[32] N. Levinson. The Wiener RMS (root mean square) error criterion in filter design and prediction. *J. Math. Phys.* **25**, pp. 261–278, 1946.

[33] B.N. Parlett. *The Symmetric Eigenvalue Problem*. Prentice-Hall, Englewood Cliffs, N.J., 1980.

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

[35] B.N. Parlett, D.R. Taylor, and Z.A. Liu. A look-ahead Lanczos algorithm for unsymmetric matrices. *Math. Comp*. **44**, pp. 105–124, 1985.

[36] D.R. Taylor. Analysis of the look ahead Lanczos algorithm. Ph.D. thesis, University of California, Berkeley, California, 1982.

[37] W.F. Trench. An algorithm for the inversion of finite Toeplitz matrices. *J. Soc. Indust. Appl. Math.* **12**, pp. 515–522, 1964.

[38] W.F. Trench. An algorithm for the inversion of finite Hankel matrices. *J. Soc. Indust. Appl. Math.* **13**, pp. 1102–1107, 1965.

[16] R.W. Freund and N.M. Nachtigal. An implementation of the QMR method based on coupled two-term recurrences. Technical Report 92.15, RIACS, NASA Ames Research Center, Moffett Field, California, June 1992.

[17] R.W. Freund and H. Zha. Formally biorthogonal polynomials and a look-ahead Levinson algorithm for general Toeplitz systems. Technical Report 91.27, RIACS, NASA Ames Research Center, Moffett Field, California, December 1991.

[18] R.W. Freund and H. Zha. A look-ahead algorithm for the solution of general Hankel systems. *Numer. Math.*, to appear.

[19] G.H. Golub and C.F. Van Loan. *Matrix Computations*, Second Edition. The Johns Hopkins University Press, Baltimore, 1989.

[20] W.B. Gragg. Matrix interpretations and applications of the continued fraction algorithm. *Rocky Mountain J. Math.* **4**, pp. 213–225. 1974.

[21] W.B. Gragg and A. Lindquist. On the partial realization problem. *Linear Algebra Appl.* **50**, pp. 277–319, 1983.

[22] U. Grenander and G. Szegö. *Toeplitz Forms and their Applications*, Second Edition. Chelsea, New York, 1984.

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

[24] M.H. Gutknecht. A completed theory of the unsymmetric Lanczos process and related algorithms, Part II. *SIAM J. Matrix Anal. Appl.*, to appear.

[25] G. Heinig and K. Rost. *Algebraic Methods for Toeplitz-like Matrices and Operators*. Birkhäuser, Basel, 1984.

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

[27] T. Kailath, A. Vieira, and M. Morf. Inverses of Toeplitz operators, innovations, and orthogonal polynomials. *SIAM Rev.* **20**, pp. 106–119, 1978.

[28] S.-Y. Kung. Multivariable and multidimensional systems: analysis and design. Ph.D. thesis, Stanford University, Stanford, California, 1977.

[29] J.L. Massey. Shift-register synthesis and BCH decoding. *IEEE Trans. Inform. Theory* **IT-15**, pp. 122–127, 1969.

[30] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Nat. Bur. Standards* **45**, pp. 255–282, 1950.

[31] C. Lanczos. Solution of systems of linear equations by minimized. *J. Res. Nat. Bur. Standards* **49**, pp. 33–53, 1952.

References

[1] G. Baxter. Polynomials defined by a difference system. *J. Math. Anal. Appl.* **2**, pp. 223–263, 1961.

[2] E.R. Berlekamp. *Algebraic Coding Theory*. McGraw-Hill, New York, 1968.

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

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

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

[6] T.S. Chihara. *An Introduction to Orthogonal Polynomials*. Gordon and Breach, New York, 1978.

[7] T.K. Citron. Algorithms and architectures for error correcting codes. Ph.D. thesis, Stanford University, Stanford, California, 1986.

[8] J. Cullum and R.A. Willoughby. A practical procedure for computing eigenvalues of large sparse nonsymmetric matrices. In: J. Cullum and R.A. Willoughby (Eds.), *Large Scale Eigenvalue Problems*. North-Holland, Amsterdam, pp. 193–240, 1986.

[9] A. Draux. *Polynômes Orthogonaux Formels – Applications*. Lecture Notes in Mathematics, Vol. 974. Springer-Verlag, Berlin, 1983.

[10] R.W. Freund. Conjugate gradient-type methods for linear systems with complex symmetric coefficient matrices. *SIAM J. Sci. Statist. Comput.* **13**, pp. 425–448, 1992.

[11] R.W. Freund. Quasi-kernel polynomials and convergence results for quasi-minimal residual iterations. In: D. Braess and L.L. Schumaker (Eds.), *Numerical Methods of Approximation Theory, Vol. 9*. Birkäuser, Basel, pp. 77–95, 1992.

[12] R.W. Freund. A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems. *SIAM J. Sci. Comput.* **14**, 1993, to appear.

[13] R.W. Freund, G.H. Golub, and N.M. Nachtigal. Iterative solution of linear systems. *Acta Numerica* **1**, pp. 57–100, 1992.

[14] 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, to appear.

[15] R.W. Freund and N.M. Nachtigal. QMR: a quasi-minimal residual method for non-Hermitian linear systems. *Numer. Math.* **60**, pp. 315–339, 1991.

2) Decide whether to construct u_{n+1} and v_{n+1} as regular vectors or as inner vectors and go to 3) or 4), respectively.

3) (*Regular step*) Compute $\alpha_l^{(1)}$, μ_n, $\beta_l^{(1)}$, ν_n by solving

$$(G^{(l)})^T\alpha_l^{(1)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad F^{(l)}\mu_n = \begin{bmatrix} 0 \\ f_l \end{bmatrix}, \quad F^{(l)}\beta_l^{(1)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad (G^{(l)})^T\nu_n = \begin{bmatrix} 0 \\ g_l \end{bmatrix},$$

respectively, and set

$$\begin{aligned} u_{n+1} &= \begin{bmatrix} 0 \\ u_n \end{bmatrix} - \rho_n \begin{bmatrix} JV^{(l)} \\ 0 \end{bmatrix} \alpha_l^{(1)} - \begin{bmatrix} U^{(l)} \\ 0 \end{bmatrix} \mu_n, \\ v_{n+1} &= \begin{bmatrix} 0 \\ v_n \end{bmatrix} - \tau_n \begin{bmatrix} JU^{(l)} \\ 0 \end{bmatrix} \beta_l^{(1)} - \begin{bmatrix} V^{(l)} \\ 0 \end{bmatrix} \nu_n. \end{aligned}$$

Set $n_{l+1} = n+1$, $l = l+1$, $U^{(l)} = V^{(l)} = F^{(l)} = G^{(l)} = \emptyset$, and go to 5).

4) (*Inner step*) Set

$$u_{n+1} = \begin{bmatrix} 0 \\ u_n \end{bmatrix} - \rho_n \begin{bmatrix} JV^{(l-1)} \\ 0 \end{bmatrix} \alpha_{l-1}^{(1)}, \quad v_{n+1} = \begin{bmatrix} 0 \\ v_n \end{bmatrix} - \tau_n \begin{bmatrix} JU^{(l-1)} \\ 0 \end{bmatrix} \beta_{l-1}^{(1)}.$$

5) Set

$$U^{(l)} = \left[\begin{array}{c|c} U^{(l)} & u_{n+1} \\ 0 & \end{array}\right], \quad V^{(l)} = \left[\begin{array}{c|c} V^{(l)} & v_{n+1} \\ 0 & \end{array}\right],$$

and update $F^{(l)}$, $G^{(l)}$.

In (9.2), the vectors s_n, $r_n \in \mathbb{C}^{n+1}$ are defined by the partitioning

$$M_{n+1} = \begin{bmatrix} t_0 & s_n^T \\ r_n & M_n \end{bmatrix}$$

of the $(n+1)$st moment matrix M_{n+1}. Furthermore,

$$J = \begin{bmatrix} 0 & \cdots & 0 & 1 \\ \vdots & \iddots & 1 & 0 \\ 0 & \iddots & \iddots & \vdots \\ 1 & 0 & \cdots & 0 \end{bmatrix}$$

denotes the antidiagonal identity matrix.

For a discussion of the look-ahead strategy used in step 1) of Algorithm 9.1 and other implementation details, we refer the reader to [17]. Also, in [17], it is shown how solutions of Toeplitz systems $M_n x_n = b_n$ can be computed, based on the factorization (9.1).

Finally, we remark that Algorithm 9.1 is an extension of the classical Levinson-Trench algorithm [32, 37] for computing inverse factorizations of a sequence of nonsingular Toeplitz matrices.

2) Decide whether to construct φ_{n+1} and ψ_{n+1} as regular FBOPs or as inner quasi-FBOPs and go to 3) or 4), respectively.

3) (*Constructing a regular* FBOP) Set

$$\alpha_n = (G^{(l)})^{-T} \begin{bmatrix} 0 \\ \rho_n \end{bmatrix}, \quad \mu_n = (F^{(l)})^{-1} \begin{bmatrix} 0 \\ f_l \end{bmatrix}, \quad \varphi_{n+1} = \lambda\varphi_n - \widehat{\Psi}^{(l)}\alpha_n - \Phi^{(l)}\mu_n,$$
$$\beta_n = (F^{(l)})^{-1} \begin{bmatrix} 0 \\ \tau_n \end{bmatrix}, \quad \nu_n = (G^{(l)})^{-T} \begin{bmatrix} 0 \\ g_l \end{bmatrix}, \quad \psi_{n+1} = \lambda\psi_n - \widehat{\Phi}^{(l)}\beta_n - \Psi^{(l)}\nu_n.$$

Set $n_{l+1} = n+1$, $l = l+1$, $\Phi^{(l)} = \Psi^{(l)} = \emptyset$, and go to 5).

4) (*Constructing a quasi*-FBOP) Set

$$\alpha_n = (G^{(l-1)})^{-T} \begin{bmatrix} 0 \\ \rho_n \end{bmatrix}, \quad \varphi_{n+1} = \lambda\varphi_n - \widehat{\Psi}^{(l-1)}\alpha_n,$$
$$\beta_n = (F^{(l-1)})^{-1} \begin{bmatrix} 0 \\ \tau_n \end{bmatrix}, \quad \psi_{n+1} = \lambda\psi_n - \widehat{\Phi}^{(l-1)}\beta_n.$$

5) Set $\Phi^{(l)} = [\,\Phi^{(l)} \quad \varphi_{n+1}\,]$, $\Psi^{(l)} = [\,\Psi^{(l)} \quad \psi_{n+1}\,]$, $F^{(l)} = \langle \Lambda^{(l)}, \Phi^{(l)} \rangle$, $G^{(l)} = \langle \Psi^{(l)}, \Lambda^{(l)} \rangle$.

9. Inverse Block Factorization of Toeplitz Matrices

In view of Theorem 3.5, the polynomials $\{\varphi_n\}_{n=0}^{\infty}$ and $\{\psi_n\}_{n=0}^{\infty}$ generated by Algorithm 8.2 induce inverse block-factorizations of the Toeplitz matrices (8.1) of the form

$$V_n^T M_n U_n = D_n. \tag{9.1}$$

Here, $U_n = [\,u_{ij}\,]_{i,j=0,\ldots,n}$ and $V_n = [\,v_{ij}\,]_{i,j=0,\ldots,n}$ is a unit upper triangular matrix (3.27) and (3.28) whose entries are just the coefficients of the polynomials $\{\varphi_j\}_{j=0}^{n}$ and $\{\psi_j\}_{j=0}^{n}$, respectively, and D_n is a block-diagonal matrix defined in (3.29). The translation of Algorithm 8.2 into a look-ahead procedure for constructing the matrices U_n, V_n, and D_n in (9.1) is analogous to the derivation of Algorithm 7.1 from Algorithm 4.1. Therefore, we omit the details, and we directly state the resulting algorithm.

Algorithm 9.1 (Sketch of a look-ahead algorithm for inverse Toeplitz factorization [17])**.**

0) Set $u_0 = v_0 = 1$, $U^{(0)} = V^{(0)} = 1$, $F^{(0)} = G^{(0)} = t_0$, $n_0 = 0$, $l = 0$.

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

1) Compute

$$\rho_n = s_n^T u_n, \quad \tau_n = v_n^T r_n. \tag{9.2}$$

where $\partial(\varphi)$ is the exact degree of $\varphi \in \mathcal{P}$, *i.e.*, $\partial(\varphi)$ is the smallest integer $n \geq 0$ such that $\varphi \in \mathcal{P}_n$. Note that $\widehat{\varphi}$ is a polynomial of degree at most $\partial(\varphi)$.

In the following, we will also use the following extension of (8.3). For any row vector

$$\Phi = [\,\varphi_0 \quad \varphi_1 \quad \cdots \quad \varphi_j\,]$$

of polynomials in $\mathcal{P}$, we define its *block-reverse* by

$$\widehat{\Phi}(\lambda) \equiv \lambda^{n_\Phi}\,\Phi(1/\lambda), \quad \text{where} \quad n_\Phi := \max_{i=0,1,\ldots,j} \partial(\varphi_i).$$

Note that the entries of $\widehat{\Phi}$ are again polynomials.

In general, it cannot be guaranteed that the moment matrices (8.1) are all nonsingular. If some of the matrices M_n are singular or nonsingular, but ill-conditioned, then breakdowns or near-breakdowns will occur in Algorithm 8.1. Recently, Freund and Zha [17], derived extensions of the Szegö recursions for the stable construction of FBOPs and quasi-FBOPs associated with general Toeplitz moment matrices.

Next, we present a sketch of this algorithm. Again, we use the notation introduced in Section 3.3. In addition to the blocks (3.21) and (3.22), we define blocks of monomials

$$\Lambda^{(k)} := \begin{cases} [\,\lambda^{n_k} \quad \lambda^{n_k+1} \quad \cdots \quad \lambda^{n_{k+1}-1}\,], & \text{if } 0 \leq k < l, \\ [\,\lambda^{n_k} \quad \lambda^{n_k+1} \quad \cdots \quad \lambda^{n}\,], & \text{if } k = l. \end{cases}$$

Furthermore, we set

$$F^{(k)} := \langle \Lambda^{(k)}, \Phi^{(k)} \rangle \quad \text{and} \quad G^{(k)} := \langle \Psi^{(k)}, \Lambda^{(k)} \rangle, \quad k = 0, 1, \ldots, l.$$

Finally, for $k < l$, we define vectors f_k and g_k as follows. If $n_{k+1} - n_k > 1$, then f_k and g_k are given by

$$F^{(k)} = \begin{bmatrix} * & f_k \\ * & * \end{bmatrix} \quad \text{and} \quad (G^{(k)})^T = \begin{bmatrix} * & g_k \\ * & * \end{bmatrix}, \tag{8.4}$$

where the elements $*$ in the lower right corners in (8.4) are 1×1. If $n_{k+1} - n_k = 1$, then we set $f_k = g_k = \emptyset$.

After these preparations, our algorithm based on generalized Szegö recursions can be formulated as follows.

Algorithm 8.2 (Look-ahead process for constructing FBOPs [17]).

0) Set $\varphi_0 = \psi_0 = 1$, $\Phi^{(0)} = \Psi^{(0)} = 1$, $F^{(0)} = G^{(0)} = \langle 1, 1 \rangle$, $n_0 = 0$, $l = 0$.

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

1) Compute

$$\rho_n = \langle 1, \lambda\varphi_n \rangle \quad \text{and} \quad \tau_n = \langle \lambda\psi_n, 1 \rangle.$$

8. Constructing FBOPs Associated with Toeplitz Matrices

In this section, we consider the construction of FBOPs associated with Toeplitz matrices.

In the sequel, it is always assumed that $\langle \cdot, \cdot \rangle$ is a bilinear form whose moments m_{jk} depend only on the *difference* $j-k$ of their indices, *i.e.*,

$$m_{jk} = t_{j-k} \quad \text{for all} \quad j, k = 0, 1, \ldots .$$

Here $\{t_j\}_{j=-\infty}^{\infty}$ is any biinfinite sequence of real or complex numbers. The corresponding moment matrices (3.4) are then of the form

$$M_n = [\, t_{i-j} \,]_{i,j=0,\ldots,n} = \begin{bmatrix} t_0 & t_{-1} & t_{-2} & \cdots & t_{-n} \\ t_1 & t_0 & \ddots & & \vdots \\ t_2 & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & & t_{-1} \\ t_n & \cdots & \cdots & t_1 & t_0 \end{bmatrix}, \quad n = 0, 1, \ldots . \tag{8.1}$$

Matrices of the form (8.1) are called *Toeplitz matrices*. Note that, unless $t_{-j} = t_j$ for all j, the moment matrices (8.1) do not satisfy the symmetry condition (3.8). Therefore, in general, the bilinear form $\langle \cdot, \cdot \rangle$ is not symmetric, and we have to construct two sequences

$$\{\varphi_n\}_{n=0}^{\infty} \quad \text{and} \quad \{\psi_n\}_{n=0}^{\infty}$$

of right, respectively left, FBOPs and quasi-FBOPs.

First, we consider the case that the matrices (8.1) are all nonsingular. By Lemma 3.2, this implies that right and left regular FBOPs of any degree exist. It turns out that these polynomials can be generated easily, by means of the celebrated Szegö recursions [22, 1, 27].

Algorithm 8.1 (The classical Szegö recursions)**.**

0) Set $\varphi_0 = \psi_0 = 1$, $\delta_0 = t_0$.

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

1) Compute $\rho_n = \langle 1, \lambda\varphi_n \rangle$, $\tau_n = \langle \lambda\psi_n, 1 \rangle$.

2) Set

$$\begin{aligned} \alpha_n &= \rho_n/\delta_n, \quad \varphi_{n+1} = \lambda\varphi_n - \widehat{\psi}_n\alpha_n, \\ \beta_n &= \tau_n/\delta_n, \quad \psi_{n+1} = \lambda\psi_n - \widehat{\varphi}_n\beta_n. \end{aligned} \tag{8.2}$$

3) Set $\delta_{n+1} = \delta_n(1 - \alpha_n\beta_n)$.

Here, in (8.2), the polynomial $\widehat{\psi}_n$ and $\widehat{\varphi}_n$ is the reverse of ψ_n and φ_n, respectively. Recall that, for any polynomial $\varphi \in \mathcal{P}$, its *reverse* $\widehat{\varphi}$ is given by

$$\widehat{\varphi}(\lambda) \equiv \lambda^{\partial(\varphi)}\varphi(1/\lambda), \tag{8.3}$$

Algorithm 7.1 (Sketch of a look-ahead algorithm for inverse Hankel factorization [18]).

0) Set $u_0 = 1$, $U^{(0)} = u_0$, $\delta^{(0)} = (U^{(0)})^T M_0 U^{(0)}$.
Set $n_0 = 0$, $l = 0$, $U^{(-1)} = 0$, $\delta^{(-1)} - 1$.

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 α_n and β_n by solving

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

respectively.
Compute

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

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

3) (*Inner step*) Compute β_n by solving

$$\delta^{(l-1)}\beta_n = \begin{bmatrix} U^{(l-1)} \\ 0 \end{bmatrix}^T M_{n+1} \begin{bmatrix} 0 \\ u_n \end{bmatrix}$$

and set

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

4) Set

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

For a discussion of the look-ahead strategy used in step 1) and other implementation details of Algorithm 7.1, we refer the reader to [18]. Furthermore, in [18], we also show how solutions of Hankel systems $M_n x_n = b_n$ can be computed, based on the factorization (7.1).

We remark that Algorithm 7.1 can be viewed as an extension of a classical procedure due to Trench [38] for computing inverse factorizations of a sequence of nonsingular Hankel matrices.

is the QMR method. We refer the reader to [15, 16] 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 [15, 11]. Finally, we remark that the QMR approach can also be used to devise a *transpose-free* QMR algorithm (TFQMR) [12] that, in contrast to the standard QMR method based on Algorithm 5.1, only requires matrix-vector products with A, but no multiplications with the transpose A^T.

7. Inverse Block-Factorization of Hankel Matrices

In this section, as a second application of Algorithm 4.1, we sketch a look-ahead procedure for inverse Hankel factorization.

Let $\{M_n\}_{n=0}^{\infty}$ be a sequence of Hankel matrices (4.2), and let $\{\varphi_n\}_{n=0}^{\infty}$ be the corresponding sequence (4.4) of FOPs and quasi-FOPs generated by Algorithm 4.1. In view of Theorem 3.5, these polynomials induce, for each n, an inverse block-factorization of M_n of the form

$$U_n^T M_n U_n = D_n. \tag{7.1}$$

Here, $U_n = [\, u_{ij} \,]_{i,j=0,\ldots,n}$ is the unit upper triangular matrix (3.27) whose entries are just the coefficients of the polynomials $\{\varphi_j\}_{j=0}^{n}$, and D_n is a symmetric block-diagonal matrix defined in (3.33). More precisely, denoting the last column of U_n by

$$u_n := \begin{bmatrix} u_{0n} \\ \vdots \\ u_{n-1,n} \\ 1 \end{bmatrix},$$

we have

$$\varphi_n(\lambda) \equiv [\, 1 \quad \lambda \quad \cdots \quad \lambda^n \,]\, u_n. \tag{7.2}$$

Similarly, 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 (3.21) can be represented as follows:

$$\Phi^{(k)}(\lambda) \equiv [\, 1 \quad \lambda \quad \cdots \quad \lambda^n \,]\, U^{(k)}, \quad k = 0, 1, \ldots, l. \tag{7.3}$$

Adopting the notation introduced in Section 5, we call the vector u_n in (7.2) *regular* if it corresponds to a regular FOP φ_n, and u_n is said to be an *inner* vector, if φ_n is a quasi-FOP.

Finally, using (7.2) and (7.3) and setting $\xi_i^{(n)} = 0$ for all i, n in (4.8), we can rewrite Algorithm 4.1 in terms of the matrices U_n and D_n defining the inverse block-factorization (7.1) of M_n. This leads to the following algorithm, which was first proposed by Freund and Zha [18]. An outline of this algorithm is as follows.

always denotes the residual vector corresponding to the iterate x_{n+1}.

We now choose

$$v_0 = r_0 \quad \text{and any} \quad w_0 \in \mathbb{C}^N, \quad w_0 \neq 0, \tag{6.4}$$

as the starting vectors for Algorithm 5.1. Then, by (5.13), the Lanczos vectors $\{v_j\}_{j=0}^n$ span the Krylov subspace $K_{n+1}(r_0, A)$ in (6.2). Therefore, using the matrix V_n defined in (5.7), any iterate (6.2) can be represented in the form

$$x_{n+1} = x_0 + V_n z_n, \quad \text{where} \quad z_n \in \mathbb{C}^{n+1}. \tag{6.5}$$

It remains to select the free parameter vector z_n in (6.5). Based on the block-tridiagonal Lanczos matrix (5.17), T_n, and its extended version (5.18), $T_n^{(e)}$, there are two natural choices for z_n. The first approach is to select z_n as the solution of the linear system

$$T_n z_n = e_1, \quad \text{where} \quad e_1 := [\,1 \quad 0 \quad \cdots \quad 0\,]^T \in \mathbb{R}^{n+1}. \tag{6.6}$$

To motivate (6.6), we remark that, by (6.3), (6.4), and (5.15), the residual vector corresponding to any iterate (6.5) satisfies

$$r_{n+1} = V_n(e_1 - T_n z_n) - v_{n+1}(z_n)_n. \tag{6.7}$$

Here, e_1 is the first unit vector defined in (6.6), and $(z_n)_n$ denotes the last component of the vector z_n. Hence the choice (6.6) of z_n just eliminates all components along V_n in the representation (6.7) of r_{n+1}. It can be shown that the resulting iterative procedure is mathematically equivalent to the classical *biconjugate gradient algorithm* (BCG) [31], provided that no look-ahead steps are performed in the underlying Lanczos Algorithm 5.1. The problem with this first approach is that the coefficient matrix T_n in (6.6) 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 [15] suggested to choose z_n as the solution of the least-squares problem

$$\|f_n - \Omega_n T_n^{(e)} z_n\| = \min_{z \in \mathbb{C}^{n+1}} \|f_n - \Omega_n T_n^{(e)} z\|. \tag{6.8}$$

Here, $f_n := [\,\|r_0\| \quad 0 \quad \cdots \quad 0\,]^T \in \mathbb{R}^{n+2}$, and Ω_n is a diagonal scaling matrix given by

$$\Omega_n := \mathrm{diag}(\|v_0\|, \|v_1\|, \ldots, \|v_{n+1}\|).$$

Note that, in view of (5.19), the coefficient matrix $\Omega_n T_n^{(e)}$ of (6.8) has full column rank $n+1$, and thus (6.8) always has a unique solution z_n. The motivation for this choice of z_n is as follows. In view of (6.3), (6.4), (5.15), and (5.18), the residual vector corresponding to any iterate (6.5) satisfies

$$r_{n+1} = V_{n+1}(\Omega_n)^{-1}(f_n - \Omega_n T_n^{(e)} z_n). \tag{6.9}$$

Hence, in view of (6.8), the iterate x_{n+1} is characterized by a minimization of the second factor in the representation (6.9) of its residual vectorr_{n+1}. This is called a *quasi-minimal residual* (QMR) property, and the resulting iterative scheme for solving linear systems (6.1)

$(n+2) \times (n+1)$ matrix

$$T_n^{(e)} := \begin{bmatrix} T_n \\ e_n^T \end{bmatrix}, \quad \text{where} \quad e_n := [\,0 \quad \cdots \quad 0 \quad 1\,]^T \in \mathbb{R}^n, \tag{5.18}$$

has always full column rank:

$$\operatorname{rank} T_n^{(e)} = n+1 \tag{5.19}$$

Finally, we stress that the look-ahead Lanczos Algorithm 5.1 is different from the original look-ahead Lanczos algorithm proposed by Taylor [36] and Parlett, Taylor, and Liu [35].

6. Applications in Matrix Computations

In this section, we briefly discuss the application of Algorithm 5.1 to eigenvalue computations and to the iterative solution of linear systems.

6.1. APPROXIMATE EIGENVALUES

As we discussed in Section 2.1, relations of the type (5.15) and (5.16) suggest to use the eigenvalues of the matrix T_n as approximations to the eigenvalues of a large matrix $A \in \mathbb{C}^{N \times N}$. The resulting procedure thus consists of two steps. First, we run the look-ahead Lanczos Algorithm 5.1 for n iterations to obtain the block-tridiagonal Lanczos matrix (5.17), 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 [8], where the classical Lanczos algorithm without look-ahead is considered. Numerical results of eigenvalue computations based on the look-ahead Lanczos Algorithm 5.1 can be found in [14].

6.2. ITERATIVE SOLUTION OF LINEAR SYSTEMS

Next, we turn to systems of linear equations,

$$Ax = b, \tag{6.1}$$

with $A \in \mathbb{C}^{N \times N}$ as coefficient matrix. We are interested in computing approximate solutions of (6.1) of the form

$$x_{n+1} \in x_0 + K_{n+1}(r_0, A), \quad n = 0, 1, \ldots . \tag{6.2}$$

Here, $x_0 \in \mathbb{C}^N$ is an arbitrary initial guess for (6.1), and $r_0 := b - Ax_0$ is the associated residual vector. In the sequel,

$$r_{n+1} := b - Ax_{n+1} \tag{6.3}$$

case of the classical Lanczos algorithm, in practice, the Lanczos vectors in Algorithm 5.1 should be scaled to have unit length.

Finally, we note that a FORTRAN 77 implementation of Algorithm 5.1 is available from netlib by sending an email message consisting of the single line "send lalqmr from misc" to netlib@ornl.gov or netlib@research.att.com.

5.2. PROPERTIES

Next, we list a few properties of the look-ahead Lanczos Algorithm 5.1.

Recall that the vectors generated by the classical Lanczos Algorithm 2.1 are characterized by the properties (2.2) and (2.3). The Lanczos vectors (5.6) constructed by means of Algorithm 5.1 still satisfy (2.2), and thus, for $n = 0, 1, \ldots$, we have

$$\operatorname{span}\{v_0, v_1, \ldots, v_n\} = K_{n+1}(v_0, A), \tag{5.13}$$

$$\operatorname{span}\{w_0, w_1, \ldots, w_n\} = K_{n+1}(w_0, A^T).$$

However, instead of (2.3), the Lanczos vectors (5.6) are now only block-biorthogonal, *i.e.*,

$$(W^{(j)})^T V^{(k)} = \begin{cases} 0, & \text{if } j \neq k, \\ \delta^{(k)}, & \text{if } j = k, \end{cases} \qquad \text{for all} \quad j \neq k, \quad j, k = 0, 1, \ldots, l. \tag{5.14}$$

We note that (5.14) is just a reformulation of the block-biorthogonality condition (3.24) for polynomials.

In analogy to (2.7), the recurrences (5.9)–(5.12) for the Lanczos vectors $\{v_j\}_{j=0}^{n+1}$ and $\{w_j\}_{j=0}^{n+1}$ can be summarized in matrix form as follows:

$$AV_n = V_n T_n + [\, 0 \quad \cdots \quad 0 \quad v_{n+1} \,], \tag{5.15}$$

$$A^T W_n = W_n T_n + [\, 0 \quad \cdots \quad 0 \quad w_{n+1} \,]. \tag{5.16}$$

Here, V_n and W_n are the matrices defined in (5.7), and T_n is now an $(n+1) \times (n+1)$ block-tridiagonal matrix of the form

$$T_n := \begin{bmatrix} \mu_0 & \nu_1 & 0 & \cdots & 0 \\ 1 & \mu_1 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \nu_l \\ 0 & \cdots & 0 & 1 & \mu_l \end{bmatrix}. \tag{5.17}$$

The diagonal blocks μ_k, $k = 0, 1, \ldots, l$, in (5.17) are square matrices of order ρ_k. Recall from (5.5) that ρ_k is the length of the kth look-ahead step. If no true look-ahead steps are performed, then Algorithm 5.1 reduces to the classical Lanczos Algorithm 2.1, and the matrix (5.17) is just the scalar tridiagonal matrix (2.8) associated with Algorithm 2.1.

We remark that the diagonal block μ_k in (5.17) 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

or 3), respectively.

2) (*Regular step*) Compute

$$\alpha_n = (\delta^{(l)})^{-1}(W^{(l)})^T A v_n, \quad \beta_n = (\delta^{(l-1)})^{-1}(W^{(l-1)})^T A v_n, \tag{5.8}$$

$$v_{n+1} = A v_n - V^{(l)}\alpha_n - V^{(l-1)}\beta_n, \tag{5.9}$$

$$w_{n+1} = A^T w_n - W^{(l)}\alpha_n - W^{(l-1)}\beta_n, \tag{5.10}$$

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

3) (*Inner step*) Compute

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

$$v_{n+1} = A v_n - v_n\zeta_n - v_{n-1}\eta_n - V^{(l-1)}\beta_n, \tag{5.11}$$

$$w_{n+1} = A^T w_n - w_n\zeta_n - w_{n-1}\eta_n - W^{(l-1)}\beta_n. \tag{5.12}$$

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 5.1. In [14], 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 (4.9) that this is a necessary condition for performing a regular step. The other two checks compare the norms of α_n and β_n in (5.8) 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 α_n and β_n are not substantially bigger than $n(A)$. Roughly speaking, the motivation for these two checks comes from the recurrences (5.9) and (5.10), 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 α_n and β_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 [14, 16]. 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 (5.11) and (5.12), $\zeta_n \in \mathbb{C}$ 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 5.1 usually performs only few inner steps anyway. We mostly used the algorithm with $\zeta_n := 1$ and, if $n \neq n_k$, $\eta_n := 1$. We stress that, as in the

For each $n \geq 0$, let $l = l(n)$ be the integer defined in (3.23). Note that 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 (5.1), the blocks $\Phi^{(k)}$, $k = 0, 1, \ldots, l$, in (3.21) now correspond to blocks of Lanczos vectors defined by

$$V^{(k)} := \begin{cases} [\, v_{n_k} \quad v_{n_k+1} \quad \cdots \quad v_{n_{k+1}-1} \,], & \text{if } 0 \leq k < l, \\ [\, v_{n_l} \quad v_{n_l+1} \quad \cdots \quad v_n \,], & \text{if } k = l, \end{cases} \tag{5.2}$$

and

$$W^{(k)} := \begin{cases} [\, w_{n_k} \quad w_{n_k+1} \quad \cdots \quad w_{n_{k+1}-1} \,], & \text{if } 0 \leq k < l, \\ [\, w_{n_l} \quad w_{n_l+1} \quad \cdots \quad w_n \,], & \text{if } k = l. \end{cases} \tag{5.3}$$

Furthermore, for the blocks $\delta^{(k)}$ defined in (3.33), we obtain the following representations:

$$\delta^{(k)} = (W^{(k)})^T V^{(k)} \quad \text{for all} \quad k = 0, 1, \ldots, l. \tag{5.4}$$

Note that $V^{(k}$ and $W^{(k)}$ are $N \times \rho_k$ matrices, where

$$\rho_k := \begin{cases} n_{k+1} - n_k, & \text{if } 0 \leq k < l, \\ n + 1 - n_l, & \text{if } k = l. \end{cases} \tag{5.5}$$

The first vectors v_{n_k} and w_{n_k} in each of the blocks (5.2) and (5.3), respectively, are always regular vectors, while the remaining ones are inner vectors. The number ρ_k is called the *length* of the kth 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 (5.2) and (5.3) define a partitioning of the first $n + 1$ Lanczos vectors

$$\{v_j\}_{j=0}^n \quad \text{and} \quad \{w_j\}_{j=0}^n \tag{5.6}$$

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

$$\begin{aligned} V_n &:= [\, v_0 \quad v_1 \quad \cdots \quad v_n \,] = [\, V^{(0)} \quad V^{(1)} \quad \cdots \quad V^{(l)} \,], \\ W_n &:= [\, w_0 \quad w_1 \quad \cdots \quad w_n \,] = [\, W^{(0)} \quad W^{(1)} \quad \cdots \quad W^{(l)} \,]. \end{aligned} \tag{5.7}$$

By means of (5.1)–(5.4), we can now rewrite Algorithm 4.1 in terms of the Lanczos vectors (5.1), instead of the polynomials (4.4). The resulting procedure is just the look-ahead Lanczos algorithm proposed by Freund, Gutknecht, and Nachtigal [14]. An outline of this algorithm is as follows.

Algorithm 5.1 (Sketch of the look-ahead Lanczos algorithm [14]).

0) Choose $v_0, w_0 \in \mathbb{C}^N$ with $v_0, w_0 \neq 0$.
Set $V^{(0)} = v_0$, $W^{(0)} = w_0$, $\delta^{(0)} = (W^{(0)})^T V^{(0)}$.
Set $n_0 = 0$, $l = 0$, $v_{-1} = w_{-1} = 0$, $V^{(-1)} = v_{-1}$, $W^{(-1)} = w_{-1}$, $\delta^{(-1)} = 1$.

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)

3) (*Constructing a quasi*-FOP) Compute

$$\beta_n = (\delta^{(l-1)})^{-1} \langle \Phi^{(l-1)}, \lambda\varphi_n \rangle, \tag{4.7}$$

$$\varphi_{n+1} = \lambda\varphi_n - \Phi^{(l-1)}\beta_n - \sum_{i=n_l}^{n} \xi_i^{(n)} \varphi_i. \tag{4.8}$$

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

We remark that, in (4.8), $\xi_i^{(n)} \in \mathbb{C}$ are arbitrary recurrence coefficients.

The strategy for deciding in step 1) of Algorithm 4.1 whether to build the next polynomial as a regular FOP or as a quasi-FOP is called the *look-ahead strategy*. In Section 5, we will briefly discuss the look-ahead strategy for one particular application of Algorithm 4.1. Here, we only remark that, in view of (3.31), for constructing φ_{n+1} as a regular FOP it is necessary that

$$\delta^{(l)} \quad \text{is nonsingular.} \tag{4.9}$$

Moreover, if the look-ahead strategy is only based on (4.9), then Algorithm 4.1 constructs all existing regular FOPs. Note that the condition (4.9) guarantees that the inverse matrices in (4.5), respectively (4.7), always exist.

Algorithm 4.1 shows that regular FOPs associated with general Hankel moment matrices can always be constructed by means of short recurrences (4.6), respectively (4.8), which involve only polynomials from the last two blocks $\Phi^{(l)}$ and $\Phi^{(l-1)}$. Finally, if all polynomials are constructed as regular FOPs, then (4.6) reduces to a simple three-term of the form (2.16).

5. 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 4.1, we obtain a look-ahead Lanczos algorithm.

5.1. THE 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 defined by (4.3) and (4.1), and let $\{\varphi_n\}_{n=0}^{\infty}$ be the sequence (4.4) of FOPS and quasi-FOPs generated by Algorithm 4.1. Recall from (3.18) that the indices n_k mark the regular FOPs.

In analogy to (2.12), we use the polynomials φ_n to define Lanczos vectors by setting

$$v_n := \varphi_n(A)v_0 \quad \text{and} \quad w_n := \varphi_n(A^T)w_0. \tag{5.1}$$

If $n = n_k$ and thus φ_{n_k} is a regular FOP, then the vectors v_{n_k} and w_{n_k} are called *regular vectors*. If φ_n is a quasi-FOP, then we refer to v_n and w_n as *inner vectors*.

matrices (3.4) are then of the form

$$M_n = [\,h_{j+k}\,]_{j,k=0,\ldots,n} = \begin{bmatrix} h_0 & h_1 & h_2 & \cdots & h_n \\ h_1 & \iddots & & \iddots & \vdots \\ h_2 & & \iddots & & \vdots \\ \vdots & \iddots & & & h_{2n-1} \\ h_n & \cdots & \cdots & h_{2n-1} & h_{2n} \end{bmatrix}, \quad n = 0, 1, \ldots . \tag{4.2}$$

Matrices of the form (4.2) are called *Hankel matrices*. Note that the moment matrices (4.2) always satisfy the symmetry condition (3.8), and thus $\langle \cdot, \cdot \rangle$ is a symmetric bilinear form. Therefore, the corresponding polynomials are FOPs and quasi-FOPs. Finally, we remark that, by (3.6), the special bilinear form (2.14) underlying the Lanczos algorithm satisfies the condition (4.1) with

$$h_j = w_0^T A^j v_0. \tag{4.3}$$

In [18], Freund and Zha derived an algorithm for constructing a sequence

$$\{\varphi_n\}_{n=0}^{\infty} \tag{4.4}$$

of regular FOPs and quasi-FOPs associated with general moment matrices of the form (4.2). Next, we present a sketch of this algorithm. We use the notation introduced in Section 3.3. In particular, indices (3.18) are used to mark the regular FOPs, and the first $n+1$ polynomials $\{\varphi_j\}_{j=0}^{n}$ are partitioned into blocks $\Phi^{(k)}$, $k = 0, 1, \ldots, l$, given by (3.21). Here, $l = l(n)$ is defined in (3.23); note that n_l is just the index of the last regular FOP φ_{n_l} with index $\leq n$. Finally, $\delta^{(k)}$ are the blocks defined in (3.33).

With these preparations, the algorithm for constructing (4.4) can now be sketched as follows.

Algorithm 4.1 (Look-ahead process for constructing FOPs [18]).

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

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

1) Decide whether to construct φ_{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) Compute

$$\alpha_n = (\delta^{(l)})^{-1} \langle \Phi^{(l)}, \lambda\varphi_n \rangle, \quad \beta_n = (\delta^{(l-1)})^{-1} \langle \Phi^{(l-1)}, \lambda\varphi_n \rangle, \tag{4.5}$$

$$\varphi_{n+1} = \lambda\varphi_n - \Phi^{(l)}\alpha_n - \Phi^{(l-1)}\beta_n, \tag{4.6}$$

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

is a block-diagonal matrix. Moreover, in (3.29)*, the blocks*

$$\delta^{(k)} \quad \textit{are nonsingular for all} \quad k = 0, 1, \ldots, l-1, \tag{3.30}$$

and

$$\delta^{(l)} \quad \textit{is nonsingular if} \quad n = n_{l+1} - 1. \tag{3.31}$$

Proof. By means of (3.5), we can rewrite the block-biorthogonality conditions (3.24) in terms of M_n and the coefficients u_{ij} and v_{ij} of the polynomials (3.25). The result is just the inverse block-factorization (3.26) with matrices U_n, V_n, and D_n given by (3.27)–(3.29).

Using (3.26)–(3.29), we deduce that

$$\det M_{n_l-1} = \prod_{k=0}^{l-1} \det \delta^{(k)} \quad \text{and} \quad \det M_n = (\det M_{n_l-1})(\det \delta^{(l)}). \tag{3.32}$$

In view of Lemma 3.2, the matrix M_{n_l-1} is nonsingular, and, if $n = n_{l+1} - 1$, the matrix M_n is nonsingular. Hence (3.30) and (3.31) follow directly from (3.32). □

Finally, consider the case that $\langle \cdot, \cdot \rangle$ is a symmetric bilinear form. Then, in view of (3.17), we have $u_{ij} = v_{ij}$ for all i, j in (3.25), and (3.26) reduces to the symmetric inverse block-factorization

$$U_n^T M_n U_n = D_n$$

of the symmetric moment matrix M_n. Here, U_n is given by (3.27) and

$$D_n := \operatorname{diag}(\delta^{(0)}, \delta^{(1)}, \ldots, \delta^{(l)}), \quad \text{where} \quad \delta^{(k)} := \langle \Phi^{(k)}, \Phi^{(k)} \rangle, \tag{3.33}$$

is a symmetric block-diagonal matrix.

4. Constructing FOPs Associated with Hankel Matrices

Polynomials that are orthogonal with respect to a true inner product satisfy simple three-term recurrences, see, *e.g.*, [6]. This is no longer for FBOPs and FOPs associated with arbitrary bilinear forms. In general, the construction of these polynomials requires all previous polynomials. However, there are two important special cases, namely bilinear forms corresponding to Hankel and Toeplitz moment matrices, for which FOPs and FBOPs can be constructed by means of short recurrences involving only few of the previous polynomials. In this section, we discuss the Hankel case. The Toeplitz case is treated in Section 8.

In the following, we assume that $\langle \cdot, \cdot \rangle$ is a bilinear form whose moments m_{jk} depend only on the *sum* $j + k$ of their indices, *i.e.*,

$$m_{jk} = h_{j+k} \quad \text{for all} \quad j, k = 0, 1, \ldots . \tag{4.1}$$

Here $\{h_j\}_{j=0}^{\infty}$ is any sequence of real or complex numbers. The corresponding moment

Using the biorthogonality conditions (3.9) and (3.10) for FBOPs and the relations (3.13) and (3.14) for n_k-quasi FBOPs, one readily verifies that the blocks (3.21) and (3.22) satisfy the block-biorthogonality conditions

$$\langle \Psi^{(j)}, \Phi^{(k)} \rangle = \begin{cases} 0, & \text{if } j \neq k, \\ \langle \Psi^{(k)}, \Phi^{(k)} \rangle, & \text{if } j = k, \end{cases} \qquad \text{for all} \quad j, k = 0, 1, \ldots, l. \tag{3.24}$$

Here and in the sequel, we have used the notation

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

where

$$\Phi = [\, \phi_0 \quad \phi_1 \quad \cdots \quad \phi_m \,] \quad \text{and} \quad \Psi = [\, \psi_0 \quad \psi_1 \quad \cdots \quad \psi_i \,]$$

are row vectors of polynomials in $\mathcal{P}$.

After these preliminaries, we can now state the main result of this section.

Theorem 3.5. *Let $\{n_k\}_{k=0}^{K} \subseteq \mathcal{I}$ be a given sequence of regular indices* (3.18). *Let $n \geq 0$, and let $l = l(n)$ be defined by* (3.23). *Let*

$$\varphi_j(\lambda) \equiv \lambda^j + \sum_{i=0}^{j-1} u_{ij} \lambda^i \quad \textit{and} \quad \psi_j(\lambda) \equiv \lambda^j + \sum_{i=0}^{j-1} v_{ij} \lambda^i, \quad j = 0, 1, \ldots, n, \tag{3.25}$$

be the subsequence (3.20) *of the polynomials* (3.19) *associated with $\{n_k\}_{k=0}^{K}$. Then the nth moment matrix M_n associated with the bilinear form $\langle \cdot, \cdot \rangle$ has the following inverse block-factorization*:

$$V_n^T M_n U_n = D_n, \tag{3.26}$$

where

$$U_n := [\, u_{ij} \,]_{i,j=0,\ldots,n} = \begin{bmatrix} 1 & u_{01} & \cdots & u_{0n} \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & u_{n-1,n} \\ 0 & \cdots & 0 & 1 \end{bmatrix} \tag{3.27}$$

and

$$V_n := [\, v_{ij} \,]_{i,j=0,\ldots,n} = \begin{bmatrix} 1 & v_{01} & \cdots & v_{0n} \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & v_{n-1,n} \\ 0 & \cdots & 0 & 1 \end{bmatrix} \tag{3.28}$$

are unit upper triangular matrices whose columns are just the coefficients of the polynomials (3.25), *and*

$$D_n := \operatorname{diag}(\delta^{(0)}, \delta^{(1)}, \ldots, \delta^{(l)}), \quad \textit{where} \quad \delta^{(k)} := \langle \Psi^{(k)}, \Phi^{(k)} \rangle, \tag{3.29}$$

but fixed, bilinear form, and FBOPs and quasi-FBOPs refer to polynomials associated with this bilinear form.

First, we need to introduce some further notation, which will also be used later on. We denote by $\mathcal{I}$ the set of all integers n for which regular FBOPs of degree n exist. In view of Lemma 3.2, $\mathcal{I}$ just consists of all integers for which M_{n-1} is nonsingular. Moreover, we always have $0 \in \mathcal{I}$. Note that, for $n = 0$, the conditions (3.9) and (3.10) are void, and thus $\varphi_0(\lambda) \equiv 1$ and $\psi_0(\lambda) \equiv 1$ are regular FBOPs of degree 0. Now let $\{n_k\}_{k=0}^{K} \subseteq \mathcal{I}$ be an arbitrary subsequence of $\mathcal{I}$; here either $K = \infty$ or K is an integer. We always assume that $\{n_k\}_{k=0}^{K}$ includes 0, and that the n_k's are numbered in increasing order. Therefore, we have

$$0 =: n_0 < n_1 < \cdots < n_k < \cdots . \tag{3.18}$$

We refer to the numbers n_k in (3.18) as *regular indices*. Finally, if $K < \infty$, then we set $n_{K+1} := \infty$.

Based on the given regular indices (3.18), we now define two sequences of monic polynomials

$$\{\varphi_n\}_{n=0}^{\infty} \quad \text{and} \quad \{\psi_n\}_{n=0}^{\infty} \tag{3.19}$$

as follows. For $n = n_k$, $0 \le k \le K$, let φ_n and ψ_n be the regular right and left FBOPs of degree n. Since $n_k \in \mathcal{I}$, these regular FBOPs are guaranteed to exist. For $n_k < n < n_{k+1}$, $0 \le k \le K$, we choose φ_n and ψ_n as any pair of right and left n_k-quasi-FBOPs of degree n. Note that $n_k \in \mathcal{I}$ implies that M_{n_k-1} is nonsingular, and thus, by Lemma 3.4, n_k-quasi-FBOPs of degree n are guaranteed to exist. However, we stress that these n_k-quasi-FBOPs are not unique.

We now show that, for each n, the subsequences

$$\{\varphi_j\}_{j=0}^{n} \quad \text{and} \quad \{\psi_j\}_{j=0}^{n} \tag{3.20}$$

of (3.19) induce an *inverse block-factorization* of the nth moment matrix M_n. In the sequel, let n be arbitrary, but fixed. Using (3.18), the polynomials (3.20) can be partitioned into blocks

$$\Phi^{(k)} := \begin{cases} [\,\varphi_{n_k} \quad \varphi_{n_k+1} \quad \cdots \quad \varphi_{n_{k+1}-1}\,], & \text{if } 0 \le k < l, \\ [\,\varphi_{n_l} \quad \varphi_{n_l+1} \quad \cdots \quad \varphi_{n}\,], & \text{if } k = l, \end{cases} \tag{3.21}$$

and

$$\Psi^{(k)} := \begin{cases} [\,\psi_{n_k} \quad \psi_{n_k+1} \quad \cdots \quad \psi_{n_{k+1}-1}\,], & \text{if } 0 \le k < l, \\ [\,\psi_{n_l} \quad \psi_{n_l+1} \quad \cdots \quad \psi_{n}\,], & \text{if } k = l. \end{cases} \tag{3.22}$$

Here, $l := l(n)$ is the unique integer defined by

$$n_l \le n < n_{l+1}, \quad 0 \le l \le K. \tag{3.23}$$

Note that $\Phi^{(k)}$ and $\Psi^{(k)}$ are row vectors of polynomials. We remark that the first polynomial in each block $\Phi^{(k)}$ and $\Psi^{(k)}$ is a regular FBOP, while the remaining polynomials are n_k-quasi-FBOPs.

respect to the bilinear form (3.1)) of degree n if $m < n$ and

$$\langle \psi, \varphi_n \rangle = 0 \quad \text{for all} \quad \psi \in \mathcal{P}_{m-1}. \tag{3.13}$$

A monic polynomial $\psi_n \in \mathcal{P}_n$ is called a *left m-quasi*-FBOP (with respect to the bilinear form (3.1)) of degree n if $m < n$ and

$$\langle \psi_n, \varphi \rangle = 0 \quad \text{for all} \quad \varphi \in \mathcal{P}_{m-1}. \tag{3.14}$$

The next lemma gives a sufficient condition for the existence of m-quasi-FBOPs.

Lemma 3.4. *Let $1 \leq m < n$. If M_{m-1} is nonsingular, then right and left m-quasi-FBOPs φ_n and ψ_n of degree n exist.*

Proof. Let φ_n be a monic polynomial of the form

$$\varphi_n(\lambda) \equiv \lambda^n + \sum_{i=0}^{m-1} u_{in} \lambda^i. \tag{3.15}$$

By (3.13) and (3.5), the polynomial (3.15) is a right m-quasi-FBOP if its coefficients $u_{0n}, u_{1n}, \ldots, u_{m-1,n}$ satisfy

$$M_{m-1} \begin{bmatrix} u_{0n} \\ u_{1n} \\ \vdots \\ u_{m-1,n} \end{bmatrix} = - \begin{bmatrix} m_{0n} \\ m_{1n} \\ \vdots \\ m_{m-1,n} \end{bmatrix}. \tag{3.16}$$

If M_{m-1} is nonsingular, the linear system (3.16) is guaranteed to have a solution, and the corresponding polynomial (3.15) is a right m-quasi-FBOP.

Similarly, one shows that a left m-quasi-FBOP exists if M_{m-1} is nonsingular. □

We remark that, for symmetric bilinear forms $\langle \cdot, \cdot \rangle$, the conditions (3.9) and (3.10) are equivalent, and thus right and left regular FBOPs coincide, *i.e.*,

$$\varphi_n = \psi_n. \tag{3.17}$$

Furthermore, the conditions (3.13) and (3.14) for right and left m-quasi-FBOPs are also equivalent. Consequently, in the case of symmetric bilinear forms $\langle \cdot, \cdot \rangle$, we do not have to distinguish between right and left FBOPs, respectively right and left m-quasi FBOPs, and we can assume (3.17). Finally, for symmetric bilinear forms, the conditions (3.9) and (3.10), respectively (3.13) and (3.14), represent orthogonality relations, rather than biorthogonality relations. Hence, we refer to φ_n as a *formally biorthogonal polynomial* (FOP), respectively an m-quasi-FOP.

3.3. INVERSE BLOCK-FACTORIZATIONS OF MOMENT MATRICES

We now show how FBOPs and quasi-FBOPs can be used to construct inverse block-factorizations of moment matrices. In the sequel, we assume that $\langle \cdot, \cdot \rangle$ is an arbitrary,

A monic polynomial $\psi_n \in \mathcal{P}_n$ is called a *left* FBOP (with respect to the bilinear form (3.1)) of degree n if

$$\langle \psi_n, \varphi \rangle = 0 \quad \text{for all} \quad \varphi \in \mathcal{P}_{n-1}. \tag{3.10}$$

A right or left FBOP φ_n or ψ_n is said to be *regular* if it is uniquely determined by (3.9) or (3.10), respectively.

In contrast to orthogonal polynomials associated with a true inner product, right and left FBOPs φ_n and ψ_n need not exist or need not be unique for every degree n. Next, we derive a necessary and sufficient condition for the existence of regular FBOPs.

In view of (3.9) and (3.5), a monic polynomial

$$\varphi_n(\lambda) \equiv \lambda^n + \sum_{i=0}^{n-1} u_{in} \lambda^i$$

is a right FBOP of degree n if, and only if, its coefficients u_{in} satisfy

$$M_{n-1} \begin{bmatrix} u_{0n} \\ u_{1n} \\ \vdots \\ u_{n-1,n} \end{bmatrix} = - \begin{bmatrix} m_{0n} \\ m_{1n} \\ \vdots \\ m_{n-1,n} \end{bmatrix}. \tag{3.11}$$

Analogously, by (3.10) and (3.5), a monic polynomial

$$\psi_n(\lambda) \equiv \lambda^n + \sum_{i=0}^{n-1} v_{in} \lambda^i$$

is a left FBOP of degree n if, and only if, its coefficients v_{in} fulfill

$$[\, v_{0n} \quad v_{1n} \quad \ldots \quad v_{n-1,n} \,] M_{n-1} = - [\, m_{n0} \quad m_{n1} \quad \ldots \quad m_{n-1,n} \,]. \tag{3.12}$$

As an immediate consequence of (3.11) and (3.12), we have the following result.

Lemma 3.2. *The following conditions are equivalent*:
(*i*) *A regular right* FBOP φ_n *of degree* n *exists.*
(*ii*) *A regular left* FBOP ψ_n *of degree* n *exists.*
(*iii*) *The matrix* M_{n-1} *is nonsingular.*

In view of Lemma 3.2, no regular FBOPs of degree n exist if M_{n-1} is singular. Furthermore, the proof of Lemma 3.2 suggests that, in finite-precision arithmetic, it is not advisable to compute regular FBOPs if M_{n-1} is nonsingular, but ill-conditioned. For the case of singular or ill-conditioned moment matrices M_{n-1}, we will need the concept of m-quasi-FBOPs that only satisfy relaxed versions of the biorthogonality conditions (3.9) and (3.10). The precise definition of m-quasi-FBOPs is as follows.

Definition 3.3. A monic polynomial $\varphi_n \in \mathcal{P}_n$ is called a *right* m*-quasi*-FBOP (with

are called the *moments* of $\langle \cdot, \cdot \rangle$. For $n = 0, 1, \ldots$, we set

$$M_n := [\, m_{jk} \,]_{j,k=0,\ldots,n} = \begin{bmatrix} m_{00} & m_{01} & \cdots & m_{0n} \\ m_{10} & m_{11} & & \vdots \\ \vdots & & \ddots & \vdots \\ m_{n0} & \cdots & \cdots & m_{nn} \end{bmatrix}. \tag{3.4}$$

The matrix (3.4) is called the nth *moment matrix* associated with $\langle \cdot, \cdot \rangle$.

With (3.2)–(3.4), it follows that for any two polynomials

$$\varphi(\lambda) \equiv \sum_{j=0}^{n} \sigma_j \lambda^j, \quad \psi(\lambda) \equiv \sum_{j=0}^{n} \tau_j \lambda^j$$

of degree at most n, $n = 0, 1, \ldots$, we have

$$\langle \psi, \varphi \rangle = v^T M_n u, \quad \text{where} \quad u := \begin{bmatrix} \sigma_0 \\ \sigma_1 \\ \vdots \\ \sigma_n \end{bmatrix}, \quad v := \begin{bmatrix} \tau_0 \\ \tau_1 \\ \vdots \\ \tau_n \end{bmatrix}. \tag{3.5}$$

This shows that a bilinear form is uniquely determined by its moments.

We note that the special form (2.14) underlying the Lanczos process clearly satisfies the bilinearity conditions (3.2). In view of (3.3) and (2.14), the moments of this bilinear form are given by

$$m_{jk} = w_0^T A^{j+k} v_0, \quad j, k = 0, 1, \ldots . \tag{3.6}$$

Finally, we remark that for the Lanczos bilinear form (2.14), we have

$$\langle \psi, \varphi \rangle = \langle \varphi, \psi \rangle \quad \text{for all} \quad \varphi, \psi \in \mathcal{P}. \tag{3.7}$$

In general, a bilinear form (3.1) with the additional property (3.7) is said to be *symmetric*. In view of (3.5), a bilinear is symmetric if, and only if,

$$M_n = M_n^T \quad \text{for all} \quad n = 0, 1 \ldots . \tag{3.8}$$

3.2. FORMALLY BIORTHOGONAL AND ORTHOGONAL POLYNOMIALS

We stress that, in general, a bilinear form is not an inner product. Indeed, it is possible that a nonzero polynomial φ 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 a given bilinear form (3.1). We now give a definition of these so-called *formally biorthogonal polynomials* (FBOPs). Recall that a polynomial $\varphi_n \in \mathcal{P}_n$ is said to be *monic* if it is of the form (2.13).

Definition 3.1. A monic polynomial $\varphi_n \in \mathcal{P}_n$ is called a *right* FBOP (with respect to the bilinear form (3.1)) of degree n if

$$\langle \psi, \varphi_n \rangle = 0 \quad \text{for all} \quad \psi \in \mathcal{P}_{n-1}. \tag{3.9}$$

and thus the biorthogonality condition (2.3) of the Lanczos vectors can be rewritten as follows:

$$\langle \varphi_j, \varphi_k \rangle = \begin{cases} 0, & \text{if } j \neq k, \\ \langle \varphi_k, \varphi_k \rangle \neq 0, & \text{if } j = k, \end{cases} \qquad \text{for all} \quad j, k = 0, 1, \ldots, n. \tag{2.15}$$

In view of (2.15), the Lanczos polynomials are orthogonal with respect to the bilinear form (2.14). Furthermore, the recursions (2.5) for the Lanczos vectors are just a reformulation of the usual three-term polynomial recurrence

$$\varphi_{n+1} = \lambda \varphi_n - \alpha_n \varphi_n - \beta_n \varphi_{n-1} \tag{2.16}$$

for orthogonal polynomials. Note that, by (2.4), (2.12), and (2.14), the coefficients in (2.16) are given by

$$\alpha_n = \frac{\langle \varphi_n, \lambda \varphi_n \rangle}{\langle \varphi_n, \varphi_n \rangle}, \quad \beta_n = \frac{\langle \varphi_n, \varphi_n \rangle}{\langle \varphi_{n-1}, \varphi_{n-1} \rangle}. \tag{2.17}$$

Finally, the conditions (2.10) and (2.11) for an exact breakdown or a near-breakdown of the classical Lanczos algorithm translate into the conditions

$$\langle \varphi_n, \varphi_n \rangle = 0 \quad \text{or} \quad \langle \varphi_n, \varphi_n \rangle \approx 0, \tag{2.18}$$

which signal an exact breakdown or a near-breakdown in the recursions (2.16) and (2.17). In Section 4, we present modified recurrences that allow to continue the construction of the Lanczos polynomials even if (2.18) occurs. These recurrences are the basis for the look-ahead Lanczos process.

3. Bilinear Forms and Formally Biorthogonal Polynomials

In Section 2.2, we saw that the Lanczos polynomials are orthogonal with respect to the particular bilinear form (2.14). In this section, we consider arbitrary bilinear forms, and we introduce the more general concepts of formally biorthogonal polynomials and formally orthogonal polynomials.

3.1. BILINEAR FORMS

A complex-valued functional

$$\langle \cdot, \cdot \rangle : \mathcal{P} \times \mathcal{P} \longmapsto \mathbb{C} \tag{3.1}$$

is called a *bilinear form* if

$$\begin{aligned} \langle \psi, \sigma_1 \varphi_1 + \sigma_2 \varphi_2 \rangle &= \sigma_1 \langle \psi, \varphi_1 \rangle + \sigma_2 \langle \psi, \varphi_2 \rangle \quad \text{for all} \quad \psi, \varphi_1, \varphi_2 \in \mathcal{P}, \ \sigma_1, \sigma_2 \in \mathbb{C}, \\ \langle \tau_1 \psi_1 + \tau_2 \psi_2, \varphi \rangle &= \tau_1 \langle \psi_1, \varphi \rangle + \tau_2 \langle \psi_2, \varphi \rangle \quad \text{for all} \quad \psi_1, \psi_2, \varphi \in \mathcal{P}, \ \tau_1, \tau_2 \in \mathbb{C}. \end{aligned} \tag{3.2}$$

The numbers

$$m_{jk} := \langle \lambda^j, \lambda^k \rangle, \quad j, k = 0, 1, \ldots, \tag{3.3}$$

constitutes a *breakdown* of the Lanczos algorithm. Of course, in finite-precision arithmetic, exact breakdowns (2.10) are unlikely; however, *near-breakdowns*, *i.e.*,

$$w_n^T v_n \approx 0, \quad \text{but} \quad v_n \not\approx 0 \quad \text{and} \quad w_n \not\approx 0, \tag{2.11}$$

may occur, which can then cause numerical instabilities in subsequent iterations. Look-ahead Lanczos methods are modifications of the classical Algorithm 2.1 that allow to continue the Lanczos process even in the case of an exact breakdown (2.10) or a near-breakdown (2.11).

We remark that, for the special case of Hermitian matrices A, the possibility of breakdowns can be excluded. In fact, if $A = A^H$, then one sets $w_0 = \overline{v_0}$, and from (2.4) and (2.5), it follows that $w_n = \overline{v_n}$ for all n. Hence, we have

$$w_n^T v_n = v_n^H v_n = \|v_n\|^2 > 0 \quad \text{if} \quad v_n \neq 0,$$

and this shows that breakdowns cannot occur. Furthermore, in the Hermitian Lanczos process, only the vectors v_n are generated, using the first relation in (2.5). In particular, work and storage requirements are roughly halved for Hermitian matrices.

There is another special case for which the two recurrences (2.5) reduce to one, namely complex symmetric matrices $A = A^T$. Here, one sets $w_0 = v_0$, and this implies that $w_n = v_n$ for all n. However, we stress that, in contrast to the Hermitian Lanczos process, exact breakdowns and near-breakdowns in the resulting complex symmetric Lanczos algorithm cannot be excluded. For a detailed discussion of the complex symmetric Lanczos process, we refer the reader to [10].

2.2. POLYNOMIAL FORMULATION

There is an equivalent formulation of Algorithm 2.1 in terms of polynomials, instead of vectors. In fact, from (2.5), it follows that each pair of Lanczos vectors v_n and w_n can be represented in the form

$$v_n = \varphi_n(A)v_0 \quad \text{and} \quad w_n = \varphi_n(A^T)w_0, \tag{2.12}$$

where φ_n is a *monic* polynomial of degree n, *i.e.*,

$$\varphi_n(\lambda) \equiv \lambda^n + \cdots \in \mathcal{P}_n. \tag{2.13}$$

We note that φ_n is uniquely determined by (2.12). The polynomials φ_n defined by (2.12) are called *Lanczos polynomials*. Using (2.12), we now rewrite Algorithm 2.1 and some of its properties 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

$$\langle \psi, \varphi \rangle := w_0^T \psi(A)\varphi(A)v_0 \quad \text{for all} \quad \varphi, \psi \in \mathcal{P}. \tag{2.14}$$

From (2.12) and (2.14), we have

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

Next, we list a few properties of Algorithm 2.1. The recursions (2.5) for all the Lanczos vectors $\{v_j\}_{j=0}^{n+1}$ and $\{w_j\}_{j=0}^{n+1}$ can be written compactly in matrix form as follows:

$$\begin{aligned} AV_n &= V_n T_n + [\,0 \quad \cdots \quad 0 \quad v_{n+1}\,], \\ A^T W_n &= W_n T_n + [\,0 \quad \cdots \quad 0 \quad w_{n+1}\,]. \end{aligned} \tag{2.7}$$

Here

$$V_n := [\,v_0 \quad v_1 \quad \cdots \quad v_n\,] \quad \text{and} \quad W_n := [\,w_0 \quad w_1 \quad \cdots \quad w_n\,]$$

are matrices whose columns are just the Lanczos vectors (2.1), and

$$T_n := \begin{bmatrix} \alpha_0 & \beta_1 & 0 & \cdots & 0 \\ 1 & \alpha_1 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \beta_n \\ 0 & \cdots & 0 & 1 & \alpha_n \end{bmatrix} \tag{2.8}$$

is an $(n+1) \times (n+1)$ tridiagonal matrix whose entries are just the recurrence coefficients in (2.5).

The Lanczos process is mostly applied to large matrices A, and usually, the iteration index n is much smaller than the order N of A. From (2.7), it is apparent how the Lanczos algorithm can be used to obtain approximate eigenvalues of large matrices A or approximate solutions of large linear systems $Ax = b$. Indeed, if $v_{n+1} = 0$ or $w_{n+1} = 0$ in (2.7), then each eigenvalue of T_n would also be an eigenvalue of A. This suggests to use the eigenvalues of T_n as approximate eigenvalues of A. Similarly, an approximate solution of $Ax = b$ can be obtained by 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 6.

We stress that the termination check $w_n^T v_n = 0$ in step 1) of Algorithm 2.1 is indispensable to avoid division by 0 in (2.4). In exact arithmetic, this check will be satisfied after at most N iterations. Normally, the algorithm stops because $w_n^T v_n = 0$ is satisfied with

$$v_n = 0 \quad \text{or} \quad w_n = 0. \tag{2.9}$$

It is easy to verify that, in (2.9), the case $v_n = 0$ occurs if, and only if, $K_n(v_0, A)$ is an A-invariant subspace of $\mathbb{C}^N$. In view of (2.2), the Lanczos vectors $\{v_j\}_{j=0}^{n-1}$ then build a basis for the A-invariant subspace $K_n(v_0, A)$. Similarly, if $w_n = 0$ in (2.9), then the vectors $\{w_j\}_{j=0}^{n-1}$ span the A^T-invariant subspace $K_n(w_0, A^T)$ of $\mathbb{C}^N$. The event (2.9) is called *regular termination*.

Unfortunately, it can also happen that Algorithm 2.1 stops because

$$w_n^T v_n = 0, \quad \text{but} \quad v_n \neq 0 \quad \text{and} \quad w_n \neq 0. \tag{2.10}$$

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 (2.10)

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 [30] generates two sequences of vectors

$$\{v_j\}_{j=0}^{n} \quad \text{and} \quad \{w_j\}_{j=0}^{n}, \quad n = 0, 1, \ldots, \tag{2.1}$$

such that, for all n,

$$\begin{aligned} \operatorname{span}\{v_0, v_1, \ldots, v_n\} &= K_{n+1}(v_0, A), \\ \operatorname{span}\{w_0, w_1, \ldots, w_n\} &= K_{n+1}(w_0, A^T). \end{aligned} \tag{2.2}$$

and the two sequences are biorthogonal, *i.e.*,

$$w_j^T v_k = \begin{cases} 0, & \text{if } j \neq k, \\ \delta_k \neq 0, & \text{if } j = k, \end{cases} \qquad \text{for all} \quad j, k = 0, 1, \ldots, n. \tag{2.3}$$

It turns out that vectors (2.1) satisfying (2.2) and (2.3) 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 [30])**.**

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

$$\alpha_n = \frac{w_n^T A v_n}{\delta_n}, \quad \beta_n = \frac{\delta_n}{\delta_{n-1}}. \tag{2.4}$$

3) Set

$$\begin{aligned} v_{n+1} &= A v_n - v_n \alpha_n - v_{n-1} \beta_n, \\ w_{n+1} &= A^T w_n - w_n \alpha_n - w_{n-1} \beta_n. \end{aligned} \tag{2.5}$$

We remark that, in practice, one uses a scaled version of the recurrences (2.5), in order to avoid over– or underflow. A proven choice is to scale the Lanczos vectors to have unit length:

$$\|v_n\| = \|w_n\| = 1 \quad \text{for all} \quad n. \tag{2.6}$$

In this paper, we will always use the unscaled recurrences (2.5), since they simplify the exposition of the algorithms. However, for practical purposes, we recommend the scaling (2.6) of the Lanczos vectors.

of coding theory is given in [2, 29, 7]. We stress that, in all these cases, only the problem of exact breakdowns was addressed. Nevertheless, 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. Among the many possible approaches to the look-ahead Lanczos process, we have chosen the one based on the connection with FOPs. The reason for this is twofold. First, this approach appears to be the most elementary. Second, as a by-product, we immediately obtain a related look-ahead procedure for computing inverse block-factorizations of Hankel matrices. We also show that similar techniques, based on formally biorthogonal polynomials (FBOPs), can be used to derive a look-ahead variant of the classical Levinson algorithm for solving general Toeplitz systems.

The remainder of this paper is organized as follows. In Section 2, we briefly review the classical Lanczos algorithm. In Section 3, we introduce the concept of FBOPs and FOPs associated with general bilinear forms, and we show that these polynomials lead to inverse block-factorizations of the corresponding moment matrices. In Section 4, we present a look-ahead procedure for constructing FOPs associated with Hankel matrices. In Section 5, we describe the corresponding look-ahead Lanczos algorithm for general non-Hermitian matrices. In Section 6, we briefly discuss applications of this algorithm in matrix computations. In Section 7, we sketch a look-ahead procedure for computing inverse block-factorizations of Hankel matrices. We then turn to FBOPs associated with general Toeplitz matrices. In Section 8, we present recurrence relations for the construction of such FBOPs. Finally, in Section 9, we present a look-ahead Levinson algorithm for the solution of linear systems with Toeplitz matrices.

Throughout the paper, all vectors and matrices are allowed to have real or complex entries. As usual, $\overline{M} = [\,\overline{m_{jk}}\,]$, $M^T = [\,m_{kj}\,]$, and $M^H = \overline{M}^T = [\,\overline{m_{kj}}\,]$ denote the complex conjugate, 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 nth *Krylov subspace* of $\mathbb{C}^N$ generated by $c \in \mathbb{C}^N$ and $B \in \mathbb{C}^{N\times N}$. We denote by

$$\mathcal{P}_n := \{\varphi(\lambda) \equiv \sigma_0 + \sigma_1\lambda + \cdots + \sigma_n\lambda^n \mid \sigma_0, \sigma_1, \ldots, \sigma_n \in \mathbb{C}\}$$

the set of all complex polynomials of degree at most n, and $\mathcal{P}$ is the set of all complex polynomials of arbitrary degree. Finally, 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 for general square matrices.

to compute approximate eigenvalues of A, or to obtain approximate solutions of systems of linear equations, $Ax = b$, with A as coefficient matrix.

Unfortunately, in the classical nonsymmetric Lanczos method 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. On the other hand, Cullum and Willoughby [8] demonstrated that the Lanczos process—even in its original form—can be a powerful tool for large sparse matrix computations. For the special case of Hermitian matrices A, the Lanczos process simplifies, and work per iteration and storage requirements are roughly halved. More importantly, for the resulting Hermitian Lanczos method, breakdowns and near-breakdowns can be excluded. We remark that, unlike the general Lanczos process, the Hermitian Lanczos method has found widespread use and has been studied extensively since the 1970s; we refer the reader to [33, 19] and the references given there.

On the other hand, it is possible to modify the Lanczos process so that it skips over those iterations in which an exact breakdown or a near-breakdown would occur in the standard method. Such a modified Lanczos method, called the *look-ahead Lanczos algorithm*, was first developed by Taylor [36] and Parlett, Taylor, and Liu [35] in the early 1980s. However, their procedure is fairly complicated, and in fact, details were only given for the simplest case of look-ahead steps of length 2. 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 number of papers that were written since 1989. For example, the papers [3, 4, 5, 10, 13, 14, 18, 23, 24, 26, 34] all deal with various aspects of the look-ahead Lanczos process.

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 [20, 9, 23] and the references given there. In the context of the partial realization problem in control theory, remedies for breakdowns were given in [28, 21]. 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 [25, 18] and the references therein. Finally, a treatment of breakdowns in the context

THE LOOK-AHEAD LANCZOS PROCESS FOR LARGE NONSYMMETRIC MATRICES AND RELATED ALGORITHMS

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

November 18, 1992

ABSTRACT. The nonsymmetric Lanczos process can be used to compute approximate eigenvalues of large non-Hermitian matrices, or 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. We also discuss related algorithms for the construction of formally orthogonal and biorthogonal polynomials, and for computing inverse triangular factorizations of Hankel and Toeplitz matrices.

KEYWORDS. Lanczos process, non-Hermitian matrices, look-ahead techniques, approximate eigenvalues, systems of linear equations, orthogonal polynomials, inverse triangular factorizations, Hankel matrices, biorthogonal polynomials, Toeplitz matrices, Levinson algorithm.

1. Introduction

In 1950, Cornelius Lanczos [30] 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