Inversion and Factorization of Non-Hermitian Quasi-Toeplitz Matrices*

Yuval Bistritz[†] and Thomas Kailath
Information Systems Laboratory
Stanford University
Stanford, California 94305

Submitted by N. K. Bose

ABSTRACT

This paper considers formulas and fast algorithms for the inversion and factorization of non-Hermitian Toeplitz and quasi-Toeplitz (QT) matrices (matrices with a certain "hidden" Toeplitz structure). The results include the following generalizations: (1) A Schur algorithm that extends to non-Hermitian matrices a previous triangular factorization algorithm for Hermitian QT matrices. (2) A Levinson algorithm that generalizes to non-Hermitian matrices a previous Levinson algorithm that finds the triangularly factorized inverses of certain (so-called admissible) QT matrices. (3) The extension to QT matrices of the Gohberg-Semencul (GS) inversion formula for non-Hermitian Toeplitz matrices. Next, the paper introduces a new fast algorithm, called the *extended QT factorization algorithm*, that overcomes the restriction to admissibility matrices of the above Levinson algorithm. The new algorithm is efficient and comprehensive; it produces, for a general QT matrix $\mathbf{R}_n$ of size $(n+1)\times(n+1)$, the triangularly factorized inverses and the GS type inverses of the matrix and all its submatrices, as well as the triangular factorization of $\mathbf{R}_n$ itself, all in approximately $7n^2$ elementary operations for a non-Hermitian and $3.5n^2$ for a

*Research supported in part by the Air Force Office of Scientific Research, Air Force Systems Command, under Contract AF-83-0228; by the Department of the Navy, Office of Naval Research, under contract N00014-85-K-0612; by the U.S. Army Research Office, under Contract DAAL03-86-K-0045; and by the Air Force Office of Scientific Research, Air Force. Yuval Bistritz also gratefully acknowledges the support by a Chaim Weizmann postdoctoral fellowship award.

[†]Now with AT&T Bell Laboratories, Murray Hill, New Jersey 07974.

52 Vanderbilt Ave., New York, NY 10017 0024-3795/88/$3.50

Hermitian matrix. The fast algorithms for non-Hermitian QT matrices are shown to be associated with two discrete transmission lines (which reduce to the familiar single lattice in the Hermitian case). All the presented algorithms are illustrated and interpreted in terms of input sequences and flows of signals in the related transmission line realization.

SUMMARY OF NOTATION

Vectors are denoted by bold lowercase letters and are always associated with polynomials by the following convention:

$$\mathbf{a}_m = [a_{m0}, a_{m,1}, \ldots, a_{m,m}]^t, \qquad a_m(z) = [1, z, \ldots, z^m]\mathbf{a}_m = \sum_{i=0}^{m} a_{m,i} z^i$$

where ^{t} denotes transposition. $L(\mathbf{a}_m)$ is a lower triangular Toeplitz matrix with first column $\mathbf{a}_m$,

$$L(\mathbf{a}_m) = \begin{bmatrix} a_{m,0} & & & \\ a_{m,1} & & \mathbf{0} & \\ \vdots & & & \\ a_{m,m} & \cdots & a_{m,1} & a_{m,0} \end{bmatrix}.$$

Matrices are denoted by bold uppercase; e.g., $\mathbf{R}_m$ is a matrix of size $(m+1)\times(m+1)$, and is the leading submatrix of $\mathbf{R}_n = [r_{i,j}]$, $m \leqslant n$. $\mathbf{T}_m$ is a Toeplitz matrix $\mathbf{T}_m = [r_{i-j}]$. A circumflex $\hat{}$ distinguishes variables special to a Toeplitz matrix; e.g., if $\mathbf{a}_n$ is the last column of $\mathbf{R}_n^{-1}$, then $\hat{\mathbf{a}}_n$ is the last column of $\mathbf{T}_n^{-1}$.

The lower shift and exchange matrices are, respectively,

$$\mathbf{Z} = \begin{bmatrix} 0 & & & \\ 1 & \ddots & \mathbf{0} & \\ & \ddots & \ddots & \\ \mathbf{0} & & 1 & 0 \end{bmatrix}, \qquad \mathbf{J} = \begin{bmatrix} 0 & & 1 \\ & \iddots & \\ 1 & & 0 \end{bmatrix}$$

and are square matrices of size determined by context. Down-shifted vectors

and reversed matrices, vectors, and polynomials are, respectively,

$$\downarrow \mathbf{a}_m = \mathbf{Z}\mathbf{a}_m, \qquad \overleftrightarrow{\mathbf{R}}_m = \mathbf{J}\mathbf{R}_m\mathbf{J}, \qquad \overleftarrow{\mathbf{a}}_m = \mathbf{J}\mathbf{a}_m, \qquad \overleftarrow{a}_m(z) = [1, z, \ldots, z^m]\overleftarrow{\mathbf{a}}_m.$$

Complex conjugation is denoted by *, e.g., $\mathbf{R}_m^*$, $\mathbf{a}_m^*$, and $\mathbf{a}_m^*(z)$ mean complex conjugation of the entries of the matrix, the vector, and the coefficients (only) of the polynomial, respectively.

Subscripts within parentheses are used to label matrices, vectors, or polynomials when the index is not indicative of their (fixed at n) dimension. For example

$$\mathbf{R}_{(m)}, \quad \mathbf{u}_{(m)}, \quad u_{(m)}(z), \qquad m = 0, 1, \ldots, n,$$

are of size $(n+1)\times(n+1)$, length $n+1$, and degree n, respectively.

---

1. INTRODUCTION

The inversion and factorization of matrices of size n takes, in general, of the order of n^3 elementary operations (see e.g [12]). Consider a square matrix of size $(n+1)\times(n+1)$ over the field of complex numbers,

$$\mathbf{R}_n = \{r_{ij}, 0 \leqslant i, j \leqslant n, r_{00} = 1\}, \tag{1.1}$$

and let $\mathbf{R}_m$, $m = 0, \ldots, n$, denote the $(m+1)\times(m+1)$ leading submatrices of $\mathbf{R}_n$. We make the assumption that $\mathbf{R}_n$ is *strongly regular*, namely, that all $\mathbf{R}_m$, $m = 0, 1, \ldots, n$, are nonsingular. Therefore the two sets of equations

$$b_m^t\mathbf{R}_m = [0, \ldots, 0, D_m], \qquad \mathbf{R}_m\mathbf{a}_m = [0, \ldots, 0, D_m]^t \tag{1.2a,b}$$

have solutions for D_m and

$$\mathbf{b}_m = [b_{m0}, b_{m1}, \ldots, 1]^t, \qquad \mathbf{a}_m = [a_{m0}, a_{m1}, \ldots, 1]^t, \tag{1.3}$$

for all $m = 0, \ldots, n$. Clearly $\mathbf{b}_m^t$ and $\mathbf{a}_m$ are the last row and last column, respectively, of $\mathbf{R}_m^{-1}$, divided by their common last entry D_m^{-1}. Furthermore, it is not difficult to see that the set of solutions $\{\mathbf{b}_m, \mathbf{a}_m, m = 0, 1, \ldots, n\}$

yields an upper-lower triangular factorization for $\mathbf{R}_n^{-1}$:

$$\mathbf{R}_n^{-1} = \mathbf{A}_n \mathbf{D}_n^{-1} \mathbf{B}_n^t$$

$$= \begin{bmatrix} 1 & a_{10} & \cdots & a_{n-1,0} & a_{n,0} \\ 0 & 1 & \cdots & a_{n-1,1} & a_{n,1} \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & a_{n,n-1} \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix}$$

$$\times \mathbf{D}_n^{-1} \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ b_{10} & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ b_{n-1,0} & b_{n-1,1} & \cdots & 1 & 0 \\ b_{n0} & b_{n,1} & \cdots & b_{n,n-1} & 1 \end{bmatrix}, \tag{1.4}$$

in which

$$\mathbf{D}_n = \operatorname{diag}[1, D_1, \ldots, D_n] \tag{1.5}$$

and the two upper triangular matrices $\mathbf{A}_n$ and $\mathbf{B}_n$, with unit diagonal elements, have as their $(m+1)$th column the vectors $\mathbf{a}_m$ and $\mathbf{b}_m$, respectively.

We shall also be interested in lower-upper triangular factorization, with unit diagonal entries, of $\mathbf{R}_n$,

$$\mathbf{R}_n = \mathbf{P}_n \mathbf{D}_n \mathbf{Q}_n^t, \tag{1.6}$$

which is obviously related to the factorization (1.4) of $\mathbf{R}_n^{-1}$ by the relations

$$\mathbf{P}_n = \mathbf{B}_n^{-t}, \qquad \mathbf{Q}_n = \mathbf{A}_n^{-t}, \tag{1.7}$$

with $\mathbf{D}_n$ given by (1.5).

The computational complexity of the inversion and factorization of $\mathbf{R}_n$ may be lower than order n^3 when the matrix has some special structure. It is known that Toeplitz matrices, which have the form

$$\mathbf{T}_n = [r_{i-j}], \qquad 0 \leqslant i, j \leqslant n,$$

can be inverted by fast algorithms that require order n^2 operations [18, 23, 24, 1, 20, 25]. Toeplitz matrices, in particular Hermitian positive definite Toeplitz matrices, have been extensively studied by mathematicians and system theorists, as they appear frequently in many physical models and engineering problems. Moreover, it has been found that these situations often give rise to a set of equations that, while not Toeplitz, have a "hidden" structural proximity to Toeplitz matrices. An important class of such matrices for which the above factorizations can be carried out in order n^2 operations is the class of Hermitian quasi-Toeplitz matrices [16, 17].

In this paper we abandon the requirement of symmetry and consider the class of *non-Hermitian quasi-Toeplitz* matrices, which are those that can be written in the form[1]

$$\mathbf{R}_n = L(\tilde{\mathbf{u}}_{(0)})L^t(\mathbf{u}_{(0)}) - L(\tilde{\mathbf{v}}_{(0)})L^t(\mathbf{v}_{(0)}), \tag{1.8}$$

where, as explained in the Summary of Notation, $L(\mathbf{a}_n)$ denotes the lower triangular Toeplitz matrix with first column $\mathbf{a}_n$. The matrix $\mathbf{R}_n$ is defined by four *generating vectors*

$$\tilde{\mathbf{u}}_{(0)} = \begin{bmatrix} 1 \\ \tilde{u}_{01} \\ \vdots \\ \tilde{u}_{0n} \end{bmatrix}, \qquad \mathbf{u}_{(0)} = \begin{bmatrix} 1 \\ u_{01} \\ \vdots \\ u_{0n} \end{bmatrix}, \qquad \tilde{\mathbf{v}}_{(0)} = \begin{bmatrix} 0 \\ \tilde{v}_{01} \\ \vdots \\ \tilde{v}_{0n} \end{bmatrix}, \qquad \mathbf{v}_{(0)} = \begin{bmatrix} 0 \\ v_{01} \\ \vdots \\ v_{0n} \end{bmatrix}. \tag{1.9}$$

This class includes and generalizes the class of non-Hermitian Toeplitz matrices $\mathbf{T}_n$, which can be obtained by making the special choice $u_{0k} = v_{0k}$ and $\tilde{u}_{0k} = \tilde{v}_{0k}$ to yield

$$\mathbf{T}_n = [c_{i-j}], \qquad c_k = v_{0k}, \quad c_{-k} = \tilde{v}_{0k}. \tag{1.10}$$

An equivalent characterization of quasi-Toeplitz (QT) matrices $\mathbf{R}_n$ is that they have displacement $\Delta\{\mathbf{R}_n\}$ with rank 2 [15],

$$\Delta\{\mathbf{R}_n\} := \mathbf{R}_n - \mathbf{Z}_n\mathbf{R}_n\mathbf{Z}_n^t = [\tilde{\mathbf{u}}_{(0)}, \tilde{\mathbf{v}}_{(0)}]\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}[\mathbf{u}_{(0)}, \mathbf{v}_{(0)}]^t \tag{1.11}$$

[1] The choice of minus sign in the definition (1.8) is not restrictive in a context of non-Hermitian matrices. If $\mathbf{R}_n$ is given by sum of two lower-upper Toeplitz products, the sign of the third or fourth vector (1.9) can be changed before applying the subsequent theory. The choice made here is intended to simplify the transitions to the extensively studied Hermitian and positive definite matrices.

where $\mathbf{Z}$ is the shift matrix defined in the Summary of Notation. This property of QT matrices is also a constructive way to find whether a matrix that does not exhibit "Toeplitzness" is nevertheless QT, and if it is, to determine four vectors that can be used to generate the form (1.8). The starting point of all the algorithms in this paper will be that $\mathbf{R}_n$ is given by (1.8) in terms of four known generating vectors. When the matrix $\mathbf{R}_n$ becomes Hermitian, (1.11) implies that $\Delta\{\mathbf{R}_n\}$ can equivalently be characterized by having one positive and one negative eigenvalue [17]. In the non-Hermitian case, $\Delta\{\mathbf{R}_n\}$ may have any two (possibly complex) eigenvalues.

In this paper we shall present fast algorithms for the triangular factorization of $\mathbf{R}_n$ and $\mathbf{R}_n^{-1}$ for *any* strongly regular non-Hermitian QT matrix $\mathbf{R}_n$. Our results will include generalizations of the Hermitian Schur and the Levinson algorithm [17] to non-Hermitian matrices.[2] We shall show that these algorithms are intimately related to a *pair of* transmission lines such as those shown in Figure 1, rather than to the single lattice arising in the Hermitian case. While the mathematical derivations will not depend on it, this (double) lattice description will be useful in providing insight into the derivation and features of algorithms, as was shown also in the study of symmetric QT matrices in [13]. We should note that the double lattice picture had already been introduced at Stanford by S. Rao in connection with the Schur algorithm for non-Hermitian Toeplitz matrices [19].

The inverse of a Toeplitz matrix is a QT matrix. This result follows from the property of Toeplitz matrices that their reverse and transpose are equal, viz.,

$$\overleftrightarrow{\mathbf{T}}_n := \mathbf{J}\mathbf{T}_n\mathbf{J} = \mathbf{T}_n^t, \tag{1.12}$$

(see Summary of Notation) and from the linear algebra identity [15]

$$\operatorname{rank}\{\mathbf{M}_n - \mathbf{Z}\mathbf{M}_n\mathbf{Z}^t\} = \operatorname{rank}\{\mathbf{M}_n^{-1} - \mathbf{Z}^t\mathbf{M}_n^{-1}\mathbf{Z}\}. \tag{1.13}$$

Indeed, Gohberg and Semencul (GS) [11] showed that $\mathbf{T}_n^{-1}$ can be written as

$$\mathbf{T}_n^{-1} = \frac{1}{D_n}\left\{L\left(\overleftrightarrow{\hat{\mathbf{b}}}_n\right)L^t\left(\overleftrightarrow{\hat{\mathbf{a}}}_n\right) - L(\downarrow\hat{\mathbf{a}}_n)L^t\left(\downarrow\hat{\mathbf{b}}_n\right)\right\}, \tag{1.14}$$

where $\hat{\mathbf{b}}_n$ and $\hat{\mathbf{a}}_n$ are the first and last (normalized) columns, respectively, of

[2] We follow the usual distinction between the algorithms, which for a Toeplitz matrix is, for example, that the Schur algorithm yields the factorization of $\mathbf{T}_n$, while the Levinson algorithm gives the factorization of $\mathbf{T}_n^{-1}$.

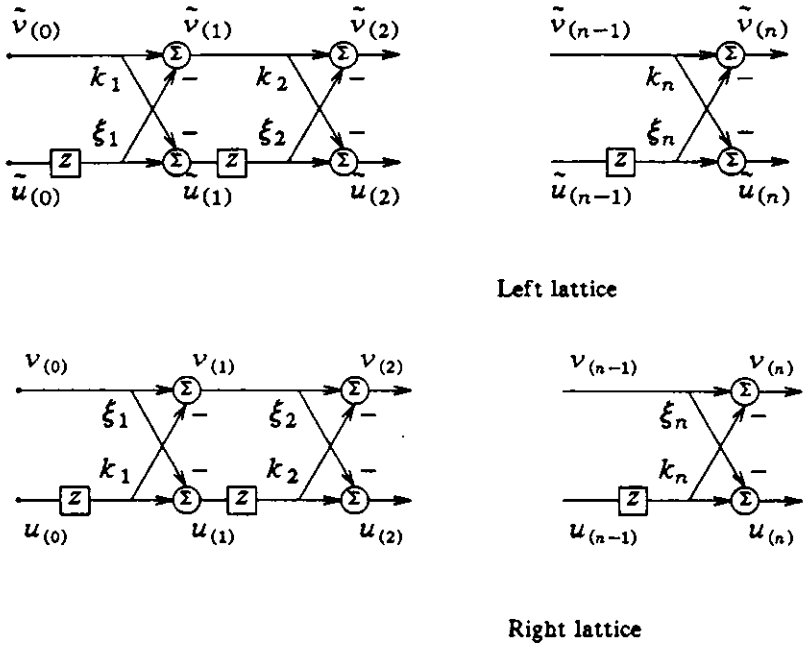

FIG. 1. Transmission lines for the Schur algorithm (Section 2).

$\mathbf{T}_n^{-1}$, or, equivalently, the vectors that solve (1.2) for the Toeplitz matrix $\mathbf{T}_n$. (In the above, $\downarrow$ denotes down-shifted vectors; see Summary of Notation.) Does the inverse of a general QT matrix $\mathbf{R}_n$ (i.e. not Toeplitz or its inverse) belong to this class? In general, $\Delta(\mathbf{R}_n)$ having rank 2 does not imply that $\Delta(\mathbf{R}_n^{-1})$ has rank 2. However, as (1.13) reveals, it does follow that $\overleftrightarrow{\mathbf{R}}_n^{-1}$ has displacement rank 2. Therefore, for any QT matrix $\mathbf{R}_n$, there exists a lower-upper GS type inversion formula for $\overleftrightarrow{\mathbf{R}}_n$. We shall derive such a formula and provide an algorithm to find the vectors it involves. The GS type inversion algorithm suggests the inversion of any QT matrix in only order n requirement of storage, rather than in order n^2, for a triangular decomposition (1.4).

An interesting property of the class of QT matrices is that any matrix $\mathbf{R}_n$ in this class is related to a Toeplitz matrix $\mathbf{T}_n$ by the following relation, which extends the notion of Toeplitz congruence (see [5], [16]) to non-Hermitian matrices:

$$\mathbf{R}_n = L(\tilde{\mathbf{h}}_{0:n})\mathbf{T}_n L^t(\mathbf{h}_{0:n}), \qquad \tilde{\mathbf{h}}_{0:n} := \tilde{\mathbf{u}}_{(0)} - \tilde{\mathbf{v}}_{(0)}, \quad \mathbf{h}_{0:n} := \mathbf{u}_{(0)} - \mathbf{v}_{(0)}. \tag{1.15}$$

This means that a QT matrix is always "close to" an intrinsic Toeplitz matrix. In fact this "hidden" matrix $\mathbf{T}_n$ is a well-defined matrix given by

$$\mathbf{T}_n = \tfrac{1}{2}\{L^{-1}(\tilde{\mathbf{u}}-\tilde{\mathbf{v}})L(\tilde{\mathbf{u}}+\tilde{\mathbf{v}}) + L^t(\mathbf{u}+\mathbf{v})L^{-t}(\mathbf{u}-\mathbf{v})\}. \tag{1.16}$$

Note that this is a Toeplitz matrix, because the product of two lower triangular Toeplitz matrices, and the inverses of such matrices, are all Toeplitz. The above relations stem from the identity, used in its symmetric form already by Schur [21],

$$\begin{aligned}\mathbf{R}_n &= L(\tilde{\mathbf{u}})L^t(\mathbf{u}) - L(\tilde{\mathbf{v}})L^t(\mathbf{v}) \\ &= \tfrac{1}{2}L(\tilde{\mathbf{u}}+\tilde{\mathbf{v}})L^t(\mathbf{u}-\mathbf{v}) + \tfrac{1}{2}L(\tilde{\mathbf{u}}-\tilde{\mathbf{v}})L^t(\mathbf{u}+\mathbf{v}) \\ &= \tfrac{1}{2}L(\tilde{\mathbf{u}}-\tilde{\mathbf{v}})\{L^{-1}(\tilde{\mathbf{u}}-\tilde{\mathbf{v}})L(\tilde{\mathbf{u}}+\tilde{\mathbf{v}}) + L^t(\mathbf{u}+\mathbf{v})L^{-t}(\mathbf{u}-\mathbf{v})\}L^t(\mathbf{u}-\mathbf{v}).\end{aligned}$$

In the Hermitian case, $\tilde{\mathbf{h}}_{0:n} = \mathbf{h}^*_{0:n}$, and (1.15) expresses a congruence relation between every QT matrix and a Toeplitz matrix [5, 16]. As we shall see, following [13], this relation provides easily the generalized GS formula. Furthermore, it also provides some crucial relations between the columns in the factorizations of $\mathbf{R}_n^{-1}$ and $\mathbf{T}_n^{-1}$ that, with appropriate interpretation, will give rise to a "recursive convolution" algorithm, which in combination with the Schur algorithm will provide a hybrid comprehensive (*extended QT factorization*) algorithm to calculate the lower-upper factorization of the matrix and the inverses of a general QT matrix, both for the GS form and for the upper-lower factorization form (1.4).

This paper is exclusively devoted to the factorization and inversion of quasi-Toeplitz matrices defined by rank two displacement. Another measure of closeness to Toeplitz has been defined in the literature in terms of shifted difference operators, and fast algorithms for matrices with low rank shifted differences were considered in [4], [6]–[9]. A QT matrix can always be expressed also as a close to Toeplitz matrix with shifted difference matrix of rank 2; however, this alternative approach requires in general more complicated lattice structures and is computationally more expensive.

Outline of Results

First we address the factorization of $\mathbf{R}_n$ and generalize the Schur algorithm [17] to non-Hermitian QT matrices. We show in Section 2 that the

Schur algorithm for QT matrices is

$$\begin{bmatrix} \tilde{u}_{(m)}(z) \\ \tilde{v}_{(m)}(z) \end{bmatrix} = \begin{bmatrix} 1 & -k_m \\ -\xi_m & 1 \end{bmatrix} \begin{bmatrix} z & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{u}_{(m-1)}(z) \\ \tilde{v}_{(m-1)}(z) \end{bmatrix}, \tag{1.17a}$$

$$\begin{bmatrix} u_{(m)}(z) \\ v_{(m)}(z) \end{bmatrix} = \begin{bmatrix} 1 & -\xi_m \\ -k_m & 1 \end{bmatrix} \begin{bmatrix} z & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_{(m-1)}(z) \\ v_{(m-1)}(z) \end{bmatrix}, \tag{1.17b}$$

$$\xi_m = \frac{\tilde{v}_{m-1,m}}{\tilde{u}_{m-1,m-1}}, \qquad k_m = \frac{v_{m-1,m}}{u_{m-1,m-1}}. \tag{1.17c,d}$$

[The above polynomial recursions are equivalent to corresponding vector recursions for the vectors $\tilde{\mathbf{u}}_{(m)}$, $\tilde{\mathbf{v}}_{(m)}$, $\mathbf{u}_{(m)}$, and $\mathbf{v}_{(m)}$ of their coefficients according the rule in the Summary if Notation.] We shall refer to the (complex) parameters of the recursions $\{k_m, \xi_m\}$, $m = 1, \ldots, n$, as *reflection coefficients*. The algorithm is initiated by the polynomials associated with the four generating vectors (1.9) of $\mathbf{R}_n$.

The recursions (1.17) can be described by the pair of transmission lines of Figure 1. If the four generating vectors (1.9) are applied as inputs to these lattices, then the algorithm produces "on line" the reflection coefficients for the two lattices, and simultaneously the columns for the factorization (1.6) of $\mathbf{R}_n$ appear (within a normalization) as the time responses of the two lattices along the lower lines of each section of the lattice. More precisely, up to normalization, $\tilde{\mathbf{u}}_{(m)}$ and $\mathbf{u}_{(m)}$ are the $(m+1)$th columns of $\mathbf{P}_n$ and $\mathbf{Q}_n$, respectively, in (1.6).

While the generalization of the Schur ($\mathbf{R}_n$-factorization) algorithm is fairly straightforward once we introduce the displacement representation, the generalization of the Levinson ($\mathbf{R}_n^{-1}$ factorization) algorithm is more complicated. This was already seen in the Hermitian case in [17], where it was also shown that a certain so-called *admissibility* constraint on the generators of $\mathbf{R}_n$ led to an algorithm of essentially the same form as the classical Levinson algorithm for Toeplitz matrices. The admissibility constraint is that the four generating vectors (1.9) can be related as

$$u_{(0)}(z) = 1 + \alpha_0 v_{(0)}(z), \tag{1.18a}$$

$$\tilde{u}_{(0)}(z) = 1 + \beta_0 \tilde{v}_{(0)}(z). \tag{1.18b}$$

Such constraints arise, for example, in reflection seismology, when the

upgoing wave suffers only a partial reflection at the surface (complete reflection leads to the Toeplitz case). For simplicity we first consider the admissible case and show in Section 3 that the generalized Levinson algorithm has the form

$$\begin{bmatrix} a_m(z) \\ \alpha_m(z) \end{bmatrix} = \begin{bmatrix} 1 & -k_m \\ -\xi_m & 1 \end{bmatrix} \begin{bmatrix} z & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} a_{m-1}(z) \\ \alpha_{m-1}(z) \end{bmatrix},$$

$$a_0(z) = 1, \quad \alpha_0(z) = \alpha_0, \quad (1.19a)$$

$$\begin{bmatrix} b_m(z) \\ \beta_m(z) \end{bmatrix} = \begin{bmatrix} 1 & -\xi_m \\ -k_m & 1 \end{bmatrix} \begin{bmatrix} z & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} b_{m-1}(z) \\ \beta_{m-1}(z) \end{bmatrix},$$

$$b_0(z) = 1, \quad \beta_0(z) = \beta_0. \quad (1.19b)$$

The *reflection coefficients* are to be calculated by *two* inner products

$$k_{m+1} = \frac{[v_{01}, \ldots, v_{0,m+1}]\mathbf{a}_m}{D_m}, \qquad \xi_{m+1} = \frac{[\tilde{v}_{01}, \ldots, \tilde{v}_{0,m+1}]\mathbf{b}_m}{D_m} \quad (1.19c,d)$$

with D_m updated by

$$D_m = (1 - \xi_m k_m) D_{m-1}, \qquad D_0 = 1. \quad (1.19e)$$

All the algorithms that will be presented in this paper are related to the (double) lattices picture and will be given interpretations as flows of signals created as responses to appropriate input sequences. The above Levinson recursion, for example, corresponds to the situation where the two lattices are excited by impulses of intensities $1, \alpha_0$ and $1, \beta_0$; the variables in the Levinson recursions appear as state vectors along the four lines as depicted in Figure 2.

In Section 3 we also go through the instructive task of deducing the Levinson algorithms for non-Hermitian Toeplitz, Hermitian QT, and Hermitian Toeplitz matrices as special cases from the general setting that we have established; see Figure 3.

In Section 4, we consider the factorization problem for $\mathbf{R}_n^{-1}$ in the general setting of not necessarily admissible non-Hermitian QT matrices. The basic approach is to exploit the extension to non-Hermitian case of the congruence relationship between $\mathbf{R}_n$ and some canonical matrix, which in the QT case is

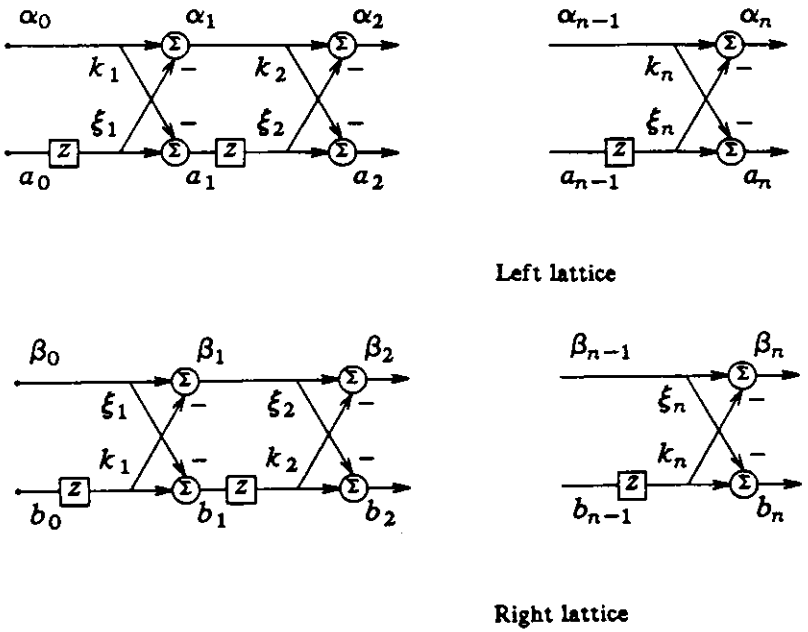

FIG. 2. Transmission lines for the Levinson algorithm (Section 3).

a Toeplitz matrix, similar to the treatment in [13] of the symmetric matrix case. More precisely, from (1.15) we see that the inverse of a general non-Hermitian QT matrix can, by (1.15), be written as

$$\mathbf{R}_n^{-1} = L^t(\Gamma_{0:n})\mathbf{T}_n^{-1}L(\tilde{\Gamma}_{0:n}) = \mathbf{A}_n\mathbf{D}_n^{-1}\mathbf{B}_n^t, \tag{1.20}$$

where the vectors $\Gamma_{0:n}$ and $\tilde{\Gamma}_{0:n}$ are defined by

$$L_n(\Gamma_{0:n}) = L^{-1}(\mathbf{h}_{0:n}), \qquad L_n(\tilde{\Gamma}_{0:n}) = L^{-1}(\tilde{\mathbf{h}}_{0:n}) \tag{1.21}$$

This suggests that we use known results for determining the various quantities related to the "hidden" $\mathbf{T}_n^{-1}$ and then modify them by using the "prefilters" $L_n(\Gamma_{0:n})$ and $L(\tilde{\Gamma}_{0:n})$. This prefiltering can be done in several ways; however, care has to be exercised to ensure that the prefiltering is done in a way that does not increase the computations by an order of magnitude. In Section 4, we shall describe one of these methods for the non-Hermitian case. First the Schur algorithm will be used to determine the reflection

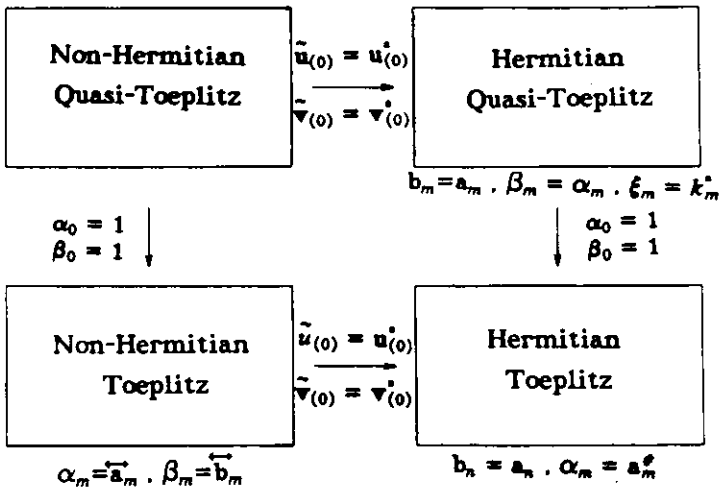

FIG. 3. Particularization routes for the generalized (admissible) Levinson algorithm (Section 3).

coefficients associated with $\mathbf{T}_n$ and $\mathbf{T}_n^{-1}$. Then, we observe that the responses to impulse inputs would determine the entries of the triangular factorization of $\mathbf{T}_n^{-1}$. Using next an appropriate interpretation of the relations (1.20)–(1.21) as convolutions, we are able to propose a fast recursive convolution algorithm to efficiently execute the prefiltering by $L_n(\Gamma_{0:n})$ and $L_n(\tilde{\Gamma}_{0:n})$. As before, the lattice picture illuminates the procedure. We show that if the two sequences defined by the vectors Γ_n and $\tilde{\Gamma}_n$ are applied to the two lattices as described in Figure 4, then the columns for $\mathbf{A}_n$ and $\mathbf{B}_n$ appear as (part of the) time responses along the upper lines of the two lattices. Furthermore, we derive in Section 4 the alternative inversion formula for $\mathbf{R}_n$,

$$\mathbf{R}_n^{-1} = \frac{1}{D_n^e}\left\{ L^t(\mathbf{e}_n)L(\tilde{\mathbf{e}}_n) - L^t(\downarrow \mathbf{g}_n)L(\downarrow \tilde{\mathbf{g}}_n)\right\}, \tag{1.22}$$

which is obtained from the generalization of the GS inversion formula for reversed QT matrices, and show that the four vectors for this formula, as well as for the inversions of all the submatrices $\mathbf{R}_m$, appear as (parts of the) time responses at points along the upper and lower lines of the two lattices in the situation described by Figure 4. We shall call this combination of the Schur algorithm with the recursive convolution algorithm *the extended quasi-Toeplitz factorization algorithm.*

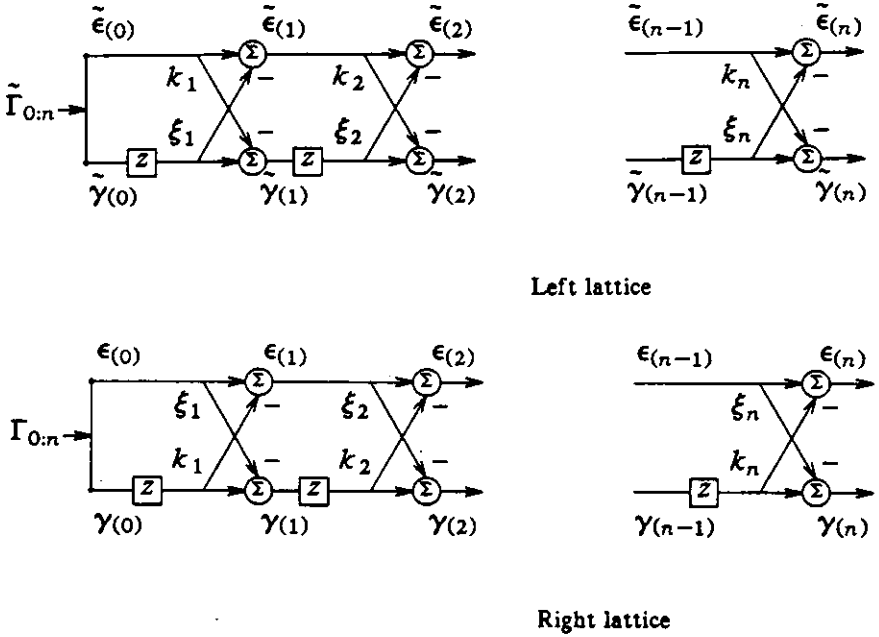

FIG. 4. Transmission lines for the recursive convolution algorithm (Section 4).

The extended QT factorization algorithm, which is summarized in Table 1, is demonstrated to be a comprehensive and efficient fast algorithm for non-Hermitian quasi-Toeplitz matrices. At a computational cost (counted in complex or real multiplications for complex or real matrices, respectively) of $O(7n^2)$ [or $O(3.5n^2)$ for Hermitian matrices],[3] it provides all of the following: the triangular factorization for $\mathbf{R}_n$, the triangular factorization of the inverses of $\mathbf{R}_m$ for all $m = 1, \ldots, n$, and the GS type inverses of $\mathbf{R}_m$ for all $m = 1, \ldots, n$.

Since a QT matrix corresponds to a well-defined rank 2 shifted matrix in the close to Toeplitz class considered in [4], [6]–[9], it is possible, in principle, to have Levinson or Schur algorithm based on the close to Toeplitz algorithms in these references. (References [6] and [7] consider also the non-Hermitian Levinson and Schur algorithms, respectively. We also note that these references all consider block matrices; therefore, to compare the two approaches one has to either extend the results in this paper to block matrices or, as we do in the following, deduce the scalar case from these

[3]We use $N = O(an^2)$ to mean that $N = an^2 + bn + c$, but order n counts are ignored for simplicity.

references using the relation between the two notions of closeness to Toeplitz; see [8].) Algorithms based on the close to Toeplitz approach would involve two (for non-Hermitian; one for Hermitian,) two-term recursions each with four polynomial variables, and would be computationally more expensive. If we apply a general count formula of operations for the inversion of a Hermitian close to Toeplitz matrix that appears in [9], to the scalar case and distance 2 ($p = 1$ and $\alpha = 2$ in the notation there), we find $O(14n^2)$ for the Hermitian Levinson algorithm and an expected double of this count for the non-Hermitian case [8]. Consequently, the new extended factorization algorithm appears to be computationally more efficient by approximately a factor of four (and it also provides more than just the inversion of $\mathbf{R}_n$). Also, a lattice realization of rank 2 close to Toeplitz matrices would be more complicated than the ones in Figures 1 or 2; it would involve two lattices with four lines (with delays along three of them) in each lattice and have multipliers interconnecting the four lines at each section of each lattice.

A complexity account for the various algorithms in this paper is given in Table 2. It indicates how, depending on the case, an amount of computation less than of the extended QT factorization algorithm may suffice. The Schur algorithm, which provides the reflection coefficients and the factorization of $\mathbf{R}_n$, requires only $O(2n^2)$ operations. In the admissible case, the generalized Levinson algorithms that provides the reflection coefficients and the triangular factorization of $\mathbf{R}^{-1}$ (only) requires only $O(3n^2)$ operations. Therefore, to obtain a factorized inverse of an admissible QT matrix, the generalized Levinson algorithm is more efficient than the extended factorization algorithm (which does not give any special allowance to the admissible subclass). We may further note that even for a general QT matrix, other methods as suggested in [17, Figure 4] may reduce the computational count for the factorization of the inverse matrix slightly below the count for the extended QT factorization algorithm.

2. NON-HERMITIAN SCHUR ALGORITHM

In this section we consider the generalization of the Schur algorithm to non-Hermitian matrices and show that it yields fast lower-upper triangular factorization for non-Hermitian Toeplitz and QT matrices (1.6). A lower-upper triangular factorization of an arbitrary strongly regular matrix, such that the factors have unit diagonal elements, is of the form

$$\mathbf{R}_n = \mathbf{P}_n \mathbf{D}_n \mathbf{Q}_n^t = \sum_{m=0}^{n} D_m \mathbf{p}_{(m)} \mathbf{q}_{(m)}^t, \tag{2.1a}$$

where $\mathbf{p}_{(m)}$ and $\mathbf{q}_{(m)}$ are the $(m+1)$th columns (of length $n+1$) of $\mathbf{P}_n$ and $\mathbf{Q}_n$ for all $m = 0, 1, \ldots, n$ of the form

$$\mathbf{p}_{(m)} = [0, \ldots, 0, 1, p_{m,m+1}, \ldots, p_{m,n}]^t, \tag{2.1b}$$

$$\mathbf{q}_{(m)} = [0, \ldots, 0, 1, q_{m,m+1}, \ldots, q_{m,n}]^t, \tag{2.1c}$$

and

$$\mathbf{D}_n = \operatorname{diag}[D_0, \ldots, D_n]. \tag{2.1d}$$

We now define for $m = 0, \ldots, n$ matrices $\mathbf{R}_{(m)}$ of size $(n+1)\times(n+1)$

$$\mathbf{R}_{(m)} = L(\tilde{\mathbf{u}}_{(m)})L^t(\mathbf{u}_{(m)}) - L(\tilde{\mathbf{v}}_{(m)})L^t(\mathbf{v}_{(m)}), \tag{2.2}$$

whose four vectors are defined by the coefficients of the polynomials in the following algorithm, which generalizes to non-Hermitian QT matrices the Schur algorithm of [17]:

$$\begin{bmatrix} \tilde{u}_{(m)}(z) \\ \tilde{v}_{(m)}(z) \end{bmatrix} = \begin{bmatrix} 1 & -k_m \\ -\xi_m & 1 \end{bmatrix} \begin{bmatrix} z & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{u}_{(m-1)}(z) \\ \tilde{v}_{(m-1)}(z) \end{bmatrix}, \tag{2.3a}$$

$$\begin{bmatrix} u_{(m)}(z) \\ v_{(m)}(z) \end{bmatrix} = \begin{bmatrix} 1 & -\xi_m \\ -k_m & 1 \end{bmatrix} \begin{bmatrix} z & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_{(m-1)}(z) \\ v_{(m-1)}(z) \end{bmatrix}, \tag{2.3b}$$

$$\xi_m = \frac{\tilde{v}_{m-1,m}}{\tilde{u}_{m-1,m-1}}, \qquad k_m = \frac{v_{m-1,m}}{u_{m-1,m-1}}. \tag{2.3c,d}$$

The algorithm is initiated by the polynomials that correspond to the generating vectors (1.9) of $\mathbf{R}_n$. Therefore it starts with (2.2) $\mathbf{R}_{(0)} = \mathbf{R}_n$. Also, it is not difficult to see that the first m entries of $\tilde{\mathbf{u}}_{(m)}$ and $\mathbf{u}_{(m)}$ and $m+1$ entries of $\tilde{\mathbf{v}}_{(m)}$ and $\mathbf{v}_{(m)}$ are zeros. In fact,

$$\tilde{\mathbf{u}}_{(m)} = [0, \ldots, 0, \tilde{u}_{m,m}, \cdots \tilde{u}_{m,n}]^t, \qquad \tilde{u}_{m,m} = D_m, \tag{2.4a}$$

$$\mathbf{u}_{(m)} = [0, \ldots, 0, u_{m,m}, \cdots u_{m,n}]^t, \qquad u_{m,m} = D_m, \tag{2.4b}$$

$$\tilde{\mathbf{v}}_{(m)} = [0, \ldots, 0, 0, \tilde{v}_{m,m+1}, \cdots \tilde{v}_{m,n}]^t, \tag{2.4c}$$

$$\mathbf{v}_{(m)} = [0, 0, 0, v_{m,m+1}, \cdots v_{m,n}]^t. \tag{2.4d}$$

Note also that the parameters ξ_m and k_m of the recursion are found as the ratios of the first two nonzero entries of the input vectors at stage m.

We proceed to show that

$$\tilde{\mathbf{u}}_{(m)} = \mathbf{p}_{(m)} D_m, \qquad \mathbf{u}_{(m)} = \mathbf{q}_{(m)} D_m \tag{2.5}$$

for all $m = 0, \ldots, n$, from which it will follow that the Schur algorithm produces the columns for the factorization (2.1a) up to multiplication by the diagonal scaling matrix $\mathbf{D}_n$. The required proof follows if we show that

$$\mathbf{R}_{(m-1)} = \tilde{\mathbf{u}}_{(m-1)} \mathbf{u}_{(m-1)}^t + (1 - \xi_m k_m)^{-1} \mathbf{R}_{(m)}, \tag{2.6}$$

because then,

$$\begin{aligned} \mathbf{R}_{(0)} = \tilde{\mathbf{u}}_{(0)} \mathbf{u}_{(0)}^t + (1 - \xi_1 k_1)^{-1} \big[& \tilde{\mathbf{u}}_{(1)} \mathbf{u}_{(1)}^t \\ & + (1 - \xi_2 k_2)^{-1} \big[\tilde{\mathbf{u}}_{(2)} \mathbf{u}_{(2)}^t \\ & \quad + \cdots + (1 - \xi_n k_n)^{-1} \big[\tilde{\mathbf{u}}_{(n)} \mathbf{u}_{(n)}^t \big] \cdots \big] \big], \end{aligned}$$

or, since $\mathbf{R}_n = \mathbf{R}_{(0)}$,

$$\mathbf{R}_n = \sum_{m=0}^{n} \frac{1}{D_m} \tilde{\mathbf{u}}_{(m)} \mathbf{u}_{(m)}^t. \tag{2.7}$$

We need the following identity for the proof:

$$\begin{aligned} & \big[\tilde{\mathbf{u}}_{(m)}, \tilde{\mathbf{v}}_{(m)} \big] \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \big[\mathbf{u}_{(m)}, \mathbf{v}_{(m)} \big]^t \\ & \quad = (1 - \xi_m k_m) \big[\mathbf{Z} \tilde{\mathbf{u}}_{(m-1)}, \tilde{\mathbf{v}}_{(m-1)} \big] \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \big[\mathbf{Z} \mathbf{u}_{(m-1)}, \mathbf{v}_{(m-1)} \big]^t \end{aligned} \tag{2.8}$$

It can be verified from the recursions (2.3) by substitution and simple evaluation. The proof of (2.6) can now be given.

The displacements of the matrices $\mathbf{R}_{(m)}$ (2.2) are

$$\Delta \{ \mathbf{R}_{(m)} \} = \tilde{\mathbf{u}}_{(m)} \mathbf{u}_{(m)}^t - \tilde{\mathbf{v}}_{(m)} \mathbf{v}_{(m)}^t. \tag{2.9}$$

Multiply the two sides of the last equality by $(1-\xi_m k_m)^{-1}$; then substituting (2.8) into the right hand side gives

$$\Delta\left\{(1-\xi_m k_m)^{-1}\mathbf{R}_{(m)}\right\} = \mathbf{Z}\tilde{\mathbf{u}}_{(m-1)}\mathbf{u}^t_{(m-1)}\mathbf{Z}^t - \tilde{\mathbf{v}}_{(m-1)}\mathbf{v}^t_{(m-1)}.$$

Subtracting this from $\Delta\{\mathbf{R}_{(m-1)}\}$, as given by (2.9), yields

$$\Delta\left\{\mathbf{R}_{(m-1)} - (1-\xi_m k_m)^{-1}\mathbf{R}_{(m)}\right\} = \tilde{\mathbf{u}}_{(m-1)}\mathbf{u}^t_{(m-1)} - \mathbf{Z}\tilde{\mathbf{u}}_{(m-1)}\mathbf{u}^t_{(m-1)}\mathbf{Z}^t,$$

or

$$\Delta\left\{\mathbf{R}_{(m-1)} - (1-\xi_m k_m)^{-1}\mathbf{R}_{(m)} - \tilde{\mathbf{u}}_{(m-1)}\mathbf{u}^t_{(m-1)}\right\} = \mathbf{0}, \qquad (2.10)$$

from which (2.6) follows.

With this we proved that the algorithm (2.3) produces the lower-upper triangular factorization for $\mathbf{R}_n$:

$$\mathbf{R}_n = \tilde{\mathbf{U}}_n \mathbf{D}_n^{-1}\mathbf{U}^t, \qquad (2.11a)$$

where the columns of $\tilde{\mathbf{U}}_n$ and $\mathbf{U}_n$ are given by the Schur variables $\tilde{\mathbf{u}}_{(m)}$ and $\mathbf{U}_{(m)}$, viz.

$$\tilde{\mathbf{U}}_n = \left[\tilde{\mathbf{u}}_{(0)}, \tilde{\mathbf{u}}_{(1)}, \ldots, \tilde{\mathbf{u}}_{(n)}\right], \qquad \mathbf{U}_n = \left[\mathbf{u}_{(0)}, \mathbf{u}_{(1)}, \ldots, \mathbf{u}_{(n)}\right], \quad (2.11b,c)$$

and $\mathbf{D}_n$ is given by (2.1d). We note that due to the nested structure of triangular factorizations and the recursiveness of the algorithm, substituting m for n in (2.11) yields the triangular factorization for the submatrices $\mathbf{R}_m$, $m = 0, \ldots, n$.

The unit diagonal factorization (2.1a) can be obtained by

$$\mathbf{P}_n = \tilde{\mathbf{U}}_n \mathbf{D}_n^{-1}, \qquad \mathbf{Q}_n = \mathbf{U}_n \mathbf{D}_n^{-1}. \qquad (2.12)$$

A unit diagonal triangular factorization can also be obtained directly by the Schur algorithm if the right hand sides of the recursions (2.3a,b) are normalized by multipliers $(1-\xi_m k_m)^{-1}$. Note that the unnormalized recursion involves $O(2n^2)$ elementary operations. The scaling (2.12) requires an extra $O(n^2)$ multiplications whereas the normalized recursions would take a total of $O(4n^2)$ operations.

The Schur algorithm in its presented (unnormalized) form is realizable by the pair of lattices of Figure 1. The algorithm presents the situation in which the four vectors (1.9) are applied as inputs to the two lattices in the way illustrated by Figure 1. Interpreting the z blocks as unit time delays, we have at the lower and upper inputs to section $m+1$ at time i in the left (right) lattice the signals $\tilde{u}_{m,i}$ and $\tilde{v}_{m,i}$ ($u_{m,i}$ and $v_{m,i}$), respectively. The mth columns of $\tilde{\mathbf{U}}_n$ and $\mathbf{U}_n$ appear as the time responses in front of the mth delay of, respectively, the left and right lattices. Also, the reflection coefficient ξ_m (resp. k_m) is the ratio of the upper line to the lower line input signals to section m at time m of the left (resp. right) lattice. It is possible to gain additional physical insight if the signal flows in the delay-free lines are reversed, so that each section forms a scatterer that relates inputs $\mathbf{u}_{(m-1)}, \mathbf{v}_{(m)}$ to outputs $\mathbf{u}_{(m)}, \mathbf{v}_{(m-1)}$ in the right lattice and inputs $\tilde{\mathbf{u}}_{(m-1)}, \tilde{\mathbf{v}}_{(m)}$ to outputs $\tilde{\mathbf{u}}_{(m)}, \tilde{\mathbf{v}}_{(m-1)}$ in the left lattice. In this rearrangement the structure of zeros in (2.4) follows from causality arguments. See [13] for such arguments in the symmetric matrix (one lattice) case.

When the matrix $\mathbf{R}_n$ becomes Hermitian, we have from (1.8)

$$\tilde{\mathbf{u}}_{(0)} = \mathbf{u}_{(0)}^*, \qquad \tilde{\mathbf{v}}_{(0)} = \mathbf{v}_{(0)}^*. \tag{2.13}$$

From these initial conditions, the recursions (2.3) imply

$$\xi_m = k_m^*, \qquad \tilde{u}_{(m)}(z) = u_{(m)}^*(z), \qquad \tilde{v}_{(m)}(z) = v_{(m)}^*(z) \tag{2.14}$$

for all $m = 1, \ldots, n$. So now D_m are real, and the factorization in (2.1) or (2.11) is indeed Hermitian with $\mathbf{U} = \tilde{\mathbf{U}}^*$. Most important, (2.14) indicates that it is sufficient to consider the familiar one two-term recursions [17]

$$\begin{bmatrix} \tilde{u}_{(m)}(z) \\ \tilde{v}_{(m)}(z) \end{bmatrix} = \begin{bmatrix} 1 & -k_m \\ -k_m^* & 1 \end{bmatrix} \begin{bmatrix} z & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{u}_{(m-1)}(z) \\ \tilde{v}_{(m-1)}(z) \end{bmatrix}, \tag{2.15a}$$

$$k_m = \frac{v_{m-1,m}}{u_{m-1,m-1}}. \tag{2.15b}$$

The amount of computation involved in the Schur algorithm is $O(2n^2)$ in the non-Hermitian case and $O(n^2)$ in the Hermitian case.

3. GENERALIZED LEVINSON ALGORITHM

We shall in this section derive the Levinson algorithm (1.17)–(1.19). For the sake of the subsequent derivation and to illustrate at least once the

transition between the polynomial and vectorial forms of the recursions (1.17) that are available in all the algorithms of this paper, we write the Levinson algorithm (1.17)–(1.19) in explicit vectorial form.

We establish in this section (and corresponding appendices) that the solutions to the two sets of equations

$$\mathbf{b}_m^t \mathbf{R}_m = [0,\ldots,0, D_m], \qquad \mathbf{R}_m \mathbf{a}_m = [0,\ldots,0, D_m]^t, \qquad (3.1a,b)$$

for $m = 0,\ldots,n$ with $\mathbf{R}_n$ an admissible QT matrix given by (1.8), (1.9), and (1.18), are produced by an algorithm that is composed of the pair of recursions (1.19) and two inner products to compute the reflection coefficients. The complete (Levinson) algorithm is: for $m = 1,\ldots,n$ compute

$$\begin{bmatrix} a_{m,0} \\ \vdots \\ a_{m,m} \end{bmatrix} = \begin{bmatrix} 0 \\ a_{m-1,0} \\ \vdots \\ a_{m-1,m-1} \end{bmatrix} - k_m \begin{bmatrix} \alpha_{m-1,0} \\ \vdots \\ \alpha_{m-1,m-1} \\ 0 \end{bmatrix}, \qquad (3.2a)$$

$$\begin{bmatrix} \alpha_{m,0} \\ \vdots \\ \alpha_{m,m} \end{bmatrix} = -\xi_m \begin{bmatrix} 0 \\ a_{m-1,0} \\ \vdots \\ a_{m-1,m-1} \end{bmatrix} + \begin{bmatrix} \alpha_{m-1,0} \\ \vdots \\ \alpha_{m-1,m-1} \\ 0 \end{bmatrix}, \qquad (3.2b)$$

$$\begin{bmatrix} b_{m,0} \\ \vdots \\ b_{m,m} \end{bmatrix} = \begin{bmatrix} 0 \\ b_{m-1,0} \\ \vdots \\ b_{m-1,m-1} \end{bmatrix} - \xi_m \begin{bmatrix} \beta_{m-1,0} \\ \vdots \\ \beta_{m-1,m-1} \\ 0 \end{bmatrix}, \qquad (3.3a)$$

$$\begin{bmatrix} \beta_{m,0} \\ \vdots \\ \beta_{m,m} \end{bmatrix} = -k_m \begin{bmatrix} 0 \\ b_{m-1,0} \\ \vdots \\ b_{m-1,m-1} \end{bmatrix} + \begin{bmatrix} \beta_{m-1,0} \\ \vdots \\ \beta_{m-1,m-1} \\ 0 \end{bmatrix}, \qquad (3.3b)$$

with initial conditions

$$a_{0,0} = b_{0,0} = 1, \qquad \alpha_{0,0} = \alpha_0, \qquad \beta_{0,0} = \beta_0 \qquad (3.4)$$

and inner products

$$k_{m+1} = \frac{[v_{01}, \ldots, v_{0,m+1}][a_{m,0}, \ldots, a_{m,m}]^t}{D_m}, \tag{3.5a}$$

$$\xi_{m+1} = \frac{[\tilde{v}_{01}, \ldots, \tilde{v}_{0,m+1}][b_{m,0}, \ldots, b_{m,m}]^t}{D_m}, \tag{3.5b}$$

$$D_m = (1 - \xi_m k_m) D_{m-1}, \qquad D_0 = 1. \tag{3.6}$$

The initial conditions in (3.4) correspond to the admissibility conditions (1.18). They also can be viewed as the most arbitrary possible zeroth degree polynomials to start the recursions with. [Note that normalizing the four initial conditions so that $a_0(z) = b_0(z) = 1$ does not restrict generality and is consistent with the normalization $r_{00} = 1$ in (1.1), which implies $a_{00} b_{00} = r_{00}$ via (1.4).] Consequently, the admissible QT class is the most general class of matrices whose inverses can be factorized as in (1.4) using the Levinson algorithm (1.19).

After proving the above algorithm, we shall show how the general Levinson algorithm simplifies for Toeplitz and Hermitian matrices; see Figure 3. The Levinson algorithm is associated to the same pair of lattices as the Schur algorithm as described in Figure 2. The admissibility values α_0 and β_0 can be viewed as connecting the two inputs by wires with these respective gains, and the variables in the recursions are then the impulse responses at (or transfer functions from the inputs to) the indicated points along the lines of the lattices, as indicated in Figure 2.

3.1. *Non-Hermitian Admissible Quasi-Toeplitz Matrices*

An inspection on the Levinson recursions (3.2)–(3.6) immediately reveals that the first and last entries of the four propagated vectors exhibit the pattern

$$a_{m,m} = 1, \qquad a_{m,0} = -k_m \alpha_0, \tag{3.7}$$

$$b_{m,m} = 1, \qquad b_{m,0} = -\xi_m \beta_0, \tag{3.8}$$

$$\alpha_{m,m} = -\xi_m, \qquad \alpha_{m,0} = \alpha_0, \tag{3.9}$$

$$\beta_{m,m} = -k_m, \qquad \beta_{m,0} = \beta_0. \tag{3.10}$$

More relations required for verifying the algorithm are the following.

LEMMA 1. *The Levinson algorithm also has the following sequence of properties.*

$$[0, v_{01}, \ldots, v_{0m}]\mathbf{a}_m = k_m, \tag{3.11}$$

$$[0, \tilde{v}_{01}, \ldots, \tilde{v}_{0m}]\mathbf{b}_m = \xi_m, \tag{3.12}$$

$$[1, u_{01}, \ldots, u_{0m}]\alpha_m = \alpha_0 D_m, \tag{3.13}$$

$$[1, \tilde{u}_{01}, \ldots, \tilde{u}_{0m}]\beta_m = \beta_0 D_m, \tag{3.14}$$

$$1 + [0, v_{01}, \ldots, v_{0m}]\alpha_m = D_m, \tag{3.15}$$

$$1 + [0, \tilde{v}_{01}, \ldots, \tilde{v}_{0m}]\beta_m = D_m. \tag{3.16}$$

We shall prove this lemma in Appendix A.

To establish our claims for the presented Levinson algorithm we need to show that the mth vector solutions to (3.1a,b) are given by the vectors $\mathbf{a}_m$ and $\mathbf{b}_m$ of (3.2a) and (3.3a). We shall give a proof by induction. This statement is trivial for $n = 1$, thus we have to show the induction step.

PROPOSITION 1. *Assume that the algorithm* (3.2)–(3.6) *produces solutions to* (3.1a,b) *for* $m = 1, \ldots, n-1$. *Then the* nth *step of the algorithm produces solutions to*

$$\mathbf{b}_n^t \mathbf{R}_n = [0, \ldots, 0, D_n], \qquad \mathbf{R}_n \mathbf{a}_n = [0, \ldots, 0, D_n]^t. \tag{3.17a,b}$$

A key observation for the proof is that the definition of $\mathbf{R}_n$ implies that it has the following nested structure:

$$\mathbf{R}_n = \tilde{\mathbf{u}}_{0:n}\mathbf{u}_{0:n}^t - \tilde{\mathbf{v}}_{0:n}\mathbf{v}_{0:n}^t + \begin{bmatrix} 0 & \mathbf{0}_{n-1}^t \\ \mathbf{0}_{n-1} & \mathbf{R}_{n-1} \end{bmatrix}, \tag{3.18}$$

in which $\mathbf{0}_n$ is vector, of length $n+1$ with zero entries (and we introduce the notation $u_{0:m} = [u_{00}, \ldots, u_{0m}]$, $m \leqslant n$, thus $u_{0:n} = u_{(0)}$). We prove Proposition 1 in Appendix B. The proof of this induction step establishes the claims made on the generalized Levinson algorithm.

3.2. *Non-Hermitian Toeplitz Matrices*

The Toeplitz case corresponds to choosing

$$\alpha_0 = 1, \qquad \beta_0 = 1 \tag{3.19}$$

The condition (3.19) means [see (1.18)] that the first and second vectors in (1.9) differ from the third and fourth only in their first entry. That is, (3.19) is equivalent to the condition

$$\tilde{\mathbf{u}}_{0:n} = \begin{bmatrix} 1 \\ \tilde{v}_{01} \\ \vdots \\ \tilde{v}_{0n} \end{bmatrix}, \qquad \mathbf{u}_{0:n} = \begin{bmatrix} 1 \\ v_{01} \\ \vdots \\ v_{0:n} \end{bmatrix}. \tag{3.20}$$

It is seen from (3.18) that if (3.19) and (3.20) hold, then $r_{ij} = r_{i-1,j-1}$ for any $i, j > 1$, which expresses the fact that the entries in the first column $\tilde{\mathbf{u}}_{0:n}$ and first row $\mathbf{u}_{0:n}^t$ determine completely the Toeplitz matrix

$$\mathbf{T}_n = (r_{ij}), \qquad r_{ij} = \begin{cases} u_{0,j-i}, & j \geqslant i, \\ \tilde{u}_{0,i-j}, & j \leqslant i. \end{cases} \tag{3.21}$$

Note that $\mathbf{T}_n$ is not Hermitian, because $\tilde{\mathbf{u}}_{0:n} \neq \mathbf{u}_{0:n}^*$. The condition (3.19) or (3.20) is also necessary for a QT matrix to be Toeplitz because $r_{ij} = r_{i-1,j-1}$ implies, using (3.18), that $\tilde{u}_{0i} = \tilde{v}_{0i}$ and $u_{0j} = v_{0j}$, $i, j > 1$.

The Levinson algorithm for non-Hermitian Toeplitz matrices $\mathbf{T}_n$ solves recursively the set of equations (3.1) with $\mathbf{T}_n$ replacing $\mathbf{R}_n$, i.e.,

$$\mathbf{b}_m^t \mathbf{T}_m = [0,\ldots,0, D_m], \qquad \mathbf{T}_m \mathbf{a}_m = [0,\ldots,0, D_m]^t, \tag{3.22}$$

which solves for the last row and last column of $\mathbf{R}_m$ for $m = 0,\ldots,n$. However, the property (1.12) of Toeplitz matrices makes this set replaceable by any of the following alternative pairs: find the last row and first row of $\mathbf{R}_m^{-1}$,

$$\mathbf{b}_m^t \mathbf{T}_m = [0,\ldots,0, D_m], \qquad \overleftarrow{\mathbf{a}}_m^t \mathbf{T}_m = [D_m, 0,\ldots,0,]; \tag{3.23a}$$

find the first column and last column of $\mathbf{R}_m^{-1}$,

$$\mathbf{T}_m \overleftrightarrow{\mathbf{b}}_m = [D_m, 0,\ldots,0]^t, \qquad \mathbf{T}_m \mathbf{a}_m = [0,\ldots,0, D_m]^t; \tag{3.23b}$$

or find the first column and first row of $\mathbf{R}_n^{-1}$,

$$\mathbf{T}_m \overleftrightarrow{\mathbf{b}}_m = [D_m, 0,\ldots,0]^t, \qquad \overleftarrow{\mathbf{a}}_m^t \mathbf{T}_m = [D_m, 0,\ldots,0,]. \tag{3.23c}$$

Correspondingly, we shall see that the triangular factorization can now be given either as an upper-lower or as a lower-upper factorization.

A Levinson algorithm for the Toeplitz case can be obtained by merely changing the initial conditions in the general algorithm (3.2)–(3.6) to comply with (3.19). Returning to the more compact polynomial notation (1.17) and distinguishing quantities related to Toeplitz matrices by a $\hat{}$, the Levinson recursions for a Toeplitz matrix are given by

$$\begin{bmatrix} \hat{a}_m(z) \\ \hat{\alpha}_m(z) \end{bmatrix} = \begin{bmatrix} 1 & -k_m \\ -\xi_m & 1 \end{bmatrix} \begin{bmatrix} z & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{a}_{m-1}(z) \\ \hat{\alpha}_{m-1}(z) \end{bmatrix},$$

$$\hat{a}_0(z) = 1, \quad \hat{\alpha}_0(z) = 1, \quad (3.24a)$$

$$\begin{bmatrix} \hat{b}_m(z) \\ \hat{\beta}_m(z) \end{bmatrix} = \begin{bmatrix} 1 & -\xi_m \\ -k_m & 1 \end{bmatrix} \begin{bmatrix} z & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{b}_{m-1}(z) \\ \hat{\beta}_{m-1}(z) \end{bmatrix},$$

$$\hat{b}_0(z) = 1, \quad \hat{\beta}_0(z) = 1, \quad (3.24b)$$

with inner products still given by (3.5).

Since a Toeplitz matrix is simpler than the non-Toeplitz matrix, one might expect some corresponding reduction in the complexity of the algorithms. Indeed there is redundancy in (3.24), and it is revealed as follows. If we apply reverse operations on the polynomials in (3.24b) and subsequently interchange the first and the second rows, we obtain

$$\begin{bmatrix} \hat{\vec{\beta}}_m(z) \\ \hat{\vec{b}}_m(z) \end{bmatrix} = \begin{bmatrix} 1 & -k_m \\ -\xi_m & 1 \end{bmatrix} \begin{bmatrix} z & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{\vec{\beta}}_{m-1}(z) \\ \hat{\vec{b}}_{m-1}(z) \end{bmatrix},$$

$$\hat{\vec{\beta}}_0(z) = 1, \quad \hat{\vec{b}}_0(z) = 1.$$

The comparison of this recursions with (3.24a) reveals that

$$\hat{\vec{\beta}}_m(z) = \hat{a}_m(z), \qquad \hat{\vec{b}}_m(z) = \hat{\alpha}_m(z). \qquad (3.25)$$

Therefore the two-term recursions (3.24a) and (3.24b) contain the same information, and one of them can be discarded. The convenient way to removing the redundancy is to retain the primary variables $\hat{\mathbf{a}}_m$ and $\hat{\mathbf{b}}_m$. One

can choose recursions that propagate the pair $\{\hat{a}_m(z), \hat{\tilde{b}}_n(z)\}$ or the pair $\{\hat{b}_m(z), \hat{\tilde{a}}_m(z)\}$. Picking the first of the two choices, the Levinson algorithm for non-Hermitian Toeplitz matrices in its least complex form comprises a single two-term recursion

$$\begin{bmatrix} \hat{a}_m(z) \\ \hat{\tilde{b}}_m(z) \end{bmatrix} = \begin{bmatrix} 1 & -k_m \\ -\xi_m & 1 \end{bmatrix} \begin{bmatrix} z & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{a}_{m-1}(z) \\ \hat{\tilde{b}}_{m-1}(z) \end{bmatrix},$$

$$\hat{a}_0(z) = 1, \quad \hat{\tilde{b}}_0(z) = 1, \quad (3.26)$$

and two inner products

$$k_m = \frac{[u_{01}, \ldots, u_{0,m}]\hat{\mathbf{a}}_{m-1}}{D_{m-1}}, \qquad \xi_m = \frac{[\tilde{u}_{01}, \ldots, \tilde{u}_{0,m}]\hat{\mathbf{b}}_{m-1}}{D_{m-1}} \quad (3.27a,b)$$

with

$$D_m = (1 - \xi_m k_m) D_{m-1}, \qquad D_0 = 1 \quad (3.27c)$$

The algorithm provides the upper-lower triangular factorization of $\mathbf{T}^{-1}$

$$\mathbf{T}_n = \hat{\mathbf{A}}_n \mathbf{D}^{-1} \hat{\mathbf{B}}_n^t, \quad (3.28)$$

where

$$\hat{\mathbf{A}}_n = \begin{bmatrix} 1 & \hat{a}_{10} & \cdots & \hat{a}_{n-1,0} & \hat{a}_{n,0} \\ 0 & 1 & \cdots & \hat{a}_{n-1,1} & \hat{a}_{n,1} \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & \hat{b}_{n,n-1} \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix}, \quad (3.29a)$$

$$\hat{\mathbf{B}}_m = \begin{bmatrix} 1 & \hat{b}_{10} & \cdots & \hat{b}_{n-1,0} & \hat{b}_{n,0} \\ 0 & 1 & \cdots & \hat{b}_{n-1,1} & \hat{b}_{n,1} \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & b_{n,n-1} \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \quad (3.29b)$$

$$D = \operatorname{diag}[1, D_n, \ldots, D_n]. \quad (3.30)$$

Alternatively, applying the reverse operation on this factorization and using the property (1.12) yields

$$\mathbf{T}_n^{-1} = \overleftarrow{\hat{\mathbf{B}}}_n \overrightarrow{\mathbf{D}}_n^{-1} \overleftarrow{\hat{\mathbf{A}}}_n^t, \tag{3.31}$$

that is, a lower-upper factorization for $\mathbf{T}_n^{-1}$.

3.3. *Quasi-Toeplitz Hermitian Matrices*

Consider now what happens when the QT matrix becomes Hermitian. Imposing $\mathbf{R}_n^t = \mathbf{R}_n^*$ on (1.8) and (1.9) implies

$$\tilde{\mathbf{u}}_{(0)} = \mathbf{u}_{(0)}^*, \quad \tilde{\mathbf{v}}_{(0)} = \mathbf{v}_{(0)}^*, \qquad \text{or} \quad \tilde{u}_{(0)}(z) = u_{(0)}^*(z), \quad \tilde{v}_{(0)}(z) = v_{(0)}^*(z). \tag{3.32}$$

This in turn implies $\xi_1 = k_1^*$ and (for our current admissible case) $\alpha_0^* = \beta_0$. It can be shown from here [by induction or showing that (3.32) implies that all subsequent Schur variables, presented in Section 2, also satisfy $u_i^*(z) = \tilde{u}_i(z)$ and $v_i^*(z) = \tilde{v}_i(z)$] that $\xi_m = k_m^*$ also for all subsequent reflection coefficients. Therefore the quasi-Toeplitz Hermitian Levinson algorithm is obtained from the non-Hermitian quasi-Toeplitz algorithm by setting

$$\beta_0 = \alpha_0^* \text{ and } \xi_m = k_m^*, \qquad m = 0, \ldots, n. \tag{3.33}$$

Consequently,

$$b_n(z) = a_n^*(z), \qquad \beta_n(z) = \alpha_n^*(z). \tag{3.34}$$

The last two equations indicate that the second two-term recursion (1.17b) is in the Hermitian case just the conjugate replica of the first. Similarly, the set of equations for $\mathbf{b}_m$ is the conjugate transpose of the other,

$$\left(\mathbf{b}_m^t \mathbf{R}_m\right)^H = \mathbf{R}_m \mathbf{a}_m = [0, \ldots, 0, D_m]^t, \tag{3.35}$$

where we have used (3.33) and (3.34) to deduce that D_m are now real.

We therefore deduce that the Levinson algorithm for Hermitian quasi-Toeplitz matrices concerns one set of equations only,

$$\mathbf{R}_m \mathbf{a}_m = [0, \ldots, 0, D_m]^t, \tag{3.36}$$

which is solved by one two-term recursion

$$\begin{bmatrix} a_m(z) \\ \alpha_m(z) \end{bmatrix} = \begin{bmatrix} 1 & -k_m \\ -k_m^* & 1 \end{bmatrix} \begin{bmatrix} z & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} a_{m-1}(z) \\ \alpha_{m-1}(z) \end{bmatrix},$$

$$a_0(z) = 1, \quad \alpha_0(z) = \alpha_0, \tag{3.37}$$

that requires only one inner product

$$k_{m+1} = \frac{[v_{01}, \ldots, v_{0,m+1}]\mathbf{a}_m}{D_m}, \tag{3.38a}$$

$$D_m = \left(1 - |k_m|^2\right) D_{m-1}, \qquad D_0 = 1 \tag{3.38b}$$

In the Hermitian case it is adequate to associate the recursions with a single lattice, say, only the left lattice in Figure 2 with $\xi_m = k_m^*$, which becomes the well-known form of lattice associated with fast algorithms for QT matrices [17].

The upper-lower factorization for the inverse of a QT Hermitian matrix is deduced from (1.4), and from the fact that now $\mathbf{b}_m = \mathbf{a}_m^*$, to be

$$\hat{\mathbf{R}}_n^{-1} = \mathbf{A}_n \mathbf{D}_n^{-1} \mathbf{A}_n^H, \tag{3.39}$$

where the columns of $\mathbf{A}_n$ are now determined by (3.37), superscript ^{H} denotes conjugate transposition, and $\mathbf{D}_n$ is a diagonal matrix (1.5) with real entries (3.38b).

3.4. *Hermitian Toeplitz Matrices*

The Levinson algorithm for Hermitian Toeplitz matrices is the best-known special case. In the present context we can deduce it either from Section 3.2 or 3.3 or directly from Section 3.1; see Figure 3. It emerges as the combination of the following constraints on the QT non-Hermitian algorithm [see (3.19) and (3.32)]:

$$\alpha_0 = 1, \qquad \beta_0 = 1, \qquad \tilde{\mathbf{u}}_{(0)} = \mathbf{u}_{(0)}^*, \qquad \tilde{\mathbf{v}}_{(0)} = \mathbf{v}_{(0)}^*. \tag{3.40}$$

Consequently, all the implied conditions (3.20), (3.21), (3.25) as well as (3.33) and (3.34) hold and superpose. Incorporating all these simplifications, the following situation is revealed.

The Toeplitz matrix is given by [cf. (1.10)]

$$\mathbf{T}_n = (c_{i-j}), \qquad c_{i-j} = \begin{cases} u_{0,j-i}, & j \geqslant i, \\ u^*_{0,i-j}, & j \leqslant i. \end{cases} \tag{3.41}$$

The Levinson algorithm solves recursively the set of equations

$$\mathbf{T}_m \hat{\mathbf{a}}_m = [0, \ldots, 0, D_m]^t. \tag{3.42}$$

The four variables originally in the general algorithm become now simply related to $\hat{a}_m(z) = a_m(z)$:

$$b_m(z) = \hat{a}^*_m(z), \qquad \beta_m(z) = \hat{\tilde{a}}_m(z), \quad \alpha_m(z) = \hat{a}^{\#}_m(z) \tag{3.43}$$

where we defined the *reciprocal* of a polynomial or a vector as the superposition of reversion and complex conjugation,

$$a^{\#}_m(z) = \tilde{a}^*_m(z) = z^m a^*_m(z^{-1}), \qquad \mathbf{a}^{\#}_m := \tilde{\mathbf{a}}^*_m. \tag{3.44}$$

The Levinson algorithm becomes the following:

$$\begin{bmatrix} \hat{a}_m(z) \\ \hat{a}^{\#}_m(z) \end{bmatrix} = \begin{bmatrix} 1 & -k_m \\ -k^*_m & 1 \end{bmatrix} \begin{bmatrix} z & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{a}_{m-1}(z) \\ \hat{a}^{\#}_{m-1}(z) \end{bmatrix},$$

$$\hat{a}_0(z) = 1 \quad \hat{a}^{\#}_0(z) = 1, \tag{3.45}$$

$$k_{m+1} = \frac{[u_{01}, \ldots, u_{0,m+1}]\hat{\mathbf{a}}_m}{D_m}, \tag{3.46a}$$

$$D_m = \left(1 - |k_m|^2\right) D_{m-1}, \qquad D_0 = 1. \tag{3.46b}$$

The triangular factorizations for the inverse of the Hermitian Toeplitz matrix are given, respectively, by the upper-lower and lower-upper triangular forms

$$\mathbf{T}_n^{-1} = \hat{\mathbf{A}}_n \mathbf{D}_n^{-1} \hat{\mathbf{A}}^H_n, \qquad \mathbf{T}_n^{-1} = \hat{\tilde{\mathbf{A}}}_n \tilde{\mathbf{D}}_n^{-1} \hat{\tilde{\mathbf{A}}}^t_n, \tag{3.47a,b}$$

where $\hat{\mathbf{A}}_n$ has the form (3.29a) with its columns determined by (3.45), and the diagonal matrix (1.5) has real entries (3.46b).

3.5. A Remark on Single Two-Term Recursions

We draw attention to a significant difference between the single two-term recursions (3.26) we saw for the non-Hermitian Toeplitz case and the single two-term recursions (3.37) and (3.45) for the Hermitian cases (Toeplitz or not). In the Hermitian case, the symmetry simplifies not only the Levinson algorithm, but (as we shall see) all the algorithms for the inversion and factorization of any Hermitian QT matrix; and throughout, two sets of equations, recursions, inner products, and lattices reduce to one. In contrast, for a non-Hermitian Toeplitz matrix the possibility of having a single two-term recursion for its inversion by the Levinson algorithm stands out as an anomaly (in the context of the non-Hermitian QT matrix theory that we shall continue to develop in the rest of this paper) which, if not perceived properly, may obscure the "doubleness" inherently possessed by the non-Hermitian QT structure in all cases (including the Toeplitz case): two transmission lines, two two-term recursions in all subsequent Schur and recursive convolution algorithms, etc. We shall further see in Section 4 that we need to use the two two-term recursions (3.24) rather than (3.26) as the recursions for the Toeplitz matrix in order to derive fast inversion algorithms for nonadmissible QT matrices.

The "anomalous" simplification in the Levinson recursions in the class of non-Hermitian QT matrices when the matrix becomes Toeplitz should not be taken as more than a reduction in computation that reflects the extra simplicity of the Toeplitz structure. It is properly comparable to the observation that, within the class of Hermitian QT matrices, the Levinson recursions simplify in the Toeplitz case by the fact that in (3.45) the second variable is the reverse of the first and thus does not require extra computation. However, the phenomenon has no further implications for the structure of the Hermitian QT lattice or for any corresponding simplification in the Schur algorithm for Toeplitz matrices.

The remark we are making here has further interesting aspects. The recursion (3.38) is quite well known. It can be found implicitly already in the original algorithm of Trench [23] for the fast inversion of non-Hermitian Toeplitz matrices [23, 24, 1, 20, 25], where his algorithm circumvents the inner product formulas and produces an unfactorized inverse. The connection of the Levinson polynomial for Hermitian Toeplitz matrices with the polynomials orthogonal on the unit circle is well known. Similar connections between the polynomials in the recursions (3.26) and biorthogonal polynomials on the unit circle were established in [2] and subsequent work; see [14] and references therein. Concluding a discussion on the interesting connections between variables in the recursion (3.26), biorthogonal polynomials, their Christoffel-Darboux formula, and the GS formula (1.14), the authors in [14] express the desirability of finding a generalization of the theory discussed

to QT matrices. The difficulty in deriving such an extended theory can be eliminated once the underlying recursion in the existing theory (3.26) is understood to be an anomalous collapse of the four variables (or two lattices) that are required for a true description of non-Hermitian Quasi-Toeplitz matrices. These aspects of the present theory, while not required for the algorithms that we derive, are of interest for their pure mathematical content. We shall discuss them in a separate paper.

3.6. A Remark on Convolution and "Prefiltering"

It is important to realize that the transmission lines associated with the Levinson and the Schur algorithms are indeed the same, with identical reflection coefficients. If the inverse of Toeplitz matrix $\mathbf{T}_n$, as found by the Levinson algorithm, is (3.35), and the Schur algorithm produces for $\mathbf{T}_n$ the factorization (2.1a) $\mathbf{T}_n = \hat{\mathbf{P}}_n \mathbf{D}_n \hat{\mathbf{Q}}_n^t$ then the triangular matrices involved are related by $\hat{\mathbf{P}}_n = \hat{\mathbf{B}}_n^{-t}$ and $\hat{\mathbf{Q}}_n = \hat{\mathbf{A}}_n^{-t}$. This means, for example, that it is possible to derive $\hat{\mathbf{A}}_n$ and $\hat{\mathbf{B}}_n$ by first applying the Schur algorithm to $\mathbf{T}_n$ to determine the reflection coefficients (i.e., the lattices), and then using the Levinson recursions (3.31) or (3.33) to evaluate the columns for $\hat{\mathbf{A}}_n$ and $\hat{\mathbf{B}}_n$ (that is, apply impulses to all the lattices inputs). This is a particular case (less efficient than using the Levinson recursions with inner products) of the extended factorization algorithm that we shall develop in the next section.

Utilizing this idea, it is possible (but not as efficient) to find the factorization of $\mathbf{R}_n$ starting with a Levinson algorithm, as we describe next. Assume we have a non-Toeplitz $\mathbf{R}_n$ and that we also know its "hidden" $\mathbf{T}_n = [c_{i-j}]$ as in (1.15). We can "construct the lattices" (i.e. find the reflection coefficients) using the Levinson algorithm for $\mathbf{T}_n$. Then we can attach prefilters $\hat{h}_{(0)}(z)$ to each of the two inputs of the left lattice, and prefilters $h_{(0)}(z)$ to each of the two inputs of the right lattice. If the four generating vectors of the Toeplitz matrix

$$\hat{\tilde{u}} = [1, c_{-1}, \ldots, c_{-n}], \qquad \hat{\tilde{v}} = [0, c_{-1}, \ldots, c_{-n}],$$
$$\hat{u} = [1, c_1, \ldots, c_n], \qquad \hat{v} = [0, c_1, \ldots, c_n], \tag{3.48}$$

are applied appropriately through the prefiltered inputs, the lattices will produce the Schur variables for $\mathbf{R}_n$ as in Figure 1. Here

$$\tilde{h}_0(z) = \tilde{u}_{(0)}(z) - \tilde{v}_{(0)}(z), \qquad h_0(z) = u_{(0)}(z) - v_{(0)}(z). \tag{3.49}$$

The validity of this (hypothetical) scheme is shown by substitution of $\mathbf{T}_n$

from (1.8) into (1.15):

$$\mathbf{R}_n = L\left(\tilde{\mathbf{h}}_{0:n}\right)\left\{L\left(\hat{\tilde{\mathbf{u}}}_{(0)}\right)L^t\left(\hat{\mathbf{u}}_{(0)}\right) - L\left(\hat{\tilde{\mathbf{v}}}_{(0)}\right)L^t\left(\hat{\mathbf{v}}_{(0)}\right)\right\}L^t\left(\mathbf{h}_{0:n}\right)$$

$$= L\left(\tilde{\mathbf{h}}_{0:n} * \hat{\tilde{\mathbf{u}}}_{(0)}\right)L^t\left(\mathbf{h}_{0:n} * \hat{\mathbf{u}}_{(0)}\right) - L\left(\tilde{\mathbf{h}}_{0:n} * \hat{\tilde{\mathbf{v}}}_{(0)}\right)L^t\left(\mathbf{h}_{0:n} * \hat{\mathbf{v}}_{(0)}\right). \quad (3.50)$$

Here $*$ denotes convolution and we used the fact that

$$L(\mathbf{x}_n)L(\mathbf{y}_n) = L(\mathbf{w}_n) \qquad \text{with} \quad \mathbf{w}_n = \mathbf{x}_n * \mathbf{y}_n = \mathbf{y}_n * \mathbf{x}_n, \quad (3.51a)$$

so that if

$$x_n(z) = \sum_0^n x_i z^i, \qquad y_n(z) = \sum_0^n y_i z^i, \quad (3.51b)$$

then

$$w_n(z) = x_n(z)y_n(z), \qquad w_n(z) = \sum_0^n w_i z^i, \qquad w_{nk} = \sum_{i=0}^{k} x_{n,i}y_{n,k-i}. \quad (3.51c)$$

Thus, by comparison of (3.50) with the expression (1.8), we see that

$$\tilde{u}_{(0)} = \tilde{h}_{(0)}(z)\hat{\tilde{u}}_{(0)}(z), \qquad \tilde{v}_{(0)}(z) = \tilde{h}_{(0)}(z)\hat{\tilde{v}}_{(0)}(z), \quad (3.52a)$$

$$u_{(0)} = h_{(0)}(z)\hat{u}_{(0)}(z), \qquad v_{(0)}(z) = h_{(0)}(z)\hat{v}_{(0)}(z), \quad (3.52b)$$

and the above interpretation follows.

In the next section we shall see how fast algorithms for the inversion of $\mathbf{R}_n$ that are not restricted (like the Levinson algorithm) to Toeplitz or admissible QT matrices emerge from the interpretation as convolutions of various relations between the variables in the Toeplitz and non-Toeplitz algorithms.

4. INVERSION OF GENERAL QT MATRICES

We saw that the Levinson algorithm produces the triangular factorization of the inverse of a Toeplitz matrix

$$\mathbf{T}_n^{-1} = \hat{\mathbf{A}}_n \mathbf{D}_n^{-1} \hat{\mathbf{B}}_n^t \quad (4.1)$$

at most for the admissible (1.18) subclass of QT matrices. An alternative inversion formula for non-Hermitian Toeplitz matrices is

$$\mathbf{T}_n^{-1} = \frac{1}{D_n}\left\{ L\left(\overleftrightarrow{\hat{\mathbf{b}}}_n\right) L^t\left(\overleftrightarrow{\hat{\mathbf{a}}}_n\right) - L(\downarrow \hat{\mathbf{a}}_n) L^t(\downarrow \hat{\mathbf{b}}_n)\right\}, \tag{4.2}$$

which was proposed by Gohberg and Semencul (for nonsymmetric Toeplitz matrices from the first). In this section we describe triangular factorizations, Gohberg-Semencul (GS) type formulas, and fast algorithms for the inversion of general (strongly regular) QT matrices.

4.1. Triangular Factorizations

We return to our initial problem and would like to have fast (i.e. order n^2 elementary operations) solutions to

$$\mathbf{b}_m^t \mathbf{R}_m = [0, \ldots, 0, D_m], \qquad \mathbf{R}_m \mathbf{a}_m = [0, \ldots, 0, D_m]^t \tag{4.3a,b}$$

for $m = 0, \ldots, n$. These solutions provide the factorized inversion

$$\mathbf{R}_n^{-1} = \mathbf{A}_n \mathbf{D}_n^{-1} \mathbf{B}_n^t, \tag{4.4}$$

where $\mathbf{A}_n$ and $\mathbf{B}_n$ are upper triangular matrices whose $(i+1)$th column begins with the entries of the vector $\mathbf{a}_i$ and $\mathbf{b}_i$, respectively, and whose remaining entries are zeros; see (1.4). Using (1.15), the inverse of $\mathbf{R}_n$ is also given by

$$\mathbf{R}_n^{-1} = L^{-t}(\mathbf{u}_{(0)} - \mathbf{v}_{(0)}) \mathbf{T}_n^{-1} L^{-1}(\tilde{\mathbf{u}}_{(0)} - \tilde{\mathbf{v}}_{(0)}). \tag{4.5}$$

We introduce some notation to simplify our subsequent derivations. Let

$$\mathbf{h}_{0:n} = \mathbf{u}_{(0)} - \mathbf{v}_{(0)} = [1, h_1, \ldots, h_n]^t, \qquad \tilde{\mathbf{h}}_{0:n} = \tilde{\mathbf{u}}_{(0)} - \tilde{\mathbf{v}}_{(0)} = [1, \tilde{h}_1, \ldots, \tilde{h}_n]^t, \tag{4.6}$$

and denote the lower triangular Toeplitz matrices inverses of $L(\mathbf{h}_{0:n})$ and $L(\mathbf{h}_{0:n})$ by

$$L(\Gamma_{0:n}) = L^{-1}(\mathbf{h}_{0:n}), \qquad L\left(\tilde{\Gamma}_{0:n}\right) = L^{-1}\left(\tilde{\mathbf{h}}_{0:n}\right), \tag{4.7}$$

$$\Gamma_{0:n} = [\gamma_{00}, \gamma_{01}, \ldots, \gamma_{0,n}]^t, \qquad \tilde{\Gamma}_{0:n} = [\tilde{\gamma}_{00}, \tilde{\gamma}_{01}, \ldots, \tilde{\gamma}_{0,n}]^t. \tag{4.8}$$

We note that an inverses such as $L^{-1}(\mathbf{h}_{0:n})$ requires only $O(0.5n^2)$ using the (polynomial inversion) algorithm

$$\gamma_{0,0} = 1, \qquad \gamma_{0,m} = -\sum_{i=0}^{m-1} \gamma_{0,i} h_{m-i}, \quad m = 1, \ldots, n. \tag{4.9}$$

Setting (4.7) and (4.1) into (4.5), we have

$$\mathbf{R}_n^{-1} = L^t(\Gamma_{0:n})\hat{\mathbf{A}}_n \mathbf{D}_n^{-1} \hat{\mathbf{B}}_n L(\tilde{\Gamma}_{0:n}). \tag{4.10}$$

In this factorization of $\mathbf{R}_n^{-1}$, the product of the two upper (lower) triangular unit diagonal matrices on the left (right) is itself a matrix of the same type. Therefore comparison with (4.4) reveals that

$$\mathbf{A}_n = L^t(\Gamma_{0:n})\hat{\mathbf{A}}_n, \qquad \mathbf{B}_n = L^t(\tilde{\Gamma}_{0:n})\hat{\mathbf{B}}_n. \tag{4.11}$$

An algorithm to find the vectors $\mathbf{a}_m$ and $\mathbf{b}_m$ could consist of the following steps:

(a) Use the Schur algorithm to find the parameters ξ_m, k_m, $m = 1, \ldots, n$. (Namely, *determine* the pair of lattices of Figure 2.)

(b) Find the inverses $L(\Gamma_{0:n})$ and $L(\tilde{\Gamma}_{0:n})$ (4.7), using (4.9).

(c) Find the triangular factorization of the Toeplitz matrix (4.1) by carrying out only the recursion part in the Levinson algorithm for non-Hermitian Toeplitz matrices. [This corresponds to applying impulse inputs to the lattices determined in step (a) and reading out the state responses in front of each delay element: see Section 3.2, which also shows that it is enough to consider one of the two recursions and lattices.]

(d) Carry out the matrix multiplications (4.11).

The first three steps are of order n^2 complexity, requiring $O(2n^2)$, $O(n^2)$, and $O(2n^2)$ operations, respectively. The fourth step, however, requires order n^3 operations. (The triangular matrices $\hat{\mathbf{A}}_n$ and $\hat{\mathbf{B}}_n$ are not Toeplitz). Consequently, the scheme, as a whole, is not a *fast* algorithm for (4.4). Apparently, the fault lies in the fact that the algorithm "waits" with the products (4.11) till after $\hat{\mathbf{A}}_n$ and $\hat{\mathbf{B}}_n$ have been found. Fortunately, after getting a better insight into the inherent meaning of the products (4.11) (the transmission line presentation becomes most helpful in this respect), we can now show that steps (c) and (d) are replaceable by an algorithm that produces $\mathbf{A}_m$ and $\mathbf{B}_m$, $m = 0, \ldots, n$, directly, and in order n^2 operations.

We rewrite the expressions that (4.11) induces on all the leading submatrices, for $m = 0, \ldots, n$, in the form [we recall the property (1.12) for

Toeplitz matrices]

$$\mathbf{JA}_m = L(\Gamma_{0:m})\mathbf{J}\hat{\mathbf{A}}, \qquad \mathbf{JB}_m = L(\tilde{\Gamma}_{0:m})\mathbf{J}\hat{\mathbf{B}}. \tag{4.12}$$

The last columns in the right hand side are the reversed vectors $\vec{\mathbf{a}}_m = \mathbf{Ja}_m$ and $\vec{\mathbf{b}}_m = \mathbf{Jb}_m$. The equation makes it clear that the reciprocal of the solution vectors of the QT matrix are related to corresponding reciprocals of the solution vectors of the Toeplitz matrix by the convolutions

$$[1, a_{m,m-1}, \ldots, a_{m,0}] = [\gamma_{00}, \gamma_{01}, \ldots, \gamma_{0,m}] * [1, \hat{a}_{m,m-1}, \ldots, \hat{a}_{m,0}], \tag{4.13a}$$

$$[1, b_{m,m-1}, \ldots, b_{m,0}] = [\tilde{\gamma}_{00}, \tilde{\gamma}_{0,1}, \ldots, \tilde{\gamma}_{0,m}] * [1, \hat{b}_{m,m-1}, \ldots, \hat{b}_{m,0}] \tag{4.13b}$$

for all $m = 0, \ldots, n$. We can regard (4.13a) as describing the response $\mathbf{a}_m$ of a linear time-invariant filter whose impulse response is $\vec{\mathbf{a}}_m$, to the input sequence $\Gamma_{0:m}$. However, recalling the lattice interpretation of the Levinson algorithm for a Toeplitz matrix (see Section 3.4), we have that along the upper line in the right filter the response to the impulse input is $\beta_m(z) = \vec{a}_m(z)$. In other words, the points in Figure 2 denoted by $\beta_m(z)$, $m = 1, \ldots, n$, provide filters that realize the convolutions (4.13a). Thus, applying the sequence $\Gamma_{0:n}$ as input to the right lattice will produce (in reverse order) the elements of $\mathbf{a}_m$, at the upper line of section $m+1$, for $m = 0, \ldots, n$. Similarly, (4.13b) can be interpreted as responses to inputs $\tilde{\Gamma}_{0:m}$ of filters whose impulse responses are $\vec{\mathbf{b}}_n$; such filters are provided by the transfer functions from the inputs to points along the upper line of the left lattice, which were denoted in Figure 2 by $\alpha_n(z)$, because in the Toeplitz case $\alpha_m(z) = \vec{b}_m(z)$. Therefore, applying the sequence $\tilde{\Gamma}_{0:n}$ to the left filter will produce along the upper line the columns (in reverse order) of $\mathbf{B}_n$.

We can now easily formalize the above findings into an explicit algorithm. They suggest taking the Levinson recursions (1.19a,b) and initiating the recursion (1.19a), which correspond to the left lattice, by the sequence $\tilde{\Gamma}_{0:n}$, while the recursion (1.19b), which corresponds to the right lattice, has to be initiated by the sequence $\Gamma_{0:n}$. Subsequently, it will be possible to read out the columns of $\mathbf{A}_n$ and $\mathbf{B}_n$ from the new propagated variables, and the above argument also indicates the proper way of doing this reading. We choose again to write the recursions in polynomial form and introduce the following variables:

$$\gamma_{(m)}(z) = \sum_{i=0}^{n} \gamma_{m,i} z^i \qquad \tilde{\epsilon}_{(m)}(z) = \sum_{i=0}^{n} \epsilon_{m,i} z^i, \tag{4.14a}$$

$$\tilde{\gamma}_{(m)}(z) = \sum_{i=0}^{n} \tilde{\gamma}_{m,i} z^i \qquad \tilde{\epsilon}_{(m)}(z) = \sum_{i=0}^{n} \tilde{\epsilon}_{m,i} z^i. \tag{4.14b}$$

The recursions are

$$\begin{bmatrix} \tilde{\gamma}_{(m)}(z) \\ \tilde{\epsilon}_{(m)}(z) \end{bmatrix} = \begin{bmatrix} 1 & -k_m \\ -\xi_m & 1 \end{bmatrix} \begin{bmatrix} z & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{\gamma}_{(m-1)}(z) \\ \tilde{\epsilon}_{(m-1)}(z) \end{bmatrix},$$

$$\tilde{\gamma}_{(0)}(z) = \tilde{\Gamma}_{0:n}(z), \quad \tilde{\epsilon}_{(0)}(z) = \tilde{\Gamma}_{0:n}(z), \quad (4.15a)$$

$$\begin{bmatrix} \gamma_{(m)}(z) \\ \epsilon_{(m)}(z) \end{bmatrix} = \begin{bmatrix} 1 & -\xi_m \\ -k_m & 1 \end{bmatrix} \begin{bmatrix} z & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \gamma_{(m-1)}(z) \\ \epsilon_{(m-1)}(z) \end{bmatrix},$$

$$\gamma_{(0)}(z) = \Gamma_{0:n}(z), \quad \epsilon_{(0)}(z) = \Gamma_{0:m}(z), \quad (4.15b)$$

where the initial conditions are the polynomials associated with the vectors (4.8),

$$\Gamma_{0:n}(z) = [1, z, \ldots, z^n]\Gamma_0, \qquad \tilde{\Gamma}_{0:n}(z) = [1, z, \ldots, z^n]\tilde{\Gamma}_0. \quad (4.15c)$$

We shall refer to (4.14)–(4.15) as the *recursive convolution* algorithm. We emphasize that all polynomials are of degree n without coefficients that are zero, by structure. The recursive convolution requires $O(4n^2)$ multiplications. Therefore it is fast, and it involves twice the computation in the Schur algorithm, which has variables with increasing numbers of leading zeros.

The recursive convolution algorithm produces, in reversed order, the vectors $\mathbf{a}_m$ and $\mathbf{b}_m$ for the solutions of (4.3) as the first $m+1$ coefficients of, respectively, $\epsilon_{(m)}(z)$ and $\tilde{\epsilon}_{(m)}(z)$. This follows from the preceding discussion, after noticing that these variables take the place of $\beta_m(z)$ and $\alpha_m(z)$, respectively, in the Levinson recursions (1.19). Therefore, the matrices $\mathbf{A}_n$ and $\mathbf{B}_n$ for the factorization (4.4) of a general QT matrix are given by

$$\mathbf{A}_n = \begin{bmatrix} 1 & \epsilon_{11} & \cdots & \epsilon_{n-1,n-1} & \epsilon_{n,n} \\ 0 & 1 & \cdots & \epsilon_{n-1,n-2} & \epsilon_{n,n-1} \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & \epsilon_{n,1} \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix}, \quad (4.16a)$$

$$\mathbf{B}_n = \begin{bmatrix} 1 & \tilde{\epsilon}_{11} & \cdots & \tilde{\epsilon}_{n-1,n-1} & \tilde{\epsilon}_{n,n} \\ 0 & 1 & \cdots & \tilde{\epsilon}_{n-1,n-2} & \tilde{\epsilon}_{n,n-1} \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & \tilde{\epsilon}_{n,1} \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix}. \quad (4.16b)$$

The complete fast algorithm for QT matrices therefore consists of the following three parts:

(i) The Schur algorithm (2.3) that produces the reflection coefficients ξ_m, k_m, $m = 0,\dots,n$. It also produces the triangular factorization of $\mathbf{R}_m$, $m = 0,\dots,n$. It requires $O(2n^2)$ elementary operations.

(ii) Finding initial conditions for the recursive convolution. This process uses (4.6)–(4.8) to produce the input sequences (4.15c). It requires $O(n^2)$ elementary operations.

(iii) The recursive convolution algorithm (4.15) that produces the triangular factorization for $\mathbf{R}_m^{-1}$, $m = 0,\dots,n$. It also produces the vectors for Gohberg-Semencul type inversion forms for $\mathbf{R}_m^{-1}$, as will be derived next.

We shall refer to this algorithm as the *extended quasi-Toeplitz factorization algorithm*. The algorithm is also summarized in Table 1.

4.2. *Gohberg-Semencul Type Inversion Formulas*

We proved in Section 1 [see (1.13)] that the inverse of the reversed matrix $\mathbf{J}\mathbf{R}_n\mathbf{J}$ has displacement rank 2 and therefore there exists a GS type formula for its inverse. To derive such a formula, we have from (4.5) and the property (1.12) of Toeplitz matrices that

$$\overleftrightarrow{\mathbf{R}}_n^{-1} = L^t(\Gamma_{0:n})\mathbf{T}_n^{-t}L\big(\tilde{\Gamma}_{0:n}\big). \tag{4.17}$$

Using the GS formula (1.14), we obtain

$$\overleftrightarrow{\mathbf{R}}_n^{-1} = \frac{1}{D_n}\Big\{L(\Gamma_{0:n})L\big(\overleftrightarrow{\hat{\mathbf{a}}}_n\big)L^t\big(\overleftrightarrow{\hat{\mathbf{b}}}_n\big)L^t\big(\tilde{\Gamma}_{0:n}\big) - L(\Gamma_{0:n})L\big(\downarrow\hat{\mathbf{b}}_n\big)L^t\big(\downarrow\hat{\mathbf{a}}_n\big)L^t\big(\tilde{\Gamma}_{0:n}\big)\Big\}. \tag{4.18}$$

The relations between products of two lower triangular Toeplitz matrices and the convolution (3.51) can be used. If we define the convolutions

$$\mathbf{e}_n = \Gamma_{0:n} * \overleftrightarrow{\hat{\mathbf{a}}}_n, \qquad \mathbf{g}_n = \Gamma_{0:n} * \hat{\mathbf{b}}_n, \qquad \tilde{\mathbf{e}}_n = \tilde{\Gamma}_{0:n} * \overleftrightarrow{\hat{\mathbf{b}}}_n, \qquad \tilde{\mathbf{g}}_n = \tilde{\Gamma}_{0:n} * \hat{\mathbf{a}}_n, \tag{4.19}$$

we can write (4.18) as

$$\overleftrightarrow{\mathbf{R}}_n^{-1} = \frac{1}{D_n}\big\{L(\mathbf{e}_n)L^t(\tilde{\mathbf{e}}_n) - L(\downarrow\mathbf{g}_n)L^t(\downarrow\tilde{\mathbf{g}}_n)\big\}. \tag{4.20}$$

We want to find an algorithm to derive the four vectors (4.19), and we return for this purpose to the transmission line picture.[4] We already saw that the convolutions $\mathbf{e}_n$ and $\tilde{\mathbf{e}}_n$ can be read out as the outputs of the upper lines of the right and left lattices, respectively, when fed by $\Gamma_{0:n}$ and $\tilde{\Gamma}_{0:n}$, respectively. Can the remaining two vectors in (4.19) be similarly identified? The vector $\mathbf{g}_n$ $[\tilde{\mathbf{g}}_n]$ would appear as the response to the input $\Gamma_{0:n}$ $[\tilde{\Gamma}_{0:n}]$ at a point at which the response to an impulse input would be $\hat{\mathbf{b}}_n$ $[\hat{\mathbf{a}}_n]$. A second glance at Figure 1 reveals that the right (left) lattice with $\Gamma_{0:n}$ $[\tilde{\Gamma}_{0:n}]$ applied to its input, which produces $\mathbf{e}_n$ $[\tilde{\mathbf{e}}_n]$ at the upper output, because the transfer function from the input to there is $\vec{a}_n(z)$ $[\vec{b}_n(z)]$, necessarily produces $\mathbf{g}_n$ $[\tilde{\mathbf{g}}_n]$ at the same time at the lower output, a point characterized by the transfer function $\hat{b}_n(z)$ $[\hat{a}_n(z)]$. Finally, it is also clear from the recursive and nested nature of all the structures involved that by replacing n by any $m \leqslant n$ in (4.17)–(4.20), the four vectors that determine the submatrix $\overleftrightarrow{\mathbf{R}}_m^{-1}$ can be read off as the respective outputs of the mth sections of the two lattices.

We conclude that the four vectors required for the GS formula for $\overleftrightarrow{\mathbf{R}}_n^{-1}$, as well as for all its submatrices $\overleftrightarrow{\mathbf{R}}_m^{-1}$, are provided by the recursive convolution algorithm (4.15) and given, for $m = 0, \ldots, n$, by

$$\mathbf{e}_m = [1, \epsilon_{m,1}, \ldots, \epsilon_{m,m}]^t, \qquad \downarrow \mathbf{g}_m = [0, \gamma_{m,0}, \ldots, \gamma_{m,m-1}]^t,$$

$$\tilde{\mathbf{e}}_m = [1, \tilde{\epsilon}_{m,1}, \ldots, \tilde{\epsilon}_{m,m}]^t, \qquad \downarrow \tilde{\mathbf{g}}_m = [0, \tilde{\gamma}_{m,0}, \ldots, \tilde{\gamma}_{m,m-1}]^t. \quad (4.21)$$

We note that a lower-times-upper formula like (4.20) can also be written as an upper-times-lower formula of the reversed matrix, which in this case suggests an inversion formula for $\mathbf{R}_n$:

$$\mathbf{R}_n^{-1} = \frac{1}{D_n} \left\{ L^t(\mathbf{e}_n) L(\tilde{\mathbf{e}}_n) - L^t(\downarrow \mathbf{g}_n) L(\downarrow \tilde{\mathbf{g}}_n) \right\}. \quad (4.22)$$

Two other inversion formulas can be obtained by modifying the original GS formula (4.2) using the property (1.12) of the Toeplitz matrix before substitution into (4.5).

As a final remark, we note that GS type formulas require order n storage, compared to order n^2 in other forms of the inverses. Applications of fast

[4] One straightforward way to calculate the required four vectors could be to follow steps (a)–(c) in Section 4.1 and replace step (d) there by the convolutions (4.19). This time the last step is "fast," because the four convolutions in (4.19) require $O(2n^2)$. The four vectors for the inverse (4.20) are found in this way in $O(7n^2)$ operations, but inverses for $\mathbf{R}_m$, $m < n$, are not obtained in the process.

TABLE 1
THE EXTENDED (NON-HERMITIAN) QUASI-TOEPLITZ FACTORIZATION ALGORITHM ASSUMPTIONS

The QT matrix $\mathbf{R}_n$ (1.8) is strongly regular and is given in terms of its four vectors (1.9) with $\tilde{u}_{0,i}, u_{0,i}, \tilde{v}_{0,i}, v_{0,i}$, $i = 0, 1, \ldots, n$ ($\tilde{u}_{0,0} = u_{0,0} = 1$, $\tilde{v}_{0,0} = v_{0,0} = 0$).

(i) The Schur algorithm.

Do for $m = 1, \ldots, n$

$$\xi_m = \frac{\tilde{v}_{m-1,m}}{\tilde{u}_{m-1,m-1}}, \qquad k_m = \frac{v_{m-1,m}}{u_{m-1,m-1}}$$

Do for $= m, \ldots, n$

$$\tilde{u}_{m,i} = \tilde{u}_{m-1,i-1} - k_m \tilde{v}_{m-1,i}$$

$$\tilde{v}_{m,i} = -\xi_m \tilde{u}_{m-1,i-1} + \tilde{v}_{m-1,i}$$

$$u_{m,i} = u_{m-1,i-1} - \xi_m v_{m-1,i}$$

$$v_{m,i} = -k_m u_{m-1,i-1} + v_{m-1,i}$$

(ii) Lower triangular Toeplitz inversions.

Do for $i = 0, \ldots, n$

$$\tilde{h}_i = \tilde{u}_{0,i} - \tilde{v}_{0,i}$$

$$h_i = u_{0,i} - v_{0,i}$$

Do for $m = 1, \ldots, n$

$$\tilde{\gamma}_{0,m} = \sum_{i=0}^{m-1} \tilde{\gamma}_{0,i} \tilde{h}_{m-i}, \qquad \tilde{\gamma}_{0,0} = 1$$

$$\gamma_{0,m} = \sum_{i=0}^{m-1} \gamma_{0,i} h_{m-i}, \qquad \gamma_{0,0} = 1$$

TABLE 1 (Continued)

(iii) The recursive convolution Algorithm.

Set, for $i = 0, \ldots, n$, $\tilde{\epsilon}_{0,i} = \tilde{\gamma}_{0,i}$, $\epsilon_{0,i} = \gamma_{0,i}$.
Do for $m = 1, \ldots, n$

$$\tilde{\gamma}_{m,i} = \tilde{\gamma}_{m-1,i-1} - k_m \tilde{\epsilon}_{m-1,i}$$

$$\tilde{\epsilon}_{m,i} = -\xi_m \tilde{\gamma}_{m-1,i-1} + \tilde{\epsilon}_{m-1,i}$$

$$\gamma_{m,i} = \gamma_{m-1,i-1} - \xi_m \epsilon_{m-1,i}$$

$$\epsilon_{m,i} = -k_m \gamma_{m-1,i-1} + \epsilon_{m-1,i}$$

REMARKS.

(1) The entries for the diagonal matrix $\mathbf{D}_n$ can be read from $u_{m,m}$ or $\tilde{u}_{m,m}$; see (2.4a,b).

(2) For the triangular factorization of $\mathbf{R}_m$, $m = 1, \ldots, n$, see (2.11) and (2.4a,b).

(3) For the triangular factorization of $\mathbf{R}_m^{-1}$, $m = 1, \ldots, n$, see (4.4) and (4.16).

(4) For GS type inversions $\mathbf{R}_m^{-1}$, $m = 1, \ldots, n$, see (4.22) and (4.21).

(5) For Hermitian QT matrices, set $\xi_m = k_m^*$ and perform only (or skip all) the calculations that involve variables with tilde $\tilde{}$.

TABLE 2
COMPUTATION COUNTS FOR VARIOUS FACTORIZATION ALGORITHMS FOR QUASI-TOEPLITZ MATRICES

	Count					
	Hermitian			Non-Hermitian		
Algorithm	Toeplitz	Admiss. QT	QT	Toeplitz	Admiss. QT	QT
Levinson	$O(n^2)$	$O(1.5n^2)$	N.A.	$O(2n^2)$	$O(3n^2)$	N.A.
Schur	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(2n^2)$	$O(2n^2)$	$O(2n^2)$
Extended QT factorization	$O(1.5n^2)$	$O(3.5n^2)$	$O(3.5n^2)$	$O(3n^2)$	$O(7n^2)$	$O(7n^2)$

inversion algorithms usually involve large size matrices and often impose memory and storage limitations. Both the triangular and the GS type formulas have order n requirement of memory for execution and order n^2 complexity. However, the GS type formulas require order n storage, whereas the triangular factorization's storage requirement is order n^2, which in some applications may become a disadvantage and in certain cases may even exceed the capacity of a computer.

5. CONCLUDING REMARKS

We have presented in this paper a generalization of known fast algorithms for the factorization and inversion of QT Hermitian matrices to non-Hermitian matrices, as well as new inversion formulas for such matrices. We have also shown that the fast factorization algorithms for non-Hermitian QT matrices are related to a pair of lattices, reducing to the familiar one lattice when the matrix becomes Hermitian. The use of these lattices makes the theory more intuitive and enhances the understanding of the algorithms and their possible applications.

The *extended QT factorization algorithm* presented in Section 4 and summarized in Table 1 is a comprehensive fast algorithm which computes virtually any factorization that may be required in applications of QT matrices: the triangular factorization for $\mathbf{R}_n$, the triangular factorization of the inverses of $\mathbf{R}_m$ for all $m = 1, \ldots, n$, and the GS type inverses of $\mathbf{R}_m$ for all $m = 1, \ldots, n$. Occasionally, however, depending on the required factorization, it may be sufficient and require less computation to carry out the Levinson algorithm or just the Schur algorithm. A complexity account for the various algorithms discussed in this paper is given in Table 2.

All the algorithms for the factorization and the inversion of the QT matrix were given in the form of two-term recursions. It is always possible to replace a two-term recursion by a three-term recursion for one of the variables and an auxiliary equation that may be used to recover the other (or by two three-term recursions, one for each variable) [3]. It can be shown that in fact there exist three-term versions for all the algorithms that appear in this paper. One advantage of three-term versions is that they can be used to relax the strong regularity conditions on the applicability of the algorithms. We shall discuss this topic elsewhere.

The relations between the polynomials in the Levinson recursions for Hermitian Toeplitz matrices and the polynomials orthogonal on the unit circle, studied by Szego, Geronimus, and others, are well known [22, 10]. The extension of the theory to non-Hermitian Toeplitz matrices involves biorthog-

onal polynomials and corresponding Christoffel-Darboux formulas; it was introduced by Baxter [2] and led to further research by him and others: see [14] and references therein. Concluding the discussion of the interconnections between biorthogonality, non-Hermitian Toeplitz matrices, and generalized Christoffel-Darboux formulas for reproducing kernels, the authors in [14] expressed the hope that a generalization of such connections to non-Toeplitz matrices would be discovered. The theory developed in this paper is a further contribution to the several results that have been discovered since then by using the concept of displacement structure to go beyond the purely Toeplitz case.

APPENDIX A. PROOF OF LEMMA 1 (SECTION 3.1)

We shall prove (3.11), (3.13) and (3.15). The proofs for (3.12), (3.14) and (3.16) can be deduced by obvious dual arguments.

(3.11): The first row of (1.2a) implies $[1, u_{01}, \ldots, u_{0m}]\mathbf{a}_m = 0$. Using then (1.18a) and (3.7) yields

$$0 = [1,0,\ldots,0]\mathbf{a}_m + [0, v_{01}, \ldots, v_{0m}]\mathbf{a}_m\alpha_0 = -k_m\alpha_0 + [0, v_{01}, \ldots, v_{0m}]\mathbf{a}_m\alpha_0$$

and (3.11) follows.

(3.13): Evidently (3.13) holds for $m = 0$. To show that if (3.13) holds for $m = n - 1$, then it holds for $m = n$, we use (3.2b) to calculate

$$\begin{aligned}[1, u_{01}, \ldots, u_{0m}]\alpha_m &\underset{(1.18a)}{=} -\xi_n\alpha_0[v_{01}, \ldots, v_{0m}]\mathbf{a}_{m-1} + [1, u_{01}, \ldots, u_{0m}]\alpha_{m-1} \\ &\underset{(3.13)}{=} -\xi_n\alpha_0[v_{01}, \ldots, v_{0n}]\mathbf{a}_{n-1} + \alpha_0 D_{n-1} \\ &\underset{(3.5a)}{=} -\alpha_0\xi_n k_n + \alpha_0 D_{n-1} = \alpha_0 D_{n-1}(1 - \xi_n k_n) \underset{(3.6)}{=} \alpha_0 D_n.\end{aligned}$$

(3.15): Follows immediately from (3.13) using (1.18a) and (3.9).

APPENDIX B. PROOF OF PROPOSITION 1 (SECTION 3.1)

We first prepare ourselves with the following lemma.

LEMMA 2. *If the relations* (3.2)–(3.6) *produce the solutions to* (3.1) *for* $m = 1, \ldots, n-1$, *then*

$$\begin{bmatrix} \mathbf{0}_{n-2} \\ D_n \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{n-2} \\ D_{n-1} \end{bmatrix} - k_n \begin{bmatrix} \tilde{v}_{01} \\ \vdots \\ \tilde{v}_{0n} \end{bmatrix} - k_n \mathbf{p}_{n-1}, \tag{B.1}$$

where we define

$$\mathbf{p}_{n-1} := \mathbf{R}_{n-1} [\alpha_{n-1,1}, \ldots, \alpha_{n-1,n-1}, 0]^t \tag{B.2}$$

and $\alpha_{n-1,i}$ *are elements of* $\boldsymbol{\alpha}_{n-1}$. *Similarly,*

$$[\mathbf{0}_{n-2}^t, D_n] = [\mathbf{0}_{n-2}^t, D_{n-1}] - \xi_n [v_{01}, \ldots, v_{0n}] - \xi_n \mathbf{q}_{n-1}, \tag{B.3}$$

where

$$\mathbf{q}_{n-1} := [\beta_{n-1,1}, \ldots, \beta_{n-1,n-1}, 0] \mathbf{R}_{n-1} \tag{B.4}$$

and $\beta_{n-1,i}$ *are elements of* $\boldsymbol{\beta}_{n-1}$.

Proof. Denote for each $m \leqslant n$

$$\mathbf{x}_m := \begin{bmatrix} \tilde{v}_{01} \\ \vdots \\ \tilde{v}_{0,m+1} \end{bmatrix} + \mathbf{R}_m \begin{bmatrix} \alpha_{m1} \\ \vdots \\ \alpha_{mm} \\ 0 \end{bmatrix} \tag{B.5}$$

with $\mathbf{x}_m = [x_{m0}, \ldots, x_{mm}]^t$ ($\mathbf{x}_0 = \tilde{v}_{01}$). The identity that we have to show, (B.1), becomes

$$\begin{bmatrix} \mathbf{0}_{n-2} \\ D_{n-1} \end{bmatrix} - k_n \mathbf{x}_{n-1} = \begin{bmatrix} \mathbf{0}_{n-2} \\ D_n \end{bmatrix}. \tag{B.6}$$

It is easy to check that if

$$\hat{\mathbf{x}}_m := [x_{m0}, \ldots, x_{m,m-1}] = \mathbf{0}_{m-1}, \qquad x_{mm} = \xi_{m+1} D_m \tag{B.7a,b}$$

hold for $m = n-1$, then (B.6) follows. Therefore, we proceed to prove that,

under the assumptions of Lemma 2, (B.7) holds for all $m \leqslant n-1$. First we note that (B.7) holds for $m=0$; then, assuming it holds for $m=n-2$, we evaluate $\hat{\mathbf{x}}_{n-1}$. Taking the update for α_{n-1} in vector form (3.2b),

$$\alpha_{n-1} = \xi_{n-1}\begin{bmatrix} \mathbf{0} \\ \mathbf{a}_{n-2} \end{bmatrix} + \begin{bmatrix} \alpha_{n-2} \\ \mathbf{0} \end{bmatrix},$$

and, after deleting first row, and substituting the result into $\hat{\mathbf{x}}_{n-1}$, one has

$$\hat{\mathbf{x}}_{n-1} = \begin{bmatrix} \tilde{v}_{01} \\ \vdots \\ \tilde{v}_{0,n-1} \end{bmatrix} - \xi_{n-1}\begin{bmatrix} \mathbf{0}_{n-2} \\ D_{n-2} \end{bmatrix} + \mathbf{R}_{n-2}\begin{bmatrix} \alpha_{n-2,1} \\ \vdots \\ \alpha_{n-2,n-2} \\ 0 \end{bmatrix}, \tag{B.8}$$

where (3.1b) was used in the second summand. Therefore,

$$\hat{\mathbf{x}}_{n-1} = \mathbf{x}_{n-2} - \xi_{n-1}\begin{bmatrix} \mathbf{0}_{n-2} \\ D_{n-2} \end{bmatrix} = \mathbf{0}_{n-2}, \tag{B.9}$$

where we first recognized in (B.8) $\mathbf{x}_{n-2}$, as defined in (B.5), and obtained the equality to zero from the assumption that (B.7) holds for $m=n-2$. This furnishes (B.7a) for $m=n-1$. To show (B.7b) for $m=n-1$, we multiply the two sides of equation (B.5) by the row vector $\mathbf{b}_{n-1}^t$; the left hand side gives $\mathbf{b}_{n-1}^t[\hat{\mathbf{x}}_{n-1}, x_{n-1,n-1}]^t = x_{n-1,n-1}$, using $\hat{\mathbf{x}}_{n-1}=0$ from (B.9) and $b_{n-1,n-1}=1$ from (3.8); on the right hand side, the inner product indeed yields the required result for (B.7b) to hold for $m=n-1$,

$$\mathbf{b}_{n-1}^t\begin{bmatrix} \tilde{v}_{01} \\ \vdots \\ \tilde{v}_{0,n} \end{bmatrix} + \mathbf{b}_{n-1}^t\mathbf{R}_{n-1}\begin{bmatrix} \alpha_{n-1,1} \\ \vdots \\ \alpha_{n-1,n-1} \\ 0 \end{bmatrix} = \xi_n D_{n-1} + 0,$$

where we have applied (3.5b) and (3.1a). The proof for the second part of Lemma 2 follows by replacing in the above all variables by their duals in the pairs such as $\{\mathbf{a}_m, \mathbf{b}_m\}$, $\{k_m, \xi_m\}$, $\{\tilde{v}_{i,j}, v_{i,j}\}$. ■

Proof of Proposition 1. Now we are ready to prove the proposition in Section 2. First, we have to verify that the new vector $\mathbf{a}_n$, given by (3.2a),

$$\mathbf{a}_n = \begin{bmatrix} 0 \\ \mathbf{a}_{n-1} \end{bmatrix} - k_n\begin{bmatrix} \boldsymbol{\alpha}_{n-1} \\ 0 \end{bmatrix}, \tag{B.10}$$

is a solution to (3.17b). To calculate $\mathbf{R}_n\mathbf{a}_n$ we take the product of $\mathbf{R}_n$, expressed by (3.18), with each of the two summands in (B.10):

$$\mathbf{R}_n\begin{bmatrix}0\\ \mathbf{a}_{n-1}\end{bmatrix} = \tilde{\mathbf{u}}_{0:n}\mathbf{u}^t_{0:n}\begin{bmatrix}0\\ \mathbf{a}_{n-1}\end{bmatrix} - \tilde{\mathbf{u}}_{0:n}\mathbf{u}^t_{0:n}\begin{bmatrix}0\\ \mathbf{a}_{n-1}\end{bmatrix} + \begin{bmatrix}0\\ \mathbf{R}_{n-1}\mathbf{a}_{n-1}\end{bmatrix}$$

$$\underset{(1.18a)}{=} \tilde{\mathbf{u}}_{0:n}\alpha_0[v_{01},\ldots,v_{0n}]\mathbf{a}_{n-1} - \tilde{\mathbf{v}}_{0:n}[v_{01},\ldots,v_{0n}]\mathbf{a}_{n-1} + \begin{bmatrix}\mathbf{0}\\ D_{n-1}\end{bmatrix}$$

$$\underset{(3.11)}{=} \tilde{\mathbf{u}}_{0:n}\alpha_0 D_{n-1}k_n - \tilde{\mathbf{v}}_{0:n}D_{n-1}k_n + \begin{bmatrix}\mathbf{0}\\ D_{n-1}\end{bmatrix}, \tag{B.11}$$

$$\mathbf{R}_n\begin{bmatrix}\boldsymbol{\alpha}_{n-1}\\ 0\end{bmatrix} = \tilde{\mathbf{u}}_{0:n}\mathbf{u}^t_{0:n-1}\alpha_{n-1} - \tilde{\mathbf{v}}_{0:n}\mathbf{v}^t_{0:n-1}\alpha_{n-1} + \begin{bmatrix}0\\ \mathbf{p}_{n-1}\end{bmatrix}$$

$$\underset{(3.13),(3.15)}{=} \tilde{\mathbf{u}}_{0:n}\alpha_0 D_{n-1} - \tilde{\mathbf{v}}_{0:n}(D_{n-1}-1) + \begin{bmatrix}0\\ \mathbf{p}_{n-1}\end{bmatrix} \tag{B.12}$$

where $\mathbf{p}_{n-1}$ was defined in (B.2). Substitution of (B.11) and (B.12) into (B.10) multiplied by $\mathbf{R}_n$ gives

$$\mathbf{R}_n\mathbf{a}_n = \mathbf{R}_n\begin{bmatrix}0\\ \mathbf{a}_{n-1}\end{bmatrix} - k_n\mathbf{R}_n\begin{bmatrix}\boldsymbol{\alpha}_{n-1}\\ 0\end{bmatrix}$$

$$= k_n\alpha_0 D_{n-1}\tilde{\mathbf{u}}_{0:n} - k_n D_{n-1}\tilde{\mathbf{v}}_{0:n} + \begin{bmatrix}0\\ D_{n-1}\end{bmatrix}$$

$$- k_n\left\{\alpha_0 D_{n-1}\tilde{\mathbf{u}}_{0:n} - D_{n-1}\tilde{\mathbf{v}}_{0:n} + \tilde{\mathbf{v}}_{0:n} + \begin{bmatrix}\mathbf{0}\\ \mathbf{p}_{n-1}\end{bmatrix}\right\}$$

$$= \begin{bmatrix}\mathbf{0}_{n-1}\\ D_{n-1}\end{bmatrix} - k_n\tilde{\mathbf{v}}_{0:n} - k_n\begin{bmatrix}0\\ \mathbf{p}_{n-1}\end{bmatrix} = \begin{bmatrix}\mathbf{0}_{n-1}\\ D_n\end{bmatrix}.$$

The final equality was prepared in Lemma 2, (B.1). This verifies (3.17b). A complementary derivation with the dual variables shows that $\mathbf{b}_n$ solves (3.17a), and that completes the proof. ■

Y. Bistritz thanks M. M. Sondhi from AT&T Bell Labs for stimulating him to find a fast algorithm with linear storage for an inverse acoustic problem that involved a large size non-Hermitian QT matrix.

REFERENCES

1 E. H. Bareiss, Numerical solution to linear equation with Toeplitz and vector Toeplitz matrices, *Numer. Math.* 13:404–424 (1969).

2 G. Baxter, Polynomials defined by difference systems, *J. Math. Anal. Appl.* 2:223–263 (1961).

3 Y. Bistritz, H. Lev-Ari, and T. Kailath, Immittance domain Levinson algorithms, *IEEE Trans. Inform. Theory*, to appear.

4 J. M. Delosme, Algorithms for Finite Shift Rank Processes, Ph.D. Thesis, Stanford Univ., 1982.

5 Ph. Delsarte, Y. Genin, and Y. Kamp, On the class of positive definite matrices equivalent to Toeplitz matrices, in *Proceedings of the International Symposium on Mathematical Theory of Networks and Systems*, Santa Montica, Calif., 5–7 Aug. 1981, pp. 40–45.

6 Ph. Delsarte, Y. Genin, and Y. Kamp, A polynomial approach to the generalized Levinson algorithm based on the Toeplitz distance, *IEEE Trans. Inform. Theory* IT-29:268–278 (1983).

7 P. Delsarte, Y. Genin, and Y. Kamp, On the Toeplitz embedding of an arbitrary matrix, *Linear Algebra Appl.* 51:97–119 (1983).

8 B. Friedlander, M. Morf, T. Kailath, and L. Ljung, New inversion formulas for matrices in terms of their distance from Toeplitz matrices, *Linear Algebra Appl.* 27:31–60 (1979).

9 B. Friedlander, T. Kailath, M. Morf, and L. Ljung, Extended Levinson and Chandrasekhar equations for general discrete-time linear estimation problems, *IEEE Trans. Automat. Control* AC-23:653–659 (1978).

10 Ya. L. Geronimus, *Polynomial Orthogonal on a Circle and Interval* (trans. from Russian), Pergamon, 1960.

11 I. C. Gohberg and I. A. Fel'dman, *Convolution Equations and Projection Methods for Their Solution*, Trans. Math. Monographs, Vol. 41, Amer. Math. Soc., Providence, R.I., 1974.

12 G. H. Golub and C. F. Van Loan, *Matrix Computation*, Johns Hopkin U.P., Baltimore, 1983.

13 T. Kailath, A. Bruckstein, and D. Morgan, Fast matrix factorizations via discrete transmission lines, *Linear Algebra Appl.* 75:1–25 (1986).

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

15 T. Kailath, S. Kung, and M. Morf, Displacement rank of matrices and linear equations, *J. Math. Anal. Appl.* 68:395–407 (1979).

16 H. Levi-Ari, Nonstationary Lattice Filter Modeling, Ph.D. Thesis, Stanford Univ., 1983.

17 H. Lev-Ari and T. Kailath, Lattice filter parametrization of non-stationary processes, *IEEE Trans. Inform. Theory* IT-30:2–16 (1984).

18 N. Levinson, The Wiener RMS (root-mean square) error criterion in filter design and prediction, *J. Math. and Phys.* 25:261–278 (1947).

19 S. K. Rao, *Class Notes for EE478*, Stanford, 1984.

20 J. Rissanen, Algorithm for triangular factorization of block Hankel and Toeplitz matrices with applications to factoring positive matrix polynomials, *Math. Comp.* 27:147–154 (1973).
21 I. Schur, Über Potenzreihen, die in Innern des Einheitskreises beschrankt sind, *J. Reine Angew. Math.* 147:205–232 (1917).
22 G. Szego, *Orthogonal Polynomials*, Colloquium Publications, No. 23, Amer. Math. Soc., Providence, R.I., 2nd ed., 1958; 3rd ed., 1967.
23 W. F. Trench, An algorithm for the inversion of finite Toeplitz matrices, *SIAM J. Appl. Math.* 12:515–522 (1964).
24 S. Zohar, Toeplitz matrix inversion: The algorithm of W. F. Trench, *J. Assoc. Comput. Mach.* 16:591–601 (1969).
25 S. Zohar, The solution of a Toeplitz set of linear equations, *J. Assoc. Comput. Mach.* 21:272–276 (1976).

Received 27 January 1987; revised 23 June 1987