Formally Biorthogonal Polynomials and a Look-Ahead Levinson Algorithm for General Toeplitz Systems

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

Hongyuan Zha†
Computer Science Department
The Pennsylvania State University
University Park, Pennsylvania 16802

Abstract

Systems of linear equations with Toeplitz coefficient matrices arise in many important applications. The classical Levinson algorithm computes solutions of Toeplitz systems with only $\mathcal{O}(n^2)$ arithmetic operations, as compared to $\mathcal{O}(n^3)$ operations that are needed for solving general linear systems. However, the Levinson algorithm in its original form requires that all leading principal submatrices are nonsingular. In this paper, an extension of the Levinson algorithm to general Toeplitz systems is presented. The algorithm uses look-ahead to skip over exactly singular, as well as ill-conditioned leading submatrices, and, at the same time, it still fully exploits the Toeplitz structure. In our derivation of this algorithm, we make use of the intimate connection of Toeplitz matrices with formally biorthogonal polynomials. In particular, the occurrence of singular or ill-conditioned submatrices corresponds to

*The research of this author was performed at the Research Institute for Advanced Computer Science (RIACS), NASA Ames Research Center, Moffett Field, California 94035, and it was supported by Cooperative Agreement NCC 2-387 between the National Aeronautics and Space Administration and the Universities Space Research Association.

†The research of this author was supported in part by Army contract number DAAL-03-90-G-0105 and in part by Cooperative Agreement NCC 2-387 between the National Aeronautics and Space Administration and the Universities Space Research Association.

a breakdown or near-breakdown in the standard recurrence relations for biorthogonal polynomials. We present new general recurrence relations that connect successive pairs in any given subsequence of all existing formally biorthogonal polynomials. These recurrences then immediately lead to the proposed look-ahead Levinson algorithm for solving Toeplitz systems. Implementation details for this algorithm and operations counts are given. Numerical experiments for Toeplitz systems with ill-conditioned submatrices are reported.

1 Introduction

Matrices whose entries are equal along each diagonal are called *Toeplitz matrices*. In particular, a general square Toeplitz matrix of order $n+1$ is of the form

$$T_n = [\,t_{i-j}\,]_{i,j=0,\ldots,n} = \begin{bmatrix} t_0 & t_{-1} & t_{-2} & \cdots & t_{-n} \\ t_1 & t_0 & \ddots & & \vdots \\ t_2 & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & & t_{-1} \\ t_n & \cdots & \cdots & t_1 & t_0 \end{bmatrix}, \tag{1.1}$$

where the entries t_i are real or complex numbers. In this paper, we are concerned with the solution of systems of linear equations

$$T_n x_n = b_n \tag{1.2}$$

with Toeplitz coefficient matrices (1.1). The task of solving Toeplitz systems (1.2) arises in many important applications, such as time-series analysis [13, 29], linear prediction [20, 36], spectral estimation [22, 24], system identification [26, 27, 30], Padé approximation [6, 18], and statistics [19].

There are classical *fast* algorithms for solving (1.2) that exploit the Toeplitz structure and require only $\mathcal{O}(n^2)$ operations, as compared to $\mathcal{O}(n^3)$ operations for general linear systems. These algorithms implicitly compute either an *inverse* triangular factorization of T_n of the type

$$V_n^T T_n U_n = D_n, \tag{1.3}$$

or a triangular factorization of T_n of the form

$$T_n = V_n^T D_n U_n, \tag{1.4}$$

see, e.g., [12]. Here, in (1.3) and (1.4), U_n and V_n are unit upper triangular matrices, and D_n is a diagonal matrix. The class of fast Toeplitz solvers based on (1.3) includes the Levinson algorithm [35] and its variants [13, 43, 48, 49]. Toeplitz solvers based on (1.4) are intimately connected with the classical work of Schur [39, 31], and they are called *Schur-type* methods. Algorithms in this second class were proposed by Bareiss [2], Rissanen [38], and others; we refer the reader to [31] and the references given there.

Note that, for a nonsingular matrix T_n, triangular decompositions of the type (1.3) and (1.4) exist if, and only if, all leading principal submatrices of T_n are nonsingular. Indeed, all classical fast Toeplitz solvers require that T_n is *strongly regular*, i.e., the submatrices T_m, $m = 0, 1, \ldots, n-1$, are all nonsingular. In some, but by far not all applications that lead to Toeplitz systems the coefficient matrices T_n are Hermitian positive definite and thus guaranteed to be strongly regular. However, Hermitian indefinite and non-Hermitian Toeplitz matrices are in general not strongly regular, and it cannot be excluded that singular or ill-conditioned submatrices occur. We remark that Hermitian indefinite Toeplitz systems arise, for instance, in spectral estimation [37], when inverse iteration is used to compute eigenvalues of Hermitian positive definite Toeplitz matrices, see [10, 22, 24].

It is well known that the Levinson algorithm and the Schur-type Toeplitz solvers can be extended to handle exactly singular leading principal submatrices, and numerous algorithms were proposed [11, 17, 20, 23, 40, 47]. These algorithms are again based on triangular factorizations of the type (1.3) or (1.4), where now D_n is a block diagonal matrix. More precisely, blocks of size $h_k > 1$ in D_n just correspond to $h_k - 1$ consecutive singular leading submatrices of T_n.

In finite-precision arithmetic, it is not enough to skip only over exactly singular submatrices, and a numerically robust Toeplitz solver also must be able to handle nonsingular, yet ill-conditioned leading principal submatrices. The literature on Toeplitz algorithms with this property is rather scarce. Sweet [41, 42] showed that, in principle, pivoting can be incorporated into the Bareiss algorithm, which allows to treat singular and nearly singular submatrices. However, there are some unresolved difficulties with this algorithm, such as the necessity for an a-priori choice of parameters, and the fact that a large pivot does not necessarily guarantee well-conditioned submatrices. Recently, Chan and Hansen [8] proposed a look-ahead modification of the Levinson algorithm for general Toeplitz systems. If a singular or a nonsingular ill-conditioned submatrix occurs, then the algorithm looks ahead to the next well-conditioned leading submatrix, and instead of a standard Levinson step, a block step is performed. However, this look-ahead algorithm is not entirely satisfactory. The look-ahead strategy used in [8] requires condition number estimates for all leading submatrices, and this generates overhead of the order $\mathcal{O}(n^2)$, even if it turns out that no block steps are necessary. Moreover, as we will demonstrate with an example in Section 7.2 below, there is a potential source for a breakdown of the algorithm if two or more consecutive block steps are performed.

In this paper, we propose a look-ahead Levinson algorithm for general Toeplitz systems that is different from the one in [8]. In our derivation of this algorithm, we make use of the intimate connection of Toeplitz matrices with *formally biorthogonal polynomials* (FBOPs). In particular, the occurrence of singular or ill-conditioned submatrices corresponds to a breakdown or near-breakdown in the standard recurrence relations for biorthogonal polynomials. First, we derive new general recurrence relations that connect successive pairs in any given subsequence of all existing FBOPs. These recurrences then immediately lead to the proposed look-ahead Levinson algorithm for solving Toeplitz systems.

The remainder of this paper is organized as follows. In Section 2, we introduce some notation, and we give a formal definition of FBOPs associated with general bilinear forms. We then turn to bilinear forms induced by Toeplitz matrices, and in Section 3, we collect some basic properties of the corresponding FBOPs. In Section 4, we derive general recurrence relations for FBOPs. In Section 5, we propose a look-ahead procedure for constructing FBOPs, and we describe some properties of this algorithm. In Section 6, we present our look-ahead Levinson algorithm for solving general Toeplitz systems. We give implementation details and operation counts, and we discuss the look-ahead strategy. In Section 7, we consider the look-ahead Levinson algorithm for the special case of Hermitian Toeplitz systems. Also, we show that the procedure proposed by Chan and Hansen has potential breakdowns. In Section 8, we report results of numerical experiments with Toeplitz matrices that have various kinds of ill-conditioned submatrices. Finally, in Section 9, we make some concluding remarks.

2 Preliminaries

In this section, we introduce some notation, and we give a formal definition of FBOPs associated with general bilinear forms.

2.1 Notation

Throughout the paper, all vectors and matrices are allowed to have real or complex entries. As usual, $M^T := [\, m_{kj}\,]$, $\overline{M} := [\, \overline{m}_{jk}\,]$, and $M^H := \overline{M}^T$ denote the transpose, complex conjugate, and conjugate transpose, respectively, of a matrix $M = [\, m_{jk}\,]$. The vector norm $\|x\| := \sqrt{x^H x}$ is the Euclidean norm, and $\|M\| := \max_{\|x\|=1} \|Mx\|$ is the corresponding matrix norm. For square matrices $M \in \mathbb{C}^{h\times h}$, we use the following condition number:

$$\kappa(M) := \begin{cases} 1/|M|, & \text{if } h = 1, \\ \|M\| \cdot \|M^{-1}\|, & \text{if } h > 1. \end{cases} \tag{2.1}$$

Whenever we call a square matrix *ill-conditioned*, it is with respect to the condition number (2.1). We denote by $I_k \in \mathbb{R}^{k\times k}$ the $k \times k$ identity matrix, by

$$J_k := \begin{bmatrix} 0 & \cdots & 0 & 1 \\ \vdots & \iddots & 1 & 0 \\ 0 & \iddots & \iddots & \vdots \\ 1 & 0 & \cdots & 0 \end{bmatrix} \in \mathbb{R}^{k\times k}$$

the $k \times k$ antidiagonal identity matrix, by $0_{k\times j}$ the $k \times j$ zero matrix, and by $0_k \in \mathbb{R}^k$ the zero vector of length k. We will drop subscripts and simply write I, J, or 0 if the actual dimensions are apparent from the context.

The set of all complex polynomials of degree at most n is denoted by

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

and $\mathcal{P}$ is the set of all complex polynomials. We denote by $\partial(\varphi)$ the exact degree of $\varphi \in \mathcal{P}$, i.e., $\partial(\varphi)$ is the smallest integer $n \geq 0$ such that $\varphi \in \mathcal{P}_n$. A polynomial $\varphi \in \mathcal{P}_n$ is called *monic* if it is of exact degree n with leading coefficient 1. For each $\varphi \in \mathcal{P}$, we define its *reverse* $\widehat{\varphi}$ by

$$\widehat{\varphi}(\lambda) \equiv \lambda^{\partial(\varphi)}\varphi(1/\lambda). \tag{2.2}$$

Note that $\widehat{\varphi}$ is a polynomial of degree at most $\partial(\varphi)$.

Capital Greek letters are always used to denote row vectors of polynomials in $\mathcal{P}$, e.g.,

$$\Phi = [\,\varphi_0 \quad \varphi_1 \quad \cdots \quad \varphi_j\,]\,. \tag{2.3}$$

A vector of the type (2.3) is called a *block* of polynomials. The *reverse* $\widehat{\Phi}$ of a block (2.3) is defined by

$$\widehat{\Phi}(\lambda) \equiv \lambda^{n_\Phi}\Phi(1/\lambda), \quad \text{where} \quad n_\Phi := \max_{i=0,1,\ldots,j} \partial(\varphi_i). \tag{2.4}$$

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

$$\widehat{\Phi} = [\,\lambda^{(n_\Phi - \partial(\varphi_0))}\widehat{\varphi}_0 \quad \lambda^{(n_\Phi - \partial(\varphi_1))}\widehat{\varphi}_1 \quad \cdots \quad \lambda^{(n_\Phi - \partial(\varphi_j))}\widehat{\varphi}_j\,]\,. \tag{2.5}$$

Furthermore, if $j = 0$ in (2.3), then (2.4) reduces to the usual reverse (2.2) of a single polynomial.

Finally, for each n, we denote by

$$\Lambda_n = [\,1 \quad \lambda \quad \lambda^2 \quad \cdots \quad \lambda^n\,] \tag{2.6}$$

the block whose entries are the monomials λ^i, $i = 0, 1, \ldots, n$. We remark that, with (2.6) and n_Φ from (2.4), any block (2.3) and its reverse can be represented in the form

$$\Phi = \Lambda_{n_\Phi} U \quad \text{and} \quad \widehat{\Phi} = \Lambda_{n_\Phi} JU, \tag{2.7}$$

respectively. Here $U \in \mathbb{C}^{(n_\Phi+1)\times(j+1)}$ is a matrix whose ith column just contains the coefficients of the polynomial φ_i, $i = 0, 1, \ldots, j$. In particular, each $\varphi \in \mathcal{P}_n$ can be written in the form

$$\varphi = \Lambda_n u \quad \text{for some} \quad u \in \mathbb{C}^{n+1}. \tag{2.8}$$

2.2 Bilinear Forms and FBOPs

A complex-valued functional

$$\langle\cdot,\cdot\rangle : \mathcal{P} \times \mathcal{P} \longmapsto \mathbb{C} \tag{2.9}$$

is called a *bilinear form* if

$$\begin{aligned} \langle\psi, \sigma_1\varphi_1 + \sigma_2\varphi_2\rangle &= \sigma_1\langle\psi,\varphi_1\rangle + \sigma_2\langle\psi,\varphi_2\rangle \quad \text{for all} \quad \psi,\, \varphi_1,\, \varphi_2 \in \mathcal{P},\ \sigma_1,\, \sigma_2 \in \mathbb{C}, \\ \langle\tau_1\psi_1 + \tau_2\psi_2, \varphi\rangle &= \tau_1\langle\psi_1,\varphi\rangle + \tau_2\langle\psi_2,\varphi\rangle \quad \text{for all} \quad \psi_1,\, \psi_2,\, \varphi \in \mathcal{P},\ \tau_1,\, \tau_2 \in \mathbb{C}. \end{aligned} \tag{2.10}$$

We stress that, in general, a bilinear form is not an inner product. Indeed, it is possible that a nonzero polynomials 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 (2.9). Next we give a definition of these *formally biorthogonal polynomials* (FBOPs).

Definition 2.1 *A monic polynomial $\varphi_n \in \mathcal{P}_n$ is called a right* FBOP (*with respect to the bilinear form* (2.9)) *of degree n if*

$$\langle \psi, \varphi_n \rangle = 0 \quad \textit{for all} \quad \psi \in \mathcal{P}_{n-1}. \tag{2.11}$$

A monic polynomial $\psi_n \in \mathcal{P}_n$ is called a left FBOP (*with respect to the bilinear form* (2.9)) *of degree n if*

$$\langle \psi_n, \varphi \rangle = 0 \quad \textit{for all} \quad \varphi \in \mathcal{P}_{n-1}. \tag{2.12}$$

A right or left FBOP *φ_n or ψ_n is said to be regular if it is uniquely determined by* (2.11) *or* (2.12)*, respectively.*

REMARK. In general, regular FBOPs need not exist for every degree n; for instance, see Lemma 3.1 below.

REMARK. Biorthogonal polynomials have been studied in various settings; we refer the reader to [3, 6, 32, 33, 46] and the papers cited therein. The notation "formally biorthogonal polynomials" goes back at least to van Rossum [46]. However, almost all the literature is concerned with cases where regular FBOPs of every degree n are guaranteed to exist or are assumed to exist.

In the sequel, it will be convenient to use the following extension of the bilinear form (2.9) to blocks of polynomials of the type (2.3). More precisely, for any blocks

$$\Phi = [\,\varphi_0 \quad \varphi_1 \quad \cdots \quad \varphi_j\,] \quad \text{and} \quad \Psi = [\,\psi_0 \quad \psi_1 \quad \cdots \quad \psi_k\,],$$

we define

$$\langle \Psi, \Phi \rangle := \begin{bmatrix} \langle \psi_0, \varphi_0 \rangle & \cdots & \langle \psi_0, \varphi_j \rangle \\ \vdots & & \vdots \\ \langle \psi_k, \varphi_0 \rangle & \cdots & \langle \psi_k, \varphi_j \rangle \end{bmatrix} \in \mathbb{C}^{(k+1)\times(j+1)}.$$

3 FBOPs Associated with Toeplitz Matrices

Let $\{t_i\}_{i=-\infty}^{\infty}$ be a given biinfinite sequence[1] of real or complex numbers, and let

$$T_n = [\,t_{i-j}\,]_{i,j=0,\ldots,n}, \quad n = 0, 1, \ldots,$$

be the associated family of Toeplitz matrices (1.1). The sequence $\{T_n\}_{n=0}^{\infty}$ induces a bilinear form $\langle \psi, \varphi \rangle$ on $\mathcal{P} \times \mathcal{P}$ as follows. Using the representation (2.8), for any two polynomials

$$\varphi = \Lambda_n u, \quad \psi = \Lambda_n v, \quad u,\, v \in \mathbb{C}^{n+1},$$

[1]In the case that only a finite sequence $t_{-n}, t_{-n+1}, \ldots, t_n$ is given, we can always extend it to a biinfinite one, by simply setting $t_i := t_{-i} := 0$ for all $i > n$.

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

$$\langle \psi, \varphi \rangle := v^T T_n u. \tag{3.1}$$

From now on, we assume that $\langle \cdot, \cdot \rangle$ is the bilinear form defined by (3.1). Furthermore, the term FBOP always refers to formally biorthogonal polynomials with respect to this particular bilinear form.

Next, we list some properties of $\langle \cdot, \cdot \rangle$. For the monomials $\varphi(\lambda) \equiv \lambda^j$ and $\psi(\lambda) \equiv \lambda^i$, we obtain

$$\langle \lambda^i, \lambda^j \rangle = t_{i-j}, \quad i,\, j = 0, 1, \ldots, \tag{3.2}$$

and hence the elements of $\{t_i\}_{i=-\infty}^{\infty}$ are just the *moments* associated with $\langle \cdot, \cdot \rangle$. From (3.2) and the bilinearity relations (2.10), it readily follows that

$$\langle \lambda\psi, \lambda\varphi \rangle = \langle \psi, \varphi \rangle \quad \text{for all} \quad \varphi,\, \psi \in \mathcal{P}. \tag{3.3}$$

Furthermore, for all $\varphi \in \mathcal{P}$ and all $j,\, k = 0, 1, \ldots,$ with $\partial(\varphi) + k - j \geq 0$, we have

$$\langle \lambda^k \widehat{\varphi}, \lambda^j \rangle = \langle \lambda^{\partial(\varphi)+k-j}, \varphi \rangle, \tag{3.4}$$

$$\langle \lambda^j, \lambda^k \widehat{\varphi} \rangle = \langle \varphi, \lambda^{\partial(\varphi)+k-j} \rangle. \tag{3.5}$$

For example, to verify (3.4), we set $n := \partial(\varphi)$, and we represent φ, $\widehat{\varphi}$ in the form

$$\varphi(\lambda) \equiv \sum_{i=0}^{n} \sigma_i \lambda^i, \quad \widehat{\varphi}(\lambda) \equiv \sum_{i=0}^{n} \sigma_i \lambda^{n-i}, \quad \text{with} \quad \sigma_0, \sigma_1, \ldots, \sigma_n \in \mathbb{C}.$$

Then, with (2.10) and (3.2), we obtain

$$\langle \lambda^k \widehat{\varphi}, \lambda^j \rangle = \sum_{i=0}^{n} \sigma_i t_{(k+n-i)-j} = \sum_{i=0}^{n} \sigma_i t_{(n+k-j)-i} = \langle \lambda^{n+k-j}, \varphi \rangle.$$

Similarly, one can show (3.5).

In view of (2.11), (3.1), and (1.1), a 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

$$T_{n-1} \begin{bmatrix} u_{0n} \\ u_{1n} \\ \vdots \\ u_{n-1,n} \end{bmatrix} = - \begin{bmatrix} t_{-n} \\ t_{-(n-1)} \\ \vdots \\ t_{-1} \end{bmatrix}. \tag{3.6}$$

Analogously, by (2.12), (3.1), and (1.1), a polynomial

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

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

$$[\,v_{0n} \quad v_{1n} \quad \dots \quad v_{n-1,n}\,]\,T_{n-1} = -[\,t_n \quad t_{n-1} \quad \dots \quad t_1\,]. \tag{3.7}$$

As an immediate consequence of (3.6) and (3.7), we have the following result.

Lemma 3.1 *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 T_{n-1} *is nonsingular.*

4 Recurrence Relations for FBOPs

In general, regular FBOPs φ_n and ψ_n need not exist for every n. We denote by

$$\{n_j\}_{j=0}^{J}\,, \quad \text{where} \quad 0 =: n_0 < n_1 < \cdots < n_j < \cdots\,, \tag{4.1}$$

the sequence of all integers n for which regular FBOPs of degree n exist. Here, either $J = \infty$ or, if there are only finitely many regular FBOPs, J is an integer. We remark that, for $n = 0$, the conditions (3.6) and (3.7) are void, and thus $\varphi_0(\lambda) \equiv 1$, $\psi_0(\lambda) \equiv 1$ are regular FBOPs of degree 0. Hence $n_0 = 0$ is always included in (4.1). Note that, in view of Lemma 3.1, the sequence $\{n_j\}_{j=1}^{J}$ just consists of all integers $n \geq 1$ for which T_{n-1} is nonsingular.

In this section, we derive recurrence relations for generating regular FBOPs corresponding to arbitrary subsequences of (4.1).

4.1 The Classical Szegö Recursions

If $n_j \equiv j$ and thus regular FBOPs of degree n exist for every n, then they can be generated by means of the celebrated Szegö recursions [19, 3, 32].

Algorithm 4.1 (Szegö recursions)

0) *Set* $\varphi_0 = \psi_0 = 1$, $\delta_0 = \langle 1, 1\rangle$.

For $n = 0, 1, \dots,$ *do:*

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

2) *Set*

$$\begin{aligned} \alpha_n &= \rho_n/\delta_n, \quad \varphi_{n+1} = \lambda\varphi_n - \widehat{\psi}_n\alpha_n, \\ \beta_n &= \tau_n/\delta_n, \quad \psi_{n+1} = \lambda\psi_n - \widehat{\varphi}_n\beta_n. \end{aligned} \tag{4.2}$$

3) *Set*

$$\delta_{n+1} = \delta_n(1 - \alpha_n\beta_n). \tag{4.3}$$

We remark that the δ_n's in (4.3) satisfy

$$\delta_n = \langle \psi_n, \varphi_n \rangle. \tag{4.4}$$

The Szegö recursions are no longer valid if regular FBOPs do not exist for every n. There are extensions of the relations (4.2) that connect consecutive pairs φ_{n_j}, ψ_{n_j} and $\varphi_{n_{j+1}}$, $\psi_{n_{j+1}}$ in the sequence (4.1) of *all* existing regular FBOPs. For example, such recurrences are given in [23] and, for the special case of Hermitian Toeplitz matrices $\{T_n\}_{n=0}^{\infty}$, in [11]. These recursions immediately lead to an extension of the Levinson algorithm that can skip over exactly singular leading submatrices. For the derivation of a more robust Levinson algorithm that can also skip over nonsingular, but ill-conditioned submatrices, we need more general recurrence relations that connect consecutive pairs in a subsequence of all existing regular FBOPs. In the next section, we present such general recursions for FBOPs.

4.2 General Recursions for FBOPs

Let $\{n_{j_k}\}_{k=0}^{K} \subseteq \{n_j\}_{j=0}^{J}$ be an arbitrary, but fixed, subsequence of (4.1). Here either $K = \infty$ or K is an integer. For simplicity, we set $n_k := n_{j_k}$. Moreover, in view of (4.1), we can assume that, without loss of generality, $n_0 = 0$ is included in the subsequence. Therefore, we always have

$$0 =: n_0 < n_1 < \cdots < n_k < \cdots . \tag{4.5}$$

For all $0 \le k < K$, we set

$$h_k := n_{k+1} - n_k \quad \text{and} \quad \mathcal{I}_k := \begin{cases} \emptyset, & \text{if } h_k = 1, \\ \{n \mid n_k < n < n_{k+1}\}, & \text{if } h_k > 1. \end{cases} \tag{4.6}$$

If $K < \infty$, then we set $n_{K+1} := \infty$ and $\mathcal{I}_K := \{n \mid n > n_K\}$.

The goal then is to give recurrences for generating the regular FBOPs

$$\{\varphi_{n_k}\}_{k=0}^{K} \quad \text{and} \quad \{\psi_{n_k}\}_{k=0}^{K} \tag{4.7}$$

corresponding to the prescribed indices (4.5). To this end, we also construct additional monic polynomials

$$\varphi_n, \psi_n \in \mathcal{P}_n \quad \text{for all} \quad n \in \mathcal{I}_k \quad \text{and all} \quad k. \tag{4.8}$$

The polynomials (4.8) are called *inner* polynomials. Note that the regular FBOPs (4.7) together with the inner polynomials (4.8) build two sequences of monic polynomials $\{\varphi_n\}_{n=0}^{\infty}$ and $\{\psi_n\}_{n=0}^{\infty}$ that both span $\mathcal{P}$. Of course, we still need to specify how to actually choose the inner polynomials. In order to obtain recurrence relations that involve as few as possible previous polynomials, it is crucial to construct the inner polynomials φ_n, ψ_n, $n \in \mathcal{I}_k$, as *quasi*-FBOPs, in the sense that they satisfy the relaxed biorthogonality relations

$$\begin{aligned} \langle \psi, \varphi_n \rangle &= 0 \quad \text{for all} \quad \psi \in \mathcal{P}_{n_k - 1}, \\ \langle \psi_n, \varphi \rangle &= 0 \quad \text{for all} \quad \varphi \in \mathcal{P}_{n_k - 1}. \end{aligned} \tag{4.9}$$

Next we introduce some further notation. For all $0 \le k < K$, we define blocks of polynomials

$$\Phi^{(k)} := [\,\varphi_{n_k} \quad \varphi_{n_k+1} \quad \cdots \quad \varphi_{n_{k+1}-1}\,], \quad \Psi^{(k)} := [\,\psi_{n_k} \quad \psi_{n_k+1} \quad \cdots \quad \psi_{n_{k+1}-1}\,]. \tag{4.10}$$

If $K < \infty$, then we also define infinite row vectors of polynomials

$$\Phi^{(K)} := [\,\varphi_{n_K} \quad \varphi_{n_K+1} \quad \cdots\,], \quad \Psi^{(K)} := [\,\psi_{n_K} \quad \psi_{n_K+1} \quad \cdots\,].$$

Moreover, for all $0 \le k < K$, we set

$$\Lambda^{(k)} := [\,\lambda^{n_k} \quad \lambda^{n_k+1} \quad \cdots \quad \lambda^{n_{k+1}-1}\,], \tag{4.11}$$

$$F^{(k)} := \langle \Lambda^{(k)}, \Phi^{(k)} \rangle, \quad G^{(k)} := \langle \Psi^{(k)}, \Lambda^{(k)} \rangle. \tag{4.12}$$

We remark that, with (4.11), the biorthogonality conditions (2.11) and (2.12) for the regular FBOPs (4.7) and the relaxed biorthogonality relations (4.9) for quasi-FBOPs can be summarized as follows: the polynomials φ_n, ψ_n, $n = 0, 1, \ldots$, are required to satisfy

$$\langle \Lambda^{(m)}, \varphi_n \rangle = 0 \quad \text{and} \quad \langle \psi_n, \Lambda^{(m)} \rangle = 0 \quad \text{for all} \quad m = 0, 1, \ldots, k(n) - 1, \tag{4.13}$$

where $k(n)$ is the integer such that $n_{k(n)} \le n < n_{k(n)+1}$.

Note that $F^{(k)}$ and $G^{(k)}$ defined in (4.12) are $h_k \times h_k$ matrices, with h_k from (4.6). We will need the fact that these matrices are nonsingular.

Lemma 4.2 *If the inner polynomials* (4.8) *are constructed as quasi-*FBOP*s, then the matrices $F^{(k)}$ and $G^{(k)}$ are nonsingular for all $0 \le k < K$.*

Proof. Suppose that $F^{(k)}$ is singular. Then there exists a vector $z \in \mathbb{C}^{h_k}$ such that

$$F^{(k)} z = 0 \quad \text{and} \quad z \neq 0, \tag{4.14}$$

and we set

$$\varphi := \varphi_{n_{k+1}} + \Phi^{(k)} z. \tag{4.15}$$

Clearly, φ is a monic polynomial of degree n_{k+1}, and since the polynomials in the block $\Phi^{(k)}$ are linearly independent, we have $\varphi \neq \varphi_{n_{k+1}}$. Using (4.13)–(4.15), we deduce that

$$\langle \Lambda^{(m)}, \varphi \rangle = \langle \Lambda^{(m)}, \varphi_{n_{k+1}} \rangle + \langle \Lambda^{(m)}, \Phi^{(k)} \rangle z = \begin{cases} 0, & \text{if } 0 \le m < k, \\ F^{(k)} z = 0, & \text{if } m = k. \end{cases}$$

Hence φ is a regular right FBOP of degree n_{k+1}, and this contradicts the uniqueness of regular FBOPs.

Similarly, one shows that $G^{(k)}$ is nonsingular. □

For the formulation of our recurrence relations, we will also need the following quantities. If $h_k > 1$, we define

$$f_k := [\,I_{h_k-1} \quad 0_{h_k-1}\,]\, F^{(k)} \begin{bmatrix} 0_{h_k-1} \\ 1 \end{bmatrix}, \quad g_k := [\,I_{h_k-1} \quad 0_{h_k-1}\,]\, (G^{(k)})^T \begin{bmatrix} 0_{h_k-1} \\ 1 \end{bmatrix}, \tag{4.16}$$

and, if $h_k = 1$, we set $f_k = g_k = \emptyset$. We remark that f_k and g_k just consist of the first $h_k - 1$ elements of the last columns of $F^{(k)}$ and $(G^{(k)})^T$, respectively. Thus we have

$$F^{(k)} = \begin{bmatrix} * & f_k \\ * & * \end{bmatrix} \quad \text{and} \quad (G^{(k)})^T = \begin{bmatrix} * & g_k \\ * & * \end{bmatrix}, \tag{4.17}$$

where the elements $*$ in the lower right corners in (4.17) are 1×1.

After these preparations, we can now state our recurrence relations for generating the regular FBOPs (4.7) corresponding to the prescribed indices (4.5), together with inner polynomials (4.8) satisfying the relaxed biorthogonality conditions (4.9). First, we set $\varphi_{-1} = \psi_{-1} = \lambda^{-1}$ and $\Phi^{(-1)} = \Psi^{(-1)} = \emptyset$. Then, for all $-1 \le k < K$ and $n_{k+1} \le n+1 < n_{k+2}$, we set:

$$\varphi_{n+1} = \lambda\varphi_n - \widehat{\Psi}^{(k)}\alpha_n - \begin{cases} \Phi^{(k)}\mu_n, & \text{if } n+1 = n_{k+1}, \\ \displaystyle\sum_{i=n_{k+1}}^{n} \xi_i^{(n)}\varphi_i & \text{if } n+1 \in \mathcal{I}_{k+1}, \end{cases} \tag{4.18}$$

$$\text{where} \quad \alpha_n = (G^{(k)})^{-T} \begin{bmatrix} 0_{h_k-1} \\ \rho_n \end{bmatrix}, \quad \rho_n = \langle 1, \lambda\varphi_n \rangle, \quad \mu_n = (F^{(k)})^{-1} \begin{bmatrix} 0 \\ f_k \end{bmatrix}, \tag{4.19}$$

and

$$\psi_{n+1} = \lambda\psi_n - \widehat{\Phi}^{(k)}\beta_n - \begin{cases} \Psi^{(k)}\nu_n, & \text{if } n+1 = n_{k+1}, \\ \displaystyle\sum_{i=n_{k+1}}^{n} \zeta_i^{(n)}\psi_i, & \text{if } n+1 \in \mathcal{I}_{k+1}, \end{cases} \tag{4.20}$$

$$\text{where} \quad \beta_n = (F^{(k)})^{-1} \begin{bmatrix} 0_{h_k-1} \\ \tau_n \end{bmatrix}, \quad \tau_n = \langle \lambda\psi_n, 1 \rangle, \quad \nu_n = (G^{(k)})^{-T} \begin{bmatrix} 0 \\ g_k \end{bmatrix}. \tag{4.21}$$

Here, in (4.18) and (4.20), $\xi_i^{(n)}$, $\zeta_i^{(n)} \in \mathbb{C}$ are coefficients that can be chosen arbitrarily.

Note that, by Lemma 4.2, the inverse matrices in (4.19) and (4.21) all exist. Moreover, we remark that the recurrence (4.20) can be equivalently formulated in terms of the reverse polynomials that appear in (4.18). The resulting relation is as follows:

$$\widehat{\psi}_{n+1} = \widehat{\psi}_n - \lambda^{n+2-n_{k+1}}\Phi^{(k)}\beta_n - \begin{cases} \lambda\widehat{\Psi}^{(k)}\nu_n, & \text{if } n+1 = n_{k+1}, \\ \displaystyle\sum_{i=n_{k+1}}^{n} \zeta_i^{(n)}\lambda^{n+1-i}\widehat{\psi}_i, & \text{if } n+1 \in \mathcal{I}_{k+1}. \end{cases}$$

Of course, we still need to verify that the recursions (4.18)–(4.21) indeed generate the regular FBOPs (4.7).

Theorem 4.3 *Let* $\{\varphi_n\}_{n=0}^{\infty}$ *and* $\{\psi_n\}_{n=0}^{\infty}$ *be the sequences of polynomials defined by the recurrence relations* (4.18)–(4.21)*. Then, these polynomials satisfy the biorthogonality relations* (4.13)*. In particular, the polynomials* $\{\varphi_{n_k}\}_{k=0}^{K}$ *and* $\{\psi_{n_k}\}_{k=0}^{K}$ *are the uniquely defined regular* FBOP*s corresponding to the prescribed indices* (4.5)*.*

Proof. We show (4.13) by induction on n. For $n = 0$, by (4.5), $k(0) = 0$, and the conditions (4.13) are void.

Now let $n \geq 0$, and assume that (4.13) holds for all polynomials $\varphi_0, \varphi_1, \ldots, \varphi_n$ and $\psi_0, \psi_1, \ldots, \psi_n$. We need to show that φ_{n+1} and ψ_{n+1} satisfy

$$\langle \lambda^j, \varphi_{n+1} \rangle = 0 \quad \text{for all} \quad j = 0, 1, \ldots, n_{k+1} - 1, \tag{4.22}$$

and

$$\langle \psi_{n+1}, \lambda^j \rangle = 0 \quad \text{for all} \quad m = 0, 1, \ldots, n_{k+1} - 1, \tag{4.23}$$

respectively. Here k is the integer defined by $n_{k+1} \leq n + 1 < n_{k+2}$. To simplify notation, we set $h' := h_k - 1$ and $n' := n_{k+1} - 1$. Moreover, in the following, we always assume that $j \in \{0, 1, \ldots, n'\}$.

First, we consider (4.22). Writing $\widehat{\Psi}^{(k)}$ in the form (2.5) and using (3.5), one readily verifies that

$$\langle \lambda^j, \widehat{\Psi}^{(k)} \rangle = \langle \Psi^{(k)}, \lambda^{n'-j} \rangle^T. \tag{4.24}$$

From (4.24) and (4.13), it follows that

$$\langle \lambda^j, \widehat{\Psi}^{(k)} \rangle = 0 \quad \text{for all} \quad j \quad \text{with} \quad h' < j \leq n'. \tag{4.25}$$

Using notations introduced in (2.6), (4.11), and (4.12), we can summarize (4.24) for the remaining indices $0 \leq j \leq h'$ as follows:

$$\langle \Lambda_{h'}, \widehat{\Psi}^{(k)} \rangle = \langle \Psi^{(k)}, \Lambda^{(k)} J \rangle^T = J (G^{(k)})^T. \tag{4.26}$$

Next, we note that, in view of (3.3) and (4.13), we have

$$\langle \lambda^j, \lambda\varphi_n \rangle = \begin{cases} \langle 1, \lambda\varphi_n \rangle = \rho_n, & \text{if } j = 0, \\ \langle \lambda^{j-1}, \varphi_n \rangle = 0, & \text{if } 1 \leq j \leq n' \text{ and } n+1 \in \mathcal{I}_{k+1}, \\ \langle \lambda^{j-1}, \varphi_n \rangle = 0, & \text{if } 1 \leq j \leq n_k \text{ and } n+1 = n_{k+1}, \\ \langle \lambda^{j-1}, \varphi_n \rangle, & \text{if } n_k < j \leq n' \text{ and } n+1 = n_{k+1}. \end{cases} \tag{4.27}$$

Using the vector f_k defined by (4.16) and (4.12), we can summarize the relations (4.27) for $n_k \leq j \leq n'$ and $n + 1 = n_{k+1}$ as follows:

$$\langle \Lambda^{(k)}, \lambda\varphi_n \rangle = \begin{bmatrix} 0 \\ f_k \end{bmatrix}, \quad \text{if} \quad n + 1 = n_{k+1}. \tag{4.28}$$

With these preparations, it now readily follows that φ_{n+1} satisfies (4.22). With (4.18) and (4.13), we obtain

$$\langle \lambda^j, \varphi_{n+1} \rangle = \langle \lambda^j, \lambda\varphi_n \rangle - \langle \lambda^j, \widehat{\Psi}^{(k)} \rangle \alpha_n - \begin{cases} \langle \lambda^j, \Phi^{(k)} \rangle \mu_n, & \text{if } n+1 = n_{k+1}, \\ 0, & \text{if } n+1 \in \mathcal{I}_{k+1}. \end{cases} \tag{4.29}$$

By (4.27), (4.25), and (4.13), all terms on the right-hand side of (4.29) vanish and hence $\langle \lambda^j, \varphi_{n+1} \rangle = 0$ for all j in the range $h' < j < n_k$, if $n+1 = n_{k+1}$, and for all j in the range $h' < j \leq n'$, if $n+1 \in \mathcal{I}_{n+1}$. For j in the range $0 \leq j \leq h'$, we use (4.29), (4.13), (4.27), (4.26), and the definition of α_n in (4.19) to deduce that

$$\langle \Lambda_{h'}, \varphi_{n+1} \rangle = \begin{bmatrix} \rho_n \\ 0_{h'} \end{bmatrix} - J(G^{(k)})^T \alpha_n = 0.$$

For the case that $n+1 = n_{k+1}$, it remains to prove (4.22) for all j in the range $n_k \leq j \leq n'$. Here we use (4.29), (4.28), (4.25), (4.12), and the definition of μ_n in (4.19) to verify that

$$\langle \Lambda^{(k)}, \varphi_{n+1} \rangle = \begin{bmatrix} 0 \\ f_k \end{bmatrix} - \langle \Lambda^{(k)}, \Phi^{(k)} \rangle \mu_n = \begin{bmatrix} 0 \\ f_k \end{bmatrix} - F^{(k)} \mu_n = 0.$$

This concludes the proof of (4.22).

To show (4.23) we proceed similarly. First, using (3.4) and (4.13), one verifies the relations

$$\begin{aligned} \langle \widehat{\Phi}^{(k)}, \Lambda_{h'} \rangle &= (F^{(k)})^T J, \\ \langle \widehat{\Phi}^{(k)}, \lambda^j, \rangle &= 0 \quad \text{for all} \quad j \quad \text{with} \quad h' < j \leq n'. \end{aligned} \tag{4.30}$$

With (3.3) and (4.13), we obtain

$$\langle \lambda \psi_n, \lambda^j \rangle = \begin{cases} \langle \lambda \psi_n, 1 \rangle = \tau_n, & \text{if } j = 0, \\ 0, & \text{if } 1 \leq j \leq n' \text{ and } n+1 \in \mathcal{I}_{k+1}, \\ 0, & \text{if } 1 \leq j \leq n_k \text{ and } n+1 = n_{k+1}, \end{cases} \tag{4.31}$$

$$\langle \lambda \psi_n, \Lambda^{(k)} \rangle = [\, 0 \quad g_k^T \,], \quad \text{if} \quad n+1 = n_{k+1}. \tag{4.32}$$

Here g_k is the vector defined by (4.16) and (4.12). From (4.20) and (4.13), we have

$$\langle \psi_{n+1}, \lambda^j \rangle = \langle \lambda \psi_n, \lambda^j \rangle - \beta_n^T \langle \widehat{\Phi}^{(k)}, \lambda^j \rangle - \begin{cases} \nu_n^T \langle \Psi^{(k)}, \lambda^j \rangle, & \text{if } n+1 = n_{k+1}, \\ 0, & \text{if } n+1 \in \mathcal{I}_{k+1}. \end{cases} \tag{4.33}$$

Finally, using (4.30)–(4.33) and the definitions of β_n and ν_n in (4.21), one easily verifies (4.23). $\square$

Remark. Our derivation of general recurrence relations is different from the one used in [23, 11] to obtain special recursions for all existing FBOPs. The approach in [23, 11] is based on Iohvidov's results [25] on the structure of exactly singular Toeplitz matrices, and it cannot be directly used to prove general recurrences of the type (4.18)–(4.21).

5 An Algorithm for Constructing FBOPs

In this section, we propose an algorithm for constructing regular FBOPs based on the general recurrence relations derived in Section 4.2, and we describe some properties of this algorithm. We use the notation introduced in Section 4.2.

5.1 The Algorithm

The algorithm generates two sequences $\{\varphi_n\}_{n=0}^{\infty}$ and $\{\psi_n\}_{n=0}^{\infty}$ of monic polynomials, where, for each n, φ_n and ψ_n are either regular FBOPs or inner quasi-FBOPs. As in (4.5), we use the indices n_k, $k = 0, 1, \ldots$, to mark the regular FBOPs, and we always set $n_0 = 0$, $\varphi_0 = \psi_0 = 1$. The index $n = 0, 1, \ldots$, is used as an iteration counter, where in the course of the nth iteration the algorithm generates the next pair of polynomials φ_{n+1} and ψ_{n+1}. For each fixed n, we define $l = l(n)$ by

$$n_l \le n < n_{l+1}. \tag{5.1}$$

Note that $l = l(n)$ is just the number of the last pair of regular FBOPs φ_{n_l} and ψ_{n_l} with degree $\le n$. In addition to the blocks $\Phi^{(k)}$, $\Psi^{(k)}$, $k = 0, 1, \ldots, l-1$, defined in (4.10), we set

$$\Phi^{(l)} := [\, \varphi_{n_l} \quad \varphi_{n_l+1} \quad \cdots \quad \varphi_n \,], \quad \Psi^{(l)} := [\, \psi_{n_l} \quad \psi_{n_l+1} \quad \cdots \quad \psi_n \,]. \tag{5.2}$$

We call the blocks $\Phi^{(l)}$ and $\Psi^{(l)}$ *complete* if $n + 1 = n_{l+1}$; in this case, the next polynomials φ_{n+1} and ψ_{n+1} are constructed as regular FBOPs. If $n + 1 < n_{l+1}$, then the blocks (5.2) are still *incomplete*; in this case, the next polynomials φ_{n+1} and ψ_{n+1} are constructed as inner quasi-FBOPs and added to $\Phi^{(l)}$ and $\Psi^{(l)}$, respectively. Finally, as in (4.11), (4.12), we set

$$\Lambda^{(l)} := [\, \lambda^{n_l} \quad \lambda^{n_l+1} \quad \cdots \quad \lambda^n \,], \quad F^{(l)} := \langle \Lambda^{(l)}, \Phi^{(l)} \rangle, \quad G^{(l)} := \langle \Psi^{(l)}, \Lambda^{(l)} \rangle. \tag{5.3}$$

Using these notations, we can rewrite the recurrences (4.18)–(4.21) in the form of the following algorithm.

Algorithm 5.1 (Construction of FBOPs)

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*

$$\rho_n = \langle 1, \lambda\varphi_n \rangle \quad \text{and} \quad \tau_n = \langle \lambda\psi_n, 1 \rangle. \tag{5.4}$$

2) *Decide whether to construct* φ_{n+1} *and* ψ_{n+1} *as regular* FBOP*s or as inner polynomials and go to* 3) *or* 4), *respectively.*

3) (Regular step) *Set*

$$\alpha_n = (G^{(l)})^{-T} \begin{bmatrix} 0 \\ \rho_n \end{bmatrix}, \quad \mu_n = (F^{(l)})^{-1} \begin{bmatrix} 0 \\ f_l \end{bmatrix}, \quad \varphi_{n+1} = \lambda\varphi_n - \widehat{\Psi}^{(l)}\alpha_n - \Phi^{(l)}\mu_n, \tag{5.5}$$

$$\beta_n = (F^{(l)})^{-1} \begin{bmatrix} 0 \\ \tau_n \end{bmatrix}, \quad \nu_n = (G^{(l)})^{-T} \begin{bmatrix} 0 \\ g_l \end{bmatrix}, \quad \psi_{n+1} = \lambda\psi_n - \widehat{\Phi}^{(l)}\beta_n - \Psi^{(l)}\nu_n. \tag{5.6}$$

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

4) (Inner step) *Set*

$$\alpha_n = (G^{(l-1)})^{-T} \begin{bmatrix} 0 \\ \rho_n \end{bmatrix}, \quad \varphi_{n+1} = \lambda \varphi_n - \widehat{\Psi}^{(l-1)} \alpha_n - \sum_{i=n_l}^{n} \xi_i^{(n)} \varphi_i, \tag{5.7}$$

$$\beta_n = (F^{(l-1)})^{-1} \begin{bmatrix} 0 \\ \tau_n \end{bmatrix}, \quad \psi_{n+1} = \lambda \psi_n - \widehat{\Phi}^{(l-1)} \beta_n - \sum_{i=n_l}^{n} \zeta_i^{(n)} \psi_i. \tag{5.8}$$

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

Note that the description of Algorithm 5.1 is still incomplete, since no criteria for the decision in step 2) are given. We defer a discussion of this so-called *look-ahead strategy* to Section 6.3. Here, we only remark that, in view of Lemma 4.2, the polynomials φ_{n+1} and ψ_{n+1} can be constructed as regular FBOPs only if the following necessary condition holds true:

$$F^{(l)} \quad \text{and} \quad G^{(l)} \quad \text{are nonsingular if} \quad n+1 = n_{l+1}. \tag{5.9}$$

In particular, (5.9) guarantees that the inverse matrices in (5.5)–(5.8) all exist.

5.2 Inverse Block Factorization of Toeplitz Matrices

Next, we show that Algorithm 5.1 yields a factorization of T_n of the form (1.3), where D_n is now a block diagonal matrix. In the following, let $n \geq 0$ be arbitrary, but fixed, and let $l = l(n)$ be the corresponding index defined by (5.1).

Recall that, by Theorem 4.3, the polynomials

$$\{\varphi_j\}_{j=0}^{n} \quad \text{and} \quad \{\psi_j\}_{j=0}^{n} \tag{5.10}$$

satisfy the conditions (4.13), which are equivalent to the following block biorthogonality relations:

$$\langle \Psi^{(k)}, \Phi^{(m)} \rangle = 0 \quad \text{for all} \quad k \neq m, \quad k, m = 0, 1, \ldots, l. \tag{5.11}$$

Moreover, we set

$$D^{(k)} := \langle \Psi^{(k)}, \Phi^{(k)} \rangle, \quad k = 0, 1, \ldots, l. \tag{5.12}$$

Here, in (5.11) and (5.12), $\Psi^{(k)}$ and $\Phi^{(k)}$ are the blocks defined in (4.10), if $k < l$, and in (5.2), if $k = l$. In view of (2.8), the monic polynomials (5.10) can be represented in the form

$$\begin{aligned} \varphi_j &= \Lambda_j u_j, \quad \text{where} \quad u_j := [\, u_{0j} \quad u_{1j} \quad \cdots \quad u_{j-1,j} \quad 1 \,]^T \in \mathbb{C}^{j+1}, \\ \psi_j &= \Lambda_j v_j, \quad \text{where} \quad v_j := [\, v_{0j} \quad v_{1j} \quad \cdots \quad v_{j-1,j} \quad 1 \,]^T \in \mathbb{C}^{j+1}. \end{aligned} \tag{5.13}$$

Using (5.13) and the definition of $\langle \cdot, \cdot \rangle$ in (3.1), we can rewrite the relations (5.11) and (5.12) in the following matrix formulation:

$$V_n^T T_n U_n = D_n, \tag{5.14}$$

where

$$U_n := [\, u_{ij} \,]_{i,j=0,\ldots,n} = \begin{bmatrix} 1 & u_{01} & \cdots & u_{0n} \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & u_{n-1,n} \\ 0 & \cdots & 0 & 1 \end{bmatrix} \tag{5.15}$$

and

$$V_n := [\, v_{ij} \,]_{i,j=0,\ldots,n} = \begin{bmatrix} 1 & v_{01} & \cdots & v_{0n} \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & v_{n-1,n} \\ 0 & \cdots & 0 & 1 \end{bmatrix} \tag{5.16}$$

are unit upper triangular matrices, and

$$D_n := \operatorname{diag}(D^{(0)}, D^{(1)}, \ldots, D^{(l)}) \tag{5.17}$$

is block diagonal. In other words, Algorithm 5.1 generates an inverse block factorization (5.14) of T_n, where the columns of U_n and V_n are just the coefficients of the polynomials (5.10). Furthermore, the sizes of the blocks in (5.17) are given by

$$D^{(k)} \in \mathbb{C}^{h_k \times h_k}, \quad \text{where} \quad h_k = \begin{cases} n_{k+1} - n_k, & \text{if } 0 \le k < l, \\ n + 1 - n_l, & \text{if } k = l. \end{cases} \tag{5.18}$$

In particular, $h_k = 1$ for all k if only regular steps 3) are performed in Algorithm 5.1. In the sequel, the construction of a *true* block of size $h_k > 1$ will be referred to as a *look-ahead step*, and h_k will be called the *length* of the look-ahead step.

Finally, we remark that, by (5.13), the blocks $\Phi^{(k)}$ and $\Psi^{(k)}$ have the following representations:

$$\begin{aligned} \Phi^{(k)} &= \Lambda_{n_{k+1}-1} U^{(k)}, \quad \text{where} \quad U^{(k)} := [\, u_{ij} \,]_{i=0,\ldots,n_{k+1}-1;\, j=n_k,\ldots,n_{k+1}-1}\,, \\ \Psi^{(k)} &= \Lambda_{n_{k+1}-1} V^{(k)}, \quad \text{where} \quad V^{(k)} := [\, v_{ij} \,]_{i=0,\ldots,n_{k+1}-1;\, j=n_k,\ldots,n_{k+1}-1}\,, \end{aligned} \tag{5.19}$$

if $0 \le k \le l-1$, and

$$\begin{aligned} \Phi^{(l)} &= \Lambda_n U^{(l)}, \quad \text{where} \quad U^{(l)} := [\, u_{ij} \,]_{i=0,\ldots,n;\, j=n_l,\ldots,n}\,, \\ \Psi^{(l)} &= \Lambda_n V^{(l)}, \quad \text{where} \quad V^{(l)} := [\, v_{ij} \,]_{i=0,\ldots,n;\, j=n_l,\ldots,n}\,, \end{aligned} \tag{5.20}$$

if $k = l$. Note that, by (5.14), (5.17), and (5.20), we have

$$D^{(l)} = (V^{(l)})^T T_n U^{(l)}. \tag{5.21}$$

5.3 Further Properties

In this subsection, we collect some further properties of Algorithm 5.1 that will be needed later on.

Using (5.20), (4.13), (5.3), and the definition of $\langle\cdot,\cdot\rangle$ in (3.1), one readily verifies that

$$T_n U^{(l)} = \begin{bmatrix} 0_{n_l\times h_l} \\ F^{(l)} \end{bmatrix}, \quad (V^{(l)})^T T_n = [\, 0_{h_l\times n_l} \quad G^{(l)}\,]. \tag{5.22}$$

Next, we partition the matrices $U^{(l)}$ and $V^{(l)}$ in the form

$$U^{(l)} = \begin{bmatrix} \hat{U}^{(l)} \\ \tilde{U}^{(l)} \end{bmatrix}, \quad V^{(l)} = \begin{bmatrix} \hat{V}^{(l)} \\ \tilde{V}^{(l)} \end{bmatrix}, \quad \text{where} \quad \tilde{U}^{(l)},\, \tilde{V}^{(l)} \in \mathbb{C}^{h_l\times h_l}. \tag{5.23}$$

Note that $\tilde{U}^{(l)}$ and $\tilde{V}^{(l)}$ are unit upper triangular matrices, and in particular, they are nonsingular. With (5.21) and (5.23), it follows from (5.22) that

$$F^{(l)} = (\tilde{V}^{(l)})^{-T} D^{(l)}, \quad G^{(l)} = D^{(l)} (\tilde{U}^{(l)})^{-1}. \tag{5.24}$$

By means of the relations (5.24), $F^{(l)}$ and $G^{(l)}$ can be obtained without evaluating the bilinear forms $\langle \Lambda^{(l)}, \Phi^{(l)}\rangle$ and $\langle \Psi^{(l)}, \Lambda^{(l)}\rangle$ in step 5) of Algorithm 5.1, provided that $D^{(l)}$ is available. It turns out that the elements of $D^{(l)}$ can easily be updated from step to step, see Lemma 5.2 below.

Recall that the recurrence coefficients $\xi_i^{(n)}$ and $\zeta_i^{(n)}$ in (5.7) and (5.8), respectively, are still arbitrary. From now on, we assume that they are chosen as

$$\xi_i^{(n)} = \zeta_i^{(n)} = 0 \quad \text{for all} \quad i,\, n. \tag{5.25}$$

Furthermore, it will be convenient to set

$$\alpha_k^{(1)} := (G^{(k)})^{-T} \begin{bmatrix} 0_{h_k-1} \\ 1 \end{bmatrix}, \quad \beta_k^{(1)} := (F^{(k)})^{-1} \begin{bmatrix} 0_{h_k-1} \\ 1 \end{bmatrix}. \tag{5.26}$$

Note that, with (5.26), the coefficient vectors α_n, β_n in step 3) and step 4) of Algorithm 5.1 are given by

$$\alpha_n = \rho_n \alpha_l^{(1)},\ \beta_n = \tau_n \beta_l^{(1)} \quad \text{and} \quad \alpha_n = \rho_n \alpha_{l-1}^{(1)},\ \beta_n = \tau_n \beta_{l-1}^{(1)}, \tag{5.27}$$

respectively.

By (5.12) and (5.2), the matrix $D^{(l)}$ is given by

$$D^{(l)} = [\, \langle \psi_i, \varphi_j \rangle \,]_{i,j=n_l,\dots,n}\,. \tag{5.28}$$

In the following lemma, we give formulas for updating the elements of (5.28) in the course of Algorithm 5.1.

Lemma 5.2 *Let* φ_{n+1}, ψ_{n+1} *be the polynomials constructed in step* n *Algorithm* 5.1.
a) *If* φ_{n+1}, ψ_{n+1} *are constructed as regular* FBOPs, *then*

$$\langle \psi_{n+1}, \varphi_{n+1} \rangle = \langle \psi_n, \varphi_n \rangle - \rho_n [\, \tau_n \quad 0 \quad g_l^T \,] J V^{(l)} \alpha_l^{(1)} - [\, \tau_n \quad 0 \quad g_l^T \,] U^{(l)} \mu_n. \tag{5.29}$$

b) *If* φ_{n+1}, ψ_{n+1} *are constructed as inner polynomials, then*

$$\langle \psi_{m+1}, \varphi_{n+1} \rangle = \begin{cases} v_{0,n_l}\rho_n, & \text{if} \quad m+1 = n_l, \\ \langle \psi_m, \varphi_n \rangle + v_{0,m+1}\rho_n, & \text{if} \quad n_l < m+1 \le n+1, \end{cases} \tag{5.30}$$

and

$$\langle \psi_{n+1}, \varphi_{m+1} \rangle = \begin{cases} u_{0,n_l}\tau_n, & \text{if} \quad m+1 = n_l, \\ \langle \psi_n, \varphi_m \rangle + u_{0,m+1}\tau_n, & \text{if} \quad n_l < m+1 \le n+1. \end{cases} \tag{5.31}$$

Proof. First we show part a). Here the polynomials ψ_{n+1} and φ_{n+1} are given by (5.6) and (5.5), respectively. Using (5.6), the biorthogonality conditions (4.13), (5.5), (3.3), and the first formula for α_n in (5.27), one readily verifies that

$$\begin{aligned} \langle \psi_{n+1}, \varphi_{n+1} \rangle &= \langle \lambda\psi_n - \widehat{\Phi}^{(l)}\beta_n - \Psi^{(l)}\nu_n, \varphi_{n+1} \rangle \\ &= \langle \lambda\psi_n, \varphi_{n+1} \rangle = \langle \lambda\psi_n, \lambda\varphi_n - \widehat{\Psi}^{(l)}\alpha_n - \Phi^{(l)}\mu_n \rangle \\ &= \langle \psi_n, \varphi_n \rangle - \rho_n \langle \lambda\psi_n, \widehat{\Psi}^{(l)} \rangle \alpha_l^{(1)} - \langle \lambda\psi_n, \Phi^{(l)} \rangle \mu_n. \end{aligned} \tag{5.32}$$

In view of (4.31) and (4.32), we have

$$\langle \lambda\psi_n, \Lambda_n \rangle = [\, \tau_n \quad 0 \quad g_l^T \,]. \tag{5.33}$$

Note that, by (5.20) and (2.7), $\Phi^{(l)} = \Lambda_n U^{(l)}$ and $\widehat{\Psi}^{(l)} = \Lambda_n J V^{(l)}$. Together with (5.33), it follows that

$$\langle \lambda\psi_n, \widehat{\Psi}^{(l)} \rangle = [\, \tau_n \quad 0 \quad g_l^T \,] J V^{(l)}, \quad \langle \lambda\psi_n, \Phi^{(l)} \rangle = [\, \tau_n \quad 0 \quad g_l^T \,] U^{(l)}. \tag{5.34}$$

By inserting (5.34) into (5.32), we obtain the relation (5.29).

Now we turn to part b). Since the proofs of (5.30) and (5.31) are complete analogous, we will only show (5.30). We assume that φ_{n+1} is constructed as an inner polynomial; note that φ_{n+1} is given by (5.7), where, by (5.25), $\xi_i^{(n)} = 0$ for all i. Let $n_l \le m+1 \le n+1$. From (5.6), if $m+1 = n_l$, and (5.8) (with $\zeta_i^{(m)} = 0$ for all i), if $m+1 > n_l$, it follows that ψ_{m+1} can be written in the form

$$\psi_{m+1}(\lambda) \equiv \psi_{m+1}(0) + \lambda\psi_m(\lambda) + \lambda\chi_m(\lambda) \quad \text{for some} \quad \chi_m \in \mathcal{P}_{n_l-2}. \tag{5.35}$$

Moreover, in view of the representation (5.13) of ψ_{m+1}, we have

$$\psi_{m+1}(0) = v_{0,m+1}. \tag{5.36}$$

Using (5.7), (4.13), (5.35), (3.3), (5.36), and the formula of ρ_n in (5.4), one easily verifies that

$$\begin{aligned} \langle \psi_{m+1}, \varphi_{n+1} \rangle &= \langle \psi_{m+1}, \lambda\varphi_n - \widehat{\Psi}^{(l-1)}\alpha_n \rangle = \langle \psi_{m+1}, \lambda\varphi_n \rangle \\ &= \psi_{m+1}(0)\langle 1, \lambda\varphi_n \rangle + \langle \lambda\psi_m, \lambda\varphi_n \rangle + \langle \lambda\chi_m, \lambda\varphi_n \rangle \\ &= v_{0,m+1}\rho_n + \langle \psi_m, \varphi_n \rangle. \end{aligned} \tag{5.37}$$

If $m+1 > n_l$, then (5.37) is just the desired relation (5.30). If $m+1 = n_l$, then, by (4.13) and since $n \geq n_l > m$, we have $\langle \psi_m, \varphi_n \rangle = 0$, and thus (5.37) reduces to (5.30). □

Typically, Algorithm 5.1 will perform mostly regular steps 3), and then the formulas (5.5) and (5.6) simplify somewhat. Indeed, assume that Algorithm 5.1 performs regular steps in two consecutive iterations, i.e.,

$$n+1 = n_{l+1} = n_l + 1 \quad \text{and} \quad h_l = n_{l+1} - n_l = 1. \tag{5.38}$$

Using (5.3), (5.2), (4.13), and (5.12), one easily shows that, in this case,

$$F^{(l)} = G^{(l)} = \langle \psi_n, \varphi_n \rangle = D^{(l)}. \tag{5.39}$$

With (5.39), (4.16), (5.5), and (5.6), we conclude that

$$f_l = g_l = \emptyset, \quad \mu_n = \nu_n = 0, \quad \alpha_n = \frac{\rho_n}{\langle \psi_n, \varphi_n \rangle}, \quad \beta_n = \frac{\tau_n}{\langle \psi_n, \varphi_n \rangle}. \tag{5.40}$$

In particular, in recurrences for φ_{n+1} and ψ_{n+1} in (5.5) and (5.6) reduce to

$$\varphi_{n+1} = \lambda \varphi_n - \widehat{\psi}_n \alpha_n \quad \text{and} \quad \psi_{n+1} = \lambda \psi_n - \widehat{\varphi}_n \beta_n, \tag{5.41}$$

respectively. Furthermore, with (5.40), (5.26), (5.39), and since $JV^{(l)} = Jv_n = [1 \quad \cdots \ ]^T$, it follows that the update formula (5.29) reduces to

$$\langle \psi_{n+1}, \varphi_{n+1} \rangle = \langle \psi_n, \varphi_n \rangle \left(1 - \frac{\rho_n \tau_n}{\langle \psi_n, \varphi_n \rangle^2} \right). \tag{5.42}$$

Note that, in view of (4.4), the recursions (5.41) and (5.42) are identical to the update formulas (4.2) and (4.3). In other words, the nth step of Algorithm 5.1 just reduces to the nth step of the Szegö Algorithm 4.1, if (5.38) is satisfied. In particular, for the case that $n_{k+1} = n_k + 1$ for all $k = 0, 1, \ldots$, we have the following result.

Corollary 5.3 *If all polynomials are constructed as regular* FBOP*s, then Algorithm* 5.1 *reduces to the Szegö Algorithm* 4.1.

6 A Look-Ahead Toeplitz Systems Solver

In this section, we present our look-ahead Levinson algorithm for solving general Toeplitz systems. We give implementation details and operation counts, and we describe the look-ahead strategy.

6.1 The Algorithm

By means of (5.13), Algorithm 5.1 can be rewritten in terms of the coefficients vectors u_n and v_n of the polynomials φ_n and ψ_n, $n = 0, 1, \ldots$. Recall from Section 5.2 that these vectors are just the columns of the triangular matrices U_n and V_n in the factorization (5.14)–(5.17) of T_n. We call u_n and v_n *regular vectors* if they are the coefficient vectors of regular FBOPs φ_n and ψ_n, and we refer to u_n and v_n as *inner vectors* if they correspond to a pair of inner polynomials.

In the following, we denote by $s_n,\ r_n \in \mathbb{C}^{n+1}$ the vectors defined by the partitioning

$$T_{n+1} = \begin{bmatrix} t_0 & s_n^T \\ r_n & T_n \end{bmatrix} \tag{6.1}$$

of T_{n+1}. In view of (5.13), we have

$$\lambda\varphi_n = \Lambda_{n+1} \begin{bmatrix} 0 \\ u_n \end{bmatrix} \quad \text{and} \quad \lambda\psi_n = \Lambda_{n+1} \begin{bmatrix} 0 \\ v_n \end{bmatrix}. \tag{6.2}$$

From (6.1), (6.2), and the definition of $\langle\cdot,\cdot\rangle$ in (3.1), it follows that

$$\langle 1, \lambda\varphi_n \rangle = s_n^T u_n \quad \text{and} \quad \langle \lambda\psi_n, 1 \rangle = v_n^T r_n. \tag{6.3}$$

Also, note that $\langle 1, 1 \rangle = t_0$. Finally, using (5.13), (6.3), (5.19), (5.20), (2.7), (5.25), and the vectors $\alpha_k^{(1)}$, $\beta_k^{(1)}$ defined in (5.26), we can rewrite Algorithm 5.1 as follows.

Algorithm 6.1 (Inverse block factorization of general T_n)

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*

$$\rho_n = s_n^T u_n, \quad \tau_n = v_n^T r_n. \tag{6.4}$$

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*

$$(G^{(l)})^T \alpha_l^{(1)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad F^{(l)}\mu_n = \begin{bmatrix} 0 \\ f_l \end{bmatrix}, \quad F^{(l)}\beta_l^{(1)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad (G^{(l)})^T \nu_n = \begin{bmatrix} 0 \\ g_l \end{bmatrix}, \tag{6.5}$$

respectively, and set

$$\begin{aligned} u_{n+1} &= \begin{bmatrix} 0 \\ u_n \end{bmatrix} - \rho_n \begin{bmatrix} JV^{(l)} \\ 0 \end{bmatrix} \alpha_l^{(1)} - \begin{bmatrix} U^{(l)} \\ 0 \end{bmatrix} \mu_n, \\ v_{n+1} &= \begin{bmatrix} 0 \\ v_n \end{bmatrix} - \tau_n \begin{bmatrix} JU^{(l)} \\ 0 \end{bmatrix} \beta_l^{(1)} - \begin{bmatrix} V^{(l)} \\ 0 \end{bmatrix} \nu_n. \end{aligned} \tag{6.6}$$

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

Note that Algorithm 6.1 only computes the inverse block factorization (5.14) of T_n. Next, we discuss how to obtain solutions for nested Toeplitz systems

$$T_n x_n = b_n, \quad n = 0, 1, \dots . \tag{6.7}$$

Here the right-hand sides $b_n \in \mathbb{C}^{n+1}$ are assumed to be nested, i.e.,

$$b_{n+1} = \begin{bmatrix} b_n \\ \sigma_{n+1} \end{bmatrix}, \quad \text{with} \quad \sigma_{n+1} \in \mathbb{C}, \quad \text{for all} \quad n.$$

Recall that T_n is guaranteed to be nonsingular for $n = n_k - 1$, $k = 1, 2, \dots, l$, and we only update the solution x_n of (6.7) for these values of n. To this end, we simply need to insert the following procedure at the beginning of each regular step 3) in Algorithm 6.1:

Set $n' = n_l - 1$, *and partition* $U^{(l)}$, b_n, *and* T_n *as follows*:

$$U^{(l)} = \begin{bmatrix} \hat{U}^{(l)} \\ \tilde{U}^{(l)} \end{bmatrix}, \quad b_n = \begin{bmatrix} b_{n'} \\ \sigma \end{bmatrix}, \quad T_n = \begin{bmatrix} T_{n'} & S \\ R & T_{n-n'} \end{bmatrix}, \tag{6.8}$$

where $\tilde{U}^{(l)}$ *and* σ *just contain the last* $n - n'$ *rows of* $U^{(l)}$ *and* b_n, *respectively. Compute* y *by solving*

$$F^{(l)} y = \sigma - R\, x_{n'}, \tag{6.9}$$

and set

$$x_n = \begin{bmatrix} x_{n'} \\ 0_{n-n'} \end{bmatrix} + U^{(l)} y. \tag{6.10}$$

We need to show that x_n given by (6.10) and (6.9) is the solution of (6.7). Indeed, by means of (6.8)–(6.10) and the first relation in (5.22), it follows that

$$T_n x_n = \begin{bmatrix} T_{n'} \\ R \end{bmatrix} x_{n'} + T_n U^{(l)} y = \begin{bmatrix} b_{n'} \\ R\, x_{n'} \end{bmatrix} + \begin{bmatrix} 0 \\ F^{(l)} \end{bmatrix} (F^{(l)})^{-1} (\sigma - R\, x_{n'}) = b_n.$$

6.2 Implementation Details and Operation Counts

We now discuss some implementation details for Algorithm 6.1, and we present operation counts.

Except for contrived examples, Toeplitz matrices that arise in practical applications have at most a small number of consecutive ill-conditioned leading principal submatrices. Consequently, Algorithm 6.1 mostly performs regular steps. Typically, only a few true look-ahead steps occurs, and their length h_l is usually small, mostly $h_l = 2$. This justifies the following convention that we will use for the operation count: a computation that requires only arithmetic operations of order $\mathcal{O}(h_l^3)$ or less is considered *negligible*.

We now consider steps 1)–5) of Algorithm 6.1 in more detail. Step 1) involves the computation of two inner products of vectors of length $n+1$. It turns out that these are the only two inner products that are required during the nth iteration. This is exactly the same as in the classical Levinson algorithm for strongly regular matrices.

The look-ahead strategy for the decision in step 2) will be described in Section 6.3. As we will see there, it only involves negligible work.

Next we turn to step 3). Note that $(G^{(l)})^T$ and $F^{(l)}$ are $h_l \times h_l$ matrices, and, if $h_l > 1$, we use Gaussian elimination to solve the four linear systems in (6.5). Recall from (4.17) that f_l and g_l are given as part of the last columns of $F^{(l)}$ and $(G^{(l)})^T$. If $h_l = 1$, then, by (5.40), $\mu_n = \nu_n = 0$, and the two updates in (6.6) require two SAXPYs[2] with vectors of length $n+1$. If $h_l > 1$, then we first compute the two vectors

$$JV^{(l)}\alpha_l^{(1)} \quad \text{and} \quad JU^{(l)}\beta_l^{(1)}, \tag{6.11}$$

which costs $2h_l$ SAXPYs. The two updates in (6.6) then require $2(h_l+1)$ additional SAXPYs.

Step 4) always requires two SAXPYs. This is obvious if $h_{l-1} = 1$. If $h_{l-1} > 1$, we use the vectors (6.11) (with l replaced by $l-1$), which were already computed in the course of the last regular step.

In step 5), we need to update the matrices $F^{(l)}$ and $G^{(l)}$. To this end, we first update $D^{(l)}$ using Lemma 5.2, and then we compute $F^{(l)}$ and $G^{(l)}$ by means of (5.24). Note that, by (5.23), the triangular matrices $\tilde{U}^{(l)}$ and $\tilde{V}^{(l)}$ in (5.24) are given as part of $U^{(l)}$ and $V^{(l)}$. Consequently, step 5) only involves negligible work.

Finally, we turn to the procedure for updating solutions of Toeplitz systems (6.7). To compute the right-hand side of (6.9), we need to generate $R\,x'_n$, which involves $h_l = n - n'$ inner products. The computation of the vector x_n in (6.10) requires another h_l SAXPYs. Note that x_n is only updated once within each cycle of h_l steps. Thus, in the average, the update procedure requires one inner product and one SAXPY per nth step.

In Table 1, we summarize the operation counts for one step of Algorithm 6.1, and for the updating procedure for solutions of general Toeplitz systems (6.7).

[2]A SAXPY operation is $z = x + \alpha y$, where x and y are vectors and α is a scalar, see, e.g., [16].

	regular step		inner step	update of x_n
	with $h_l = 1$	with $h_l > 1$		
inner products/step	2	2	2	1
SAXPYs/step	2	$2(2h_l + 1)$	2	1

Table 1: Operation counts for Algorithm 6.1

6.3 The Look-Ahead Strategy

The look-ahead strategy is crucial both for the accuracy and efficiency of Algorithm 6.1. Its main purpose is to skip over ill-conditioned leading principal submatrices in order to avoid breakdowns and numerical instabilities. However, as is obvious from the operation counts in Table 1, it is more expensive to perform a look-ahead step of length $h_l > 1$ than h_l classical Levinson steps. Therefore, for the sake of efficiency, it is desirable to perform look-ahead steps only when necessary. The look-ahead strategy is implemented through the criteria that are used in step 2) of Algorithm 6.1 to decide whether the next vectors are constructed as regular or inner ones. There are two quantities, denoted by $\kappa(\Gamma_l)$ and η_n, that are monitored throughout the algorithm. Both of them are obtained from local information only. In particular, we do not need to estimate the condition number of the current leading principal submatrix T_n. The decision about building regular or inner vectors is then based on a comparison of $\kappa(\Gamma^{(l)})$ and η_n with two threshold parameters `COND` (> 0) and `GFACTOR` (≥ 1), respectively. The algorithm dynamically determines `COND` and `GFACTOR`, and the only input that is required from the user is the number $h_{\max}$, which is the maximal length of a look-ahead step the algorithm is allowed to perform. Note that, in view of (5.18), we then have

$$h_k \leq h_{\max} \quad \text{for all} \quad k,$$

and Algorithm 6.1 generates an inverse block factorization (5.14) with blocks of size $\leq h_{\max}$. We remark that Algorithm 6.1 reduces to the classical Levinson algorithm if we set $h_{\max} = 1$.

The initialization phase of the algorithm is as follows. As the first block, we set $D^{(0)} = T_m$, where T_m is the matrix with the smallest condition number[3] $\kappa(T_m)$ among T_h, $h = 0, 1, \ldots, h_{\max} - 1$. Then we build the next vector u_{m+1} and v_{m+1} as regular vectors and we set $n_1 = m + 1$. Furthermore, we initialize `COND` to be this smallest condition number $\kappa(T_m)$, and we set `GFACTOR` to 1.

Now we consider a general iteration step of Algorithm 6.1. As we mentioned before, for the sake of efficiency, the look-ahead Toeplitz solver should build as many regular vectors as possible. Therefore, in each iteration step, we first pretend that u_{n+1} and v_{n+1} can actually be constructed as regular vectors. Recall from (5.9) that, for a regular step, it is necessary that $F^{(l)}$ and $G^{(l)}$ are nonsingular. To check this condition, we compute the matrix

$$\Gamma^{(l)} := (\tilde{V}^{(l)})^{-T} D^{(l)} (\tilde{U}^{(l)})^{-1},$$

[3]Recall the definition of of κ in (2.1).

where $\widetilde{U}^{(l)}$ and $\widetilde{V}^{(l)}$ are the triangular matrices given by (5.23). Note that, by (5.24), we have

$$\Gamma^{(l)} = F^{(l)}(\widetilde{U}^{(l)})^{-1} = (\widetilde{V}^{(l)})^{-T}G^{(l)},$$

and hence $F^{(l)}$ and $G^{(l)}$ are nonsingular if, and only if, $\Gamma^{(l)}$ is nonsingular. We now check whether

$$\kappa(\Gamma^{(l)}) \leq 2{*}\mathtt{COND}. \tag{6.12}$$

If (6.12) is not satisfied, then we go to step 4) in Algorithm 6.1, and we build u_{n+1} and v_{n+1} as inner vectors. To justify the criterion (6.12), recall that, in view of Lemma 3.1, T_n is required to be nonsingular for a regular step. Actually, at the end of this section, we will point out that $\kappa(\Gamma^{(l)})$ is closely related to $\kappa(T_n)$.

If (6.12) is satisfied, then we compute the quantities α_n, μ_n, β_n, and ν_n, using (6.5) and the first two relations in (5.27), and we set

$$\eta_n := \max\{\|\alpha_n\|_1, \|\mu_n\|_1, \|\beta_n\|_1, \|\nu_n\|_1\}.$$

Here $\|x\|_1 := |\xi_1| + \cdots + |\xi_h|$ denotes the 1-norm of a vector $x = [\,\xi_1 \quad \cdots \quad \xi_h\,]^T \in \mathbb{C}^h$. Then, we check whether

$$\eta_n \leq 2{*}\mathtt{GFACTOR}. \tag{6.13}$$

If the criterion (6.13) is not satisfied, we proceed with step 4) and construct u_{n+1} and v_{n+1} as inner vectors. The justification for (6.13) is as follows. Note that, by (5.14),

$$T_{n+1} = V_{n+1}^{-T} D_{n+1} U_{n+1}^{-1} \quad \text{and} \quad T_{n+1}^T = U_{n+1}^{-T} D_{n+1}^T V_{n+1}^{-1}. \tag{6.14}$$

If one would compute the decompositions (6.14) of T_{n+1} or T_{n+1}^T directly by means of Gaussian elimination, then pivoting would be used to ensure that the size of the off-diagonal elements of U_{n+1}^{-1} and V_{n+1}^{-1} is bounded by 1. Indeed, this is the key to numerical stability of Gaussian elimination. Recall that the elements of the strictly upper triangular parts of U_{n+1}^{-1} and V_{n+1}^{-1} are just the multipliers in Gaussian elimination. Now, for Toeplitz matrices pivoting would destroy the Toeplitz structure. Roughly speaking, the look-ahead Algorithm 6.1 performs a true look-ahead step of length $h_l > 1$, whenever one would encounter a small pivot in Gaussian elimination. Consequently, the look-ahead strategy should also guarantee that off-diagonal elements of U_{n+1}^{-1} and V_{n+1}^{-1} are not too large. Since

$$U_{n+1}^{-1} = I + C + C^2 + \cdots + C^{n+1}, \quad \text{where} \quad C := I - U_{n+1},$$

a large off-diagonal element of U_{n+1} usually leads to a large off-diagonal element of U_{n+1}^{-1}. A similar conclusion also holds for V_{n+1}^{-1}. Therefore, in each step of the Algorithm 6.1, we limit the growth in the newly generated off-diagonal elements of U_{n+1} and V_{n+1}, which are just the components of u_{n+1} and v_{n+1}. This is the purpose of imposing check (6.13).

If both criteria (6.12) and (6.13) are satisfied, then we proceed with step 3) in Algorithm 6.1 and construct u_{n+1} and v_{n+1} as regular vectors.

It cannot be excluded that the algorithm has reached the maximal block size $h_{\max}$, but the two checks for building the next vectors as regular ones are still not satisfied. More precisely, the algorithm has built a pair of blocks $U^{(l)}$ and $V^{(l)}$ of size $h_{\max}$, both starting with index n_l and ending with index $n_l + h_{\max} - 1$, and the criteria (6.12) and (6.13) for constructing $u_{n_l+h_{\max}}$ and $v_{n_l+h_{\max}}$ as regular vectors are not fulfilled. In this case, we adjust the values of the threshold parameters `COND` and `GFACTOR` in such a way that the criteria for building regular vectors are satisfied within the maximal look-ahead size $h_{\max}$. To this end, we first determine a step size $h \in \{2, 3, \ldots, h_{\max}\}$ such that

$$\kappa(\Gamma_h^{(l)}) + |\Gamma_1^{(l)}| \eta_{n_l+h-1} \tag{6.15}$$

is minimal. Here $\Gamma_j^{(l)}$, $j = 1, \ldots, h_{\max}$, denotes the $j \times j$ leading principal submatrix of $\Gamma^{(l)}$. We then set `COND` to be the condition number of the corresponding $\Gamma_h^{(l)}$, and the second parameter `GFACTOR` is set to the value of the corresponding η_{n_l+h-1}. This choice of h guarantees that the vectors with index $n_l + h$ can be constructed as regular vectors. The motivation for the choice (6.15) is as follows. From the above discussion, it is clear that the goal is to minimize simultaneously $\kappa(\Gamma_h^{(l)})$ and η_{n_l+h-1}; this is exactly what (6.15) attempts to ensure. The weighting factor $|\Gamma_1^{(l)}|$ in (6.15) was chosen based on extensive numerical tests, and the choice (6.15) was found to work satisfactory in practice. With the described look-ahead strategy, the algorithm can be expected to produce accurate solutions of Toeplitz systems as long as the coefficient matrix has at most $h_{\max} - 1$ consecutive ill-conditioned leading principal submatrices.

We conclude this section with a discussion of the connection of the condition numbers of $\Gamma^{(l)}$ and T_n. We set $n' := n_l - 1$, and we partition T_n as follows:

$$T_n = \begin{bmatrix} T_{n'} & S \\ R & \tilde{T} \end{bmatrix}.$$

Here $T_{n'}$ is the last well-conditioned Toeplitz submatrix. With this notation, $\Gamma^{(l)}$ is, in fact, the Schur complement of $T_{n'}$ in T_n, and we have the decomposition

$$T_n = \begin{bmatrix} I & 0 \\ R T_{n'}^{-1} & I \end{bmatrix} \begin{bmatrix} T_{n'} & 0 \\ 0 & \Gamma^{(l)} \end{bmatrix} \begin{bmatrix} I & T_{n'}^{-1} S \\ 0 & I \end{bmatrix},$$

which implies that

$$\begin{aligned} \Gamma^{(l)} &= [\,0 \quad I\,] \begin{bmatrix} I & 0 \\ -R T_{n'}^{-1} & I \end{bmatrix} T_n \begin{bmatrix} I & -T_{n'}^{-1} S \\ 0 & I \end{bmatrix} \begin{bmatrix} 0 \\ I \end{bmatrix} \\ &= [\,-R T_{n'}^{-1} \quad I\,]\, T_n \begin{bmatrix} -T_{n'}^{-1} S \\ I \end{bmatrix}. \end{aligned} \tag{6.16}$$

From (6.16), it follows that

$$\|\Gamma^{(l)}\| \le \|[\,-R T_{n'}^{-1} \quad I\,]\| \cdot \left\| \begin{bmatrix} -T_{n'}^{-1} S \\ I \end{bmatrix} \right\| \cdot \|T_n\|.$$

It can be verified that

$$\|[\,-R\,T_{n'}^{-1} \quad I\,]\| \le \sqrt{1 + (\|R\|\kappa(T_{n'})/\|T_{n'}\|)^2},$$

$$\left\|\begin{bmatrix} -T_{n'}^{-1}S \\ I \end{bmatrix}\right\| \le \sqrt{1 + (\|S\|\kappa(T_{n'})/\|T_{n'}\|)^2}.$$

Therefore, setting

$$\varrho := \sqrt{1 + (\|R\|\kappa(T_{n'})/\|T_{n'}\|)^2}\sqrt{1 + (\|S\|\kappa(T_{n'})/\|T_{n'}\|)^2},$$

we have

$$\|\Gamma^{(l)}\| \le \varrho\,\|T_n\|\,. \tag{6.17}$$

Similarly, we can show

$$\left\|(\Gamma^{(l)})^{-1}\right\| \le \varrho\,\left\|T_n^{-1}\right\|. \tag{6.18}$$

To get bounds for the condition numbers, we distinguish two cases. First, assume that $\Gamma^{(l)}$ is of size > 1. Then, by combining (6.17) and (6.18), we arrive at

$$\kappa(T_n) \le \varrho^2\kappa(\Gamma^{(l)}).$$

With the same technique, we can show that

$$\kappa(\Gamma^{(l)}) \le \varrho^2\kappa(T_n),$$

and hence

$$\kappa(T_n)/\varrho^2 \le \kappa(\Gamma^{(l)}) \le \varrho^2\kappa(T_n). \tag{6.19}$$

For the case that $\Gamma^{(l)}$ is a scalar, from (6.17) and (6.18), we obtain

$$\frac{1}{\varrho\,\|T_n\|} \le \kappa(\Gamma^{(l)}) \le \varrho\,\|T_n^{-1}\|. \tag{6.20}$$

Roughly speaking, the inequalities (6.19), respectively (6.20), state that, if $T_{n'}$ is well conditioned, then T_n is well conditioned if, and only if, $\Gamma^{(l)}$ is well conditioned.

7 The Special Case of Hermitian Toeplitz Matrices

In this section, we consider the special case of Hermitian Toeplitz matrices.

7.1 A Look-Ahead Levinson Algorithm

Suppose that the elements of the biinfinite sequence $\{t_i\}_{i=-\infty}^{\infty}$ satisfy $t_{-i} = \overline{t}_i$ for all i. Then the Toeplitz matrices (1.1) are all Hermitian, i.e.,

$$T_n = T_n^H, \quad n = 0, 1, \ldots\ . \tag{7.1}$$

Using (7.1) and the definition of $\langle\cdot,\cdot\rangle$, one easily verifies that regular FBOPs and inner quasi-FBOPs associated with the bilinear form $\langle\cdot,\cdot\rangle$ are connected by

$$\psi_n = \overline{\varphi}_n. \tag{7.2}$$

Here $\overline{\varphi}$ denotes the polynomial whose coefficients are just the complex conjugates of the coefficients of φ. In other words, the coefficient vectors u_n and v_n of φ_n and ψ_n satisfy

$$v_n = \overline{u}_n. \tag{7.3}$$

Consequently, for Hermitian Toeplitz matrices, Algorithm 6.1 simplifies, since we only need to update the vectors u_n. Furthermore, note that, in (6.1), $s_n = \overline{r}_n$, and together with (7.3) and (6.4), we obtain

$$\tau_n = v_n^T r_n = u_n^H r_n = (r_n^H u_n)^H = \overline{\rho}_n. \tag{7.4}$$

In view of (7.2), the matrices (4.12) are now connected by

$$F^{(k)} = (G^{(k)})^H. \tag{7.5}$$

Finally, using (7.3)–(7.5) and setting

$$\gamma_k^{(1)} := \overline{\alpha_k^{(1)}} = (F^{(k)})^{-1}\begin{bmatrix}0\\1\end{bmatrix},$$

we obtain from Algorithm 6.1 the following look-ahead Levinson algorithm for Hermitian Toeplitz matrices.

Algorithm 7.1 (Inverse block factorization of Hermitian T_n)

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

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

1) *Compute* $\tau_n = u_n^H r_n$.

2) *Decide whether to construct* u_{n+1} *as a regular vector or as an inner vector and go to* 3) *or* 4), *respectively.*

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

$$F^{(l)}\gamma_l^{(1)} = \begin{bmatrix}0\\1\end{bmatrix}, \quad F^{(l)}\mu_n = \begin{bmatrix}0\\f_k\end{bmatrix},$$

respectively, and set

$$u_{n+1} = \begin{bmatrix}0\\u_n\end{bmatrix} - \tau_n \begin{bmatrix}J\overline{U}^{(l)}\\0\end{bmatrix}\gamma_l^{(1)} - \begin{bmatrix}U^{(l)}\\0\end{bmatrix}\mu_n.$$

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

4) (Inner step) *Set*

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

5) *Set*

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

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

7.2 A Counterexample

Chan and Hansen proposed a look-ahead Levinson procedure that is different from our algorithm. Their method was first presented for the special case of real symmetric Toeplitz matrices [9], and then extended to general nonsymmetric Toeplitz systems in [8]. The derivation of their algorithm is actually based on the assumption that only *isolated* look-ahead steps occur, which are preceded and followed by standard Levinson steps. However, in general it cannot be excluded that T_n has two or more *consecutive* blocks of singular or ill-conditioned leading principal submatrices, which then requires two or more *consecutive* look-ahead steps. In both papers [9] and [8], this case is treated separately, and special recurrences are derived for handling two consecutive look-ahead steps, see [9, Theorem 3] and [8, Theorem2]. However, the proposed approach involves division by the first component c_1 of a coefficient vector c, see [9, Equation (5.9)] and [8, Equation (50)]. Unfortunately, it is not guaranteed that $c_1 \neq 0$, and thus division by 0 can occur. Indeed, we now present a real symmetric Toeplitz matrix for which $c_1 = 0$. Consequently, the look-ahead Levinson algorithm proposed by Chan and Hansen can break down for general Toeplitz systems, as well as for the special case of Hermitian Toeplitz matrices.

Consider the 7×7 Toeplitz matrix

$$T_6 = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}. \tag{7.6}$$

This matrix has exactly singular leading principal submatrices T_0, T_2, T_3, and T_4, while the remaining submatrices T_1, T_5, and T_6, are all nonsingular with condition numbers $\kappa(T_1) = 1$, $\kappa(T_5) \approx 18.1$, and $\kappa(T_6) \approx 7.2$. In particular, T_1 is optimally conditioned, and the algorithm in [9] starts with a look-ahead step of length 2, followed by another look-ahead step of length 4 or 5. For the update of quantities corresponding to the second look-ahead step, the algorithm [9] requires division by the first component c_1 of a vector c that, using the

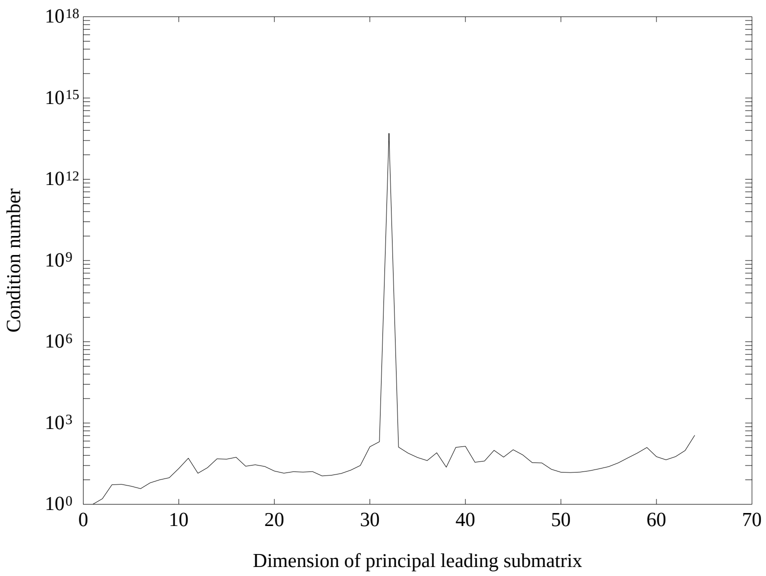

Figure 1: Condition number profile of a 64×64 Toeplitz matrix

notations from [9], is given as follows:

$$r_4^{(2)} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}, \quad y_2 = -\begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad R_4 = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}, \quad c = -r_4^{(2)} - R_4^T J y_2 = \begin{bmatrix} 0 \\ -1 \\ 1 \\ 0 \end{bmatrix}.$$

Thus we have $c_1 = 0$, and the algorithm breaks down in the course of the second look-ahead step.

8 Numerical Experiments

In this section, we report results of numerical experiments with the look-ahead Algorithms 6.1 and 7.1, and the classical Levinson algorithm. All computations were carried out using MATLAB on a DEC 3100 workstation with machine precision of order $\mathcal{O}(10^{-16})$. For all the examples, we generated the right-hand side b_n such that the vector of all 1's is the exact solution x_{exact} of $T_n x_n = b_n$. In the following, we always list the relative error defined as

$$\text{relative error} \;=\; \frac{\|x_{\text{compt}} - x_{\text{exact}}\|}{\|x_{\text{exact}}\|},$$

where x_{compt} is the computed solution.

Example 1. This test set consists of 100 nonsymmetric 64×64 matrices with at least one ill-conditioned leading principal submatrix. The off-diagonal entries t_i of these matrices

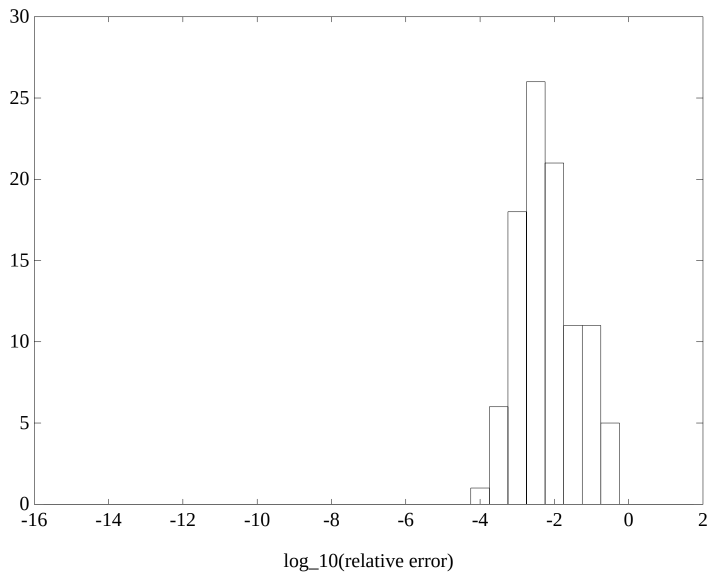

Figure 2: Histogram of the relative errors for the classical Levinson algorithm, Example 1

$n+1$	classical	look-ahead	overhead
64	12096	12740	3.7 %
200	119400	120010	3.5 %

Table 2: Average number of multiplications for Examples 1 and 2

were generated as random numbers in $[-1,1]$, and t_0 was then chosen so that at least on submatrix is ill-conditioned. A typical condition number profile of a matrix in this test set is shown in Figure 1. First, we use the classical Levinson algorithm, and in Figure 2, we show—in the form of a histogram—the relative errors for the 100 test matrices. We see that the relative errors are rather poor. In Figure 3, we plot the relative errors for the look-ahead Algorithm 6.1 (with $h_{\max}=2$). We see a substantial improvement of the relative errors.

EXAMPLE 2. This test set consists of 100 nonsymmetric 200×200 matrices with at least one ill-conditioned leading principal submatrix. The matrices were generated as in Example 1. In Figure 4 and Figure 5, we plot the histograms of the relative errors for the classical Levinson algorithm and the look-ahead Algorithm 6.1 (with $h_{\max}=2$), respectively. In Table 2, we list, for both Example 1 and Example 2, the average number of multiplications required to solve one system in the test set by the classical Levinson algorithm and the look-ahead algorithm, and we state the corresponding overhead for the look-ahead algorithm.

EXAMPLE 3. This test set consists of symmetric matrices given by

$$t_0=\epsilon, \quad t_i=t_{-i}=(1/2)^i, \quad i=1,2,\ldots,n. \tag{8.1}$$

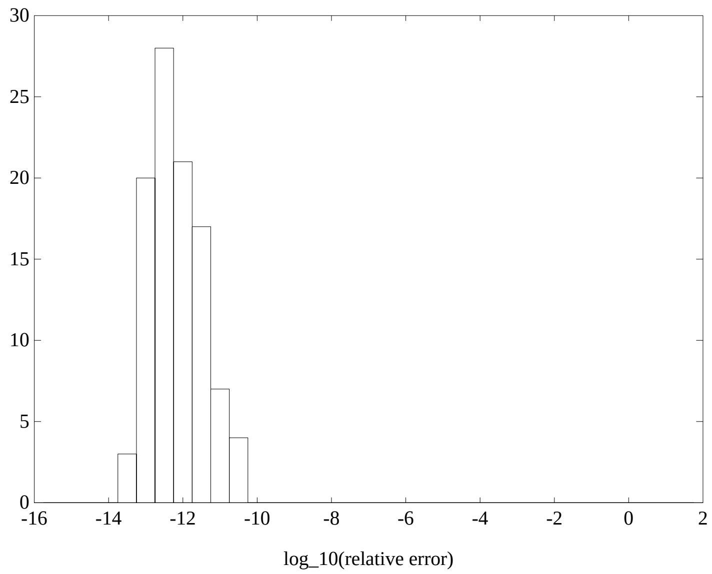

Figure 3: Histogram of the relative errors for the look-ahead algorithm, Example 1

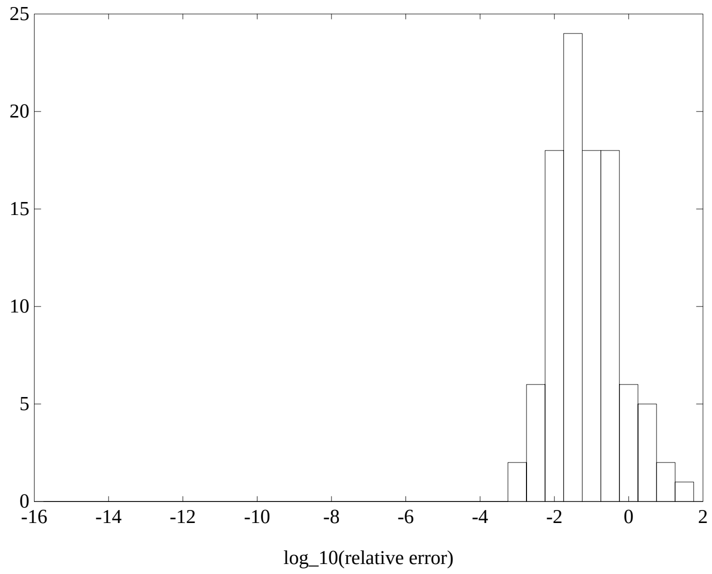

Figure 4: Histogram of the relative errors for the classical Levinson algorithm, Example 2

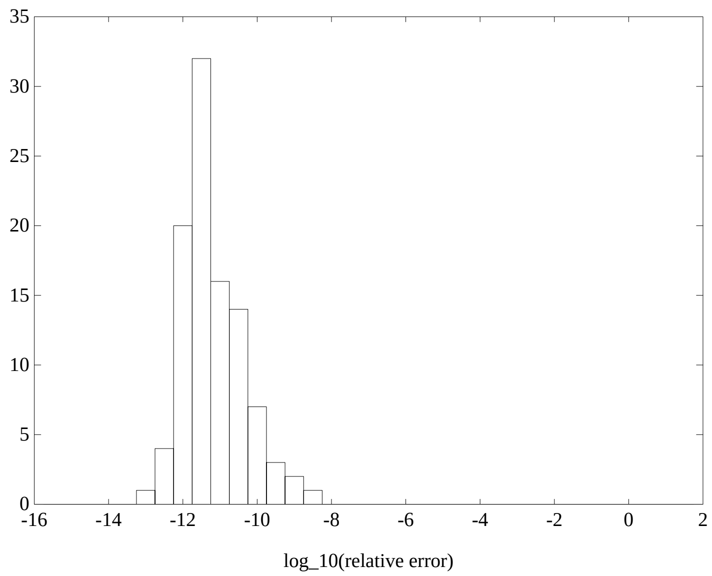

Figure 5: Histogram of the relative errors for the look-ahead algorithm, Example 2

These are special cases of a class of Toeplitz matrices called Kac-Murdock-Szegö (KMS) matrices (with $\rho = 1/2$), see [28]. The eigenvalues of the KMS matrices (8.1) can be easily computed [44]. It turns out that, for $\epsilon = 0$, each third principal submatrix, i.e., T_{3m}, $m = 0, 1, \ldots,$ is exactly singular, while the remaining submatrices are well conditioned. Consequently, if ϵ is set to a small number, then every third principal leading submatrices is ill-conditioned. The KMS matrices (8.1), with $\epsilon = 10^{-14}$, were also used as test examples in [9], and we chose the same parameter $\epsilon = 10^{-14}$. We ran the classical Levinson algorithm and the look-ahead Levinson Algorithm 7.1 for KMS matrices T_n of order $n + 1 = 15$, 30, 60, 120, 240, 480. The relative errors and the number of multiplications are listed in Table 3. We see that the look-ahead procedures yields solutions with nearly full accuracy. Considering that one third of the iteration steps are inner steps, the overhead of the look-ahead procedure is still reasonable.

Example 4. This test set consists of three small nonsymmetric matrices that have at least two *consecutive* ill-conditioned leading principal submatrices. To list the entires $\{t_j\}_{j=-n}^{n}$ of a Toeplitz matrix (1.1), T_n, of order $n+1$, we use the following convention:

$$t := t_0, \quad t1 := [\, t_{-1} \quad t_{-2} \quad \cdots \quad t_{-n} \,], \quad t2 := [\, t_1 \quad t_2 \quad \cdots \quad t_n \,].$$

Example 4.1. This is a 5×5 matrix whose entries are listed in Table 4. As the condition number profile in Table 5 shows, this matrix has two consecutive ill-conditioned leading principal submatrices.

Example 4.2. This is a 6×6 matrix with three consecutive ill-conditioned leading principal submatrices. Its entries are listed in Table 6, and its condition number profile is shown in Table 7.

	relative error		multiplications		
$n+1$	classical	look-ahead	classical	look-ahead	overhead
15	0.0191	1.20^{-15}	630	960	52%
30	0.0184	1.79^{-15}	2610	3870	48%
60	0.0185	1.98^{-15}	10620	15340	46%
120	0.0186	4.61^{-15}	42840	62280	45%
240	0.0186	6.85^{-15}	172080	248333	44%
480	0.0187	3.69^{-14}	689760	995853	44%

Table 3: Test results for the KMS matrices, Example 3

t	t1	t2
-1.00000000000001	1.27324683138786	0.78539366864947
	-1.62115749363923	3.41046741401696
	1.06413364195684	-17.92422495778239
	1.21785304238395	38.20692196916536

Table 4: Elements of the matrix in Example 4.1

m	0	1	2	3	4
$\kappa(T_m)$	1.00	1.25e+15	7.46e+14	4.00e+1	4.70e+2

Table 5: Condition number profile of the matrix in Example 4.1

t	t1	t2
-0.99999999999998	1.05288024249153	0.94977563415339
	-1.10855680502906	3.85673107101965
	1.16717755769466	-13.61721591570147
	-2.22889818997626	3.81850412563076
	4.51853189291597	73.05176317918625

Table 6: Elements of the matrix in Example 4.2

m	0	1	2	3	4	5
$\kappa(T_m)$	1.00	7.70e+14	6.33e+14	3.91e+15	4.00e+1	4.80e+2

Table 7: Condition number profile of the matrix in Example 4.2

t	5											
t1	-1	6	2	5.697	5.850	3	-5	-2	-7	1	10	-15
t2	1	-3	12.755	-19.656	28.361	-7	-1	2	1	-6	1	-0.5

Table 8: Elements of the matrix in Example 4.3

EXAMPLE 4.3. This example is taken from [41], and it was also used in [8]. It is a 13×13 matrix with five consecutive ill-conditioned leading principal submatrices. Its entries are listed in Table 8, and the condition number profile is shown in Table 9.

We ran the classical Levinson algorithm and the look-ahead Algorithm 6.1 for the three matrices in Example 4. The relative errors are presented in Table 10.

EXAMPLE 5. Finally, we consider the symmetric 7×7 matrix (7.6) from Section 7.2. The classical Levinson algorithm breaks down for this matrix, while the look-ahead Algorithm 7.1 generates the exact solution. Next, we perturb the matrix slightly by adding a Toeplitz matrix with random entries in $[-10^{-14}, 10^{-14}]$ to (7.6). The condition number profile of the resulting matrix is shown in Table 11. Note that none of the submatrices are exactly singular. Nevertheless, the classical Levinson algorithm still breaks down. The look-ahead Algorithm 7.1 computes a solution with relative error `1.33e-14`.

9 Concluding Remarks

We studied formally biorthogonal polynomials (FBOPs) for bilinear forms induced by general Toeplitz matrices. We presented new recurrence relations that connect successive pairs in any given subsequence of all existing FBOPs. Based on these recursions, we proposed a

m	0	1	2	3	4	5
$\kappa(T_m)$	2.00e-1	1	1.63	9.45e+5	2.40e+6	3.61e+5
6	7	8	9	10	11	12
4.76e+6	3.60e+6	3.34e+2	1.71e+2	1.63e+1	4.02e+1	2.05e+1

Table 9: Condition number profile of the matrix in Example 4.3

	classical	look-ahead
Example 4.1	0.50112484863612	5.232908767834996e-15
Example 4.2	0.66589668180614	4.028860512358659e-14
Example 4.3	3.204212636311192e-10	7.089249323695304e-14

Table 10: Comparison of the relative errors for the three matrices in Example 4

m	0	1	2	3	4	5	6
$\kappa(T_m)$	1.09e+15	1.00	3.98e+14	6.53e+14	8.74e+14	1.81e+1	7.21

Table 11: Condition number profile of the matrix in Example 5

look-ahead algorithm for solving general Toeplitz systems. This procedure is an extension of the classical Levinson algorithm for strongly regular Toeplitz matrices. We stress that our look-ahead algorithm is different from other generalizations of the Levinson algorithm that have been proposed. Except for the procedure devised by Chan and Hansen [8], all these algorithms can only skip over exactly singular submatrices. The algorithm in [8] allows to skip over ill-conditioned submatrices; however, as we showed, the procedure can break down if two or more consecutive blocks of ill-conditioned submatrices occur. In contrast to these other proposed extensions of the Levinson algorithm, our look-ahead procedure skips over exactly singular, as well as ill-conditioned leading principal submatrices, and it can handle arbitrary consecutive blocks of ill-conditioned submatrices. We reported results of numerical experiments, which demonstrate that our look-ahead Levinson algorithm generates solutions of Toeplitz systems with ill-conditioned submatrices to nearly full accuracy.

We remark that similar techniques can be used to derive a look-ahead algorithm for solving general Hankel systems. A detailed description of such a look-ahead Hankel solver can be found in [15]. We stress that the Hankel case is fundamentally different from the Toeplitz case. The Hankel case is actually simpler in the sense that bilinear forms associated with Hankel matrices are always symmetric, and consequently one only has to deal with one sequence of *formally orthogonal polynomials* (FOPs), rather than two sequences of FBOPs as in the Toeplitz case. We note that FOPs associated with Hankel matrices are intimately connected with the nonsymmetric Lanczos process [34] for matrix computations; see, e.g., [14] and the references given there.

In future work, we intend to give a rigorous stability analysis of the look-ahead Levinson algorithm proposed in this paper. Furthermore, we plan to develop a software package with FORTRAN and MATLAB implementations of this algorithm, as well as the Hankel solver described in [15].

In recent years, various so-called *super-fast* algorithms were developed that solve Toeplitz systems with only $\mathcal{O}(n \log^2 n)$ operations, see, e.g., [1, 4, 5, 45]. Bunch [7] discussed the stability of some of these algorithms, and he pointed out that they are unstable in the case of general Toeplitz systems. It is natural to ask whether look-ahead techniques can also be used to enhance the stability of super-fast Toeplitz solvers. This question is addressed in a recent paper by Gutknecht [21] who devised two super-fast Toeplitz algorithms with look-ahead. However, no numerical results are given in [21], and it remains to be seen whether these algorithms can be turned into practical numerical procedures.

References

[1] G. S. Ammar and W. B. Gragg, Superfast solution of real positive definite Toeplitz systems, *SIAM J. Matrix Anal. Appl.* 9:61–76 (1988).

[2] E. H. Bareiss, Numerical solution of linear equations with Toeplitz and vector Toeplitz matrices, *Numer. Math.* 13:404–424 (1969).

[3] G. Baxter, Polynomials defined by a difference system, *J. Math. Anal. Appl.* 2:223–263 (1961).

[4] R. R. Bitmead and B. D. O. Anderson, Asymptotically fast solution of Toeplitz and related systems of linear equations, *Linear Algebra Appl.* 34:103–116 (1980).

[5] R. P. Brent, F. G. Gustavson, and D. Y. Y. Yun, Fast solution of Toeplitz systems of equations and computation of Padé approximants, *J. Algorithms* 1:259–295 (1980).

[6] A. Bultheel, *Laurent Series and their Padé Approximations*, Birkhäuser, Basel, 1987.

[7] J. R. Bunch, Stability of methods for solving Toeplitz systems of equations, *SIAM J. Sci. Stat. Comput.* 6:349–364 (1985).

[8] T. F. Chan and P. C. Hansen, A Stable Levinson Algorithm for General Toeplitz Systems, Technical Report CAM 90–11, University of California, Los Angeles, May 1990.

[9] T. F. Chan and P. C. Hansen, A look-ahead Levinson algorithm for indefinite Toeplitz systems, *SIAM J. Matrix Anal. Appl.* 13:490–506 (1992).

[10] G. Cybenko and C. F. Van Loan, Computing the minimum eigenvalue of a symmetric positive definite Toeplitz matrix, *SIAM J. Sci. Stat. Comput.* 7:123–131 (1986).

[11] P. Delsarte, Y. V. Genin, and Y. G. Kamp, A generalization of the Levinson algorithm for Hermitian Toeplitz matrices with any rank profile, *IEEE Trans. Acoust. Speech Signal Process.* ASSP-33:964–971 (1985).

[12] C. J. Demeure and L. L. Scharf, Linear statistical models for stationary sequences and related algorithms for Cholesky factorization of Toeplitz matrices, *IEEE Trans. Acoust. Speech Signal Process.* ASSP-35:29–42 (1987).

[13] J. Durbin, The fitting of time-series models, *Rev. Inst. Internat. Statist.* 28:233–243 (1960).

[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. Stat. Comput.* 14 (1993), to appear.

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

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

[17] M. J. C. Gover and S. Barnett, Inversion of Toeplitz matrices which are not strongly non-singular, *IMA J. Numer. Anal.* 5:101–110 (1985).

[18] P. R. Graves-Morris, The numerical calculation of Padé approximants, in *Padé Approximation and its Applications, Proceedings, 1979* (L. Wuytack, Ed.), Lecture Notes in Math. 765, Springer-Verlag, Berlin, 1979, pp. 231–245.

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

[20] G. Gueguen, Linear prediction in the singular case and the stability of eigen models, in *Proc. 1981 IEEE Int. Conf. Acoust., Speech, Signal Process.*, Atlanta, GA, pp. 881–885.

[21] M. H. Gutknecht, Stable Row Recurrences for the Padé Table and Generically Superfast Look-Ahead Solvers for Non-Hermitian Toeplitz Systems, IPS Research Report 92–14, ETH, Zürich, August 1992.

[22] M. H. Hayes and M. A. Clements, An efficient algorithm for computing Pisarenko's harmonic decomposition using Levinson's recursion, *IEEE Trans. Acoust. Speech Signal Process.* ASSP-34:485–491 (1986).

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

[24] Y. H. Hu and S.-Y. Kung, Toeplitz eigensystem solver, *IEEE Trans. Acoust. Speech Signal Process.* ASSP-33:1264–1271 (1985).

[25] I. S. Iohvidov, *Hankel and Toeplitz matrices and forms*, Birkhäuser, Boston, 1982.

[26] E. Jonckheere and C. Ma, Combined sequence of Markov parameters and moments in linear systems, *IEEE Trans. Automat. Control* AC-34:379–382 (1989).

[27] E. Jonckheere and C. Ma, Recursive partial realization from the combined sequence of Markov parameters and moments, *Linear Algebra Appl.* 122/123/124:565–590 (1989).

[28] M. Kac, W. L. Murdock, and G. Szegö, On the eigen-values of certain Hermitian forms, *J. Rat. Mech. Anal.* 2:767–800 (1953).

[29] T. Kailath, A view of three decades of linear filtering theory, *IEEE Trans. Inform. Theory* IT-20:146–181 (1974).

[30] T. Kailath, *Linear Systems*, Prentice-Hall, Englewood Cliffs, 1980.

[31] T. Kailath, A theorem of I. Schur and its impact on modern signal processing, in *I. Schur Methods in Operator Theory and Signal Processing* (I. Gohberg, Ed.), Operator Theory Adv. Appl. 18, Birkhäuser, Basel, 1986, pp. 9–30.

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

[33] J. D. E. Konhauser, Some properties of biorthogonal polynomials, *J. Math. Anal. Appl.* 11:242–260 (1965).

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

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

[36] J. Makhoul, Linear prediction: a tutorial review, *Proc. IEEE* 63:561–580 (1975).

[37] V. F. Pisarenko, The retrieval of harmonics from a covariance function, *Geophys. J. Royal Astron. Soc.* 33:347–366 (1973).

[38] J. Rissanen, Algorithms for triangular decomposition of block Hankel and Toeplitz matrices with application to factoring positive matrix polynomials, *Math. Comp.* 27:147–154 (1973).

[39] I. Schur, Über Potenzreihen, die im Innern des Einheitskreises beschränkt sind, Parts I and II, *J. Reine Angew. Math.* 147:205–232 (1917) and 148:122–145 (1918).

[40] Y. Sugiyama, An algorithm for solving discrete-time Wiener-Hopf equations based upon Euclid's algorithm, *IEEE Trans. Inform. Theory* IT-32:394–409 (1986).

[41] D. R. Sweet, Numerical Methods for Toeplitz Matrices, Ph.D. thesis, University of Adelaide, Australia, 1982.

[42] D. R. Sweet, The use of pivoting to improve the numerical performance of Toeplitz solvers, In *Advanced Algorithms and Architectures for Signal Processing* (J. M. Speiser, Ed.), Proc. SPIE 696, 1986, pp. 8–18.

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

[44] W. F. Trench, Numerical solution of the eigenvalue problem for Hermitian Toeplitz matrices, *SIAM J. Matrix Anal. Appl.* 10:135–146 (1989).

[45] E. E. Tyrtyshnikov, New cost-effective and fast algorithms for special classes of Toeplitz systems, Sov. J. Numer. Anal. Math. Modelling 3:63–76 (1988).

[46] H. van Rossum, Formally biorthogonal polynomials, in *Padé Approximation and its Applications, Proceedings, 1980* (M. G. de Bruin and H. van Rossum, Eds.), Lecture Notes in Math. 888, Springer-Verlag, Berlin, 1981, pp. 341–351.

[47] C. J. Zarowski, Schur algorithms for Hermitian Toeplitz, and Hankel matrices with singular leading principal submatrices, *IEEE Trans. Signal Process.* 39:2464–2480 (1991).

[48] S. Zohar, Toeplitz matrix inversion: the algorithm of W. F. Trench, *J. Assoc. Comput. Mach.* 16:592–601 (1969).

[49] S. Zohar, The solution of a Toeplitz set of linear equations, *J. Assoc. Comput. Mach.* 21:272–276 (1974).