A look-ahead Bareiss algorithm for general Toeplitz matrices

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

Summary. The Bareiss algorithm is one of the classical fast solvers for systems of linear equations with Toeplitz coefficient matrices. The method takes advantage of the special structure, and it computes the solution of a Toeplitz system of order N with only $\mathcal{O}(N^2)$ arithmetic operations, instead of $\mathcal{O}(N^3)$ operations. However, the original Bareiss algorithm requires that all leading principal submatrices be nonsingular, and the algorithm is numerically unstable if singular or ill-conditioned submatrices occur. In this paper, an extension of the Bareiss algorithm to general Toeplitz systems is presented. Using look-ahead techniques, the proposed algorithm can skip over arbitrary blocks of singular or ill-conditioned submatrices, and at the same time, it still fully exploits the Toeplitz structure. Implementation details and operations counts are given, and numerical experiments are reported. We also discuss special versions of the proposed look-ahead Bareiss algorithm for Hermitian indefinite Toeplitz systems and banded Toeplitz systems.

Mathematics Subject Classification (1991): 65F05, 15A57, 42C05

1 Introduction

A matrix is called *Toeplitz* if its entries are constant along each diagonal. Thus a general square Toeplitz matrix of order $N+1$ is of the form

$$T = [\, t_{i-j} \,]_{i,j=0,1,\ldots,N} = \begin{bmatrix} t_0 & t_{-1} & t_{-2} & \cdots & t_{-N} \\ t_1 & t_0 & t_{-1} & \cdots & t_{-(N-1)} \\ t_2 & t_1 & t_0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & t_{-1} \\ t_N & t_{N-1} & \cdots & t_1 & t_0 \end{bmatrix}, \tag{1.1}$$

where $t_j \in \mathbb{C}$, $j = -N, -N+1, \ldots, N$. In this paper, we are concerned with the solution of systems of linear equations

$$T\,x = b, \tag{1.2}$$

where T is a general Toeplitz matrix (1.1). Such Toeplitz systems arise in numerous important applications, including time-series analysis, linear prediction, image processing, statistics, probability theory, solution of integral equations, and Padé approximation; see [7, 16, 20] and the references given there.

1.1 *Classical fast Toeplitz solvers*

There are several well-known *fast* Toeplitz solvers that take advantage of the special structure and require only $\mathcal{O}(N^2)$ arithmetic operations for the solution of a nonsingular system (1.2), instead of $\mathcal{O}(N^3)$ operations. Fast Toeplitz solvers can be divided into two categories. *Levinson-type* algorithms are based on an *inverse triangular factorization* of T of the form

$$V^T T U = D, \tag{1.3}$$

where U and V are unit upper triangular matrices and D is a nonsingular diagonal matrix. The classical Levinson algorithm [28] and its variants [12, 37, 39, 40] all belong to this class. On the other hand, *Schur-type* algorithms are based on a *direct triangular factorization* of T of the form

$$T = S D^{-1} R^T, \tag{1.4}$$

where R and S are lower triangular matrices and D is the diagonal matrix from (1.3). The first fast Toeplitz solver of this type was proposed by Bareiss [2]. Later on, closely related algorithms were presented in [27, 29, 31]. Today, Toeplitz solvers based on (1.4) are usually referred to as Schur-type algorithms because of their intimate connection [1, 25] with Schur's seminal work [32]. We remark that the matrices R and S in (1.4) are unique only up to a scaling of their columns. It turns out to be convenient to scale R and S so that they both have the same diagonal elements as D. In particular, the triangular matrices in (1.3) and (1.4) are then connected as follows:

$$R^T = D U^{-1} \quad \text{and} \quad S = V^{-T} D.$$

1.2 *Levinson-type versus Schur-type algorithms*

There are a number of differences between Levinson-type and Schur-type algorithms, and both approaches have their merits. Levinson-type algorithms require the computation of SAXPYs[1] *and* of inner products of long vectors, while Schur-type algorithms involve *only* SAXPYs. Consequently, on parallel machines with $\mathcal{O}(N)$ processors, Schur-type methods require only $\mathcal{O}(N)$ computing time, as compared to $\mathcal{O}(N \log N)$ computing time for Levinson-type methods. Actual implementations of Schur-type algorithms on a linear array of $\mathcal{O}(N)$ processors are described in [6, 10, 26]. If T is a band matrix, then the triangular matrices R and S in (1.4) are also banded, while the factors U and V in (1.3) are in general full upper triangular matrices. As a result, if the Toeplitz matrix T is banded, then Schur-type algorithms—unlike Levinson-type methods—can take advantage of this additional structure. Finally, a recent analysis [5] of the Bareiss algorithm for the special case of symmetric positive definite Toeplitz systems suggests that Schur-type methods have slightly better stability properties than Levinson-type methods. On the other hand, the actual computation of the solution of (1.2) from the factorization (1.3), respectively (1.4), is somewhat simpler for Levinson-type algorithms, than for Schur-type methods. Furthermore, in the case of Levinson-type algorithms, as a by-product, one obtains the solutions of all leading principal subsystems of (1.2); this is important in certain applications where the solutions of these subsystems also yield useful information.

1.3 *Fast solvers for general Toeplitz systems*

Unfortunately, not every matrix T admits a decomposition of the type (1.3) or (1.4). In fact, for a nonsingular Toeplitz matrix (1.1), factorizations of the form (1.3) and (1.4) exist if, and only if, T is *strongly regular*, i.e., *all* its leading principal submatrices T_n, $n = 0, 1, \ldots, N$, are nonsingular. Here, T_n is the $(n+1) \times (n+1)$ Toeplitz matrix defined by

$$T_n := [\, t_{i-j} \,]_{i,j=0,1,\ldots,n} \,. \tag{1.5}$$

Indeed, the fast Toeplitz solvers require that T be strongly regular, and if one encounters a singular submatrix T_n, then a *breakdown*—triggered by division by 0—occurs in all algorithms. Furthermore, in finite-precision arithmetic, if one encounters a nonsingular, yet ill-conditioned submatrix T_n, then a *near-breakdown*, i.e., division by a number close to 0, occurs. Such near-breakdowns then usually cause numerical instabilities in subsequent steps of the algorithm, resulting in inaccurately computed solutions of (1.2). Of course, the very same problem arises in the factorization of general matrices by means of Gaussian elimination, and there the remedy is to use pivoting. The point is that standard pivoting techniques cannot be incorporated into any of the fast Toeplitz solvers since pivoting would destroy the special Toeplitz structure. We stress that—except for special cases, such as Hermitian positive definite matrices—it cannot be excluded that general Toeplitz matrices T indeed do have singular or ill-conditioned leading principal submatrices, even if T itself is well conditioned.

[1]A SAXPY operation is the computation of $z := x + \alpha y$, where x and y are vectors of the same length and α is a scalar.

The problem of how to avoid breakdowns of the fast Toeplitz solvers by skipping over singular submatrices has been studied extensively, and various such algorithms have been proposed; see [11, 17, 18, 21, 22, 30, 34, 38] and the references given there. In finite-precision arithmetic however, a numerically stable solver for general Toeplitz matrices must avoid exact breakdowns as well as near-breakdowns by skipping over exactly singular as well as nonsingular, but ill-conditioned submatrices. Surprisingly, it was only recently that extensions of the fast Toeplitz solvers with this property have been developed. Sweet [35] proposed to incorporate some form of restricted pivoting into the Bareiss algorithm, and he showed how to restore the Toeplitz structure after pivoting. However, the resulting algorithm is very complicated, and it is not clear if his approach is viable for Toeplitz matrices with consecutive blocks of singular or ill-conditioned submatrices. Chan and Hansen [8] were the first to devise a *look-ahead* variant of the Levinson algorithm. The basic idea is to relax the inverse triangular decomposition (1.3) slightly and to compute an inverse block-factorization of the form (1.3) with D a block-diagonal matrix, instead of a scalar diagonal matrix. More precisely, each block in D of size $h > 1$ just corresponds to the occurrence of $h-1$ consecutive singular or ill-conditioned submatrices of T. The size of such a block is usually not known beforehand, and it is determined in the course of the algorithm, using so-called look-ahead techniques. Another look-ahead extension of the classical Levinson algorithm to general Toeplitz matrices was recently proposed by Freund and Zha [16]. Although the look-ahead Levinson algorithm in [16] is also based on an inverse block-factorization (1.3), this algorithm and the one in [8] differ considerably; for instance, the block steps and the look-ahead strategy are different. Furthermore, in the derivation of their algorithm, Freund and Zha make use of the intimate connection between fast Toeplitz solvers and recurrence relations for *formally biorthogonal polynomials* (FBOPs) associated with Toeplitz matrices. Due to the use of these recursions for FBOPs, the resulting look-ahead Levinson algorithm still fully exploits the Toeplitz structure, even within the block steps. Finally, we remark that, in [19], Gutknecht developed various look-ahead algorithms for general Toeplitz matrices. However, none of the methods in [19] is a direct extension of any of the classical fast Toeplitz solvers.

In this paper, we present a Schur-type counterpart to the look-ahead Levinson algorithm in [16]. As in [16], we make use of the connection between fast Toeplitz solvers and FBOPs, and our algorithm is based on the recurrence relations for FBOPs given in [16]. The resulting algorithm is a look-ahead extension of the Bareiss algorithm for strongly regular matrices to general Toeplitz matrices T, and it reduces to the standard Bareiss algorithm if no look-ahead steps are taken. The proposed procedure is based on a *block-triangular factorization* of T of the form (1.4) where R and S are lower block-triangular matrices and D is a block-diagonal matrix. Actually, we will see that our algorithm can also handle singular matrices T; in this case, it generates a block-triangular factorization (1.4) with D^{-1} replaced by the pseudoinverse of D.

1.4 *Outline*

The organization of the remainder of this article is as follows. In Section 1.5, some notation is introduced. In Section 2, we recall a few results from [16] on FBOPs associated with general Toeplitz matrices, and on their connection with inverse block-factorizations (1.3). In Section 3, we show that these FBOPs also yield a direct block-triangular factorization of the type (1.4), and we derive recurrence relations for the entries of the matrices R, S, and D in (1.4). In Section 4, we formulate our look-ahead Bareiss algorithm for computing block-triangular factorizations (1.4) of general square Toeplitz matrices, and we give operation counts. In Section 5, we consider two special cases, namely Hermitian indefinite Toeplitz matrices and banded Toeplitz matrices. In Section 6, we discuss the actual computation of the solution of nonsingular Toeplitz systems (1.2) from the block-triangular factorization generated by the proposed look-ahead Bareiss algorithm. In Section 7, we present results of numerical experiments with Toeplitz systems that have various kinds of ill-conditioned submatrices. Finally, we make some concluding remarks in Section 8.

1.5 *Notation*

Throughout this paper, all matrices are allowed to have real or complex entries. As in (1.1), we always count rows and columns starting with 0; for example, a general $(n+1)\times(m+1)$ matrix M is of the form

$$M = [\, m_{ij} \,]_{\substack{i=0,1,\dots,n \\ j=0,1,\dots,m}}\,, \quad \text{where} \quad m_{ij} \in \mathbb{C} \quad \text{for all} \quad i,j.$$

As usual, $M^T := [\, m_{ji} \,]$, $\overline{M} := [\, \overline{m}_{ij} \,]$, and $M^H := \overline{M}^T$ denote the transpose, complex conjugate, and conjugate transpose, respectively, of a matrix $M = [\, m_{ij} \,]$. 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:

$$\kappa(M) := \begin{cases} 1/|M|, & \text{if } h = 1, \\ \|M\| \cdot \|M^{-1}\|, & \text{if } h > 1. \end{cases} \tag{1.6}$$

Whenever we call a square matrix *ill-conditioned*, it is with respect to the condition number (1.6). We denote by

$$Z_n := [\, \delta_{i,j+1} \,]_{i,j=0,1,\dots,n-1}\,, \quad \text{where} \quad \delta_{ij} := \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{if } i \neq j, \end{cases} \tag{1.7}$$

the $n \times n$ down-shift matrix, by $I_n := [\, \delta_{ij} \,]_{i,j=0,1,\dots,n-1}$ the $n\times n$ identity matrix, by $0_{n\times m}$ the $n \times m$ zero matrix, and by $0_n \in \mathbb{R}^n$ the zero vector of length n. We will drop subscripts and simply write Z, I, or 0 whenever the actual dimensions are apparent from the context. We denote by

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

the set of all complex polynomials of degree at most n. For each polynomial $\varphi \in \mathcal{P}_n$ of exact degree n, we define its *reverse* $\widehat{\varphi}$ by

$$\widehat{\varphi}(\lambda) \equiv \lambda^n \varphi(1/\lambda).$$

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

Finally, throughout this paper, T is always assumed to be a general $(N+1) \times (N+1)$ Toeplitz matrix (1.1) with entries $t_j \in \mathbb{C}$, $j = -N, -N+1, \ldots, N$. We stress that—unless stated otherwise—T can be singular or nonsingular. As in (1.5), we always denote by T_n, $n = 0, 1, \ldots, N$, the leading principal submatrices of T.

2 Formally biorthogonal polynomials

In this section, we recall some results on formally biorthogonal polynomials that will be used later on in the derivation of our look-ahead Bareiss algorithms. We refer the reader to [13, 16] for proofs of these results. From now on, we abbreviate *formally biorthogonal polynomial* by FBOP.

2.1 *The Toeplitz bilinear form*

We associate with the Toeplitz matrix T a complex-valued *bilinear form*

$$\langle \cdot, \cdot \rangle : \mathcal{P}_N \times \mathcal{P}_N \longmapsto \mathbb{C} \tag{2.1}$$

by setting

$$\langle \psi, \varphi \rangle := v^T T u \tag{2.2}$$

for any pair of polynomials

$$\varphi(\lambda) \equiv [1 \quad \lambda \quad \cdots \quad \lambda^N]\, u, \quad \psi(\lambda) \equiv [1 \quad \lambda \quad \cdots \quad \lambda^N]\, v, \quad u, v \in \mathbb{C}^{N+1}, \tag{2.3}$$

of degree at most N. We will refer to (2.1) as the *Toeplitz bilinear form*. Note that, for the special polynomials $\varphi(\lambda) \equiv \lambda^j$ and $\psi(\lambda) \equiv \lambda^i$, we have

$$\langle \lambda^i, \lambda^j \rangle = t_{i-j}, \quad i, j = 0, 1, \ldots, N, \tag{2.4}$$

and hence the entries of T are just the *moments* of the Toeplitz bilinear form. Using (2.4) and the bilinearity of (2.1), one readily verifies that

$$\langle \lambda\psi, \lambda\varphi \rangle = \langle \psi, \varphi \rangle \quad \text{for all} \quad \varphi, \psi \in \mathcal{P}_{N-1}. \tag{2.5}$$

The "shift" property (2.5) of the Toeplitz bilinear form will be used frequently in this paper.

There are several possible approaches to fast algorithms for general Toeplitz matrices. Here, we will use the one based on polynomials that are biorthogonal with respect to the Toeplitz bilinear form (2.1). We will construct two sequences of polynomials

$$\{\varphi_n\}_{n=0}^N \quad \text{and} \quad \{\psi_n\}_{n=0}^N, \tag{2.6}$$

where, for each n, $\varphi_n = \lambda^n + \cdots$ and $\psi_n = \lambda^n + \cdots$ are *monic* polynomials of degree n. The monic polynomial φ_n is called a *regular right* FBOP (of degree n) if

$$\langle \psi, \varphi_n \rangle = 0 \quad \text{for all} \quad \psi \in \mathcal{P}_{n-1}, \tag{2.7}$$

and if φ_n is uniquely determined by (2.7). The monic polynomial ψ_n is called a *regular left* FBOP (of degree n) if

$$\langle \psi_n, \varphi \rangle = 0 \quad \text{for all} \quad \varphi \in \mathcal{P}_{n-1}, \tag{2.8}$$

and if ψ_n is uniquely determined by (2.8). Note that, for $n = 0$, the conditions (2.7) and (2.8) are void, and thus

$$\varphi_0 = 1 \quad \text{and} \quad \psi_0 = 1 \tag{2.9}$$

are always regular FBOPs of degree 0.

Part a) of Lemma 2.1 below shows that regular FBOPs of degree $n \geq 1$ do not exist if T_{n-1} is singular. Furthermore, if T_{n-1} is nonsingular, but ill-conditioned, then it would not be numerically safe to compute regular FBOPs of degree n. For such cases, we relax the biorthogonality relations (2.7) and (2.8) slightly. The monic polynomial φ_n is called a *right m-quasi*-FBOP (of degree n) if $0 \leq m < n$ and

$$\langle \psi, \varphi_n \rangle = 0 \quad \text{for all} \quad \psi \in \mathcal{P}_{m-1}. \tag{2.10}$$

The monic polynomial ψ_n is called a *left m-quasi*-FBOP (of degree n) if $0 \leq m < n$ and

$$\langle \psi_n, \varphi \rangle = 0 \quad \text{for all} \quad \varphi \in \mathcal{P}_{m-1}. \tag{2.11}$$

We remark that, for $m = 0$, the conditions (2.10) and (2.11) are void, and thus all monic polynomials φ_n and ψ_n of degree $n \geq 1$ are 0-quasi FBOPs.

The following lemma gives conditions for the existence of regular FBOPs and m-quasi-FBOPs.

Lemma 2.1 ([13, Section 3.2]) *Let $1 \leq n \leq N$.*

a) *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.

b) *Let $1 \leq m < n$. If T_{m-1} is nonsingular, then right and left m-quasi*-FBOP*s φ_n and ψ_n of degree n exist.*

2.2 *Recurrence relations for* FBOP*s*

If T is strongly regular then, in view of part a) of Lemma 2.1, regular FBOPs φ_n and ψ_n of degree n exist for all $n = 0, 1, \ldots, N$. Furthermore, these polynomials satisfy simple recurrence relations, namely the celebrated Szegő recursions [3, 36]. In [16], we derived generalized recurrences for constructing regular FBOPs and quasi-FBOPs associated with arbitrary Toeplitz matrices. In this section, we review these generalized Szegő recursions.

By part a) of Lemma 2.1 and (2.9), the set

$$\mathcal{I} := \{0\} \cup \{\, n \mid 1 \leq n \leq N \text{ and } T_{n-1} \text{ is nonsingular} \,\} \tag{2.12}$$

consists of all integers $0 \leq n \leq N$ for which regular FBOPs of degree n exist. Let

$$\{n_k\}_{k=0}^{K} \subseteq \mathcal{I}, \quad \text{where} \quad 0 =: n_0 < n_1 < \cdots < n_k < n_{k+1} < \cdots < n_K, \tag{2.13}$$

be an *arbitrary*, but fixed, monotone subsequence of $\mathcal{I}$ that includes $n_0 = 0$. It will be convenient to set $n_{K+1} := N + 1$ and

$$h_k := n_{k+1} - n_k \quad \text{for all} \quad k = 0, 1, \ldots, K. \tag{2.14}$$

Note that, since $n_0 = 0$, we have

$$h_0 = n_1. \tag{2.15}$$

We refer to (2.13) as a sequence of *regular indices*, since, by (2.12) and (2.13), all polynomials in the corresponding subsequences

$$\{\varphi_{n_k}\}_{k=0}^{K} \quad \text{and} \quad \{\psi_{n_k}\}_{k=0}^{K} \tag{2.16}$$

of (2.6) can be constructed as regular FBOPs. In order to complete (2.16) to sequences (2.6), for each $k \in \{\, 0, 1, \ldots, K \,\}$ with $h_k > 1$ we need to add polynomials

$$\varphi_n \quad \text{and} \quad \psi_n \quad \text{for all} \quad n = n_k + 1, n_k + 2, \ldots, n_{k+1} - 1. \tag{2.17}$$

We will construct the polynomials (2.17) as n_k-quasi-FBOPs; part b) of Lemma 2.1 shows that this is always possible. We remark that such n_k-quasi-FBOPs are not uniquely determined. Here, we choose the polynomials (2.17) so that the recurrence relations (2.24)–(2.27) below are as short as possible.

In the sequel, the following notation will be used. For each $k = \{\, 0, 1, \ldots, K \,\}$, we define $h_k \times h_k$ matrices

$$\begin{aligned} F_k &:= \left[\, f_{lm}^{(k)} \,\right]_{l,m=0,1,\ldots,h_k-1}, \quad \text{where} \quad f_{lm}^{(k)} := \langle \psi_{n_k+m}, \lambda^{n_k+l} \rangle, \\ \text{and} \quad G_k &:= \left[\, g_{lm}^{(k)} \,\right]_{l,m=0,1,\ldots,h_k-1}, \quad \text{where} \quad g_{lm}^{(k)} := \langle \lambda^{n_k+l}, \varphi_{n_k+m} \rangle, \end{aligned} \tag{2.18}$$

and, if $h_k > 1$, we define vectors

$$f_k := \left[\, f_{l,h_k-1}^{(k)} \,\right]_{l=0,1,\ldots,h_k-2} \quad \text{and} \quad g_k := \left[\, g_{l,h_k-1}^{(k)} \,\right]_{l=0,1,\ldots,h_k-2}. \tag{2.19}$$

Note that the entries of the vectors f_k and g_k are just the first $h_k - 1$ elements of the last columns of F_k and G_k, respectively. We denote by $(\eta)_m$, $m = 0, 1, \ldots, h-1$, the mth element of a vector $\eta \in \mathbb{C}^h$.

After these preliminaries, we can now state our generalized Szegő recursions for constructing the polynomials (2.16) and (2.17).

Algorithm 2.2 (Generalized Szegő Recursions [16, Section 4.2].)

0) Set

$$\varphi_n = \lambda^n \quad \text{and} \quad \psi_n = \lambda^n \quad \text{for all} \quad n = 0, 1, \dots, n_1 - 1. \tag{2.20}$$

For $k = 0, 1, \dots, K-1$ do:

1) Set $h_k = n_{k+1} - n_k$.

2) For $n = n_{k+1} - 1, n_{k+1}, \dots, n_{k+2} - 2$ do:

Set

$$\rho_n = \langle 1, \lambda\varphi_n \rangle, \quad \tau_n = \langle \lambda\psi_n, 1 \rangle, \tag{2.21}$$

$$\alpha_n = F_k^{-1} \begin{bmatrix} 0_{h_k-1} \\ \rho_n \end{bmatrix}, \quad \beta_n = G_k^{-1} \begin{bmatrix} 0_{h_k-1} \\ \tau_n \end{bmatrix}, \tag{2.22}$$

and, if $h_k > 1$,

$$\mu_k = G_k^{-1} \begin{bmatrix} 0 \\ g_k \end{bmatrix}, \quad \nu_k = F_k^{-1} \begin{bmatrix} 0 \\ f_k \end{bmatrix}. \tag{2.23}$$

If $n + 1 = n_{k+1}$ and $h_k > 1$, then set

$$\varphi_{n+1} = \lambda\varphi_n - \sum_{m=0}^{h_k-1} \lambda^{h_k-1-m} \widehat{\psi}_{n_k+m} (\alpha_n)_m - \sum_{m=0}^{h_k-1} \varphi_{n_k+m} (\mu_k)_m, \tag{2.24}$$

$$\psi_{n+1} = \lambda\psi_n - \sum_{m=0}^{h_k-1} \lambda^{h_k-1-m} \widehat{\varphi}_{n_k+m} (\beta_n)_m - \sum_{m=0}^{h_k-1} \psi_{n_k+m} (\nu_k)_m. \tag{2.25}$$

Otherwise, set

$$\varphi_{n+1} = \lambda\varphi_n - \sum_{m=0}^{h_k-1} \lambda^{h_k-1-m} \widehat{\psi}_{n_k+m} (\alpha_n)_m, \tag{2.26}$$

$$\psi_{n+1} = \lambda\psi_n - \sum_{m=0}^{h_k-1} \lambda^{h_k-1-m} \widehat{\varphi}_{n_k+m} (\beta_n)_m. \tag{2.27}$$

Remark. Suppose that T is strongly regular and the regular indices in (2.13) are chosen as $n_k = k$, $k = 0, 1, \dots, N$. Then, by (2.14), $h_k = 1$ for all k, and in Algorithm 2.2 the polynomials φ_{n+1} and ψ_{n+1} are always generated by means of the recurrences (2.26) and (2.27). Furthermore, (2.26), (2.27), and (2.22) now reduce to the relations

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

which are just the classical Szegő recursions.

In [16], we showed that the matrices F_k and G_k whose inverses appear in (2.22) and (2.23) are indeed nonsingular, and that the polynomials generated by Algorithm 2.2 satisfy the required biorthogonality conditions (2.7) and (2.8), respectively (2.10) and (2.11), for regular FBOPs, respectively n_k-quasi-FBOPs. We summarize these results in the following theorem.

Theorem 2.3 ([16, Section 4.2]) *Let $\{n_k\}_{k=0}^{K}$ be a given sequence of regular indices* (2.13). *Let $\{\varphi_n\}_{n=0}^{N}$ and $\{\psi_n\}_{n=0}^{N}$ be the sequences of monic polynomials generated by Algorithm* 2.2. *Then:*

a) *The matrices F_k and G_k are nonsingular for all $k = 0, 1, \ldots, K-1$.*

b) *For all $k = 0, 1, \ldots, K$ and all $n = n_k, n_k + 1, \ldots, n_{k+1} - 1$:*

$$\langle \psi, \varphi_n \rangle = \langle \psi_n, \varphi \rangle = 0 \quad \textit{for all} \quad \psi, \varphi \in \mathcal{P}_{n_k - 1}. \tag{2.28}$$

In particular, the polynomials φ_n and ψ_n are regular FBOP*s if $n = n_k$, and n_k-quasi-*FBOP*s if $n_k < n < n_{k+1}$.*

2.3 *Inverse block-factorization of Toeplitz matrices*

We now denote by $u_{jn}, v_{jn} \in \mathbb{C}$, $j = 0, 1, \ldots, n-1$, the coefficients of the nth pair of monic polynomials

$$\varphi_n(\lambda) \equiv \lambda^n + \sum_{j=0}^{n-1} u_{jn} \lambda^j \quad \text{and} \quad \psi_n(\lambda) \equiv \lambda^n + \sum_{j=0}^{n-1} v_{jn} \lambda^j. \tag{2.29}$$

We denote by

$$U := [\, u_{jn} \,]_{j,n=0,1,\ldots,N} = \begin{bmatrix} 1 & u_{01} & \cdots & u_{0n} \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & u_{n-1,n} \\ 0 & \cdots & 0 & 1 \end{bmatrix} \tag{2.30}$$

and

$$V := [\, v_{jn} \,]_{j,n=0,1,\ldots,N} = \begin{bmatrix} 1 & v_{01} & \cdots & v_{0n} \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & v_{n-1,n} \\ 0 & \cdots & 0 & 1 \end{bmatrix} \tag{2.31}$$

the unit upper triangular matrices whose nth columns just consist of the coefficients of (2.29). Furthermore, we define a block-diagonal matrix D by setting

$$\begin{aligned} D &:= \operatorname{diag}(D_0, D_1, \ldots, D_K), \\ &\text{where} \quad D_k := [\, \langle \psi_{n_k+l}, \varphi_{n_k+m} \rangle \,]_{l,m=0,1,\ldots,h_k-1}, \quad k = 0, 1, \ldots, K. \end{aligned} \tag{2.32}$$

Next, we show that the matrices (2.30)–(2.32) define an inverse block-factorization of the Toeplitz matrix T.

Theorem 2.4 *Let $\{n_k\}_{k=0}^{K}$ be a given sequence of regular indices (2.13). Let $\{\varphi_n\}_{n=0}^{N}$ and $\{\psi_n\}_{n=0}^{N}$ be the sequences of monic polynomials generated by Algorithm 2.2. Then, the Toeplitz matrix T has the following inverse block-factorization :*

$$V^T T U = D, \tag{2.33}$$

where U and V are the unit upper triangular matrices defined by (2.29)–(2.31), and D is the block-diagonal matrix given by (2.32). Moreover, in (2.32), the blocks

$$D_k \quad \text{are nonsingular for all} \quad k = 0, 1, \ldots, K-1, \tag{2.34}$$

and

$$D_K \quad \text{is nonsingular} \quad \Longleftrightarrow \quad T \quad \text{is nonsingular.} \tag{2.35}$$

Proof. By (2.28), we have, for all $j, n = 0, 1, \ldots, N$,

$$\langle \psi_j, \varphi_n \rangle = \begin{cases} \langle \psi_{n_k+l}, \varphi_{n_k+m} \rangle, & \text{if } j = n_k + l,\ n = n_k + m,\ 0 \leq l, m < h_k, \\ 0, & \text{otherwise.} \end{cases} \tag{2.36}$$

Using (2.1)–(2.3), we now rewrite the biorthogonality relations (2.36) in terms of the Toeplitz matrix T and the coefficients of the polynomials (2.29). The result is the desired factorization (2.33).

Since U and V are unit upper triangular, the decomposition (2.33) implies that

$$\det T_{n_{k+1}-1} = \left(\det T_{n_k-1}\right)(\det D_k) \quad \text{for all} \quad k = 0, 1, \ldots, K, \tag{2.37}$$

where, for $k = 0$, we set $\det T_{-1} := 1$. Note that, by (2.12) and (2.13), we have $\det T_{n_k-1} \neq 0$ for all $k = 0, 1, \ldots, K$. Also, recall that $T_{n_{K+1}-1} = T_N = T$. Therefore, the statements (2.34) and (2.35) follow immediately from (2.37). □

Remark. By rewriting the recurrence relations for FBOPs in Algorithm 2.2 in terms of the matrices U, V, and D, one obtains simple recursions for computing the inverse block-factorization (2.33). The resulting procedure is just the look-ahead Levinson algorithm proposed in [16].

We stress that the triangular matrices U and V in (2.33) are neither Toeplitz, nor do they have any other similar structure. However, certain principal submatrices of U and V still exhibit a Toeplitz structure. In Lemma 2.5 below, we give a precise statement of this fact for the case of the matrix U; this result will be used in Section 3.3. First, we introduce some notation. For each $k \in \{1, 2, \ldots, K\}$, we define a vector $w_k \in \mathbb{C}^{h_k}$ and an $h_k \times h_k$ matrix Y_k as follows:

$$\begin{bmatrix} 1 & w_k^T \\ 0_{h_k} & Y_k \end{bmatrix} := [\, u_{n_k-1+l, n_k-1+m} \,]_{l,m=0,1,\ldots,h_k}. \tag{2.38}$$

For $k = 0$, we set

$$Y_0 := [\, u_{lm} \,]_{l,m=0,1,\ldots,h_0-1}. \tag{2.39}$$

Note that the matrix (2.38) is the principal submatrix of U that lies in the rows and columns of U indexed by $n_k - 1, n_k, \ldots, n_{k+1} - 1$. Furthermore, Y_k is the principal submatrix of U that just corresponds to the kth block D_k of the block-diagonal matrix D in (2.32).

Lemma 2.5 *The submatrices* (2.39) *and* (2.38) *of U have the following properties*:

a) *$Y_0 = I_{h_0}$ is the $h_0 \times h_0$ identity matrix.*

b) *For all $k = 1, 2, \ldots, K$, the submatrix* (2.38) *is a unit upper triangular Toeplitz matrix*:

$$\begin{bmatrix} 1 & w_k^T \\ 0 & Y_k \end{bmatrix} = \begin{bmatrix} 1 & \omega_1^{(k)} & \omega_2^{(k)} & \cdots & \omega_{h_k}^{(k)} \\ 0 & 1 & \omega_1^{(k)} & \cdots & \omega_{h_k-1}^{(k)} \\ 0 & 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \omega_1^{(k)} \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix}. \tag{2.40}$$

Proof. Recall from (2.29) and (2.30) that the entries of the matrix U are just the coefficients of the polynomials $\{\varphi_n\}_{n=0}^N$. Thus, part a) is an immediate consequence of (2.20); note that, by (2.15), $h_0 = n_1$. Similarly, by means of the recurrence relations (2.24), respectively (2.26), one readily verifies the Toeplitz structure of the submatrix (2.40) of U. ▯

3 Block-triangular factorization of Toeplitz matrices

In this section, we show that the polynomials generated by Algorithm 2.2 also define a block-triangular factorization of the Toeplitz matrix T, and we derive recurrence relations for the elements of this decomposition.

3.1 *The factorization*

We now use the polynomials $\{\varphi_n\}_{n=0}^N$ and $\{\psi_n\}_{n=0}^N$ from Algorithm 2.2 to define the following matrices:

$$\begin{aligned} R &:= [\, r_{jn} \,]_{j,n=0,1,\ldots,N}\,, \quad \text{where} \quad r_{jn} := \langle \psi_n, \lambda^j \rangle, \\ S &:= [\, s_{jn} \,]_{j,n=0,1,\ldots,N}\,, \quad \text{where} \quad s_{jn} := \langle \lambda^j, \varphi_n \rangle. \end{aligned} \tag{3.1}$$

Moreover, we denote by $D^\dagger$ the *pseudoinverse* of the block-diagonal matrix D in (2.32); see, e.g., [33, Section 4.8.5] for basic properties of the pseudoinverse. In view of (2.34), $D^\dagger$ is given by

$$D^\dagger = \operatorname{diag}\left(D_0^{-1}, D_1^{-1}, \ldots, D_{K-1}^{-1}, D_K^\dagger \right). \tag{3.2}$$

After these preparations, our main result on the bock-triangular factorization of Toeplitz matrices can be stated as follows.

Theorem 3.1 *Let* $\{n_k\}_{k=0}^K$ *be a given sequence of regular indices* (2.13). *Let* $\{\varphi_n\}_{n=0}^N$ *and* $\{\psi_n\}_{n=0}^N$ *be the sequences of monic polynomials generated by Algorithm* 2.2, *and let* R *and* S *be the corresponding matrices defined by* (3.1). *Then:*

a) *The matrices* R *and* S *are lower block-triangular:*

$$R = \begin{bmatrix} F_0 & 0 & \cdots & 0 \\ \star & F_1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \star & \cdots & \star & F_K \end{bmatrix} \quad \text{and} \quad S = \begin{bmatrix} G_0 & 0 & \cdots & 0 \\ \star & G_1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \star & \cdots & \star & G_K \end{bmatrix}, \tag{3.3}$$

with $h_k \times h_k$ *diagonal blocks* F_k *and* G_k, $k = 0, 1, \ldots, K$, *given by* (2.18).

b) *The matrices* R *and* S *and the matrices* U, V, *and* D *defined by* (2.29)–(2.32) *are connected as follows:*

$$S = TU = V^{-T}D \quad \text{and} \quad R^T = V^T T = DU^{-1}. \tag{3.4}$$

c) *The Toeplitz matrix* T *has the following inverse block-factorization:*

$$T = SD^\dagger R^T, \tag{3.5}$$

where $D^\dagger$ *is the block-diagonal matrix given by* (3.2) *and* (2.32). *Moreover,* D *is nonsingular if, and only if,* T *is nonsingular, and then* $D^\dagger = D^{-1}$ *is the inverse of* D.

Proof. From (2.28) (with $\psi = \varphi = \lambda^j$) and (3.1), we have

$$r_{jn} = \langle \psi_n, \lambda^j \rangle = 0 \quad \text{and} \quad s_{jn} = \langle \lambda^j, \varphi_n \rangle = 0 \quad \text{for all} \quad 0 \le j < n_k,\ n_k \le n < n_{k+1},$$

and hence R and S are lower block-triangular matrices whose kth diagonal blocks start at the index $j = n = n_k$. Comparing (3.1) and (2.18), we see that the diagonal blocks are just the matrices F_k and G_k. Thus R and S are indeed lower block-triangular matrices of the form (3.3).

We now turn to the proof of (3.4). Recall from (2.29)–(2.31) that the columns of the matrices U and V are the coefficients of the polynomials $\{\varphi_n\}_{n=0}^N$ and $\{\psi_n\}_{n=0}^N$. Together with (3.1) and by the definition of the Toeplitz bilinear form in (2.1)–(2.3), it follows that

$$S = ITU = TU \quad \text{and} \quad R^T = V^T TI = V^T T. \tag{3.6}$$

Note that the unit triangular matrices U and V are nonsingular, and from (2.33), we obtain

$$TU = V^{-T}D \quad \text{and} \quad V^T T = DU^{-1}. \tag{3.7}$$

Combining (3.6) and (3.7) gives (3.4).

Using (2.33), the fact that the pseudoinverse satisfies $D = DD^\dagger D$, and (3.4), we get

$$T = V^{-T}DU^{-1} = V^{-T}DD^\dagger DU^{-1} = SD^\dagger R^T,$$

which is just (3.5). By (2.34) and (2.35), the matrix D is nonsingular if, and only if, T is nonsingular. Finally, for any nonsingular D, the pseudoinverse $D^\dagger$ of D coincides with the inverse D^{-1}. ∎

3.2 *Computing the triangular factors*

Next, we derive recurrence relations for the efficient computation of the columns of the block-triangular factors R and S in the decomposition (3.5). To this end, we define two strictly lower triangular matrices as follows:

$$
\begin{aligned}
P &:= [\, p_{jn} \,]_{j,n=0,1,\ldots,N}\,, \quad \text{where} \quad p_{jn} := \begin{cases} 0, & \text{if } j \le n, \\ \langle \widehat{\varphi}_n, \lambda^j \rangle, & \text{if } j > n, \end{cases} \\
Q &:= [\, q_{jn} \,]_{j,n=0,1,\ldots,N}\,, \quad \text{where} \quad q_{jn} := \begin{cases} 0, & \text{if } j \le n, \\ \langle \lambda^j, \widehat{\psi}_n \rangle, & \text{if } j > n. \end{cases}
\end{aligned}
\tag{3.8}
$$

Here, $\widehat{\varphi}_n$ and $\widehat{\psi}_n$ are the reverses of the polynomials generated by Algorithm 2.2. The purpose of the auxiliary matrices (3.8) is twofold. Firstly, the columns of P and Q appear in the recurrence relations for generating the columns of R and S; see Lemma 3.3 below. Secondly, the subdiagonal elements of P and Q are used in the computation of coefficients for these recursions. In fact, the following proposition shows that these elements are just the quantities ρ_n and τ_n in (2.21).

Proposition 3.2 *For all* $n = 0, 1, \ldots, N-1$,

$$p_{n+1,n} = \langle 1, \lambda\varphi_n \rangle \ \ (=: \rho_n) \quad \textit{and} \quad q_{n+1,n} = \langle \lambda\psi_n, 1 \rangle \ \ (=: \tau_n)\,. \tag{3.9}$$

Proof. Using the representation (2.29) of φ_n and the property (2.4) of the Toeplitz bilinear form $\langle \cdot, \cdot \rangle$, we obtain

$$\langle 1, \lambda\varphi_n \rangle = \sum_{j=0}^{n} u_{jn} t_{-j-1} = \sum_{j=0}^{n} u_{jn} t_{n-j-(n+1)} = \langle \widehat{\varphi}_n, \lambda^{n+1} \rangle = p_{n+1,n}.$$

In the same way, one shows the second relation in (3.9). □

In our look-ahead Bareiss algorithm, the matrices R, S, P, and Q are generated by columns. Since the matrices R and S are lower block-triangular, we only need to compute the "lower block-triangular part" of the columns of R and S. In view of (3.3), the lower block-triangular part of the nth column of R and S is given by

$$r_n := [\, r_{n_k+l,n} \,]_{l=0,1,\ldots,N-n_k} \quad \text{and} \quad s_n := [\, s_{n_k+l,n} \,]_{l=0,1,\ldots,N-n_k}\,, \tag{3.10}$$

respectively, where k is the unique block index with $n_k \le n < n_{k+1}$. Similarly, since the matrices P and Q are strictly lower triangular, we only compute the "strictly lower triangular part"

$$p_n := [\, p_{n+1+l,n} \,]_{l=0,1,\ldots,N-n-1} \quad \text{and} \quad q_n := [\, q_{n+1+l,n} \,]_{l=0,1,\ldots,N-n-1} \tag{3.11}$$

of the nth column of P and Q, respectively. Note that r_n and s_n are vectors of length $N - n_k + 1$, while p_n and q_n are vectors of length $N - n$. In view of Proposition 3.2, the vectors (3.11) can be partitioned as follows:

$$p_n = \begin{bmatrix} \rho_n \\ p_n^{(l)} \end{bmatrix} \quad \text{and} \quad q_n = \begin{bmatrix} \tau_n \\ q_n^{(l)} \end{bmatrix}, \tag{3.12}$$

where $p_n^{(l)}, q_n^{(l)} \in \mathbb{C}^{N-n-1}$.

We remark that, in view of (3.1) and (3.8), the vectors r_n, s_n, p_n, and q_n depend only on the nth pair of polynomials φ_n and ψ_n. Recall that for the regular indices $n = n_k$, $k = 0, 1, \ldots, K$, the polynomials φ_n and ψ_n are regular FBOPs. For that reason, we call the vectors r_n, s_n, p_n, and q_n *regular* if $n = n_k$ is a regular index. For the remaining indices n, i.e., if $n_k < n < n_{k+1}$, we refer to r_n, s_n, p_n, and q_n as *inner* vectors.

For the formulation of our recurrence relations, it will be convenient to use the following notation. For each $k \in \{0, 1, \ldots, K\}$, we define $(N - n_k + 1) \times h_k$ matrices

$$R_k := [\, r_{n_k+l, n_k+m} \,]_{\substack{l=0,1,\ldots,N-n_k \\ m=0,1,\ldots,h_k-1}}, \quad S_k := [\, s_{n_k+l, n_k+m} \,]_{\substack{l=0,1,\ldots,N-n_k \\ m=0,1,\ldots,h_k-1}}, \tag{3.13}$$

and $(N - n_{k+1} + 1) \times h_k$ matrices

$$P_k := [\, p_{n_k+1+l+m, n_k+m} \,]_{\substack{l=0,1,\ldots,N-n_{k+1} \\ m=0,1,\ldots,h_k-1}}, \quad Q_k := [\, q_{n_k+1+l+m, n_k+m} \,]_{\substack{l=0,1,\ldots,N-n_{k+1} \\ m=0,1,\ldots,h_k-1}}. \tag{3.14}$$

Note that, by (3.3), R_k and S_k are just the kth block columns of the lower block-triangular part of R and S. Moreover, in view of (3.3), we can partition R_k and S_k as follows:

$$R_k = \begin{bmatrix} F_k \\ R_k^{(l)} \end{bmatrix} \quad \text{and} \quad S_k = \begin{bmatrix} G_k \\ S_k^{(l)} \end{bmatrix}, \tag{3.15}$$

where $R_k^{(l)}$ and $S_k^{(l)}$ are $(N - n_{k+1} + 1) \times h_k$ matrices. We remark that the columns of P_k and Q_k are just truncated versions of the vectors p_n and q_n defined in (3.11). More precisely, we have

$$\begin{aligned} &P_k = \begin{bmatrix} p_{n_k}^{(u)} & p_{n_k+1}^{(u)} & \cdots & p_{n_{k+1}-1}^{(u)} \end{bmatrix}, \quad Q_k = \begin{bmatrix} q_{n_k}^{(u)} & q_{n_k+1}^{(u)} & \cdots & q_{n_{k+1}-1}^{(u)} \end{bmatrix}, \\ &\text{where} \quad p_j^{(u)} := [\, I_{N-n_{k+1}+1} \quad 0 \,]\, p_j, \quad q_j^{(u)} := [\, I_{N-n_{k+1}+1} \quad 0 \,]\, q_j, \quad n_k \le j < n_{k+1}. \end{aligned} \tag{3.16}$$

Finally, recall from (1.7) that Z denotes the *down-shift* matrix; it shifts down the elements of a vector by one position:

$$Zv = \begin{bmatrix} 0 \\ v_0 \\ \vdots \\ v_{n-2} \end{bmatrix} \quad \text{for all} \quad v = \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_{n-1} \end{bmatrix} \in \mathbb{C}^n.$$

After these preliminaries, we can now state the recurrence relations for computing the columns of R, S, P, and Q.

Lemma 3.3 *Let* $\{n_k\}_{k=0}^K$ *be a given sequence of regular indices* (2.13). *Let* $\{\varphi_n\}_{n=0}^N$ *and* $\{\psi_n\}_{n=0}^N$ *be the sequences of monic polynomials generated by Algorithm* 2.2, *and let* R, S, P, *and* Q *be the corresponding matrices defined by* (3.1) *and* (3.8). *Then:*

a) *If* $0 \le n < n_1$, *then*

$$\begin{aligned} r_n &= [\, t_{n-l} \,]_{l=0,1,\dots,N}\,, \quad s_n = [\, t_{l-n} \,]_{l=0,1,\dots,N}\,, \\ p_n &= [\, t_{-(n+1+l)} \,]_{l=0,1,\dots,N-n-1}\,, \quad q_n = [\, t_{n+1+l} \,]_{l=0,1,\dots,N-n-1}\,. \end{aligned} \tag{3.17}$$

b) (Regular vectors.) *If* $n+1 = n_{k+1}$, *then*

$$r_{n+1} = [\,0 \quad I_{N-n}\,]\, Z r_n - P_k \beta_n - \begin{cases} R_k^{(l)} \nu_k, & \textit{if } h_k > 1, \\ 0, & \textit{if } h_k = 1, \end{cases} \tag{3.18}$$

$$s_{n+1} = [\,0 \quad I_{N-n}\,]\, Z s_n - Q_k \alpha_n - \begin{cases} S_k^{(l)} \mu_k, & \textit{if } h_k > 1, \\ 0, & \textit{if } h_k = 1, \end{cases} \tag{3.19}$$

$$p_{n+1} = p_n^{(l)} - [\, I_{N-n-1} \quad 0\,]\, R_k^{(l)} \alpha_n - \begin{cases} [\, I_{N-n-1} \quad 0\,]\, P_k \mu_k, & \textit{if } h_k > 1, \\ 0, & \textit{if } h_k = 1, \end{cases} \tag{3.20}$$

$$q_{n+1} = q_n^{(l)} - [\, I_{N-n-1} \quad 0\,]\, S_k^{(l)} \beta_n - \begin{cases} [\, I_{N-n-1} \quad 0\,]\, Q_k \nu_k, & \textit{if } h_k > 1, \\ 0, & \textit{if } h_k = 1, \end{cases} \tag{3.21}$$

where α_n, β_n, μ_k, *and* ν_k *are given by* (2.22) *and* (2.23).

c) (Inner vectors.) *If* $n_k < n+1 < n_{k+1}$, *then*

$$r_{n+1} = Z r_n - P_{k-1} \beta_n, \quad s_{n+1} = Z s_n - Q_{k-1} \alpha_n, \tag{3.22}$$

$$p_{n+1} = p_n^{(l)} - [\, I_{N-n-1} \quad 0\,]\, R_{k-1}^{(l)} \alpha_n, \quad q_{n+1} = q_n^{(l)} - [\, I_{N-n-1} \quad 0\,]\, S_{k-1}^{(l)} \beta_n, \tag{3.23}$$

where α_n *and* β_n *are given by* (2.22).

Proof. Let $0 \le n < n_1$. By (2.20), $\varphi_n = \psi_n = \lambda^n$, and thus $\widehat{\varphi}_n = \widehat{\psi}_n = 1$. Together with (3.1), (3.8), and (2.4), it follows that, for all $l = 0, 1, \dots, N$,

$$r_{ln} = \langle \lambda^n, \lambda^l \rangle = t_{n-l}, \quad s_{ln} = \langle \lambda^l, \lambda^n \rangle = t_{l-n}, \tag{3.24}$$

and, for all $l = 0, 1, \dots, N-n-1$,

$$p_{n+1+l,n} = \langle 1, \lambda^{n+1+l} \rangle = t_{-(n+1+l)}, \quad q_{n+1+l,n} = \langle \lambda^{n+1+l}, 1 \rangle = t_{n+1+l}. \tag{3.25}$$

Rewriting (3.24) and (3.25) in terms of the vectors r_n, s_n, p_n, and q_n gives (3.17).

Next, we establish the relations (3.18) and (3.20). The proofs of the remaining identities (3.19) and (3.21) in part b), as well as the proofs of (3.22) and (3.23) are completely analogous, and will be omitted here.

We now assume that $n+1 = n_{k+1}$. In order to avoid the distinction between the two cases in (3.18) and (3.20), we formally set $\mu_k := \nu_k := 0$ if $h_k = 1$. First, we show (3.18). To this end, we rewrite (3.18), by means of (3.10) and (3.13)–(3.15), in terms of the coefficients of r_{n+1}, r_n, P_k, and $R_k^{(l)}$, and thus we need to verify that

$$r_{j,n+1} = r_{j-1,n} - \sum_{m=0}^{h_k-1} p_{1+j-h_k+m,n_k+m}(\beta_n)_m - \sum_{m=0}^{h_k-1} r_{j,n_k+m}(\nu_k)_m \tag{3.26}$$

holds true for all indices $j := n_{k+1} + l$ in the range $n_{k+1} \le j \le N$. Note that, in the derivation of (3.26), we used the relation $n_k - n_{k+1} = -h_k$ from (2.14). By means of the definitions (3.1) and (3.8) of r_{jn} and p_{jn}, we can rewrite (3.26) as follows:

$$\langle \psi_{n+1}, \lambda^j \rangle = \langle \psi_n, \lambda^{j-1} \rangle - \sum_{m=0}^{h_k-1} \langle \widehat{\varphi}_{n_k+m}, \lambda^{1+j-h_k+m} \rangle (\beta_n)_m - \sum_{m=0}^{h_k-1} \langle \psi_{n_k+m}, \lambda^j \rangle (\nu_k)_m. \tag{3.27}$$

On the other hand, from the relation (2.25), respectively (2.27), and the bilinearity of the form $\langle \cdot, \cdot \rangle$, we have that, for all $j = n_{k+1}, n_{k+1}+1, \ldots, N$,

$$\langle \psi_{n+1}, \lambda^j \rangle = \langle \lambda\psi_n, \lambda^j \rangle - \sum_{m=0}^{h_k-1} \langle \lambda^{h_k-1-m} \widehat{\varphi}_{n_k+m}, \lambda^j \rangle (\beta_n)_m - \sum_{m=0}^{h_k-1} \langle \psi_{n_k+m}, \lambda^j \rangle (\nu_k)_m. \tag{3.28}$$

Using the shift property (2.5) of the form $\langle \cdot, \cdot \rangle$, it follows that (3.28) and (3.27) are equivalent. This completes the proof of (3.18).

Finally, we turn to (3.20). Here, we first rewrite the recurrence relation (2.24), respectively (2.26), in terms of the reverse polynomials. This gives

$$\widehat{\varphi}_{n+1} = \widehat{\varphi}_n - \sum_{m=0}^{h_k-1} \lambda\psi_{n_k+m}(\alpha_n)_m - \sum_{m=0}^{h_k-1} \lambda^{h_k-m} \widehat{\varphi}_{n_k+m}(\mu_k)_m. \tag{3.29}$$

Using (3.29), (2.5), (3.8), (3.1), and the fact that $h_k = n+1-n_k$, one readily verifies that, for all $j = n+2, n+3, \ldots, N$,

$$\begin{aligned} p_{j,n+1} &= \langle \widehat{\varphi}_{n+1}, \lambda^j \rangle \\ &= \langle \widehat{\varphi}_n, \lambda^j \rangle - \sum_{m=0}^{h_k-1} \langle \psi_{n_k+m}, \lambda^{j-1} \rangle (\alpha_n)_m - \sum_{m=0}^{h_k-1} \langle \widehat{\varphi}_{n_k+m}, \lambda^{j-h_k+m} \rangle (\mu_k)_m \\ &= p_{jn} - \sum_{m=0}^{h_k-1} r_{j-1,n_k+m}(\alpha_n)_m - \sum_{m=0}^{h_k-1} p_{n_k+1+j-(n+2)+m,n_k+m}(\mu_k)_m. \end{aligned} \tag{3.30}$$

Finally, using (3.11)– (3.15), we rewrite the relations (3.30) in terms of p_{n+1}, $p_n^{(l)}$, $R_k^{(l)}$, and P_k. The result is the desired identity (3.20). □

3.3 *Computing the diagonal blocks*

It remains to show how to compute the blocks D_k of the matrix D in the block-triangular factorization (3.5) of T. By (3.4), we have

$$D = R^T U, \tag{3.31}$$

where U is the unit upper triangular matrix in the inverse block-factorization (2.33) of T. From (3.31), it follows that

$$D_k = F_k^T Y_k \quad \text{for all} \quad k = 0, 1, \ldots, K, \tag{3.32}$$

where F_k is the kth diagonal block of R in (3.3), and Y_k is the submatrix of U defined in (2.38), respectively (2.39). Note that the matrix F_k is available as part of R. While the matrix U itself is not generated explicitly within the look-ahead Bareiss algorithm, it turns out that the submatrices Y_k of U can be obtained easily from R. In the following lemma, we state the resulting procedure for computing the blocks D_k.

Lemma 3.4 *Let D_k, $k = 0, 1, \ldots, K$, be the blocks of the block-diagonal matrix D in* (3.5). *Then:*

a) $D_0 = T_{n_1-1}$.

b) *For $k = 1, 2, \ldots, K$,*

$$D_k = F_k^T Y_k, \quad \textit{where} \quad Y_k = \begin{bmatrix} 1 & \omega_1^{(k)} & \cdots & \omega_{h_k-1}^{(k)} \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \omega_1^{(k)} \\ 0 & \cdots & 0 & 1 \end{bmatrix}. \tag{3.33}$$

The elements of Y_k satisfy the recurrences

$$\omega_{j+1}^{(k)} = -\begin{bmatrix} \omega_j^{(k)} & \omega_{j-1}^{(k)} & \cdots & \omega_1^{(k)} & 1 & 0_{N-n_k-j}^T \end{bmatrix} d_{k-1}, \quad j = 0, 1, \ldots, h_k - 2, \tag{3.34}$$

where

$$d_{k-1} := R_{k-1}^{(l)} F_{k-1}^{-1} \begin{bmatrix} 0_{h_{k-1}-1} \\ 1 \end{bmatrix}, \tag{3.35}$$

and $R_{k-1}^{(l)}$ is the subblock of R defined by (3.15) (*with k replaced by $k-1$*).

Proof. First, let $k = 0$. From the form of the vectors r_n in (3.17), it follows that $F_0 = T_{n_1-1}^T$. Furthermore, by part a) of Lemma 2.5, we have $Y_0 = I$, and thus $D_0 = F_0^T I = T_{n_1-1}$.

We now turn to part b). Let $1 \le k \le K$. The first relation in (3.33) was already shown in (3.32), and the Toeplitz structure of Y_k was established in part b) of Lemma 2.5. It only remains to prove (3.34). To this end, consider the part of the matrix equation (3.31) that lies in the $(k-1)$st block row and the kth block column. By (2.32), the corresponding block of D is zero, and we obtain

$$0 = F_{k-1}^T \begin{bmatrix} \star \\ w_k^T \end{bmatrix} + (R_{k-1}^{(l)})^T \begin{bmatrix} Y_k \\ 0_{(N-n_{k+1}+1)\times h_k} \end{bmatrix}, \tag{3.36}$$

where w_k is the vector defined in (2.38). Together with (2.40), it follows from (3.36) that

$$\begin{bmatrix} \omega_1^{(k)} \\ \vdots \\ \omega_{h_k}^{(k)} \end{bmatrix} = w_k = -\left[\, Y_k^T \quad 0_{h_k \times (N-n_{k+1}+1)} \,\right] d_{k-1}, \tag{3.37}$$

with d_{k-1} given by (3.35). Finally, by writing (3.37) in terms of the coefficients of w_k, we get (3.34). □

4 A look-ahead Bareiss algorithm

In this section, we give a detailed description of the proposed look-ahead Bareiss algorithm for the block-triangular factorization of general square Toeplitz matrices.

4.1 *The algorithm*

In Lemmas 3.3 and 3.4, we presented recurrence relations for generating the factors of the block-triangular factorization (3.5) of a general square Toeplitz matrix T. In order to obtain our look-ahead Bareiss algorithm, we simply need to state these recursions in the form of a computational procedure; the result is Algorithm 4.1 below. Note that the triangular matrices R, S, P, and Q are generated by columns, where, at the nth step of Algorithm 4.1, the lower-triangular parts r_{n+1}, s_{n+1}, p_{n+1}, and q_{n+1} of the $(n+1)$st columns of R, S, P, and Q are computed.

One important issue remains to be addressed. So far, we have assumed that the sequence $\{n_k\}_{k=0}^{K}$ of regular indices in (2.13) is given. Of course, in practice, this sequence is not known beforehand, and it needs to be determined in the course of the algorithm. More precisely, at the beginning of the nth step of Algorithm 4.1, we need to decide whether the vectors r_{n+1}, s_{n+1}, p_{n+1}, and q_{n+1} are built as regular vectors or as inner vectors. In the first case, we call the nth step *regular*; otherwise, the nth step is referred to as *inner*. In Algorithm 4.1 below, we use k to denote the number of the last regular index n_k at the beginning of the nth step. If the nth step is regular, then $n_{k+1} := n+1$ is the next regular index, and we set $k := k+1$. If the nth step is inner, then k remains unchanged; in this case, we have $n_k < n+1 < n_{k+1}$.

The criteria on which the decision "regular step or inner step" is based is called the *look-ahead strategy*. In Section 4.2 below, we will sketch the look-ahead strategy used in our look-ahead Bareiss algorithm. Here, we merely remark that our look-ahead strategy requires only one input parameter, namely an upper bound $h_{\max}$ on the maximal size of the diagonal blocks in the factorization (3.5) of T. In other words, the sizes of the diagonal blocks built by Algorithm 4.1 are constrained by the condition

$$h_k = n_{k+1} - n_k \le h_{\max} \quad \text{for all} \quad k. \tag{4.1}$$

Recall that the regular indices $\{n_k\}_{k=0}^{K}$ correspond to the well-conditioned leading principal submatrices of T. Thus the requirement (4.1) means that our look-ahead Bareiss algorithm can handle Toeplitz matrices T that have at most $h_{\max} - 1$ consecutive singular or ill-conditioned leading principal submatrices. Except for contrived examples, Toeplitz matrices that arise in practice have at most a small number of consecutive singular or ill-conditioned leading principal submatrices, and hence we can usually assume that $h_{\max}$ is small compared to N. If no information on the leading principal submatrices of T is available, then we recommend to use $h_{\max} = 5$ as a default value for the maximal block size.

In order to reduce the number of SAXPYs in our look-ahead Bareiss algorithm, we actually use slightly modified versions of the recursions (3.18)–(3.21) if $h_k > 1$, and of the relations (3.22) and (3.23) if $h_{k-1} > 1$. The key observation here is that, in view of (2.22), the coefficient vectors α_n and β_n satisfy

$$\alpha_n = \rho_n \tilde{\alpha}_k \quad \text{and} \quad \beta_n = \tau_n \tilde{\beta}_k \quad \text{for all} \quad n \quad \text{with} \quad n_{k+1} \leq n + 1 < n_{k+2}, \tag{4.2}$$

where

$$\tilde{\alpha}_k := F_k^{-1} \begin{bmatrix} 0_{h_k-1} \\ 1 \end{bmatrix} \quad \text{and} \quad \tilde{\beta}_k := G_k^{-1} \begin{bmatrix} 0_{h_k-1} \\ 1 \end{bmatrix}. \tag{4.3}$$

Now assume that $h_k > 1$, and set

$$\tilde{r}_k := R_k^{(l)} \, \tilde{\alpha}_k, \quad \tilde{s}_k := S_k^{(l)} \, \tilde{\beta}_k, \quad \tilde{p}_k := P_k \, \tilde{\beta}_k, \quad \tilde{q}_k := Q_k \, \tilde{\alpha}_k, \tag{4.4}$$

From (4.2) and (4.4), it follows that

$$R_k^{(l)} \, \alpha_n = \rho_n \, \tilde{r}_k, \quad S_k^{(l)} \, \beta_n = \tau_n \, \tilde{s}_k, \quad P_k \, \beta_n = \tau_n \, \tilde{p}_k, \quad Q_k \, \alpha_n = \rho_n \, \tilde{q}_k \tag{4.5}$$

for all n with $n_{k+1} \leq n + 1 < n_{k+2}$. Thus, for each k with $h_k > 1$, we first precompute the vectors (4.4), and then we use the formulas (4.5) to obtain the terms $R_k^{(l)} \alpha_n$, $S_k^{(l)} \beta_n$, $P_k \beta_n$, and $Q_k \alpha_n$ in (3.18)–(3.21), instead of computing them directly. Moreover, if $h_{k-1} > 1$, then we obtain the terms $R_{k-1}^{(l)} \alpha_n$, $S_{k-1}^{(l)} \beta_n$, $P_{k-1} \beta_n$, and $Q_{k-1} \alpha_n$,in (3.23) and (3.22) by means of the relations (4.5) (with k replaced by $k-1$). Note that, so far, the vectors (4.4) are only defined for the indices k with $h_k > 1$. For the formulation of our look-ahead Bareiss algorithm, it will be convenient to set

$$\tilde{r}_k := R_k^{(l)}, \quad \tilde{s}_k := S_k^{(l)}, \quad \tilde{p}_k := p_{n_k}, \quad \tilde{q}_k := q_{n_k} \tag{4.6}$$

if k is an index with $h_k = 1$. We remark that, by (4.3), (4.4), and (4.6), the vector d_{k-1} defined in (3.35) satisfies

$$d_{k-1} = \gamma_{k-1} \tilde{r}_{k-1}, \quad \text{where} \quad \gamma_{k-1} := \begin{cases} 1/F_{k-1}, & \text{if } h_{k-1} = 1, \\ 1, & \text{if } h_{k-1} > 1. \end{cases} \tag{4.7}$$

Finally, by rewriting the recursions in Lemmas 3.3 and 3.4 in the form of an algorithm, and by utilizing the relations (3.12), (2.19), (2.18), (3.15), (2.22), (3.16), and (4.3)—(4.7), one readily obtains the following procedure for computing a block-triangular factorization of T.

Algorithm 4.1 (Block-Triangular Factorization of a General Square Toeplitz Matrix T.)

INPUT: Toeplitz matrix $T = [\, t_{i-j} \,]_{i,j=0,1,\ldots,N}$;
Maximal size $h_{\max}$ of the blocks D_k of the block-diagonal matrix D.

OUTPUT: Matrices R, S, and D of a block-triangular factorization $T = SD^{\dagger}R^T$ of T.

0) Determine $n \in \{\, 0, 1, \ldots, h_{\max} - 1 \,\}$ such that

$$\kappa(T_n) = \min_{h=0,1,\ldots,h_{\max}-1} \kappa(T_h). \tag{4.8}$$

Set

$$k = 0, \quad h_0 = n + 1, \quad n_0 = 0, \quad Y_0 = I_{h_0},$$

$$R_0 = [\, r_0 \quad r_1 \quad \cdots \quad r_n \,] = T^T \begin{bmatrix} I_{h_0} \\ 0 \end{bmatrix}, \quad S_0 = [\, s_0 \quad s_1 \quad \cdots \quad s_n \,] = T \begin{bmatrix} I_{h_0} \\ 0 \end{bmatrix},$$

$$p_j = \begin{bmatrix} t_{-(j+1)} \\ t_{-(j+2)} \\ \vdots \\ t_{-N} \end{bmatrix} \quad \text{and} \quad q_j = \begin{bmatrix} t_{j+1} \\ t_{j+2} \\ \vdots \\ t_N \end{bmatrix} \quad \text{for} \quad j = 0, 1, \ldots, n.$$

For $n = h_0 - 1, h_0, \ldots, N - 1$ do:

1) Obtain ρ_n, $p_n^{(l)}$, τ_n, $q_n^{(l)}$ by partitioning p_n and q_n as follows:

$$p_n = \begin{bmatrix} \rho_n \\ p_n^{(l)} \end{bmatrix} \begin{matrix} \}\,1 \\ \end{matrix}, \quad q_n = \begin{bmatrix} \tau_n \\ q_n^{(l)} \end{bmatrix} \begin{matrix} \}\,1 \\ \end{matrix}.$$

If $k = 0$, then go to 3).

2) Decide whether to perform a regular or an inner step, and go to 3) or 4), respectively.

3) (Regular step.)

a) Obtain f_k, $r_n^{(l)}$, g_k, $s_n^{(l)}$, F_k, $R_k^{(l)}$, G_k, $S_k^{(l)}$ by partitioning Zr_n, Zs_n, R_k, and S_k as follows:

$$Z\,r_n = \left[\begin{array}{c} 0 \\ f_k \\ \hline r_n^{(l)} \end{array}\right] \begin{matrix} \}\,1 \\ \}\,h_k - 1 \\ \end{matrix}, \quad Z\,s_n = \left[\begin{array}{c} 0 \\ g_k \\ \hline s_n^{(l)} \end{array}\right] \begin{matrix} \}\,1 \\ \}\,h_k - 1 \\ \end{matrix},$$

$$R_k = \left[\begin{array}{c} F_k \\ \hline R_k^{(l)} \end{array}\right] \begin{matrix} \}\,h_k \\ \end{matrix}, \quad S_k = \left[\begin{array}{c} G_k \\ \hline S_k^{(l)} \end{array}\right] \begin{matrix} \}\,h_k \\ \end{matrix}.$$

Set

$$D_k = F_k^T Y_k. \tag{4.9}$$

b) (Previous block is of size 1.) If $h_k = 1$, then set

$$\begin{aligned}
\gamma_k &= \frac{1}{F_k}, \quad \alpha_n = \rho_n\,\gamma_k, \quad \delta_k = \frac{1}{G_k}, \quad \beta_n = \tau_n\,\delta_k, \\
\tilde{r}_k &= R_k^{(l)}, \quad \tilde{s}_k = S_k^{(l)}, \quad \tilde{p}_k = p_n, \quad \tilde{q}_k = q_n, \\
r_{n+1} &= r_n^{(l)} - \tilde{p}_k\,\beta_n, \quad s_{n+1} = s_n^{(l)} - \tilde{q}_k\,\alpha_n, \\
p_{n+1} &= p_n^{(l)} - [\,I_{N-n-1} \;\; 0\,]\,\tilde{r}_k\,\alpha_n, \quad q_{n+1} = q_n^{(l)} - [\,I_{N-n-1} \;\; 0\,]\,\tilde{s}_k\,\beta_n.
\end{aligned}$$

c) (Previous block is of size > 1.) If $h_k > 1$, then obtain $\tilde{\alpha}_k$, $\tilde{\beta}_k$, μ_k, ν_k by solving

$$F_k\,\tilde{\alpha}_k = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad G_k\,\tilde{\beta}_k = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad G_k\,\mu_k = \begin{bmatrix} 0 \\ g_k \end{bmatrix}, \quad F_k\,\nu_k = \begin{bmatrix} 0 \\ f_k \end{bmatrix}, \tag{4.10}$$

respectively, and set

$$\begin{aligned}
p_j^{(u)} &= [\,I_{N-n} \;\; 0\,]\,p_j \quad \text{and} \quad q_j^{(u)} = [\,I_{N-n} \;\; 0\,]\,q_j \\
&\qquad\qquad\qquad\qquad \text{for} \quad j = n_k, n_k+1, \ldots, n, \\
P_k &= \begin{bmatrix} p_{n_k}^{(u)} & p_{n_k+1}^{(u)} & \cdots & p_n^{(u)} \end{bmatrix}, \quad Q_k = \begin{bmatrix} q_{n_k}^{(u)} & q_{n_k+1}^{(u)} & \cdots & q_n^{(u)} \end{bmatrix}, \\
\tilde{r}_k &= R_k^{(l)}\,\tilde{\alpha}_k, \quad \tilde{s}_k = S_k^{(l)}\,\tilde{\beta}_k, \quad \tilde{p}_k = P_k\,\tilde{\beta}_k, \quad \tilde{q}_k = Q_k\,\tilde{\alpha}_k, \quad \gamma_k = \delta_k = 1, \\
r_{n+1} &= r_n^{(l)} - \tilde{p}_k\,\tau_n - R_k^{(l)}\,\nu_k, \quad s_{n+1} = s_n^{(l)} - \tilde{q}_k\,\rho_n - S_k^{(l)}\,\mu_k, \\
p_{n+1} &= p_n^{(l)} - [\,I_{N-n-1} \;\; 0\,]\,(\tilde{r}_k\,\rho_n + P_k\,\mu_k), \\
q_{n+1} &= q_n^{(l)} - [\,I_{N-n-1} \;\; 0\,]\,(\tilde{s}_k\,\tau_n + Q_k\,\nu_k).
\end{aligned}$$

d) Set

$$n_{k+1} = n+1, \quad k = k+1, \quad h_k = 1, \quad R_k = S_k = Y_k = \emptyset, \quad \Omega_k = 1,$$

and go to 5).

4) (Inner step.) Set

$$\begin{aligned}
r_{n+1} &= Z\,r_n - \tilde{p}_{k-1}\,\tau_n\,\delta_{k-1}, \quad s_{n+1} = Z\,s_n - \tilde{q}_{k-1}\,\rho_n\,\gamma_{k-1}, \\
p_{n+1} &= p_n^{(l)} - [\,I_{N-n-1} \;\; 0\,]\,\tilde{r}_{k-1}\,\rho_n\,\gamma_{k-1}, \quad q_{n+1} = q_n^{(l)} - [\,I_{N-n-1} \;\; 0\,]\,\tilde{s}_{k-1}\,\tau_n\,\delta_{k-1}, \\
\omega_{h_k}^{(k)} &= -\gamma_{k-1}\,\tilde{r}_{k-1}^T \begin{bmatrix} \Omega_k \\ 0 \end{bmatrix}, \quad \Omega_k = \begin{bmatrix} \omega_{h_k}^{(k)} \\ \Omega_k \end{bmatrix}, \quad h_k = h_k + 1.
\end{aligned}$$

5) Set

$$R_k = [\,R_k \quad r_{n+1}\,], \quad S_k = [\,S_k \quad s_{n+1}\,], \quad Y_k = \left[\begin{array}{c|c} \begin{matrix} Y_k \\ 0 \end{matrix} & \Omega_k \end{array}\right].$$

If $n+1 = N$, then set

$$K = k, \quad F_K = R_K, \quad D_K = F_K^T Y_K.$$

Remark. If $h_{\max} = 1$, then, by (4.1), $h_k = 1$ for all k, and all vectors are built as regular vectors. In this case, Algorithm 4.1 reduces to the symmetric version of the classical Bareiss algorithm described in [2].

4.2 *The look-ahead strategy*

In [16], Freund and Zha devised a look-ahead strategy for their look-ahead Levinson algorithm for computing an inverse block-factorization (2.33) of T. In view of Theorems 2.4 and 3.1, the block sizes in the block-triangular factorization $T = SD^{\dagger}R^T$ of T generated by the look-ahead Bareiss Algorithm 4.1 should be the same as the block sizes in the inverse block-factorization $V^T T U = D$ of T. Therefore, for Algorithm 4.1, we simply use the look-ahead strategy from [16]. This is indeed possible, since all quantities needed for the look-ahead strategy in [16] are also available in our look-ahead Bareiss Algorithm 4.1. Next, we briefly sketch this look-ahead strategy; for a more detailed discussion, the reader is referred to [16, Section 6.3].

The look-ahead strategy requires from the user only the maximal block size $h_{\max}$ as an input parameter. Based on $h_{\max}$, two threshold parameters $\texttt{COND}$ (> 0) and $\texttt{GFACTOR}$ (≥ 1) are then determined adaptively in the course of the algorithm. The decision "regular step or inner step" is based on the two parameters `COND` and `GFACTOR`. Obviously, the goal is to perform regular steps whenever possible, and to resort to inner steps only in the case of singular or ill-conditioned submatrices. In our look-ahead strategy, we first pretend that the nth step of Algorithm 4.1 can be performed as a regular step, and we compute the matrices G_k and Y_k, as well as the quantities α_n and β_n if $h_k = 1$, respectively $\rho_n \tilde{\alpha}_k$, $\tau_n \tilde{\beta}_k$, μ_k, and ν_k if $h_k > 1$. We then compute

$$\Gamma_k := G_k Y_k^{-1},$$

$$\vartheta_n := \begin{cases} \max\{\, |\alpha_n|, |\beta_n| \,\}, & \text{if } h_k = 1, \\ \max\{\, \|\rho_n \tilde{\alpha}_k\|_1, \|\tau_n \tilde{\beta}_k\|_1, \|\mu_k\|_1, \|\nu_k\|_1 \,\}, & \text{if } h_k > 1. \end{cases}$$

Here, $\|\xi\|_1 := |\xi_0| + |\xi_1| + \cdots + |\xi_{h-1}|$ is the 1-norm of a vector $\xi \in \mathbb{C}^h$. Finally, we check whether

$$\kappa(\Gamma_k) \leq 2 \times \texttt{COND} \quad \text{and} \quad \vartheta_n \leq 2 \times \texttt{GFACTOR}. \tag{4.11}$$

If both conditions in (4.11) are satisfied, then Algorithm 4.1 actually performs a regular step; otherwise, an inner step is performed.

We remark that the matrix Γ_k is a suitably rescaled version of the diagonal block D_k. Recall that, in view of (2.34), for a regular step it is necessary that D_k is nonsingular, and this motivates the first check in (4.11). Furthermore, in [16], it is shown that $\kappa(\Gamma_k)$ is in some sense related to the condition number of the leading principal submatrix T_n of T. For a motivation of the second check in (4.11), we refer the reader to [16].

4.3 *Operation counts*

In this section, we present an operation count for Algorithm 4.1. To simplify matters, we assume that

$$h_{\max} \ll N. \tag{4.12}$$

As we mentioned before, Toeplitz matrices T that arise in practice usually have at most a small number of consecutive singular or ill-conditioned leading principal submatrices, and thus the size h_k of the blocks in the block-triangular factorizations of T is small, with most blocks of size $h_k = 1$. In particular, the assumption (4.12) on the maximal block size $h_{\max}$ will typically be satisfied in practice.

The operations in Algorithm 4.1 can be divided into two categories: (i) the updates of the vectors r_{n+1}, s_{n+1}, p_{n+1}, and q_{n+1}, which require only SAXPYs of vectors of approximate length $N-n$, and (ii) various operations with matrices and vectors of order h_k, such as the determination of the minimal condition number in (4.8), the computation of D_k in (4.9), and the solution of the $h_k \times h_k$ systems in (4.10). All operations in (ii) require at most $\mathcal{O}(h_k^3)$ multiplications and additions, and hence, in view of (4.1) and (4.12), the work for (ii) is *negligible* compared to (i). Therefore, in our operation count, we only focus on the work for (i), and in Table 1, we list the numbers of SAXPYs and their lengths required in the nth step of the look-ahead Bareiss Algorithm 4.1. In Table 1, we also list the numbers of SAXPYs and inner products involved in the nth step of the look-ahead Levinson algorithm proposed in [16]. Recall that the look-ahead Bareiss algorithm generates the lower triangular matrices R, S, P, and Q by columns, from left to right, and this is why the lengths of the SAXPYs decrease with n. The look-ahead Levinson algorithm generates the upper triangular matrices U and V by columns, from left to right, and thus, the lengths of the vectors increase with n. Furthermore, note that

$$n_k \le n < n_{k+1} = n_k + h_k \le n_k + h_{\max},$$

and hence, for the lengths $N-n_k-1$ and n_k in Table 1, we have the following upper bounds:

$$N - n_k - 1 \le N - n + h_{\max} - 2 \quad \text{and} \quad n_k \le n.$$

If $h_k = 1$ for all k, then the look-ahead Bareiss algorithm and the look-ahead Levinson algorithm both reduce to the classical versions of these algorithms for strongly regular Toeplitz matrices. The nth step of the classical Bareiss algorithm involves two SAXPYs of length $N-n$ and two SAXPYs of length $N-n-1$, while the nth step of the classical Levinson algorithm requires two SAXPYs and two inner products of length $n+1$. Table 1 shows that each block of size $h_k > 1$ built by the look-ahead Bareiss algorithm causes an overhead of $8h_k$ SAXPYs, compared to the classical algorithm. The overhead could actually be reduced to $8h_k - 4$ SAXPYs, by working with certain rescaled versions of $\tilde{r}_k$, $\tilde{s}_k$, $\tilde{p}_k$, and $\tilde{q}_k$ in Algorithm 4.1. Finally, we remark that the overhead for the look-ahead Bareiss algorithm is higher than for the look-ahead Levinson algorithm, which has an overhead of $4h_k$ SAXPYs of length $n+1$ for each block of size $h_k > 1$.

LOOK-AHEAD BAREISS	regular step		inner step
	with $h_k = 1$	with $h_k > 1$	
SAXPYs of length $N-n-1$	2	2	2
SAXPYs of length $N-n$	2	$2+8h_k$	0
SAXPYs of length $N-n_k-1$	0	0	2
LOOK-AHEAD LEVINSON	regular step		inner step
	with $h_k = 1$	with $h_k > 1$	
inner products of length $n+1$	2	2	2
SAXPYs of length $n+1$	2	$2+4h_k$	0
SAXPYs of length n_k	0	0	2

Table 1: Operation counts for the nth step of the look-ahead Bareiss Algorithm 4.1 and the look-ahead Levinson algorithm [16, Algorithm 6.1]

5 Special Toeplitz matrices

In this section, we consider two important special cases, namely Hermitian Toeplitz matrices and banded Toeplitz matrices.

5.1 *Hermitian Toeplitz matrices*

Assume that the Toeplitz matrix T is Hermitian:

$$T = T^H.$$

As we pointed out in [16, Section 7], the left and right FBOPs and quasi-FBOPs generated by Algorithm 2.2 are connected as follows:

$$\psi_n = \overline{\varphi}_n \quad \text{for all} \quad n = 0, 1, \ldots, N. \tag{5.1}$$

Here $\overline{\varphi}$ denotes the polynomial whose coefficients are the complex conjugates of the coefficients of φ. The relation (5.1) is reflected in the matrices R and S, respectively P and Q. Indeed, using (5.1), (3.1), and (3.8), one readily verifies that

$$R = \overline{S} \quad \text{and} \quad P = \overline{Q}. \tag{5.2}$$

In view of (5.2), we only need to compute the columns s_{n+1} and q_{n+1} of S and Q, and we can delete the updates of the columns r_{n+1} and q_{n+1} of R and Q in Algorithm 4.1. Moreover, by (5.2), we have the following relations:

$$\alpha_n = \overline{\beta_n}, \quad \rho_n = \overline{\tau_n}, \quad \gamma_k = \overline{\delta_k}, \quad \tilde{\alpha}_k = \overline{\tilde{\beta}_k}, \quad \nu_k = \overline{\mu_k}, \quad F_k^T = G_k^H.$$

With these preliminaries, one easily verifies that, for Hermitian Toeplitz matrices T, Algorithm 4.1 reduces to the Hermitian look-ahead Bareiss Algorithm 5.1 below. We stress that this procedure requires only half the work of Algorithm 4.1. Furthermore, note that, by (5.2), we now obtain a Hermitian block-triangular factorization of T of the form

$$T = SD^{\dagger}S^H, \quad \text{where} \quad D = D^H,$$

instead of the decomposition (3.5) for general T.

Our look-ahead Bareiss algorithm for Hermitian Toeplitz matrices can be summarized as follows.

Algorithm 5.1 (Block-Triangular Factorization of a Hermitian Toeplitz Matrix T.)

INPUT: Hermitian Toeplitz matrix $T = [\, t_{i-j} \,]_{i,j=0,1,\ldots,N}$;
Maximal size $h_{\max}$ of the blocks D_k of the block-diagonal matrix D.

OUTPUT: Matrices S and D of a block-triangular factorization $T = SD^{\dagger}S^H$ of T.

0) Determine $n \in \{\, 0, 1, \ldots, h_{\max} - 1 \,\}$ such that

$$\kappa(T_n) = \min_{h=0,1,\ldots,h_{\max}-1} \kappa(T_h).$$

Set

$$k = 0, \quad h_0 = n + 1, \quad n_0 = 0, \quad Y_0 = I_{h_0},$$

$$S_0 = [\, s_0 \quad s_1 \quad \cdots \quad s_n \,] = T \begin{bmatrix} I_{h_0} \\ 0 \end{bmatrix}, \quad q_j = \begin{bmatrix} t_{j+1} \\ t_{j+2} \\ \vdots \\ t_N \end{bmatrix} \quad \text{for} \quad j = 0, 1, \ldots, n.$$

For $n = h_0 - 1, h_0, \ldots, N - 1$ do:

1) Obtain τ_n and $q_n^{(l)}$ by partitioning q_n as follows:

$$q_n = \begin{bmatrix} \tau_n \\ q_n^{(l)} \end{bmatrix} \begin{matrix} \}\, 1 \\ \end{matrix}.$$

If $k = 0$, then go to 3).

2) Decide whether to perform a regular or an inner step, and go to 3) or 4), respectively.

3) (Regular step.)

a) Obtain g_k, $s_n^{(l)}$, G_k, $S_k^{(l)}$ by partitioning Zs_n and S_k as follows:

$$Z\,s_n = \begin{bmatrix} 0 \\ g_k \\ \hline s_n^{(l)} \end{bmatrix} \begin{matrix} \}\,1 \\ \}\,h_k - 1 \\ \end{matrix}, \quad S_k = \begin{bmatrix} G_k \\ \hline S_k^{(l)} \end{bmatrix} \begin{matrix} \}\,h_k \\ \end{matrix}.$$

Set

$$D_k = G_k^H Y_k.$$

b) (Previous block is of size 1.) If $h_k = 1$, then set

$$\delta_k = \frac{1}{G_k}, \quad \beta_n = \tau_n\,\delta_k, \quad \tilde{s}_k = S_k^{(l)}, \quad \tilde{q}_k = q_n,$$
$$s_{n+1} = s_n^{(l)} - \tilde{q}_k\,\overline{\beta_n}, \quad q_{n+1} = q_n^{(l)} - [\,I_{N-n-1} \quad 0\,]\,\tilde{s}_k\,\beta_n.$$

c) (Previous block is of size > 1.) If $h_k > 1$, then obtain $\tilde{\beta}_k$ and μ_k by solving

$$G_k\,\tilde{\beta}_k = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \text{and} \quad G_k\,\mu_k = \begin{bmatrix} 0 \\ g_k \end{bmatrix},$$

respectively, and set

$$\begin{aligned} q_j^{(u)} &= [\,I_{N-n} \quad 0\,]\,q_j \quad \text{for} \quad j = n_k, n_k+1, \ldots, n, \\ Q_k &= \left[\, q_{n_k}^{(u)} \quad q_{n_k+1}^{(u)} \quad \cdots \quad q_n^{(u)} \,\right], \\ \tilde{s}_k &= S_k^{(l)}\,\tilde{\beta}_k, \quad \tilde{q}_k = Q_k\,\overline{\tilde{\beta}_k}, \quad \delta_k = 1, \\ s_{n+1} &= s_n^{(l)} - \tilde{q}_k\,\overline{\tau_n} - S_k^{(l)}\,\mu_k, \\ q_{n+1} &= q_n^{(l)} - [\,I_{N-n-1} \quad 0\,]\,(\tilde{s}_k\,\tau_n + Q_k\,\overline{\mu_k}). \end{aligned}$$

d) Set

$$n_{k+1} = n+1, \quad k = k+1, \quad h_k = 1, \quad S_k = Y_k = \emptyset, \quad \Omega_k = 1,$$

and go to 5).

4) (Inner step.) Set

$$\begin{aligned} s_{n+1} &= Z\,s_n - \tilde{q}_{k-1}\,\overline{\tau_n\,\delta_{k-1}}, \\ q_{n+1} &= q_n^{(l)} - [\,I_{N-n-1} \quad 0\,]\,\tilde{s}_{k-1}\,\tau_n\,\delta_{k-1}, \\ \omega_{h_k}^{(k)} &= -\,\overline{\delta_{k-1}}\,\tilde{s}_{k-1}^H \begin{bmatrix} \Omega_k \\ 0 \end{bmatrix}, \quad \Omega_k = \begin{bmatrix} \omega_{h_k}^{(k)} \\ \Omega_k \end{bmatrix}, \quad h_k = h_k + 1. \end{aligned}$$

5) Set

$$S_k = [\, S_k \quad s_{n+1} \,], \quad Y_k = \left[\begin{array}{c|c} Y_k & \Omega_k \\ 0 & \end{array} \right].$$

If $n+1 = N$, then set

$$K = k, \quad G_K = S_K, \quad D_K = G_K^H Y_K.$$

5.2 *Banded Toeplitz matrices*

Next, we consider the special case of banded Toeplitz matrices T. More precisely, we assume that the elements of T in (1.1) satisfy

$$\begin{aligned} t_j &= 0 \quad \text{for all} \quad j = \eta+1, \eta+2, \ldots, N, \\ t_{-j} &= 0 \quad \text{for all} \quad j = \xi+1, \xi+2, \ldots, N, \end{aligned} \tag{5.3}$$

for some given integers $\eta, \xi \in \{\, 0, 1, \ldots, N \,\}$. The numbers η and ξ are called the *lower bandwidth* and *upper bandwidth* of T, respectively. It turns out that, for banded Toeplitz matrices T, the matrices R, S, P, and Q generated by the look-ahead Bareiss Algorithm 4.1 are also banded. More precisely, we have the following result.

Theorem 5.2 *Let $\eta, \xi \in \{\, 0, 1, \ldots, N \,\}$ be given, and assume that T is a banded Toeplitz matrix with lower bandwidth η and upper bandwidth ξ. Then:*

a) *The lower block-triangular matrix R and the strictly lower triangular matrix P are both banded with lower bandwidth ξ, i.e.,*

$$r_{jn} = p_{jn} = 0 \quad \textit{if} \quad n + \xi < j \le N. \tag{5.4}$$

b) *The lower block-triangular matrix S and the strictly lower triangular matrix Q are both banded with lower bandwidth η, i.e.,*

$$s_{jn} = q_{jn} = 0 \quad \textit{if} \quad n + \eta < j \le N. \tag{5.5}$$

Proof. Let n be an arbitrary, but fixed integer with $0 \le n < N - \xi$. Using the bilinearity of the form $\langle \cdot, \cdot \rangle$, the property (2.4), and (5.3), it follows that, for any polynomial

$$\phi(\lambda) \equiv \sum_{i=0}^{n} \sigma_i \lambda^i$$

of degree at most n, we have

$$\langle \phi, \lambda^j \rangle = \sum_{i=0}^{n} \sigma_i \langle \lambda^i, \lambda^j \rangle = \sum_{i=0}^{n} \sigma_i t_{i-j} = \sum_{i=0}^{n} \sigma_i t_{-(j-i)} = 0 \tag{5.6}$$

for all j with $j > n + \xi$. Recall from (3.1) and (3.8) that $r_{jn} = \langle \psi_n, \lambda^j \rangle$ and $p_{jn} = \langle \widehat{\varphi}_n, \lambda^j \rangle$. Hence, by choosing $\phi = \psi_n$, respectively $\phi = \widehat{\varphi}_n$, in (5.6), we obtain (5.4).

The proof of (5.5) is completely analogous. □

Remark. Theorem 5.2 shows that the look-ahead Bareiss Algorithm 4.1 can exploit the additional band structure of T. In fact, we only need to update the "essential" parts of the vectors r_{n+1}, s_{n+1}, p_{n+1}, and q_{n+1} corresponding to the band of R, S, P, and Q. Consequently, Algorithm 4.1 only requires SAXPYs of vectors of approximate lengths η and ξ.

6 Solving Toeplitz systems

In this section, we sketch various possibilities for the computation of the solution of the Toeplitz system $Tx = b$ from the block-triangular factorization generated by the look-ahead Bareiss algorithm. From now on, we assume that the matrix T is nonsingular. By part c) of Theorem 3.1, the matrix D is then guaranteed to be nonsingular, and the block-triangular factorization (3.5) now reads:

$$T = SD^{-1}R^T. \tag{6.1}$$

All operation counts given in this section assume that (4.12) holds true, so that additional work caused by blocks of size $h_k > 1$ is negligible.

6.1 *Explicit triangular solves*

In view of (6.1), the solution $x = T^{-1}b$ of the Toeplitz system (1.2) can be obtained by solving the lower and upper block-triangular systems

$$S\,c = b \quad \text{and} \quad R^T\,x = D\,c, \tag{6.2}$$

using forward elimination and back-substitution, respectively. The solution of each of the two systems in (6.2) requires approximately $N^2/2$ multiplications and additions, and in terms of work this is the most efficient approach. Recall that Algorithm 4.1 generates the lower triangular matrix S by columns, and consequently, the forward elimination for solving the first system in (6.2) can be done "on the fly" within Algorithm 4.1, without ever storing the matrix S. Unfortunately, the back-substitution for solving the second system in (6.2) cannot be started before the complete solution c of the first system has been obtained, and therefore, we need to store the matrix R, which requires $\mathcal{O}(N^2)$ storage in the general case. For large N, this is a severe drawback of the simple approach (6.2). In Section 6.2 below, we will briefly sketch two options for avoiding the excessive storage requirements for the back-substitution in (6.2).

Recall from Theorem 5.2 that R is a band matrix with bandwidth ξ if T is banded with upper bandwidth ξ, and then only $\mathcal{O}(N\xi)$ storage is required for R. In particular, we recommend to use the approach (6.2) if T is a banded matrix with upper bandwidth $\xi \ll N$.

6.2 *Avoiding the back-substitution*

For the special case of symmetric positive definite Toeplitz matrices, Kung and Hu [26] and Delosme and Ipsen [10, 23] proposed two different procedures for avoiding the explicit storage of R and the back-substitution in (6.2). In this section, we outline extensions of both procedures that can be used in conjunction with our look-ahead Bareiss algorithm for general Toeplitz matrices.

In both procedures, the solution c of the first system in (6.2) is obtained by forward elimination within the look-ahead Bareiss Algorithm 4.1. Next, we sketch two options for obtaining the solution of the second system in (6.2) without storing R. Both approaches are based on the following observation. Using the relation $R^T = DU^{-1}$ from part b) of Theorem 3.1, we see that the solution of the second system in (6.2) is given by

$$x = U\,c. \tag{6.3}$$

Here, U is the unit upper triangular matrix from the inverse block-factorization (2.33) of T. While U is not explicitly generated by the look-ahead Bareiss algorithm, it can be constructed from the quantities

$$\rho_n, \quad \tau_n, \quad \tilde{\alpha}_k, \quad \tilde{\beta}_k, \quad \gamma_k, \quad \delta_k, \quad \mu_k, \quad \text{and} \quad \nu_k \tag{6.4}$$

computed in Algorithm 4.1. The reason is that the look-ahead Levinson algorithm [16] for generating the triangular factors U and V of the decomposition of (2.33) of T uses the same recurrence coefficients (6.4) as Algorithm 4.1. Therefore, by running, in addition to the look-ahead Bareiss Algorithm 4.1, the corresponding recurrences for U from the look-ahead Levinson algorithm [16] one can generate the columns of U and update x by means of (6.3) on the fly, without ever storing the full triangular factors U or R. We refer to the resulting procedure as the *Bareiss-Levinson* approach. We remark that this procedure for obtaining x requires approximately $3N^2/2$ multiplications and additions for general Toeplitz matrices T, and approximately N^2 multiplications and additions for Hermitian Toeplitz systems. Hence, the Bareiss-Levinson approach is more expensive than simply solving the second system in (6.2) by back-substitution, which involves only approximately $N^2/2$ multiplications and additions. However, the Bareiss-Levinson approach requires only $\mathcal{O}(N)$ storage, instead of $\mathcal{O}(N^2)$ for (6.2). For the special case of symmetric positive definite Toeplitz systems, the described procedure reduces to the one proposed in [26].

For the case of symmetric positive definite Toeplitz matrices, Delosme and Ipsen [10, 23] proposed a "recursive back-substitution" for updating the vector x in (6.3) directly from the set of all recurrence coefficients generated by the classical Bareiss algorithm, without explicitly generating the columns of U. It turns out that this recursive back-substitution can be extended to our look-ahead Bareiss algorithm for general Toeplitz matrices. The resulting procedure is quite involved, and we refer the reader to [14] for a detailed description. Here, we only present the basic idea. Recall from (2.29) and (2.30) that the columns of the matrix U are just the coefficients of the polynomials $\{\varphi_n\}_{n=0}^{N}$. Thus, we can rewrite the relation (6.3) in terms of polynomials:

$$\sum_{j=0}^{N} x_j \lambda^j \equiv \sum_{n=0}^{N} c_n \varphi_n(\lambda). \tag{6.5}$$

Here, x_j, $j = 0, 1, \ldots, N$, denote the components of the unknown vector x in (6.3), and c_n, $n = 0, 1, \ldots, N$, are the components of the known vector c. In view of (6.5), determining x is equivalent to performing a change of basis: we are given a polynomial, namely the right-hand side in (6.5), represented in the basis $\{\varphi_n\}_{n=0}^{N}$, and we seek the coefficients of this polynomial for the usual monomial basis $\{\lambda^j\}_{j=0}^{N}$. As we show in [14], this change of basis can be done efficiently, by essentially running the generalized Szegő recursions in Algorithm 2.2 *backwards*. We remark that the resulting recursive back-substitution for computing the vector x in (6.3) requires approximately N^2 multiplications and additions, as well as $\mathcal{O}(N)$ storage for all the quantities (6.4) generated in the course of the look-ahead Bareiss Algorithm 4.1. For general Toeplitz matrices, we recommend to use recursive back-substitution, since it is cheaper than the Bareiss-Levinson approach. For Hermitian Toeplitz matrices, both procedures involve approximately N^2 multiplications and additions, and we recommend to use the Bareiss-Levinson approach, since it does not require the storage of all coefficients (6.4).

6.3 *The case of several right-hand sides*

In this section, we consider the case that we wish to solve the Toeplitz system (1.2) repeatedly, where the coefficient matrix T is fixed and the right-hand side b varies. For example, this situation arises when iterative refinement [33, Section 8.1] is used to improve the accuracy of a computed solution of a single Toeplitz system (1.2).

In the case of several right-hand sides, we recommend to use a variant of the Gohberg-Semencul formula given in [21, Page 18]. This requires the solution of the two special Toeplitz systems

$$T \begin{bmatrix} \sigma_0 \\ \sigma_1 \\ \vdots \\ \sigma_N \end{bmatrix} = \begin{bmatrix} 0_N \\ 1 \end{bmatrix} \quad \text{and} \quad T \begin{bmatrix} \varrho_0 \\ \varrho_1 \\ \vdots \\ \varrho_N \end{bmatrix} = - \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \\ \alpha \end{bmatrix}. \tag{6.6}$$

Here, $\alpha \in \mathbb{C}$ is a parameter that can be chosen arbitrarily; for example, set $\alpha := 0$. In order to obtain the solutions of the two special systems (6.6), we first generate a block-triangular factorization $T = SD^{-1}R^T$ of T by running the look-ahead Bareiss Algorithm 4.1. Then, we use forward elimination followed by the recursive back-substitution sketched in Section 6.2. Note that, in view of the special right-hand side of the first system in (6.6), the work for forward elimination for this system is negligible. Thus computing the solutions of both systems (6.6) from the block-triangular factorization requires approximately $N^2 + 3N^2/2 = 5N^2/2$ multiplications and additions.

Based on the solutions of (6.6), the inverse of T is given by the Gohberg-Semencul formula

$$T^{-1} = \begin{bmatrix} \sigma_N & \sigma_{N-1} & \sigma_{N-2} & \cdots & \sigma_0 \\ 0 & \sigma_N & \sigma_{N-1} & \cdots & \sigma_1 \\ 0 & 0 & \sigma_N & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \sigma_{N-1} \\ 0 & 0 & \cdots & 0 & \sigma_N \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ \varrho_0 & 1 & 0 & \cdots & 0 \\ \varrho_1 & \varrho_0 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ \varrho_{N-1} & \varrho_{N-2} & \cdots & \varrho_0 & 1 \end{bmatrix}$$
$$- \begin{bmatrix} \varrho_N & \varrho_{N-1} & \varrho_{N-2} & \cdots & \varrho_0 \\ 0 & \varrho_N & \varrho_{N-1} & \cdots & \varrho_1 \\ 0 & 0 & \varrho_N & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \varrho_{N-1} \\ 0 & 0 & \cdots & 0 & \varrho_N \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ \sigma_0 & 0 & 0 & \ddots & 0 \\ \sigma_1 & \sigma_0 & 0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ \sigma_{N-1} & \sigma_{N-2} & \cdots & \sigma_0 & 0 \end{bmatrix}. \tag{6.7}$$

Finally, the solution $x = T^{-1}b$ of (1.2) for any right-hand side $b \in \mathbb{C}^{N+1}$ is obtained by computing the matrix-vector product $T^{-1} \cdot b$, using the representation (6.7) of T^{-1} . Exploiting the special structure of (6.7), the calculation of the matrix-vector product $T^{-1} \cdot b$ can be done by means of FFTs, and it requires only $\mathcal{O}(N \log N)$ operations per product; see, e.g., [4].

7 Numerical examples

The classical Bareiss algorithm without look-ahead, the look-ahead Bareiss Algorithm 4.1, and its Hermitian variant, Algorithm 5.1, have been implemented in MATLAB™. In this section, we present results of numerical experiments with these algorithms. All our numerical tests were run on an SGI computer with machine precision of order $\mathcal{O}(10^{-16})$.

For all our examples, we first generated a block-triangular factorization $T = SD^{-1}R^T$ by means of Algorithm 4.1, respectively 5.1 if T is Hermitian. We then computed the solution of the Toeplitz system $Tx = b$ by simply solving the two triangular systems (6.2). The right-hand side b was always chosen so that the vector of all ones,

$$x_{\text{exact}} := \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}^T,$$

is the exact solution of $Tx = b$. For all examples, we list the relative error

$$\mathcal{F} := \frac{\left\| T - SD^{-1}R^T \right\|}{\|T\|}$$

of the computed block-triangular factorization $T = SD^{-1}R^T$, as well as the relative error

$$\mathcal{S} := \frac{\|x_{\text{compt}} - x_{\text{exact}}\|}{\|x_{\text{exact}}\|}$$

of the computed solution x_{compt} of the Toeplitz system.

7.1 *Example 1*

Here we use a family of 100×100 matrices

$$T = T^{(0)} + \epsilon T^{(r)} \tag{7.1}$$

that depend on a parameter $\epsilon \geq 0$. In (7.1), the first term

$$T^{(0)} = \left[t_{i-j}^{(0)} \right]_{i,j=0,1,\ldots,99}$$

is a real symmetric Toeplitz matrix given by

$$t_j^{(0)} = \begin{cases} 0, & \text{if } |j| = 0, 2, 5, \\ 1, & \text{if } |j| = 1, 3, 4, 6, \\ \varrho_{|j|}, & \text{if } 7 \leq |j| \leq 99, \end{cases}$$

where $\varrho_7, \varrho_8, \ldots, \varrho_{99}$ are uniformly distributed random numbers in $[0,1]$. The matrix $T^{(0)}$ is constructed so that its submatrices $T_0^{(0)}$, $T_2^{(0)}$, $T_3^{(0)}$, and $T_4^{(0)}$ are exactly singular. The second term

$$T^{(r)} = \left[t_{i-j}^{(r)} \right]_{i,j=0,1,\ldots,99}$$

in (7.1) is a nonsymmetric Toeplitz matrix whose entries $t_j^{(r)}$ are uniformly distributed random numbers in $[0,1]$. Note that the matrix T is nonsymmetric if $\epsilon > 0$. Moreover, by construction of the matrix T, the submatrices T_0, T_2, T_3, and T_4 are ill-conditioned if ϵ is small. For various values of ϵ, we ran the look-ahead Bareiss Algorithm 4.1, as well as the classical Bareiss algorithm. In Table 2, we list the relative errors $\mathcal{F}$ and $\mathcal{S}$ for all these runs, together with the condition numbers of the matrices T. Note that, in all cases, the look-ahead Bareiss algorithm generates both the block-triangular factorization of T and the solution of $Tx = b$ to essentially full machine precision. In contrast, the classical Bareiss algorithm breaks down (indicated by "∞" in Table 2) for $\epsilon \leq 10^{-8}$. This example clearly shows that, for general Toeplitz systems, the use of look-ahead is crucial.

Finally, we remark that, for all runs listed in Table 2, the look-ahead Bareiss algorithm generated a block of size 2 starting at $n = 0$, a block of size 4 starting at $n = 2$, and three blocks of size 2 starting at $n = 56$, $n = 79$, and $n = 89$, respectively. The first two blocks reflect the ill-conditioned submatrices T_0, T_2, T_3, and T_4, while the remaining three blocks were caused by a failure of the second check in (4.11).

7.2 *Example 2*

In this example, we consider a family of 312×312 matrices T of the form (7.1) that again depend on a parameter $\epsilon \geq 0$. The matrix $T^{(r)}$ in (7.1) is now a nonsymmetric 312×312 Toeplitz matrix whose entries are uniformly distributed random numbers in $[0,1]$. The matrix

$$T^{(0)} = \left[t_{|i-j|}^{(0)} \right]_{i,j=0,1,\ldots,311} \tag{7.2}$$

	$\mathcal{F}$ (factorization)		$\mathcal{S}$ (linear system)		
ϵ	classical	look-ahead	classical	look-ahead	$\kappa(T)$
1.0e-02	9.87e-12	5.47e-14	1.50e-10	1.39-12	6.52e+03
1.0e-04	4.80e-08	3.81e-14	4.68e-07	3.82e-12	4.16e+03
1.0e-06	5.93e-04	6.82e-14	7.46e-03	7.75e-13	4.14e+03
1.0e-08	∞	4.85e-14	∞	2.05e-12	4.14e+03
1.0e-10	∞	3.70e-14	∞	4.64e-12	4.14e+03
1.0e-12	∞	4.54e-14	∞	1.46e-12	4.14e+03
1.0e-14	∞	4.38e-14	∞	5.39e-12	4.14e+03
0.0e+00	∞	4.10e-14	∞	1.17e-12	4.14e+03

Table 2: Relative errors and condition numbers for the family of matrices in Example 1

in (7.1) is a matrix taken from [9]. In [9], Córdova, Freund, and Ruscheweyh, study a family of infinite Toeplitz matrices depending on two integer parameters m and q, and they explicitly describe the eigenvalues of all leading principal submatrices of all matrices in this family. The matrix $T^{(0)}$ in (7.2) is chosen as the leading principal submatrix of order 312 of the infinite Toeplitz matrix corresponding to the parameters $m = 315$ and $q = 3$. The entries $t_j^{(0)}$, $j = 0, 1, \ldots, 311$, of (7.2) are given as the coefficients of the Taylor expansion

$$f(z) \equiv \sum_{j=0}^{\infty} t_j^{(0)} z^j$$

of the function

$$f(z) \equiv 18 + \frac{1}{1-z} - \frac{3}{1-z^3} + \frac{6}{1-z^6} - \frac{24}{1-z^{24}} + \frac{48}{1-z^{48}} - \frac{96}{1-z^{96}}.$$

From the results in [9], it follows that the matrix $T^{(0)}$ has 4 distinct eigenvalues, namely 318 (of multiplicity 4), 342 (of multiplicity 20), 270 (of multiplicity 44), and -18 (of multiplicity 244), and thus $\kappa(T^{(0)}) = 19$. Moreover, the submatrices $T_n^{(0)}$, $n = 50, 51, \ldots, 56$, of $T^{(0)}$ of $T^{(0)}$ are exactly singular, while the remaining leading principal submatrices are all well conditioned. Again, we ran the look-ahead Bareiss Algorithm 4.1 and the classical Bareiss algorithm for various values of ϵ. In Table 3, we list the relative errors $\mathcal{F}$ and $\mathcal{S}$ for all these runs. We remark that, for all runs, the look-ahead Bareiss algorithm generated precisely one block of size bigger than 1, namely a block of size 8, starting at $n = 50$. This block is caused by the 7 consecutive ill-conditioned submatrices T_n, $n = 50, 51, \ldots, 56$, of T.

7.3 *Example 3*

In this example, we use symmetric indefinite Toeplitz matrices T of the form

$$T = [\,t_{|i-j|}\,]_{i,j=0,1,\ldots,N}, \quad \text{where} \quad t_j = \begin{cases} \epsilon, & \text{if } j = 0, \\ 1/2^j, & \text{if } j = 1, 2, \ldots, N. \end{cases} \tag{7.3}$$

	$\mathcal{F}$ (factorization)		$\mathcal{S}$ (linear system)	
ϵ	classical	look-ahead	classical	look-ahead
1.0e-02	1.29e-11	1.85e-13	4.29e-12	1.60e-13
1.0e-04	1.73e-09	6.66e-14	5.18e-10	5.96e-14
1.0e-06	1.43e-07	1.16e-13	7.05e-08	1.23e-13
1.0e-08	1.77e-05	1.84e-13	6.43e-06	1.33e-13
1.0e-10	9.27e-04	1.97e-13	4.06e-04	2.63e-13
1.0e-12	1.92e-01	2.56e-13	9.75e-02	1.08e-13
1.0e-14	2.80e+00	1.33e-13	1.22e+00	5.67e-14
0.0e+00	1.91e-01	1.17e-13	3.83e+00	2.21e-14

Table 3: Relative errors for the family of matrices in Example 2

$N+1$	$\mathcal{F}$ (factorization)	$\mathcal{S}$ (linear system)	$\kappa(T)$
15	2.13e-16	5.51e-16	25.52
30	3.67e-16	1.25e-15	51.35
60	3.78e-16	1.47e-15	101.62
120	3.79e-16	1.83e-15	201.23
240	3.79e-16	3.09e-15	399.91
480	3.79e-16	2.09e-15	796.97

Table 4: Relative errors and condition numbers for the KMS matrices with $\epsilon = 10^{-14}$ in Example 3, using the look-ahead Bareiss algorithm

The family of matrices (7.3) was studied by Kac, Murdock, and Szegő [24], and therefore, we refer to T as a KMS matrix. It is well known that, for $\epsilon = 0$, each third leading principal submatrix T_{3m}, $0 \leq m \leq N/3$, of T is exactly singular. For $\epsilon \approx 0$, the submatrices T_{3m} with $0 \leq m \leq N/3$ are ill-conditioned, while the remaining submatrices are all well conditioned. We ran the Hermitian look-ahead Bareiss Algorithm 5.1 and a Hermitian version of the classical Bareiss algorithm for two different values of the parameter ϵ and various matrix sizes N. First, we chose $\epsilon = 10^{-14}$. In Table 4, we list the relative errors $\mathcal{F}$ and $\mathcal{S}$ for the runs with the look-ahead Bareiss algorithm, together with the condition numbers of the KMS matrices T. In all cases, the look-ahead Bareiss algorithm built blocks of size 2, starting at $n = 3m$, for each m with $0 \leq m \leq N/3$. Table 4 shows that the look-ahead Bareiss algorithm generates the block-triangular factorization and the solution of the Toeplitz system to full machine precision. In contrast, breakdowns occurred in all corresponding runs with the classical Bareiss algorithm.

Next, we set $\epsilon = 10^{-8}$. In Table 5, we list the relative errors $\mathcal{F}$ and $\mathcal{S}$ for runs with the look-ahead Bareiss algorithm and the classical Bareiss algorithm. In all cases, no breakdowns

	$\mathcal{F}$ (factorization)		$\mathcal{S}$ (linear system)	
$N+1$	classical	look-ahead	classical	look-ahead
15	1.04e-08	9.48e-16	1.12e-08	9.23e-16
30	1.03e-08	1.85e-15	1.26e-08	2.69e-15
60	1.01e-08	1.90e-15	8.68e-09	3.32e-15
120	1.00e-08	2.36e-15	7.01e-09	7.35e-15
240	1.00e-08	5.92e-15	5.34e-09	1.39e-14
480	1.00e-08	5.92e-15	4.73e-09	1.68e-14

Table 5: Relative errors for the KMS matrices with $\epsilon = 10^{-8}$ in Example 3

occurred in the classical algorithm; however, the look-ahead algorithm gives much more accurate results.

8 Concluding remarks

We proposed a Schur-type Toeplitz solver that extends the classical Bareiss algorithm for strongly regular Toeplitz matrices to general Toeplitz systems. The algorithm uses a look-ahead technique to detect and skip over exactly singular as well as nonsingular, yet ill-conditioned leading principal submatrices. The resulting look-ahead Bareiss algorithm can handle Toeplitz systems with arbitrary consecutive blocks of singular or ill-conditioned submatrices.

The proposed look-ahead Bareiss algorithm generates a block-triangular factorization of the Toeplitz matrix. The procedure is a counterpart to the look-ahead Levinson algorithm developed in [16] that computes an inverse block-factorization of the Toeplitz matrix. While both algorithms have comparable work and storage requirements on serial machines, the look-ahead Bareiss algorithm is preferable on parallel and vector architectures since it only involves SAXPYs of long vectors. In contrast, the look-ahead Levinson algorithm requires both SAXPYs and inner products of long vectors. In a forthcoming report, we will describe an implementation of our look-ahead Bareiss algorithm on a massively parallel machine.

We are planning to develop a software package with FORTRAN codes for the proposed look-ahead Bareiss algorithm and the look-ahead Levinson algorithm presented in [16]. The package will also include implementations of the look-ahead algorithm for the fast solution of general Hankel systems described in [15] as well as of a Schur-type look-ahead Hankel solver.

Finally, we note that techniques similar to the ones employed in this paper can also be used to derive look-ahead versions of fast Schur-type algorithms for other classes of structured matrices, such as block-Toeplitz matrices, Toeplitz-plus-Hankel matrices, and matrices with low displacement rank. Such look-ahead algorithms will be described elsewhere.

Acknowledgment

The author is indebted to Ilse Ipsen for bringing the recursive back-substitution in [10, 23] to his attention.

References

[1] Ammar, G.S., Gragg, W.B. (1987): The generalized Schur algorithm for the superfast solution of Toeplitz systems. In: J. Gilewicz, M. Pindor, and W. Siemaszko, eds., Rational Approximation and its Applications in Mathematics and Physics. Lecture Notes in Mathematics, Vol. 1237, pp. 315–330. Springer-Verlag, Berlin Heidelberg New York

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

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

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

[5] Bojanczyk, A.W., Brent, R.P., de Hoog, F.R. (1991): On stability of Bareiss algorithm. In: F.T. Luk, ed., Advanced Algorithms and Architectures for Signal Processing II. Proc. SPIE, Vol. 1566, pp. 23–34

[6] Brent, R.P., Luk, F.T. (1983): A systolic array for the linear-time solution of Toeplitz systems of equations. J. VLSI Comput. Syst. **1**, 1–22

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

[8] Chan, T.F., Hansen, P.C. (1992): A look-ahead Levinson algorithm for general Toeplitz systems. IEEE Trans. Signal Process. **40**, 1079–1090

[9] Córdova, A., Freund, R.W., Ruscheweyh, S. (1993): On the eigenstructure of a class of symmetric Toeplitz matrices. AT&T Numerical Analysis Manuscript, Bell Laboratories, Murray Hill

[10] Delosme, J.-M., Ipsen, I.C.F. (1986): Parallel solution of symmetric positive definite systems with hyperbolic rotations. Linear Algebra Appl. **77**, 75–111

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

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

[13] Freund, R.W. (1993): The look-ahead Lanczos process for large nonsymmetric matrices and related algorithms. In: M.S. Moonen, G.H. Golub, and B.L.R. de Moor, eds., Linear Algebra for Large Scale and Real-Time Applications, pp. 137–163. Kluwer Academic Publishers, Dordrecht

[14] Freund, R.W. (1993): Generalized Szegő recurrences and Schur parametrization of general Toeplitz matrices. AT&T Numerical Analysis Manuscript, Bell Laboratories, Murray Hill, in preparation

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

[16] Freund, R.W., Zha, H. (1993): Formally biorthogonal polynomials and a look-ahead Levinson algorithm for general Toeplitz systems. Linear Algebra Appl. **188/189**, 255–303

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

[18] Gueguen, C. (1981): 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

[19] Gutknecht, M.H. (1993): Stable row recurrences for the Padé table and generically superfast look-ahead solvers for non-Hermitian Toeplitz systems. Linear Algebra Appl. **188/189**, 351–421

[20] Grenander, U., Szegő, G. (1984): Toeplitz Forms and their Applications, Second Edition. Chelsea, New York

[21] Heinig, G., Rost, K. (1984): Algebraic Methods for Toeplitz-like Matrices and Operators. Birkhäuser, Basel

[22] Henkel, W. (1992): An extended Berlekamp-Massey algorithm for the inversion of Toeplitz matrices. IEEE Trans. Comm. **40**, 1557–1561

[23] Ipsen, I.C.F. (1988): Systolic algorithms for the parallel solution of dense symmetric positive-definite Toeplitz systems. In: M. Schultz, ed., Numerical Algorithms for Modern Parallel Computer Architectures, pp. 85–108. Springer-Verlag, New York

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

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

[26] Kung, S.-Y., Hu, Y.H. (1983): A highly concurrent algorithm and pipelined architecture for solving Toeplitz systems. IEEE Trans. Acoust. Speech Signal Process. **ASSP-31**, 66–76

[27] Le Roux, J., Gueguen, C. (1977): A fixed point computation of partial correlation coefficients. IEEE Trans. Acoust. Speech Signal Process. **ASSP-25**, 257–259

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

[29] Morf, M. (1974): Fast algorithms for multivariable systems. Ph.D. Thesis, Department of Electrical Engineering, Stanford University

[30] Pal, D. (1990): Fast algorithms for structured matrices with arbitrary rank profile. Ph.D. Thesis, Department of Electrical Engineering, Stanford University

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

[32] Schur, I. (1917/18): Über Potenzreihen, die im Innern des Einheitskreises beschränkt sind, Parts I and II. J. Reine Angew. Math. **147**, 205–232, and **148**, 122–145

[33] Stoer, J., Bulirsch, R. (1993): Introduction to Numerical Analysis, Second Edition. Springer-Verlag, New York

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

[35] Sweet, D. (1993): The use of pivoting to improve the numerical performance of algorithms for Toeplitz matrices. SIAM J. Matrix Anal. Appl. **14**, 468–493

[36] Szegő, G. (1975): Orthogonal Polynomials, Fourth Edition. Amer. Math. Soc., Providence, R.I.

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

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

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

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