectors, but we need to replace these vectors by product vectors w_n^l; see (1.2).

We introduce the blocks of product vectors

$$\widehat{W}_j^l := \begin{bmatrix} w_{n_j}^l & \dots & w_n^l \end{bmatrix}, \quad W_{j-1}^l := \begin{bmatrix} w_{n_{j-1}}^l & \dots & w_{n_j-1}^l \end{bmatrix}$$

and the *auxiliary product vector* $w'^{\,l}_{j-1}$ defined by

$$w'^{\,l}_{j-1} := \tau_l(A)y'_{j-1} = \tau_l(A)Y_{j-1}\left(D_{j-1}^{-1}\ell\right) = W_{j-1}^l\left(D_{j-1}^{-1}\ell\right). \tag{4.1}$$

Again $\widehat{W}_j^l = W_j^l$ if $n+1 = n_{j+1}$ is a regular index, while $\widehat{W}_j^l$ denotes the not yet completed jth block if $n+1$ is an inner index. Now, multiplying (2.21) by $\tau_l(A)$ from the left, we obtain

$$w_{n+1}^l = \left(Aw_n^l - \widehat{W}_j^l\alpha_n - w'^{\,l}_{j-1}\beta'_n\right)/\gamma_n \qquad (n_j < n+1 \le n_{j+1}). \tag{4.2}$$

As in section 3, the coefficient vectors β'_n and, in the regular case, α_n can be expressed in terms of the product vectors by rewriting all inner products in such a way that the part $\overline{\tau}_l(A^H)$ of $\widetilde{z}_l$ on the left side is transferred to the right side of the inner product. For the diagonal blocks D_j of the Gramian this yields

$$D_j = \left[\langle \widetilde{z}_0, w_k^i\rangle\right]_{i,k=n_j}^{n_{j+1}-1}, \tag{4.3}$$

inner case

	n_{j-1}			n_j	$\dots$	l
n_{j-1}	d	d	d			
	d	d	d			
	d	d	d			v′
n_j						V
$\vdots$						V
n						$\underline{\mathrm{V}}$ A
$n+1$						$*$

regular case

	n_{j-1}			n_j	$\dots$	l	n	n_{j+1}
n_{j-1}	d	d	d					
	d	d	d					
	d	d	d	α'	α'	v′	α'	
n_j				D	D	V	D	
$\vdots$				D	D	V	D	
$\vdots$				D	D	V	D	
n				$\underline{\mathrm{D}}$ A	$\underline{\mathrm{D}}$ A	$\underline{\mathrm{V}}$ A	$\underline{\mathrm{D}}$ A	
n_{j+1}						$*$		

FIG. 4.1. *Entries in the w-table needed for the construction of a new inner or regular vector marked by $*$ by recurrence formula* (4.2)*. Here, v′ indicates entries needed for ${w'}_{j-1}^{l}$, V those for $\widehat{W}_j^l$, d those for D_{j-1}, D those additionally required for D_j. Moreover, α' marks auxiliary vectors ${w'}_{j-1}^{k}$ $(n_j \le k < n_{j+1})$ in the formula* (4.5) *for α_n, and A stands for a matrix-vector product (MV) with A in this formula. One such product also appears explicitly in* (4.2).

and β'_n of (2.18) becomes

$$\beta'_n = \widetilde{z}_0^H A w_n^{n_j-1} \quad (n_j < n+1 \le n_{j+1}). \tag{4.4}$$

Likewise, α_n of (2.22) turns into

$$\alpha_n = D_j^{-1} \begin{bmatrix} \langle \widetilde{z}_0, A w_n^{n_j} - {w'}_{j-1}^{n_j} \beta'_n \rangle \\ \vdots \\ \langle \widetilde{z}_0, A w_n^{n_{j+1}-1} - {w'}_{j-1}^{n_{j+1}-1} \beta'_n \rangle \end{bmatrix} \quad (n+1 = n_{j+1}). \tag{4.5}$$

For advancing in the horizontal direction of the w-table we will still use (3.4), (3.8), the combination (3.12), or (3.16). Substituting $A w_n^{n_j-1}$ in (4.4) according to these formulas and letting $\delta_n^{n_j} = \langle \widetilde{z}_0, w_n^{n_j} \rangle$ be the (n_j, n)-element of D, (4.4) simplifies to

$$\beta'_n = \begin{cases} \delta_n^{n_j} / \eta_{n_j-1} & \text{if (3.4) holds for } l = n_j - 1, \\ -\delta_n^{n_j} / \chi_{n_j-1} & \text{if (3.8) holds for } l = n_j - 1, \\ \delta_n^{n_j} \gamma_{n_j-1} & \text{if (3.16) holds for } l = n_j - 1. \end{cases} \tag{4.6}$$

We remark that by using (4.6) instead of $\langle \widetilde{z}_0, A w_n^{n_j-1} \rangle$ we avoid computing and storing the complete W_j^{j-1} block. Now only a few elements of this block are needed; see Figure 4.1. In this figure we display those entries w_n^l and products $A w_n^l$ in the w-table that are needed to compute by (4.2) a new vector w_{n+1}^l marked by $*$.

Note that the horizontal recursions are valid for the auxiliary product vectors ${w'}_{j-1}^{l}$ too: for example, (3.4) and (4.1) imply

$${w'}_{j-1}^{l+1} = A {w'}_{j-1}^{l} \eta_l + {w'}_{j-1}^{l} \xi_l + {w'}_{j-1}^{l-1} (1 - \xi_l), \tag{4.7}$$

while (3.8) and (4.1) provide

$${w'}_{j-1}^{l+1} = {w'}_{j-1}^{l} - A {w'}_{j-1}^{l} \chi_l \,. \tag{4.8}$$

Thus, the auxiliary vector can be updated in the horizontal direction at the cost of one matrix-vector product (MV) per step.

Combining the recursions for the vertical movement with those for the horizontal movement leads to the look-ahead version of an LTPM. (Actually, we will also need to compute approximations to the solution $A^{-1}b$ of the linear system; but we defer this until section 6.) Because of the additional vectors involved in the above recurrence formulas, it seems that such a look-ahead LTPM requires much more computational work and storage than its unstable, standard version. However, the number of look-ahead steps as well as the size of their blocks is small in practice, so that the overhead is moderate. Moreover, we describe in the following for various choices of polynomials $\tau_n(\zeta)$ how MVs that are needed can be computed indirectly by applying the recurrence formulas. In the same way also the values of inner products can be obtained indirectly at nearly no cost.

4.1. Look-ahead LTPMs based on a three-term recursion for $\{\tau_l\}$: LABiOxCheb and LABiOxMR2. For methods incorporating a horizontal three-term recurrence, such as BiOxCheb and BiOxMR2, applying the above principles for look-ahead LTPMs leads to the general Loop 4.1. Note that the first four substeps are identical in both cases. We will see in section 5 that the decision between a regular and an inner loop is made during substep 5.

In Figure 4.2 we display the action of this loop in the w-table, but now we use a different format than before, which specifies what has been known before the current sweep through the loop and what is being computed in this sweep. In particular, the following symbols and indicators are used:

- V indicates that the corresponding product vector is already known;
- A as denominator indicates that the product of the vector represented by V with the matrix A was needed;
- a solid box around a fraction means that this product by A required (or requires) an MV;
- no box around a fraction means that this product can be obtained by applying a recurrence formula;
- a number as entry specifies in which substep of the current sweep this entry is calculated;
- a prime indicates that the vector is an auxiliary one, as defined in (4.1) (these vectors are displayed in the last row of the corresponding block of the w-table);
- A′ as denominator indicates that the product of the vector represented by V′ with the matrix A was needed;
- double primes will indicate that the vector is an auxiliary one of the type defined in (4.11) used in LABiCGS (these vectors are displayed in the lower-right corner of the corresponding block of the w-table);
- S will indicate that the vector or the MV is obtained for free due to the symmetry of the w-table of LABiCGS.

In the first two w-tables of Figure 4.2 we display what happens in a first inner step (at left) and in a regular step (at right) that follows directly a regular step (so that $h_j = 1$). In the second pair of w-tables of Figure 4.2 we consider an inner and a regular step in the case of a large block size. Note that substeps 2–4 of the loop do not appear in the first pair, and that substep 4 would still not be active in the regular loop at the end of a look-ahead step of length $h_j = 3$, while, when $h_j \geq 4$, it becomes active.

For such a look-ahead step of length h_j, *the cost in terms of MVs is* $4h_j - 3$ MVs if $h_j > 1$. In fact, in the first substep of the $h_j - 1$ inner and the one regular

LOOP 4.1 (look-ahead for $(3,3)$-type LTPM). *Let* $\mu := \min\{n_j, n-1\}$.

Inner Loop: ($n_j < n+1 < n_{j+1}$)

1. *Compute* Aw_n^n.
2. *If* $n > n_j$, *use* (4.2) *to compute indirectly* $Aw_{n_j}^n, \ldots, Aw_{n-1}^n$.
3. *If* $n \geq n_j + 2$, *compute* Aw_n^μ.
4. *If* $n \geq n_j + 3$, *use* (3.4) *to compute indirectly* $Aw_n^{\mu+1}, \ldots, Aw_n^{n-2}$.
5. *Use* (4.2) *to compute* $w_{n+1}^n, w_{n+1}^{n-1}, \ldots, w_{n+1}^\mu$.
6. *Compute* Aw_{n+1}^n *and* ξ_n, η_n.
7. *Use* (3.4) *to compute* $w_{n_j}^{n+1}, \ldots, w_{n+1}^{n+1}$.
8. *Compute* $Aw'^{\,n}_{j-1}$ *and use* (4.7) *to compute* $w'^{\,n+1}_{j-1}$.

Regular Loop: ($n+1 = n_{j+1}$)

1. *Compute* Aw_n^n.
2. *If* $n > n_j$, *use* (4.2) *to compute indirectly* $Aw_{n_j}^n, \ldots, Aw_{n-1}^n$.
3. *If* $n \geq n_j + 2$, *compute* Aw_n^μ.
4. *If* $n \geq n_j + 3$, *use* (3.4) *to compute indirectly* $Aw_n^{\mu+1}, \ldots, Aw_n^{n-2}$.
5. *Use* (4.2) *to compute* w_{n+1}^n *and* w_{n+1}^{n-1}.
6. *Compute* Aw_{n+1}^n *and* ξ_n, η_n.
7. *Use* (3.4) *to compute* $w_{n_j}^{n+1}, \ldots, w_{n+1}^{n+1}$.
8. *Compute* $w'^{\,n}_j$ *and* $w'^{\,n+1}_j$ *according to definition* (4.1).

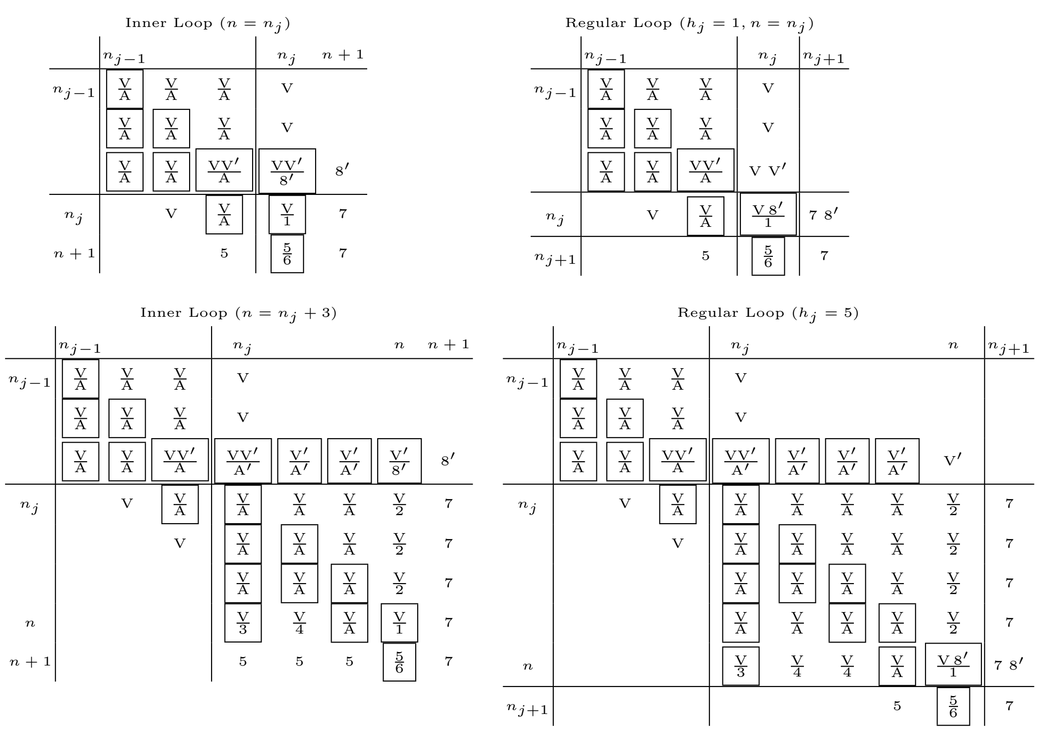

FIG. 4.2. *Action of Loop* 4.1 *(look-ahead for* $(3,3)$*-type LTPM) in the* w*-table. For blocks of different sizes an inner step (at left) and the following regular step (at right) are shown.*

loops h_j MVs are consumed; another $h_j - 2$ MVs are needed in substep 3, a further h_j MVs in substep 6, and the remaining $h_j - 1$ MVs are used in substep 6 of the inner loops. If no look-ahead is needed, that is, if $h_j = 1$, then 2 MVs are required, both in our procedure and in the standard one that does not allow for look-ahead. Hence, in a step of length 2 we have 25% overhead, in a step of length 3 there is 50% overhead, and for even longer steps, which are very rare in practice, the overhead grows gradually toward 100%.

4.2. Look-ahead for BiOStab and BiOStab2. Since the horizontal recurrence (3.8) for BiOStab is only a two-term one, there is no need to compute elements of the second subdiagonal of the w-table as long as we are not in a look-ahead step. In the case of a look-ahead step, this remains true for those elements that lie in a subdiagonal block, but, of course, not for those in a diagonal block. For Loop 4.1 this means that μ simplifies to $\mu := n_j$ and that in substep 5 of a regular loop there is no need to compute w_{n+1}^{n-1}. All the other changes refer to equation numbers or the coefficients ξ_n and η_n. In summary, we obtain Loop 4.2. Again the first four substeps are the same in both cases, and the choice between them will be made in substep 5. In Figure 4.3 we display for this loop the two sections of w-tables that correspond to the second pair in Figure 4.2.

Since those product vectors from Loop 4.1 that are no longer needed in Loop 4.2 were found without an extra MV before, the overhead in terms of MVs remains the same here.

Look-ahead for BiOStab2 could be defined along the same lines by alternating between steps of Loop 4.1 and Loop 4.2. At this point it also becomes clear how to obtain a look-ahead version of BiOStab(ℓ), an algorithm analogous to BiCGStab(ℓ) of Sleijpen and Fokkema [38], but based on the three-term Lanczos process instead of coupled two-term BiCG formulas.

4.3. Look-ahead (bi)conjugate gradient squared: LABiOS. For BiOS, which will be the underlying process for our BiOResS version of BiCGS, the horizontal recurrence (3.4) has to be substituted by the Lanczos recurrence given in (3.16), which may need to be replaced by the look-ahead formula that is analogous to (4.2) with l and n exchanged and with suitably defined auxiliary vectors and blocks of vectors. But since the w-table is symmetric, we can build it up by vertical recursions only and reflections at the diagonal.

We only formulate the recursion for $w_{j-1}'^{\,l}$ as a horizontal one that replaces (4.7): if $n_k < l+1 \le n_{k+1}$,

$$w_{j-1}'^{\,l+1} = \left(A w_{j-1}'^{\,l} - \widehat{W}_{j-1}'^{\,k} \alpha_l - W_{j-1}'^{\,k-1} \beta_l \right) / \gamma_l \,, \tag{4.9}$$

where now

$$\widehat{W}_{j-1}'^{\,k} := \left[\begin{array}{ccc} w_{j-1}'^{\,n_k} & \dots & w_{j-1}'^{\,l} \end{array} \right], \quad W_{j-1}'^{\,k-1} := \left[\begin{array}{ccc} w_{j-1}'^{\,n_{k-1}} & \dots & w_{j-1}'^{\,n_k - 1} \end{array} \right]. \tag{4.10}$$

Recurrence (4.9) follows from the horizontal Lanczos look-ahead recurrences for computing $w_{n_{j-1}}^{l+1}, \dots, w_{n_j - 1}^{l+1}$, derived from (2.13) instead of (2.21), which can be gathered into one recurrence for these h_{j-1} columns of W_{j-1}^{l+1} and then postmultiplied by $D_{j-1}^{-1}\ell$, as in (4.1).

Again, (4.9) can be simplified: if we define for block $k-1$ and the needed values of j the auxiliary vector

$$w_{j-1}''^{\,k-1} := W_{j-1}'^{\,k-1} D_{k-1}^{-1} \ell, \tag{4.11}$$

LOOP 4.2 (look-ahead for BiOStab).

Inner Loop: ($n_j < n+1 < n_{j+1}$)

1. *Compute* Aw_n^n.
2. *If* $n > n_j$, *use* (4.2) *to compute indirectly* $Aw_{n_j}^n, \ldots, Aw_{n-1}^n$.
3. *If* $n \geq n_j + 2$, *compute* $Aw_n^{n_j}$.
4. *If* $n \geq n_j + 3$, *use* (3.8) *to compute indirectly* $Aw_n^{n_j+1}, \ldots, Aw_n^{n-2}$.
5. *Use* (4.2) *to compute* $w_{n+1}^n, w_{n+1}^{n-1}, \ldots, w_{n+1}^{n_j}$.
6. *Compute* Aw_{n+1}^n *and* χ_n.
7. *Use* (3.8) *to compute* $w_{n_j}^{n+1}, \ldots, w_{n+1}^{n+1}$.
8. *Compute* $Aw_{j-1}^{\prime\, n}$ *and use* (4.8) *to compute* $w_{j-1}^{\prime\, n+1}$.

Regular Loop: ($n+1 = n_{j+1}$)

1. *Compute* Aw_n^n.
2. *If* $n > n_j$, *use* (4.2) *to compute indirectly* $Aw_{n_j}^n, \ldots, Aw_{n-1}^n$.
3. *If* $n \geq n_j + 2$, *compute* $Aw_n^{n_j}$.
4. *If* $n \geq n_j + 3$, *use* (3.8) *to compute indirectly* $Aw_n^{n_j+1}, \ldots, Aw_n^{n-2}$.
5. *Use* (4.2) *to compute* w_{n+1}^n.
6. *Compute* Aw_{n+1}^n *and* χ_n.
7. *Use* (3.8) *to compute* $w_{n_j}^{n+1}, \ldots, w_{n+1}^{n+1}$.
8. *Compute* $w_j^{\prime\, n}$ *and* $w_j^{\prime\, n+1}$ *according to definition* (4.1).

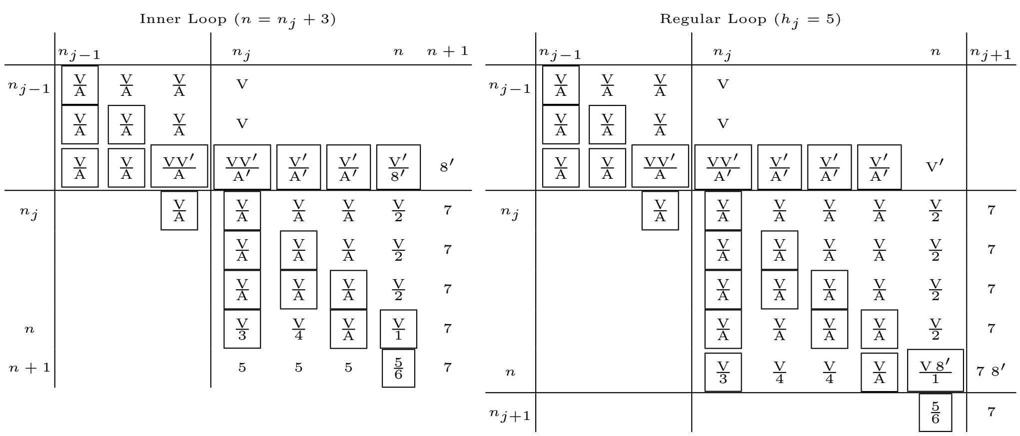

FIG. 4.3. *Action of Loop* 4.2 *(look-ahead for BiOStab) in the w-table. An inner step (at left) and the following regular step (at right) are shown.*

and for each l with $n_k \leq l < n_{k+1}$ the same coefficient $\beta_l' := \delta_l^{n_k}\gamma_{n_k-1}$ as in (4.6), then in view of (2.17) and (4.6),

$$W_{j-1}^{\prime\, k-1}\beta_l = W_{j-1}^{\prime\, k-1} D_{k-1}^{-1} \ell \beta_l' = w_{j-1}^{\prime\prime\, k-1}\beta_l' \,. \tag{4.12}$$

Consequently, (4.9) simplifies to

$$w_{j-1}^{\prime\, l+1} = \left(Aw_{j-1}^{\prime\, l} - \widehat{W}_{j-1}^{\prime\, k}\alpha_l - w_{j-1}^{\prime\prime\, k-1}\beta_l' \right) / \gamma_l \,. \tag{4.13}$$

A step of the resulting LABiOS algorithm is summarized as Loop 4.3.

Here, the decision between a regular and an inner loop is made in substep 3 (which corresponds to substep 5 in Loops 4.1 and 4.2). Now even the first five substeps of the two versions of the loop are identical. In Figure 4.4 we display for LABiOS the same pair of loops as we did in Figure 4.3 for LABiOStab.

LOOP 4.3 (look-ahead for BiOS).

Inner Loop: ($n_j < n+1 < n_{j+1}$)

1. *Compute* Aw_n^n.
2. *If* $n \geq n_j+2$, *use* (4.2) *to compute indirectly* $Aw_{n_j}^n, \ldots, Aw_{n-2}^n$.
3. *Use* (4.2) *to compute* $w_{n+1}^n, w_{n+1}^{n-1}, \ldots, w_{n+1}^{n_j}$.
4. *Compute* Aw_{n+1}^n *and* $Aw'^{\,n}_{j-1}$.
5. *Use* (4.13) *to compute* $w'^{\,n+1}_{j-1}$.
6. *Use* (4.2) *to compute* w_{n+1}^{n+1}.

Regular Loop: ($n+1 = n_{j+1}$)

1. *Compute* Aw_n^n.
2. *If* $n \geq n_j+2$, *use* (4.2) *to compute indirectly* $Aw_{n_j}^n, \ldots, Aw_{n-2}^n$.
3. *Use* (4.2) *to compute* $w_{n+1}^n, w_{n+1}^{n-1}, \ldots, w_{n+1}^{n_j}$.
4. *Compute* Aw_{n+1}^n *and* $Aw'^{\,n}_{j-1}$.
5. *Use* (4.13) *to compute* $w'^{\,n+1}_{j-1}$.
6. *Use definition* (4.1) *to compute* $w'^{\,n_j}_j, \ldots, w'^{\,n_{j+1}-1}_j$ *and use definition* (4.11) *to compute* $w''^{\,j}_j$.
7. *Use* (4.2) *to compute* w_{n+1}^{n+1}.
8. *Use definition* (4.1) *to compute* $w'^{\,n+1}_j$.

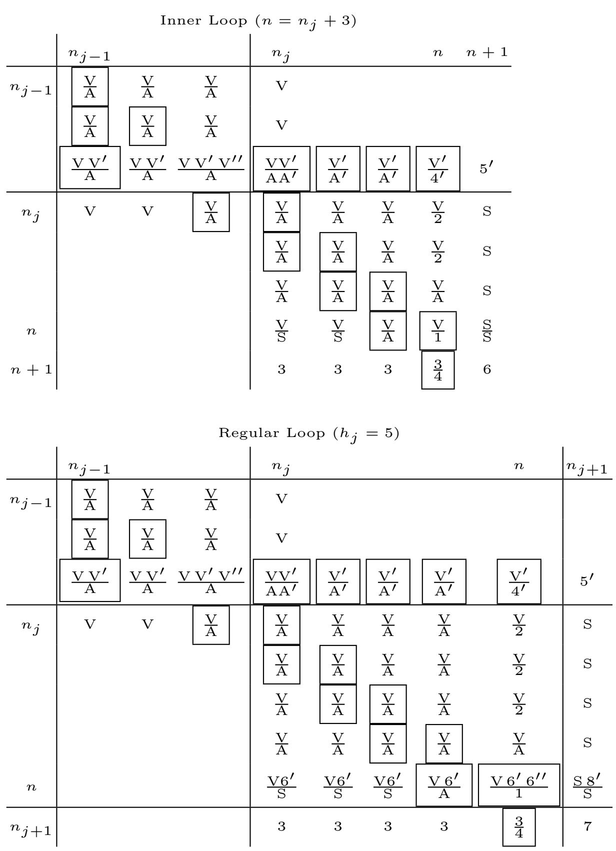

FIG. 4.4. *Action of Loop* 4.3 *(look-ahead for BiOS) in the w-table. An inner loop (at the top) and the following regular loop (at the bottom) are shown.*

TABLE 4.1
Cost and overhead of look-ahead LTPMs when constructing a look-ahead step of length $h_j > 1$. The construction of iterates is not yet included. By further capitalizing upon storage locations that become available during a look-ahead step, the storage overhead could be reduced by roughly 50%.

Method	Total cost	Cost overhead		Storage overhead
	(MVs, IPs)	(MVs, IPs)	relative	(N-vectors)
$(3, 3)$-type LA-LTPM	$4h_j - 3$	$2h_j - 3$	25%–100%	$2h_j^2 + 3h_j + 5$
LABiOStab	$4h_j - 3$	$2h_j - 3$	25%–100%	$2h_j^2 + 3h_j + 3$
LABiOS (if $h_{j-1} > 1$)	$3h_j$	h_j	50%	$h_j^2 + 4h_j + 4$
LABiOS (if $h_{j-1} = 1$)	$3h_j - 1$	$h_j - 1$	25%–50%	$h_j^2 + 4h_j + 4$

For a look-ahead step of length h_j, *the cost in terms of MVs is now* $3h_j$ MVs if $h_j > 1$ and $h_{j-1} > 1$, while it is only $3h_j - 1$ if $h_{j-1} = 1$. As the standard, non-look-ahead algorithm requires 2 MVs per step, this means that the overhead is at most 50%. Here, in the first substep of the $h_j - 1$ inner and the one regular loops h_j MVs are consumed, and another $2h_j$ MVs are needed in substep 4.

4.4. Overhead for look-ahead. In this subsection we further discuss the overhead of the look-ahead process in terms of MVs, inner products, and the necessary storage of N-vectors.

First we note that all inner products required in the look-ahead algorithms have the fixed initial vector $\widetilde{z}_0$ as their first argument. Therefore, if the second argument of an inner product was computed by a recurrence formula, then also the inner product of this vector with $\widetilde{z}_0$ can be computed indirectly by applying the same recurrence formula. Thus, only inner products of the form $\langle \widetilde{z}_0, Aw_n^l \rangle$ for which Aw_n^l is computed directly need to be computed explicitly. This means that in all algorithms the number of required inner products is equal to the number of required MVs. We must admit, however, that such recursively computed inner products, as well as the recursively computed MVs, may be the source of additional roundoff, which may cause instability.

Actually, to understand how to compute all the inner products needed, the reader may want to introduce a δ-table and a σ-table with entries $\delta_n^l := \langle \widetilde{z}_0, w_n^l \rangle$ and $\sigma_n^l := \langle \widetilde{z}_0, Aw_n^l \rangle$, respectively. Our loops and figures for generating the w-table then hold as well for these two tables. Note that the δ-table just contains the transpose of the matrix D.

In terms of MVs, the cost of a look-ahead step of length $h_j > 1$ has been specified in the previous subsection. By subtracting the cost for h_j non-look-ahead steps, that is, $2h_j$ MVs, we obtain the overhead summarized in Table 4.1, where IP stands for inner product. We stress that when $h_j = 1$ our algorithm has no overhead except for the necessary test of regularity, which, if it fails, would reveal an upcoming instability and initiate a look-ahead step. The table lists additionally the overhead in storage of N-vectors in a straightforward implementation that is not optimized with respect to memory usage.

For comparison, we cite from page 60 of [5] or page 180 of [6] that the CGS look-ahead procedure of Brezinski and Redivo Zaglia requires $6h_j - 3$ MVs if $h_j \leq n_j + 1$ (the typical case) and $5h_j + n_j - 2$ MVs if $h_j > n_j + 1$ (which means that a relatively large look-ahead step is needed in one of the first few iterations). Therefore, compared to our numbers, the overhead in terms of MVs is about four times (if h_j is large, but $h_j \leq n_j + 1$) to five times (if $h_j = 2$) larger than in our LABiOS (assuming as the basis a standard CGS implementation requiring 2 MVs per ordinary iteration). However, we also note that, according to the above numbers, for a step without look-ahead ($h_j = 1$), the methods of [5] and [6] need 3 MVs instead of 2 MVs.

5. Look-ahead strategies. In this section we address the delicate issue of when to perform a look-ahead step, which means in an LTPM to decide whether the new vertical index $n+1$ is a regular index or an inner index, i.e., whether the required product vectors w_{n+1}^l in the $(n+1)$th row of the w-table should be computed as regular or as inner vectors. Therefore, the look-ahead procedure in LTPMs serves to stabilize the underlying Lanczos method, the vertical movement of the product method in the w-table. Consequently, the criterion of when to carry out a look-ahead step in an LTPM can be based on the criterion given in [15] for the Lanczos algorithm. However, since the Lanczos vectors $\widetilde{y}_n$ and y_n are not computed explicitly in a product method, we need to rewrite the conditions of this criterion in terms of the product vectors w_n^l. Let us first motivate these conditions.

In the case of an *exact breakdown*, where $0 = \delta_n^\tau = \langle \widetilde{z}_0, w_n^n \rangle = \langle \widetilde{z}_n, y_n \rangle$ and $\widetilde{z}_n \neq 0, y_n \neq 0$, a division by zero would occur in the next Lanczos step. The first task of the look-ahead process is to circumvent these exact breakdowns without the necessity of restarting the Lanczos process and losing its superlinear convergence.

In finite-precision arithmetic, exact breakdowns are very unlikely. However, *near-breakdowns*, where $|\delta_n^\tau|$ is very small, may occur and cause large relative roundoff errors in the Lanczos coefficients α_n and β_n given by (3.2). To be more precise, we recall that the relative roundoff error in the computation of the inner product $\delta_n^\tau = \langle \widetilde{z}_0, w_n^n \rangle$ is bounded by [18, p. 64]

$$\frac{|\mathrm{fl}(\delta_n^\tau) - \delta_n^\tau|}{|\delta_n^\tau|} \leq 1.01 N\varepsilon \frac{\langle |\widetilde{z}_0|, |w_n^n| \rangle}{|\delta_n^\tau|} \leq 1.01 N\varepsilon \frac{\|\widetilde{z}_0\| \, \|w_n^n\|}{|\delta_n^\tau|},$$

where ε denotes the roundoff unit. Thus, a small value of $|\delta_n^\tau|$ leads in finite-precision arithmetic to a big relative roundoff error in the computation of the inner product δ_n^τ, which also causes a perturbation of the Lanczos coefficients α_n and β_n, since they depend on δ_n^τ and δ_{n-1}^τ, respectively. The second task of the look-ahead process is therefore to avoid a convergence deterioration due to perturbed Lanczos coefficients. Of course, similar roundoff effects may come up in the numerators of the formulas (3.2) for α_n and β_n, but large relative errors in those will only be harmful if the denominators are small too.

We would like to point out that in an LTPM the inner products δ_n^τ can be enlarged to a certain extent by an appropriate adaptive choice of the polynomials τ_n [39], [40] as long as $\langle (A^H)^n \widetilde{z}_0, y_n \rangle \neq 0$. For example, considering the BiOStab case, where $w_n^n = w_n^{n-1} - A w_n^{n-1} \chi_{n-1}$, we obtain, since $\langle \widetilde{z}_0, w_n^{n-1} \rangle = 0$,

$$\frac{\|\widetilde{z}_0\| \, \|w_n^n\|}{|\delta_n^\tau|} = \frac{\left\| w_n^{n-1} - A w_n^{n-1} \chi_{n-1} \right\|}{|\chi_{n-1}|} \frac{\|z_0\|}{\langle \widetilde{z}_0, A w_n^{n-1} \rangle}.$$

Thus, minimizing the relative roundoff error in the calculation of δ_n^τ is equivalent to choosing χ_{n-1} such that it minimizes $\left\| w_n^{n-1} - A w_n^{n-1} \chi_{n-1} \right\| / |\chi_{n-1}|$. This leads to $\chi_{n-1}^{OR} := \left\| w_n^{n-1} \right\|^2 / \langle w_n^{n-1}, A w_n^{n-1} \rangle$, which corresponds to the use of orthogonal residual polynomials of degree 1 instead of minimal residual polynomials of degree 1 in the recursive definition of $\tau_l(\zeta)$. Therefore, minimizing the relative roundoff error in the inner product δ_n^τ conflicts often with the objective of avoiding large intermediate residuals in order to prevent the recursive residual from drifting apart from the true residual [40]. Performing a look-ahead step is then the only possible remedy.

We need now to find a criterion for deciding when a look-ahead step should be performed so that both the above objectives can be attained. In view of the recursion (4.2) for the product vectors in the case of look-ahead, it follows that a block can be closed and a new product vector can be computed as a regular vector only if the diagonal blocks D_{j-1} and D_j of the Gramian are numerically nonsingular. Thus, the first condition that needs to be fulfilled in order to compute a new product vector w_{n+1}^l as a regular vector ($n+1 = n_{j+1}$) is given by

$$\sigma_{\min}(D_j) \geq \varepsilon, \tag{5.1}$$

where $\sigma_{\min}(D_j)$ denotes the minimal singular value of the block D_j. Note that this does not mean that D_j is well conditioned; it just guarantees numerical nonsingularity. In practice, (5.1) can be replaced by some other condition that implies this nonsingularity, for example by one implemented in a linear solver used for computing $D_{j-1}^{-1}\ell$ in (2.19) and α_n in (2.22).

Freund et al. [15] use a second condition to guarantee that the Krylov space is stably extended in the next Lanczos step in the sense that the basis of Lanczos vectors is sufficiently well conditioned. In terms of product vectors, this second condition amounts to computing, for any l, the new product vector w_{n+1}^l as a regular vector if in addition to (5.1) the following conditions for the coefficients α_n and β_n are fulfilled:

$$\|\alpha_n\|_1 \leq n(A), \qquad \|\beta_n\|_1 \leq n(A). \tag{5.2}$$

Here $\|\cdot\|_1$ denotes the ℓ_1-norm and $n(A)$ is an estimate for $\|A\|$ which is updated dynamically to ensure that the blocks W_j^n do not become larger than a user-specified maximal size [15]. The motivation for (5.2) was to ensure that in the new regular vector

$$w_{n+1}^n = \left(Aw_n^n - W_j^n \alpha_n - W_{j-1}^n \beta_n\right)/\gamma_n = \left(Aw_n^n - W_j^n \alpha_n - w_{j-1}'^{\,n} \beta_n'\right)/\gamma_n$$

(obtained from (4.2) with $\widehat{W}_j^n = W_j^n$ since $n+1 = n_{j+1}$ is regular) the component in the new direction Aw_n^n is sufficiently large, which will be the case if

$$\|Aw_n^n\| \geq \mathrm{tol}_2 \, \|w_t\|, \tag{5.3}$$

where $w_t := W_j^n \alpha_n + W_{j-1}^n \beta_n = W_j^n \alpha_n - w_{j-1}'^{\,n} \beta_n'$ and tol_2 is a chosen tolerance.

Compared to (5.2), condition (5.3) costs two additional inner products and the calculation of w_t, which can only be reused in the regular case for the computation of the new product vector w_{n+1}^n. Since we have

$$\|w_t\| \leq C\left(\|\alpha_n\|_1 + |\beta_n'|\right) \quad \text{with} \quad C := \max\left\{\left\|w_{j-1}'^{\,n}\right\|;\ \|w_k^n\| : n_j \leq k < n_{j+1}\right\}, \tag{5.4}$$

we could replace (5.3) by the less expensive condition

$$\|Aw_n^n\| \geq \mathrm{tol}_2\, C\left(\|\alpha_n\|_1 + |\beta_n'|\right). \tag{5.5}$$

However, in an LTPM it is not possible to normalize all product vectors w_n^l (see section 6), so C is not equal to 1. Moreover, (5.5) is less strict than (5.3) and (5.2).

Since a look-ahead step is more expensive than regular steps providing the same increase of the Krylov space dimension, a tight look-ahead criterion can save overall computational cost. Therefore, it is reasonable to spend extra effort for it. For this reason we favor criterion (5.3).

A drawback of (5.3) is that it does not take the angle between Aw_n^n and w_t into account. If $C_c := 1 - |\langle Aw_n^n, w_t\rangle| / (\|Aw_n^n\| \, \|w_t\|)$ is small, tol_2 in (5.3) should be chosen larger than in the case where it is nearly 1. This motivates the choice

$$\text{tol}_2 = \frac{C_1}{1-(1-C_2)\left|\frac{\langle Aw_n^n, w_t\rangle}{\|Aw_n^n\|\|w_t\|}\right|} \tag{5.6}$$

with suitably chosen constants $C_1 = C_1(\varepsilon)$ and $C_2 = C_2(\varepsilon)$ depending on the roundoff unit ε. This criterion requires an extra inner product and an appropriate choice for C_1 and C_2. For many small problems $C_1 = 10^{-3}$ and $C_2 = 10^{-2}$ worked well, but for larger problems we observed that C_c decays dramatically with the block size. Therefore, the probability that (5.3) with tol_2 as defined in (5.6) will be fulfilled decreases with the block length and leads very often (especially in BiOS) to situations where the maximal user-specified block size was reached. Further investigations are needed to see if this problem can be solved by a better choice of C_1 and C_2 or by a more appropriate selection for α_n in the inner case, instead of using, as in [15], $\alpha_n^n = 1, \alpha_n^{n-1} = 1$, $\alpha_n^k = 0$ for $k = 1, \ldots, n-2$.

6. Obtaining the solution of $Ax = b$.

So far we have only introduced various algorithms for constructing a sequence of product vectors w_n^l that provide a basis for $\mathcal{K}_m$. However, our goal is to solve the linear system $Ax = b$. We now describe how to accomplish this with these algorithms. There are several basic approaches to constructing approximate solutions of linear systems from a Krylov-space basis. Ours is related to the Galerkin method, but avoids the difficulty that arises when the Galerkin solution does not exist. (This difficulty causes, for example, the so-called *pivot breakdown* of the BiCG method.) Our approach is a natural generalization of the one that led to "unnormalized BiORes" introduced in [20] and renamed "inconsistent BiORes" in [25]. In contrast to the BiOMin, BiODir, and BiORes versions of the BiCG method, the inconsistent BiORes variant is not endangered by pivot breakdowns. An alternative would be to construct approximate solutions based on the QMR approach [16]. For a combination of this approach with LTPMs we refer the reader to [36].

Let the doubly indexed sequence of scalars $\dot\rho_n^l$ be given by

$$\dot\rho_n^l := \tau_l(0)\rho_n(0). \tag{6.1}$$

We define a doubly indexed sequence of *product iterates* x_n^l as follows. Starting with an arbitrary $x_0^0 \in \mathbb{C}^N$, we choose the initial product vector w_0^0 so that

$$w_0^0 = b\dot\rho_0^0 - Ax_0^0. \tag{6.2}$$

Here, $\dot\rho_0^0 = 1$ since $\tau_0(\zeta) = \rho_0(\zeta) \equiv 1$. For $l, n > 0$ the product iterates are now implicitly defined by

$$b\dot\rho_n^l - Ax_n^l := w_n^l \qquad \text{so that} \quad b - A\frac{x_n^l}{\dot\rho_n^l} = \frac{w_n^l}{\dot\rho_n^l} \ \text{ if } \ \dot\rho_n^l \neq 0. \tag{6.3}$$

Of course, x_n^l will be constructed only when w_n^l is. If $\dot\rho_n^l \neq 0$, it follows from (6.3) that $x_n^l/\dot\rho_n^l$ can be considered as an approximate solution of $Ax = b$, whose corresponding residual is $w_n^l/\dot\rho_n^l$. In order to derive recursions for the scalars $\dot\rho_n^l$ and the product iterates x_n^l, we introduce the blocks

$$\begin{aligned} X_{j-1}^l &:= \left[\begin{array}{ccc} x_{n_{j-1}}^l & \cdots & x_{n_j-1}^l \end{array}\right], & \widehat{X}_j^l &:= \left[\begin{array}{ccc} x_{n_j}^l & \cdots & x_n^l \end{array}\right], \\ P_{j-1}^l &:= \left[\begin{array}{ccc} \dot\rho_{n_{j-1}}^l & \cdots & \dot\rho_{n_j-1}^l \end{array}\right], & \widehat{P}_j^l &:= \left[\begin{array}{ccc} \dot\rho_{n_j}^l & \cdots & \dot\rho_n^l \end{array}\right], \end{aligned}$$

as well as the *auxiliary product iterates* x'^l_{j-1} and *auxiliary scalars* $\dot\rho'^l_{j-1}$ defined by

$$x'^l_{j-1} := X^l_{j-1} D^{-1}_{j-1} \ell, \qquad \dot\rho'^l_{j-1} := P^l_{j-1} D^{-1}_{j-1} \ell. \tag{6.4}$$

Again, $\widehat{X}^l_j = X^l_j$ and $\widehat{P}^l_j = P^l_j$ if $n+1$ is a regular index.

Then, by (6.3), (4.1), and (6.4),

$$\widehat{W}^l_j = b\widehat{P}^l_j - A\widehat{X}^l_j, \quad w'^{\,l}_{j-1} = b\dot\rho'^l_{j-1} - Ax'^l_{j-1}. \tag{6.5}$$

Next, using (4.2) we conclude that

$$\begin{aligned} b\dot\rho^l_{n+1} - Ax^l_{n+1} &= w^l_{n+1} = \left(Aw^l_n - \widehat{W}^l_j \alpha_n - w'^{\,l}_{j-1}\beta'_n \right)/\gamma_n \\ &= \left(Aw^l_n - \left[b\widehat{P}^l_j - A\widehat{X}^l_j \right]\alpha_n - \left[b\dot\rho'^l_{j-1} - Ax'^l_{j-1} \right]\beta'_n \right)/\gamma_n \\ &= b\left[-\widehat{P}^l_j\alpha_n - \dot\rho'^l_{j-1}\beta'_n \right]/\gamma_n + A\left[w^l_n + \widehat{X}^l_j\alpha_n + x'^l_{j-1}\beta'_n \right]/\gamma_n. \end{aligned}$$

This shows that

$$x^l_{n+1} = -\left[w^l_n + \widehat{X}^l_j\alpha_n + x'^l_{j-1}\beta'_n \right]/\gamma_n, \quad \dot\rho^l_{n+1} = -\left[\widehat{P}^l_j + \dot\rho'^l_{j-1}\beta'_n \right]/\gamma_n. \tag{6.6}$$

If we arrange the product iterates x^l_n and the scalars $\dot\rho^l_n$ in two tables analogous to the w-table (with the n-axis pointing downward and the l-axis to the right), these two recursions can be used to proceed in the vertical direction.

To obtain recursions for a horizontal movement, we assume first that the polynomials $\tau_n(\zeta)$ are given by the normalized three-term recurrence (3.3). This covers all algorithms described in this paper except LABiOS. Using (3.4) and (6.3) we see that the product iterates satisfy

$$x^{l+1}_n = -w^l_n\eta_l + x^l_n\xi_l + x^{l-1}_n(1-\xi_l). \tag{6.7}$$

For the scalars $\dot\rho^l_n$ the recursion $\dot\rho^{l+1}_n = \dot\rho^l_n\xi_l + \dot\rho^{l-1}_n(1-\xi_l)$ is valid, but since the polynomials τ_l are normalized (that is, $\tau_l(0) = 1$ for all l), the scalars $\dot\rho^l_n$ do not change with the index l, and we have simply $\dot\rho^l_n = \dot\rho^0_n = \rho_n(0)$.

In LABiOS only one horizontal movement is explicitly computed per step, namely in substep 5 of Loop 4.3 based on the recurrence (4.13). If we define in analogy to (4.10) and (4.11)

$$\begin{aligned} \widehat{X}'^k_{j-1} &:= \left[\; x'^{n_k}_{j-1} \;\; \dots \;\; x'^l_{j-1} \;\right], \qquad & X'^{k-1}_{j-1} &:= \left[\; x'^{n_{k-1}}{}_{j-1} \;\; \dots \;\; x'^{n_k-1}_{j-1} \;\right], \\ \widehat{P}'^k_{j-1} &:= \left[\; \dot\rho'^{n_k}_{j-1} \;\; \dots \;\; \dot\rho'^l_{j-1} \;\right], \qquad & P'^{k-1}_{j-1} &:= \left[\; \dot\rho'^{n_{k-1}}{}_{j-1} \;\; \dots \;\; \dot\rho'^{n_k-1}_{j-1} \;\right], \end{aligned}$$

and

$$x''^{k-1}_{j-1} := X'^{k-1}_{j-1} D^{-1}_{k-1}\ell, \quad \dot\rho''^{k-1}_{j-1} := P'^{k-1}_{j-1} D^{-1}_{k-1}\ell, \tag{6.8}$$

recurrences for the auxiliary iterates and the corresponding scalars are given by

$$\begin{aligned} x'^{l+1}_{j-1} &= -\left[w'^l_{j-1} + \widehat{X}'^k_{j-1}\alpha_l + x''^{k-1}_{j-1}\beta'_l \right]/\gamma_l, \\ \dot\rho'^{l+1}_{j-1} &= -\left[\widehat{P}'^k_{j-1}\alpha_l + \dot\rho''^{k-1}_{j-1}\beta'_l \right]/\gamma_l. \end{aligned} \tag{6.9}$$

Using for the scaling parameter γ_n in (4.2) the special choice

$$\gamma_n = \begin{cases} -\mathbf{1}_{h_j}^T \alpha_n - \mathbf{1}_{h_{j-1}}^T \beta_n & \text{if } n+1 = n_{j+1}, \\ -\mathbf{1}_{n-n_j+1}^T \alpha_n - \mathbf{1}_{h_{j-1}}^T \beta_n & \text{if } n_j < n+1 = n_{j+1}, \end{cases} \tag{6.10}$$

with $\mathbf{1}_m := [1 \cdots 1]_{m\times 1}^T$, the Lanczos polynomials ρ_n also could be normalized ($\rho_n(0) = 1$) so that $\dot{\rho}_n^l = 1$ for all $n, l \geq 0$. However, as we mentioned before, some γ_n might turn out to be zero, which would lead to a so-called pivot breakdown. Moreover, to avoid overflow or underflow, in the Lanczos process the scaling parameter γ_n is often used to normalize the Lanczos vectors y_n. But since the Lanczos vectors y_n are not explicitly computed in an LTPM, we cannot base the choice of γ_n here on their norm. However, independent of the size of the blocks generated by the look-ahead process, it is always necessary to compute the product vectors w_{n+1}^n in an LTPM. Therefore, we choose here γ_n to normalize w_{n+1}^n, that is,

$$\gamma_n := \left\| A w_n^n - \widehat{W}_j^n \alpha_n - W_{j-1}^n \beta_n \right\| \quad \text{if } n_j < n+1 \leq n_{j+1}. \tag{6.11}$$

7. Numerical examples. In this section we demonstrate the practical performance of our look-ahead versions of LTPMs in numerical examples. The tests are restricted to BiOStab, BiOxMR2, and BiOS. The look-ahead versions of these LTPMs are denoted by LABiOStab, LABiOxMR2, and LABiOS, respectively. For all tests the initial iterate $x_0 = 0$ is used, and the iteration is terminated when the norm of the recursive residual is less than $\sqrt{\varepsilon}$, the square root of the roundoff unit. The test programs were written in FORTRAN90/95 and run on workstations with 64-bit IEEE arithmetic. We start with small, artificially constructed model problems and move gradually to large real-world problems.

Example 1. The following small test example was proposed by Joubert [29] and also used by Brezinski and Redivo Zaglia [6]:

$$A := \begin{bmatrix} 1 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 3 & -1 \\ 0 & 0 & 1 & 3 \end{bmatrix}, \quad b := \begin{bmatrix} 0 \\ 2 \\ 2 \\ 4 \end{bmatrix}, \tag{7.1}$$

$x_0 = 0$, and $\widetilde{z}_0 = [1, 1, 1, 1]^T$. The Lanczos process and hence BiOStab, BiOxMR2, and BiOS without look-ahead break down at step 2. On the contrary, all look-ahead versions avoid this breakdown and converge after four iterations as shown in our plots of the true residual norms $\|b - Ax_n^l \frac{1}{\dot{\rho}_n^l}\|$ in Figure 7.1.

Example 2. Our second example is taken from [6, Example 5.2], the matrix

$$A := \begin{bmatrix} 2 & 1 & & & \\ 0 & 2 & 1 & & \\ 1 & 0 & 2 & 1 & \\ & 1 & 0 & 2 & \ddots \\ & & \ddots & \ddots & \ddots \end{bmatrix} \tag{7.2}$$

of order 400 and the right-hand side $b = (3, 3, 4, \ldots, 4, 3, 3)^T$ imply that the solution is $x = (1, 1, \ldots, 1)^T$. With $x_0 = 0$, $\widetilde{z}_0 = (0, 0, 0, -1, 1, \ldots, 0)^T$ the Lanczos process, and thus the LTPMs, break down in the first iteration. Our look-ahead versions perform

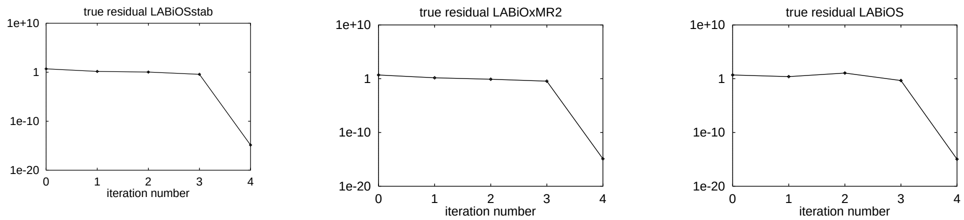

FIG. 7.1. *The true residual norm history (i.e.,* $\log(\|b - Ax_n^l \frac{1}{\dot\rho_n^l}\|)$ *vs. n) for the linear system defined in* (7.1) *solved by different LTPMs with look-ahead.*

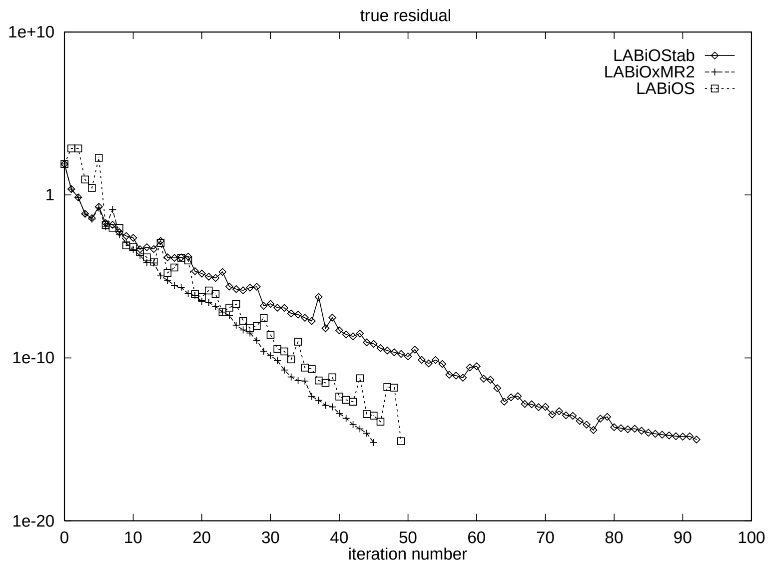

FIG. 7.2. *The true residual norm history (i.e.,* $\log(\|b - Ax_n^l \frac{1}{\dot\rho_n^l}\|)$ *vs. n) for the linear system defined in* (7.2) *solved by different LTPMs with look-ahead.*

two inner steps at the first and second iterations. All following iterations are regular steps. This is a typical behavior: there are only few and short look-ahead steps, and therefore the resulting mean overhead per iteration is nearly negligible. The residual norm history is shown in Figure 7.2.

In the next set of examples we consider p-cyclic matrices of the form

$$A := \begin{bmatrix} I_1 & & & B_1 \\ B_2 & I_2 & & \\ & \ddots & \ddots & \\ & & B_p & I_p \end{bmatrix}. \tag{7.3}$$

Hochbruck [27] showed that the computational work for solving a linear system with a p-cyclic system matrix by QMR with look-ahead can be reduced by approximately a factor $1/p$ (compared to a straightforward implementation using sparse matrix-vector multiplications with A) if the initial Lanczos vectors have only one nonzero block conforming to the block structure of A, if the inner vectors are chosen so that the nonzero structure of $(A - I)y_n$ is not destroyed, and if the blocks B_k are used for generating only possibly nonzero components of the Krylov space basis. Then it can

TABLE 7.1
Indices of regular steps in LTPMs for three problems with a p-cyclic system matrix.

Example	LABiOStab	LABiOxMR2	LABiOS
3	1,5,6,10,11,15	1,5,6,10,11,15	1,5,6,10,11,15
4	1,4,5,8,9,12,13,16	1,4,5,8,9,12,13	1,4,5,8,9,12,13,16
5	1,8,9,16,17	1,8,9,16,17	1,8,9,16,17

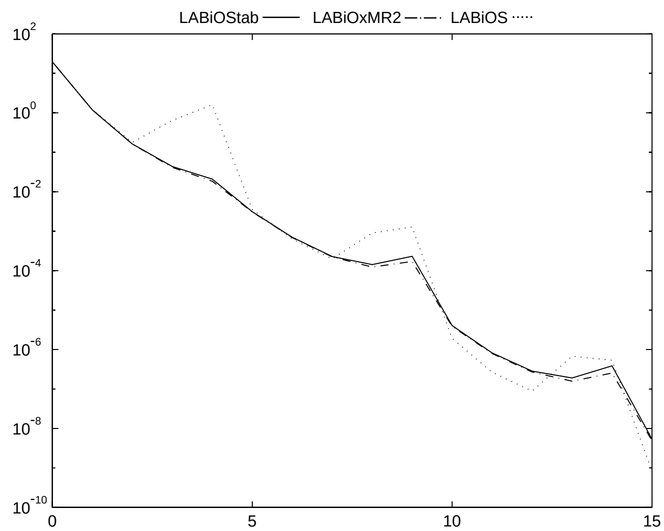

FIG. 7.3. *The true residual norm history (i.e.,* $\log(\|b - Ax_n^l \frac{1}{\rho_n^l}\|)$ *vs. n) for the linear system with the* 5*-cyclic system matrix defined in Example* 3 *solved by different LTPMs with look-ahead.*

be proved that in each cycle of p steps there are at least $p-2$ consecutive exact breakdowns for $p > 2$. But when using directly the system matrix A to generate the Krylov subspace, we have only in the first cycle of p steps $p-2$ consecutive exact breakdowns, while in the following cycles these will, in general, no longer persist, but must be expected to become near-breakdowns. Therefore, such problems provide good test examples for look-ahead algorithms.

Example 3. In this example we consider a 5-cyclic matrix with $B_k = B$ for $k = 1, \ldots, 5$, where B is a randomly generated 10×10 matrix. As right-hand side and as initial left Lanczos vector we choose

$$b = \begin{bmatrix} f_1 \\ 0 \end{bmatrix}, \qquad \widetilde{z}_0 = \begin{bmatrix} g_1 \\ 0 \end{bmatrix}, \tag{7.4}$$

where f_1, g_1 have random entries. The convergence history for the different LTPMs applied to this problem are shown in Figure 7.3, and the indices of the regular steps are listed in Table 7.1. Those are found exactly where predicted.

Example 4. We move now to a bigger 4-cyclic system matrix with $B_k = B$ for $k = 1, \ldots, 4$, where B is a 100×100 matrix with random entries. The results for this problem are shown in Figure 7.4, and the indices of the regular steps are depicted

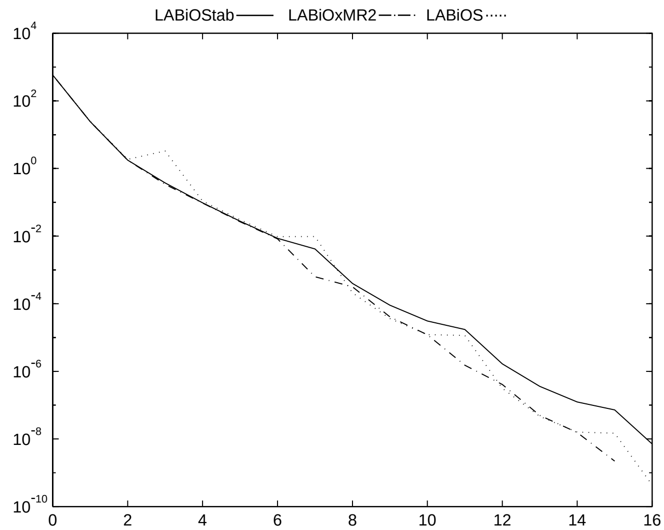

FIG. 7.4. *The true residual norm history (i.e.,* $\log(\|b - Ax_n^l \frac{1}{\rho_n^l}\|)$ *vs. n) for the linear system with the* 4*-cyclic system matrix defined in Example* 4 *solved by different LTPMs with look-ahead.*

also in Table 7.1. Again, they occur where predicted.

Example 5. Finally, we consider an 8-cyclic system matrix with B defined as in Example 4. The convergence history plotted in Figure 7.5 shows oscillations in the residual norm history of LABiOS, but overall LABiOS needs one iteration step fewer than LABiOStab and LABiOxMR2 to fulfill the convergence condition. For all methods the same look-ahead criterion ((5.3) with tol_2 defined as in (5.6) and $C_1 = 10^{-3}$, $C_2 = 10^{-2}$) is used. Especially for LABiOS the correct choice of the look-ahead criterion seems to be crucial. While with the above values of C_1 and C_2 the breakdowns occurred only where expected, we discovered for this larger problem that in BiOS the maximal user-specified block length of 10 was reached very often for $C_1 \gg 10^{-3}$ and $C_2 \ll 10^{-2}$, which indicates that the constructed inner vectors become more and more linearly dependent. Therefore, further investigations are needed to figure out a better choice for the coefficient vector α_n in the inner case. For example, following the proposal in [21], Hochbruck used in [27] a Chebyshev iteration for the generation of the inner vectors instead of $\alpha_n^n = 1, \alpha_n^{n-1} = 1$, and $\alpha_n^k = 0$ for $k = 1, \ldots, n-1$, which we adapted from [15].

Example 6. In our last example we take a real-world problem from the Harwell-Boeing Sparse Matrix Collection, namely SHERMAN1, a matrix of order 1000 with 3750 nonzero entries. The right-hand side b and the initial left Lanczos vector $\widetilde{z}_0$ were generated as different unit vectors with random entries. Without look-ahead, all LTPMs introduced here break down (BiOStab at step 186, BiOStab2 at step 176, BiOxMR2 at step 145, and BiOS at step 352). On the contrary, the look-ahead versions in combination with the look-ahead criterion (5.3) with tol_2 defined as in (5.6) and $C_1 = 10^{-3}$ and $C_2 = 10^{-2}$ converge as shown in Figure 7.6. Due to the real spectrum of the system matrix A, there is only a slight difference in the convergence of LABiOStab and LABiOxMR2. It was reported to us that BiCGStab(ℓ) with $\ell = 8$ and no look-ahead can handle this problem in about the same number of MVs as our BiCGStab with look-ahead.

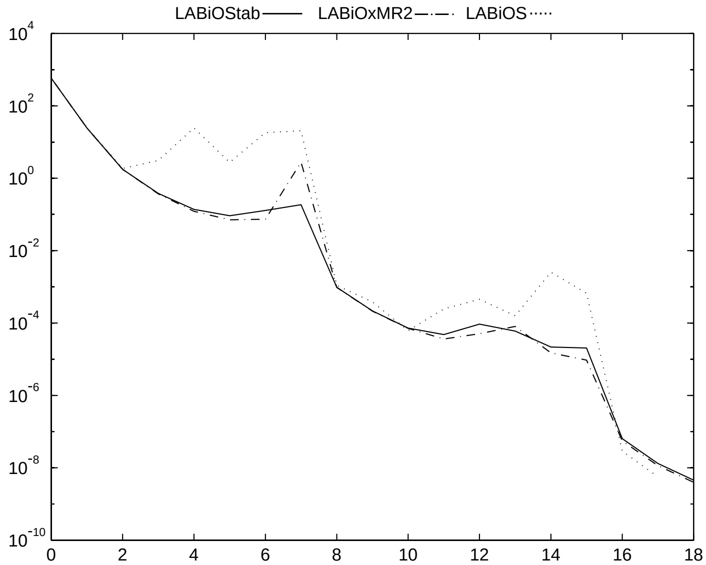

FIG. 7.5. *The true residual norm history (i.e.,* $\log(\|b - Ax_n^l \frac{1}{\dot{\rho}_n^l}\|)$ *vs.* n*) for the linear system with the* 8*-cyclic system matrix defined in Example* 5 *solved by different LTPMs with look-ahead.*

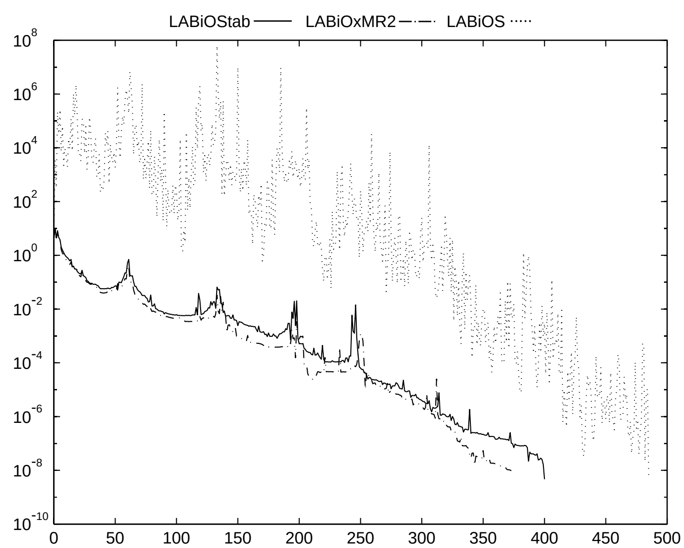

FIG. 7.6. *The true residual norm history (i.e.,* $\log(\|b - Ax_n^l \frac{1}{\dot{\rho}_n^l}\|)$ *vs.* n*) for a real-world problem with the SHERMAN*1 *matrix from the Harwell-Boeing collection solved by different LTPMs with look-ahead.*

8. Conclusions. We have proposed look-ahead versions for various Lanczos-type product methods that make use of the Lanczos three-term recurrences. Since they are based on the Lanczos look-ahead version of Gutknecht [23] and Freund et al. [15], they can handle look-ahead steps of any length and avoid steps that are longer than needed. The algorithms proposed in this work should be easy to understand due to the introduction of an array of product vectors, symbolically displayed in the w-table, and the visualization of the progress in this w-table. Furthermore, the w-table proved to be a useful tool to derive optimal variants for which the computational work in terms of MVs is minimized.

A variety of numerical examples demonstrate the practical performance of the proposed algorithms. However, larger problems indicate that further work should be directed to finding an improved look-ahead criterion that more reliably avoids critical perturbations of the Lanczos coefficients by roundoff errors. Moreover, one should investigate if there is a better way of constructing inner vectors than the choice adapted from [15]. Alternatives would be to orthogonalize them within each block [2] or to construct them by Chebyshev iteration [21], but it is not clear if the additional cost involved pays off.

The look-ahead process in an LTPM stabilizes primarily the vertical movement in the w-table, except in the BiOS algorithm where the w-table is symmetric. For the horizontal movement it is also important to generate the Krylov space stably, and both BiOStab2 and BiOxMR2 (in particular when suitably modified) do that more reliably than BiCGStab, since the two-dimensional steps offer more flexibility. This has also a lasting positive effect on the roundoff in the vertical movement. To stabilize the horizontal movement further, a local minimal residual polynomial of degree $\ell \geq 1$ with an adaptive choice of ℓ as in BiCGStab(ℓ) could be used. A further possibility is to adapt ℓ to the size h_j of the current Lanczos block, which would mean to perform in each regular step an h_j-dimensional local minimization of the residual. An alternative is to trade in the local residual minimization for a more stable Krylov-space generation whenever the former causes a problem. Yet another possibility, indicated in section 4.2, is to combine the Lanczos process with a hybrid Chebyshev iteration.

It is known that in finite-precision arithmetic BiORes is usually more affected by roundoff than the standard BiOMin version of BiCG, at least with regard to the gap between recursively and explicitly computed residuals. Therefore, we are in the process of extending this work to look-ahead procedures for LTPMs that are based on coupled two-term recurrences.

REFERENCES

[1] E. Ayachour, *Avoiding the look-ahead in the Lanczos method*, Appl. Math. (Warsaw), 26 (1999), pp. 33–62.

[2] 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), pp. 21–43.

[3] C. Brezinski, *CGM: A Whole Class of Lanczos-Type Solvers for Linear Systems*, Tech. Report ANO-253, Université Lille Flandres Artois, France, 1991.

[4] C. Brezinski and M. Redivo Zaglia, *Breakdowns in the computation of orthogonal polynomials*, in Nonlinear Numerical Methods and Rational Approximation II, A. Cuyt, ed., Kluwer, Dordrecht, The Netherlands, 1994, pp. 49–59.

[5] C. Brezinski and M. Redivo Zaglia, *Treatment of near-breakdown in the CGS algorithms*, Numer. Algorithms, 7 (1994), pp. 33–73.

[6] C. Brezinski and M. Redivo Zaglia, *Look-ahead in Bi-CGSTAB and other product methods for linear systems*, BIT, 35 (1995), pp. 169–201.

[7] C. Brezinski, M. Redivo Zaglia, and H. Sadok, *New look-ahead Lanczos-type algorithms for linear systems*, Numer. Math., 83 (1999), pp. 53–85.

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

[9] Z.-H. Cao, *Avoiding breakdown in variants of the BI-CGSTAB algorithm*, Linear Algebra

Appl., 263 (1997), pp. 113–132.

[10] Z.-H. Cao, *On the QMR approach for iterative methods including coupled three-term recurrences for solving nonsymmetric linear systems*, Appl. Numer. Math., 27 (1998), pp. 123–140.

[11] T. F. Chan, E. Gallopoulos, V. Simoncini, T. Szeto, and C. H. Tong, *A quasi-minimal residual variant of the Bi-CGSTAB algorithm for nonsymmetric systems*, SIAM J. Sci. Comput., 15 (1994), pp. 338–347.

[12] V. Eijkhout, *LAPACK Working Note* 78: *Computational Variants of the CGS and BiCGstab Methods*, Tech. Report UT-CS-94-241, Computer Science Department, University of Tennessee, Knoxville, TN, 1994.

[13] D. A. Flanders and G. Shortly, *Numerical determination of fundamental modes*, J. Appl. Phys., 21 (1950), pp. 1326–1332.

[14] R. W. Freund, *A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems*, SIAM J. Sci. Comput., 14 (1993), pp. 470–482.

[15] 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.

[16] 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.

[17] R. W. Freund and H. Zha, *A look-ahead algorithm for the solution of general Hankel systems*, Numer. Math., 64 (1993), pp. 295–321.

[18] G. H. Golub and C. F. van Loan, *Matrix Computations*, 2nd ed., Johns Hopkins University Press, Baltimore, MD, 1989.

[19] P. R. Graves-Morris, *A "look-around Lanczos" algorithm for solving a system of linear equations*, Numer. Algorithms, 15 (1997), pp. 247–274.

[20] M. H. Gutknecht, *The unsymmetric Lanczos algorithms and their relations to Padé approximation, continued fractions, and the qd algorithm*, in Preliminary Proceedings of the Copper Mountain Conference on Iterative Methods, Copper Mountain, CO, 1990.

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

[22] M. H. Gutknecht, *Variants of BiCGStab for matrices with complex spectrum*, SIAM J. Sci. Comput., 14 (1993), pp. 1020–1033.

[23] M. H. Gutknecht, *A completed theory of the unsymmetric Lanczos process and related algorithms*, II, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 15–58.

[24] M. H. Gutknecht, *The Lanczos process and Padé approximation*, in Proceedings of the Cornelius Lanczos International Centenary Conference, J. D. Brown, M. T. Chu, D. C. Ellison, and R. J. Plemmons, eds., SIAM, Philadelphia, PA, 1994, pp. 61–75.

[25] M. H. Gutknecht, *Lanczos-type solvers for nonsymmetric linear systems of equations*, Acta Numer., 6 (1997), pp. 271–397.

[26] M. H. Gutknecht and K. J. Ressel, *Look-Ahead Procedures for Lanczos-Type Product Methods Based on Three-Term Recurrences*, Tech. Report TR-96-19, Swiss Center for Scientific Computing, Manno, Switzerland, 1996.

[27] M. Hochbruck, *Lanczos- und Krylov-Verfahren für nicht-Hermitesche lineare Systeme*, Ph.D. thesis, Fakultät für Mathematik, Universität Karlsruhe, Karlsruhe, Germany, 1992.

[28] M. Hochbruck, *The Padé table and its relation to certain numerical algorithms*, Habilitationsschrift, Universität Tübingen, Germany, 1996.

[29] W. D. Joubert, *Generalized Conjugate Gradient and Lanczos Methods for the Solution of Nonsymmetric Systems of Linear Equations*, Ph.D. thesis, Center for Numerical Analysis, University of Texas at Austin, 1990.

[30] W. D. Joubert, *Iterative Methods for the Solution of Nonsymmetric Systems of Linear Equations*, Tech. Report CNA-239, Center for Numerical Analysis, University of Texas at Austin, 1990.

[31] W. Joubert, *Lanczos methods for the solution of nonsymmetric systems of linear equations*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 926–943.

[32] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Research Nat. Bur. Standards, 45 (1950), pp. 255–281.

[33] T. A. Manteuffel, *The Tchebyshev iteration for nonsymmetric linear systems*, Numer. Math., 28 (1977), pp. 307–327.

[34] B. N. Parlett, *Reduction to tridiagonal form and minimal realizations*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 567–593.

[35] 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.

[36] K. J. Ressel and M. H. Gutknecht, *QMR smoothing for Lanczos-type product methods based on three-term recurrences*, SIAM J. Sci. Comput., 19 (1998), pp. 55–73.
[37] W. Schönauer, *Scientific Computing on Vector Computers*, Elsevier, Amsterdam, 1987.
[38] G. L. G. Sleijpen and D. R. Fokkema, *BiCGstab(l) for linear equations involving unsymmetric matrices with complex spectrum*, Electron. Trans. Numer. Anal., 1 (1993), pp. 11–32.
[39] G. L. G. Sleijpen and H. A. van der Vorst, *Maintaining convergence properties of BiCGstab methods in finite precision arithmetic*, Numer. Algorithms, 10 (1995), pp. 203–223.
[40] G. L. G. Sleijpen and H. A. van der Vorst, *An overview of approaches for the stable computation of hybrid BiCG methods*, Appl. Numer. Math., 19 (1995), pp. 235–254.
[41] G. L. G. Sleijpen, H. A. van der Vorst, and D. R. Fokkema, *BiCGstab(l) and other hybrid Bi-CG methods*, Numer. Algorithms, 7 (1994), pp. 75–109.
[42] P. Sonneveld, *CGS, a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 36–52.
[43] D. R. Taylor, *Analysis of the Look Ahead Lanczos Algorithm*, Ph.D. thesis, Department of Mathematics, University of California, Berkeley, 1982.
[44] H. A. van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 631–644.
[45] H. E. Wrigley, *Accelerating the Jacobi method for solving simultaneous equations by Chebyshev extrapolation when the eigenvalues of the iteration matrix are complex*, Comput. J., 6 (1963), pp. 169–176.
[46] S.-L. Zhang, *GPBI-CG: Generalized product-type methods based on Bi-CG for solving nonsymmetric linear systems*, SIAM J. Sci. Comput., 18 (1997), pp. 537–551.
[47] L. Zhou and H. F. Walker, *Residual smoothing techniques for iterative methods*, SIAM J. Sci. Comput., 15 (1994), pp. 297–312.
[48] M. Ziegler, *Generalized biorthogonal bases and tridiagonalisation of matrices*, Numer. Math., 77 (1977), pp. 407–421.