Immittance- Versus Scattering-Domain Fast Algorithms for Non-Hermitian Toeplitz and Quasi-Toeplitz Matrices*

Yuval Bistritz
Department of Electronic Systems
Tel-Aviv University
Tel-Aviv 69978, Israel

and

Hanoch Lev-Ari and Thomas Kailath
Information Systems Laboratory
Stanford University
Stanford, California 94305

Submitted by Paul A. Fuhrmann

ABSTRACT

The classical algorithms of Schur and Levinson are efficient procedures to solve sets of Hermitian Toeplitz linear equations or to invert the corresponding coefficient matrices. They propagate pairs of variables that may describe incident and scattered waves in an associated cascade-of-layered-media model, and thus they can be viewed as *scattering-domain* algorithms. It was recently found that a certain transformation of these variables followed by a change from two-term to three-term recursions results in reduction in computational complexity in the abovementioned algorithms roughly by a factor of two. The ratio of such pairs of transformed variables can be interpreted in the above layered-media model as an *im*pedance or ad*mittance*; hence the name *immittance-domain* variables. This paper provides extensions for previous immittance Schur and Levinson algorithms from Hermitian to non-Hermitian matrices. It consid-

*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 of a Chaim Weizmann postdoctoral fellowship award.

© Elsevier Science Publishing Co., Inc., 1989
655 Avenue of the Americas, New York, NY 10010 0024-3795/89/$3.50

ers both Toeplitz and *quasi-Toeplitz* matrices (matrices with certain "hidden" Toeplitz structure) and compares two- and three-term recursion algorithms in the two domains. The comparison reveals that for non-Hermitian matrices the algorithms are equally efficient in both domains. This observation adds new comprehension to the source and value of algorithms in the immittance domain. The immittance algorithms, like the scattering algorithms, exploit the (quasi-)Toeplitz structure to produce fast algorithms. However, unlike the scattering algorithms, they can respond also to symmetry of the underlying matrix when such extra structure is present, and yield algorithms with improved efficiency.

SUMMARY OF NOTATION

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

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

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

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

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 the 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,

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

and are square matrices of size determined by the context. Downshifted vectors and reversed matrices, vectors, and polynomials are, respectively,

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

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, $\mathbf{u}_{(m)}$, $u_{(m)}(z)$, $m = 0, 1, \dots, n$, are all of length $n+1$ and degree n, respectively.

1. INTRODUCTION

The classical algorithms of Schur and Levinson are efficient procedures to solve and factorize a Hermitian Toeplitz matrix and its inverse [25, 27, 19, 17]. They involve two-term polynomial or vector recursive updates in the so-called *scattering variables*, namely variables that may describe incident and scattered waves in an associated cascade-of-layered-media model [22, 24, 20]. However, it is also possible to treat these algorithms in some transformed variables. In particular it was recently shown [7, 8] that a certain transformation of variables to a so-called *immittance-variable* form, followed by a change from two-term to three-term recursions, results in reduction in computational complexity in the abovementioned algorithms. The ratio of the pair of transformed variables can be viewed as the impedance or admittance of the associated layered-media model; hence the label *immittance domain* given to all the new algorithms (the term *immittance*, for *im*pedance/ad*mittance*, is due to Bode [10]).

The redundancy in computation in the classical algorithms associated with a Toeplitz matrix was first observed in several forms of a new test for the zero location of polynomials with respect to the unit circle [1–3] and in a modified Levinson algorithm for Toeplitz matrices [11, 7] that it inspired. The Schur-Cohn algorithm for zero location of polynomials (also familiar in tabular form as the Jury-Marden test [18]), the conditions for the stability of the linear prediction filter [26], the classical Levinson algorithm to solve sets of Toeplitz equations [25, 29, 30], testing whether a power series in z is bounded [27, 19], and finding the inertia of a Toeplitz matrix [23] are only a partial list of apparently diverse problems in mathematics and system theory that are intimately related and can be resolved via similar two-term recursions of asymmetric (no particular structure) polynomials. In contrast, the new zero-location and Levinson algorithms in [1, 2, 3, 7, 11] have introduced a new formulation that, in polynomial notation, involves a certain three-term recursion of symmetric (or antisymmetric) polynomials. It was surprising to discover that the new algorithms were able to solve these classical problems in, roughly, half the amount of computation. Thus, the new formulation detected some inherent redundancy in the above classical algorithms by revealing that the symmetric (or antisymmetric) parts of the polynomials involved in the problem contain essentially the information needed to solve it. Indeed, Delsarte and Genin called their algorithms in [11, 12] the "split Levinson" and "split Schur" algorithms. The adjective "split" arises from the ability to work with the odd and even (or symmetric and skew-symmetric) parts of the polynomials involved in the usual Levinson algorithm. Bistritz, Lev-Ari, and Kailath [7] analyzed the new Levinson algorithm in a framework that studied the possible effect of a general transformation of variables and a change from two-term to three-term recursions on the efficiency of the algorithm. This detailed study proved that the new *immittance* approach applies not only to Toeplitz but also to certain symmetric *quasi-Toeplitz* matrices, where the polynomials in the improved Levinson algorithm are not symmetric or skew-symmetric and can*not* be viewed as an even-odd split of the polynomials in the usual Levinson algorithm. They also found that there are *three* computationally efficient versions of the Levinson algorithm for real Toeplitz and quasi-Toeplitz matrices, which differ in the form of the recursion: the *balanced*, the *monic*, and the *comonic* forms. The zero-location and Toeplitz-Levinson algorithms in the balanced recursion format were extended also to the complex case and found to retain the same relative efficiency over the classical algorithms in terms of the number of real-arithmetic operations [14, 4, 21]. A continuation of the study in [7] with a similarly systematic study of complex Hermitian quasi-Toeplitz matrices was made in [8]. It showed that in the complex case there exist five three-term different recursions that have better efficiency than the scattering recursions, but that only one, the balanced recursion, achieves the same improved efficiency that was found before for algorithms associated with real Toeplitz and quasi-Toeplitz matrices.

The efficiency of immittance algorithms for real or complex Hermitian quasi-Toeplitz matrices in [7, 8] cannot be explained by symmetry of polynomials, because in the non-Toeplitz case the polynomials in the Levinson algorithm have no particular structure in the immittance domain either. In fact, the better performance of immittance algorithms is also apparent in algorithms related to the Levinson algorithm, such as the Schur and the autocorrelation algorithms [12, 9], where even in the Toeplitz case the variables involved are as structureless in the immittance domain as they are in the scattering domain. This kind of evidence indicates that the advantage of the immittance algorithms over the scattering algorithms must stem from something more fundamental than splitting the classical algorithms into their symmetric components, even if the possibility of working with just the symmetric part of polynomials in some of the algorithms associated with Toeplitz matrices suffices to explain the improved efficiency.

This paper deals with immittance-domain algorithms for Toeplitz and quasi-Toeplitz (QT) real or complex matrices that are not Hermitian, and compares them with the corresponding scattering algorithm studied recently in another paper in this journal [6]. The most general form of a non-Hermitian QT matrix can be written as

$$\mathbf{R}_n = L(\tilde{\mathbf{u}}_{(0)})L^t(\mathbf{u}_{(0)}) - L(\tilde{\mathbf{v}}_{(0)})\mathbf{L}^t(v_{(0)}), \tag{1.1}$$

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

$$\tilde{\mathbf{u}}_{(0)} = \begin{bmatrix} 1 \\ \tilde{u}_{01} \\ \vdots \\ \tilde{u}_{0n} \end{bmatrix}, \qquad \mathbf{u}_{(0)} = \begin{bmatrix} 1 \\ u_{01} \\ \vdots \\ u_{0n} \end{bmatrix}, \qquad \tilde{\mathbf{v}}_{(0)} = \begin{bmatrix} 0 \\ \tilde{v}_{01} \\ \vdots \\ \tilde{v}_{0n} \end{bmatrix}, \qquad \mathbf{v}_{(0)} = \begin{bmatrix} 0 \\ v_{01} \\ \vdots \\ v_{0n} \end{bmatrix}. \tag{1.2}$$

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

$$\mathbf{T}_n = [c_{i-j}], \qquad c_k = v_{0k}, \qquad c_{-k} = \tilde{v}_{0k}. \tag{1.3}$$

An important subclass of so-called admissible QT matrices is obtained when the four generating vectors are related by the constraints

$$u_{(0)}(z) = 1 + \alpha_0 v_{(0)}(z), \tag{1.4a}$$

$$\tilde{u}_{(0)}(z) = 1 + \beta_0 \tilde{v}_{(0)}(z), \tag{1.4b}$$

where we have associated the generating vectors with polynomials by the standard convention (see Summary of Notation). Obviously, the Toeplitz matrix is a special case of an admissible QT matrix that corresponds to the choice $\alpha_0 = \beta_0 = 1$.

The fast algorithms for non-Hermitian QT matrices are associated with two discrete transmission lines [6] that reduce to the familiar single "parcor" lattice (e.g. [26]) in the Hermitian case. One transmission line can be thought of as taking care of the "left" and the other of the "right" triangular factors of $\mathbf{R}_n$ or its inverse. Suppose we want to solve the following two sets of *normal equations*:

$$\mathbf{b}_n^t \mathbf{R}_n = [0,\dots,0,D_n], \qquad \mathbf{R}_n \mathbf{a}_n = [0,\dots,0,D_n]^t \tag{1.5}$$

for an admissible QT matrix $\mathbf{R}_n = (a_{nn} = b_{nn} = 1)$. This can be done by the following ("scattering") Levinson algorithm for admissible QT matrices [6]. The algorithm consists of a pair of two-term recursions

$$\begin{pmatrix} a_m(z) \\ \alpha_m(z) \end{pmatrix} = K_m(z) \begin{pmatrix} a_{m-1}(z) \\ \alpha_{m-1}(z) \end{pmatrix}, \qquad \begin{pmatrix} b_m(z) \\ \beta_m(z) \end{pmatrix} = \tilde{K}_m(z) \begin{pmatrix} b_{m-1}(z) \\ \beta_{m-1}(z) \end{pmatrix}, \tag{1.6a}$$

where

$$K_m(z) := \begin{pmatrix} 1 & -k_m \\ -\xi_m & 1 \end{pmatrix} \begin{pmatrix} z & 0 \\ 0 & 1 \end{pmatrix}, \qquad \tilde{K}_m(z) := \begin{pmatrix} 1 & -\xi_m \\ -k_m & 1 \end{pmatrix} \begin{pmatrix} z & 0 \\ 0 & 1 \end{pmatrix}. \tag{1.6b}$$

The initial values are

$$\begin{Bmatrix} a_0(z) = 1 \\ \alpha_0(z) = \alpha_0 \end{Bmatrix}, \qquad \begin{Bmatrix} b_0(z) = 1 \\ \beta_0(z) = \beta_0 \end{Bmatrix}, \tag{1.6c}$$

while the *reflection coefficients* are computed from the inner products

$$k_{m+1} = \frac{[v_{01},\dots,v_{0,m+1}]\mathbf{a}_m}{D_m}, \qquad \xi_{m+1} = \frac{[\tilde{v}_{01},\dots,\tilde{v}_{0,m+1}]\mathbf{b}_m}{D_m}, \tag{1.6d}$$

$$D_m = (1 - \xi_m k_m) D_{m-1}, \qquad D_0 = 1. \tag{1.6e}$$

The algorithm may be described by the pair of transmission lines in Figure 1.

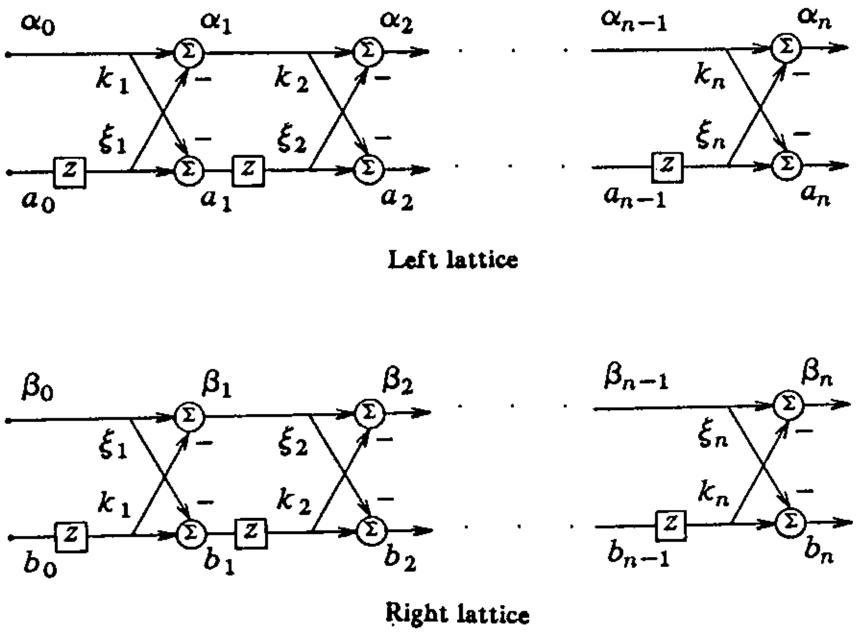


FIG. 1. Scattering-domain transmission lines for the non-Hermitian Levinson algorithm.

In the Hermitian case $\xi_m = k_m$, $\beta_0 = \alpha_0^*$. Consequently, the two recursions and the two lines become the (Hermitian) replicas of each other, and the algorithm reduces to the admissible QT Levinson [22, 24] (or Toeplitz Levinson for $\alpha_0 = 1$) with the familiar single ("parcor" [26]) transmission line.

The solution of the two sets of normal equations (1.5) for a Toeplitz matrix $\mathbf{T}_n$ also provides the generating vectors for a Gohberg-Semencul formula for the inversion of $\mathbf{T}_n$. Let $\hat{\mathbf{a}}_n$ and $\hat{\mathbf{b}}_n$ be the solutions to (1.5) for a Toeplitz matrix, i.e. let $\mathbf{R}_n = \mathbf{T}_n$. Then Gohberg and Semencul showed [16] that $\mathbf{T}_n^{-1}$ can be written as

$$\mathbf{T}_n^{-1} = \frac{1}{D_n}\left\{ L\left(\overleftrightarrow{\hat{\mathbf{b}}}_n\right) L^t\left(\overleftrightarrow{\hat{\mathbf{a}}}_n\right) - L(\downarrow \hat{\mathbf{a}}_n) L^t(\downarrow \hat{\mathbf{b}}_n)\right\}, \tag{1.7}$$

where $\downarrow$ denotes downshifted vectors (see Summary of Notation). The solution of the normal equations (1.5) for a general QT matrix is possible by an *extended QT algorithm* proposed in [6]. It is based on the fact that a QT matrix $\mathbf{R}_n$ is always related to some well-defined "hidden" Toeplitz matrix $\mathbf{T}_n$ by the relation

$$\mathbf{R}_n = L(\tilde{\mathbf{h}}_{0:n})\mathbf{T}_n L^t(\mathbf{h}_{0:n}), \qquad \tilde{\mathbf{h}}_{0:n} := \tilde{\mathbf{u}}_{(0)} - \tilde{\mathbf{v}}_{(0)}, \quad \mathbf{h}_{0:n} := \mathbf{u}_{(0)} - \mathbf{v}_{(0)}, \quad (1.8)$$

which we shall refer to as *congruence*, as it extends a similar congruency relation between Hermitian QT and Toeplitz matrices, studied in [13, 22, 24]. The extended QT algorithm in [6] comprises two stages. First, a Schur algorithm for the non-Hermitian QT matrix (to be reviewed in the next section) creates the set of reflection coefficients $\{\xi_m, k_m\}$, $m = 1,\dots,n$, which characterize *all* QT matrices that are related by congruence to a common Toeplitz matrix. Next, a *recursive convolution algorithm* uses to advantage the interpretation of multiplications of the form $L(\mathbf{h}_{0:n})\mathbf{a}_n$ as convolution between two vectors, $\mathbf{h}_{0:n} * \mathbf{a}_n$ to produce the solution vectors to (1.5) and its submatrices. These congruence relations led us in [6] also to the following GS-type inversion formula for an arbitrary QT matrix:

$$\overleftrightarrow{\mathbf{R}}_n^{-1} = \frac{1}{D_n}\left\{ L(\mathbf{e}_n)L^t(\tilde{\mathbf{e}}_n) - L(\downarrow\mathbf{d}_n)L^t(\downarrow\tilde{\mathbf{d}}_n)\right\}, \qquad (1.9a)$$

in which the generating vectors are given by

$$\mathbf{e}_n = L^{-1}(\mathbf{h}_{0:n})\overleftrightarrow{\hat{\mathbf{a}}}_n, \qquad \mathbf{d}_n = L^{-1}(\mathbf{h}_{0:n})\hat{\mathbf{b}}_n,$$

$$\tilde{\mathbf{e}}_n = L^{-1}(\tilde{\mathbf{h}}_{0:n})\overleftrightarrow{\hat{\mathbf{b}}}, \qquad \tilde{\mathbf{d}}_n = L^{-1}(\mathbf{h}_{0:n})\hat{\mathbf{a}}_n. \qquad (1.9b)$$

In the above $\overleftrightarrow{\mathbf{R}}_n$ is the reversed matrix (see Summary of Notation) and $\hat{\mathbf{a}}_n$ and $\hat{\mathbf{b}}_n$ are the solutions of the normal equations (1.5) for the "hidden" Toeplitz matrix $\mathbf{T}_n$ of (1.8). The extended QT algorithm produces recursively the generating vectors for (1.9) and solutions to (1.5) to all submatrices $\mathbf{R}_m$, $m = 1,\dots,n$. Admissible QT matrices admit a more direct GS formula whose generating vectors are produced by the Levinson algorithm (1.6) [5].

This paper carries out a systematic study of algorithms for the solution of the normal equations obtained by transformation of the scattering algorithms to immittance variables and by passing from two-term to three-term recursions. We obtain, for example, that the above (1.6) Levinson algorithm for admissible QT matrices transforms into a pair of three-term recursions for linear combinations of the scattering Levinson variables. For efficient algorithms, the immittance variables in the pair of three-term recursions have to be linear combinations of the scattering variables of the forms

$$f_m(z) = \psi_m a_m(z) + \upsilon_m \alpha_m(z), \qquad g_m(z) = \upsilon_m b_m(z) + \psi_m \beta_m(z), \quad \text{(1.10a,b)}$$

where ψ_m and υ_m are some scalars (complex numbers) common, in the manner shown, to the two combinations and obey some well-defined additional constraints. For example, the Levinson algorithm (1.6) has as one of its possible immittance versions the following form:

$$f_{m+1}(z) = (\delta_m z + \zeta_m) f_m(z) - z f_{m-1}(z), \quad \text{(1.11a)}$$

$$g_{m+1}(z) = (\zeta_m z + \delta_m) g_m(z) - z g_{m-1}(z) \quad \text{(1.11b)}$$

with

$$\delta_m = \frac{\tau(f_{m-1})}{\tau(f_m)}, \qquad \zeta_m = \frac{\tilde{\tau}(g_{m-1})}{\tilde{\tau}(g_m)}, \qquad m \geqslant 1, \quad \text{(1.11c,d)}$$

$$\tau(f_m) := [1, u_{01}, \dots, u_{0m}]\mathbf{f}_m, \qquad \tilde{\tau}(g_m) := [1, \tilde{u}_{01}, \dots, \tilde{u}_{0m}]\mathbf{g}_m. \quad \text{(1.11e)}$$

The algorithm is initiated by the values

$$z f_{-1}(z) = \tfrac{1}{2}(1 - z)(1 - \alpha_0), \qquad f_0(z) = \tfrac{1}{2}(1 + \alpha_0), \qquad \delta_0 = 1, \quad \text{(1.11f)}$$

$$z g_{-1}(z) = \tfrac{1}{2}(1 - z)(1 - \beta_0), \qquad g_0(z) = \tfrac{1}{2}(1 + \beta_0), \qquad \zeta_0 = 1. \quad \text{(1.11g)}$$

The above algorithm features a pair of so-called *balanced recursions*. Our study reveals that there are five different pairs of efficient recursions. We found similarly five efficient recursions also for the Hermitian case [8]. However, unlike the case there, where one of the recursions (the balanced recursion) was more efficient than the other four, in the non-Hermitian case all the five are of equal efficiency. If required, at the end of n recursion steps the scattering variables $a_n(z)$, $b_n(z)$ that form the solutions to the normal sets (1.5) as well as the two complementary variables $\alpha_n(z)$ and $\beta_n(z)$ can be recovered at a negligible (order n) extra cost in computation. For example, we show that the solutions for (1.5) are given by

$$a_n(z) = \frac{1}{f_{nn} + \phi_{nn}} \{ f_n(z) + \phi_n(z) \}, \tag{1.12a}$$

$$b_n(z) = \frac{1}{g_{nn} + \gamma_{nn}} \{ g_n(z) + \gamma_n(z) \} \tag{1.12b}$$

where

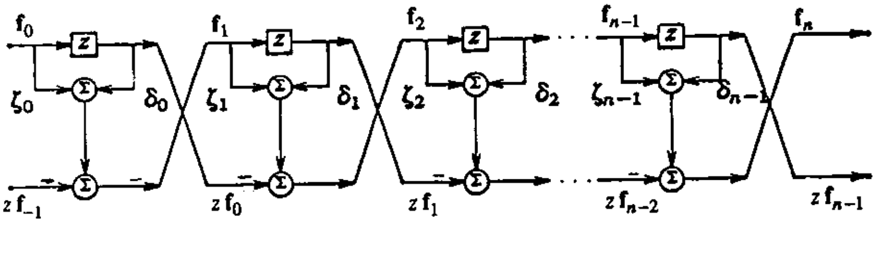
$$(z-1)\phi_n(z) = \frac{2f_n(1)}{f_{n+1}(1)} f_{n+1}(z) - (z+1) f_n(z), \tag{1.13a}$$

$$(z-1)\gamma_n(z) = \frac{2g_n(1)}{g_{n+1}(1)} g_{n+1}(z) - (z+1) g_n(z). \tag{1.13b}$$

The non-Hermitian immittance algorithms are associated with two transmission lines as illustrated in Figure 2 for the above balanced recursions. The other four versions have similar transmission lines that differ only in the location of the multipliers. Similar pairs of lattices are associated also with the immittance versions of the Schur and the extended QT algorithms, which are also studied in this paper.

We examine in detail the computational complexity of the best possible immittance Schur, Levinson, and extended QT algorithms for non-Hermitian quasi-Toeplitz matrices. In all cases there are five versions of equally efficient algorithms. The comparison of the count of arithmetic is summarized in Table 1. The table may be compared with Table 2, reproduced from [8], which similarly summarizes algorithms for Hermitian matrices. It is observed that in the non-Hermitian case, in contrast with the Hermitian case, the algorithms are roughly of the same efficiency in both domains. As we have already discussed, this observation has significance of its own in relating the "mysterious" factor two (roughly) of computational saving found in all the previously reported immittance algorithms to the symmetry in the underlining matrices, as all previous reports considered Hermitian problems.

The outline of the paper is as follows. The next section (Section 2) reviews the scattering-domain Schur algorithm to complement the Levinson algorithm already shown above. Then it describes the general technique of moving from two-term to three-term recursions and demonstrates it by converting the above Levinson algorithm for admissible QT matrices to a three-term recur-



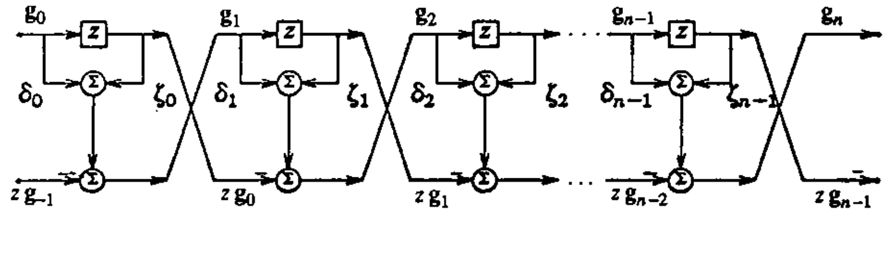


FIG. 2. Immittance-domain transmission lines for the non-Hermitian Levinson algorithm (balanced form).

sion version. The immittance transformation is introduced in Section 3, which carefully studies the set of all possible pairs of recursions of highest efficiency and shows five essentially different pairs of recursions of equal efficiency. The relations that exist between the coefficients of the five recursions are also exposed. In the subsequent Section 4 we first develop the rest of the formulae required to complete each of the five pairs of recursions into a Schur and a Levinson algorithm. Then, we show how the extended QT factorization algorithm can also be transformed into immittance algorithms. Finally, we address the problem of recovering conveniently the immittance variables that were suppressed during the conversion to three-term recursions and the reconstruction of the original scattering variables as necessary. The counts of arithmetic operations in the different algorithms is summarized conveniently in a table, Table 1, whose comparison with the corresponding Table 2 for Hermitian matrices exhibits, on one hand, the immittance compatability with the scattering-domain algorithms in the general non-Hermitian case, and on

TABLE 1
COMPUTATION COUNTS[a] FOR VARIOUS IMMITTANCE ALGORITHMS FOR QUASI-TOEPLITZ MATRICES

	Counts					
	Hermitian			Non-Hermitian		
Algorithm	Toeplitz	Admiss. QT	QT	Toeplitz	Admiss. QT	QT
Levinson	$O(0.5n^2)$	$O(n^2)$	N.A.	$O(2n^2)$	$O(3n^2)$	N.A.
Schur	$O(0.5n^2)$	$O(0.5n^2)$	$O(0.5n^2)$	$O(2n^2)$	$O(2n^2)$	$O(2n^2)$
Extended QT	$O(1.25n^2)$	$O(2n^2)$	$O(2n^2)$	$O(3n^2)$	$O(7n^2)$	$O(7n^2)$

[a] Number of multiplications. Number of additions as in Table 2; see text for details.

TABLE 2
COMPUTATION COUNTS[a] FOR VARIOUS SCATTERING ALGORITHMS FOR QUASI-TOEPLITZ MATRICES

	Counts					
	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	$O(1.5n^2)$	$O(3.5n^2)$	$O(3.5n^2)$	$O(3n^2)$	$O(7n^2)$	$O(7n^2)$

[a] Number of additions equals number of multiplications.

the other hand links its relative advantage on all previous reports, which always dealt with Hermitian matrices, to its further ability to use to advantage the additional structure that exist in Hermitian matrices by virtue of their symmetry.

2. NON-HERMITIAN SCATTERING ALGORITHMS

2.1. *The Schur and Levinson Algorithms*

The Schur algorithm for a non-Hermitian QT matrix [6] is given by the recursions $m = 0, \dots, n$:

$$\begin{bmatrix} \tilde{u}_{(m)}(z) \\ \tilde{v}_{(m)}(z) \end{bmatrix} = K_m(z) \begin{bmatrix} \tilde{u}_{(m-1)}(z) \\ \tilde{v}_{(m-1)}(z) \end{bmatrix}, \qquad \begin{bmatrix} u_{(m)}(z) \\ v_{(m)}(z) \end{bmatrix} = \tilde{K}_m(z) \begin{bmatrix} u_{(m-1)}(z) \\ v_{(m-1)}(z) \end{bmatrix} \tag{2.1a,b}$$

with transmission matrices $K_m(z)$ and $\tilde{K}_m(z)$ identical to those already introduced in (1.6b), and where the coefficients ξ_m and k_m are computed by

$$\xi_m = \frac{\tilde{v}_{m-1,m}}{\tilde{u}_{m-1,m-1}}, \qquad k_m = \frac{v_{m-1,m}}{u_{m-1,m-1}} \tag{2.2c,d}$$

The algorithm is initiated by the polynomials defined by the generating vectors (1.2) of $\mathbf{R}_n$. All vectors are of length $n+1$ with increasing number of leading zeros, viz.,

$$\tilde{\mathbf{u}}_{(m)} = [0,\dots,0, \tilde{u}_{m,m}, \dots, \tilde{u}_{m,n}]^t, \qquad \tilde{u}_{m,m} = D_m, \tag{2.3a}$$

$$\mathbf{u}_{(m)} = [0,\dots,0, u_{m,m}, \dots, u_{m,n}]^t, \qquad u_{m,m} = D_m, \tag{2.3b}$$

$$\tilde{\mathbf{v}}_{(m)} = [0,\dots,0,0, \tilde{v}_{m,m+1}, \dots, \tilde{v}_{m,n}]^t, \tag{2.3c}$$

$$\mathbf{v}_{(m)} = [0,\dots,0,0, v_{m,m+1}, \dots, v_{m,n}]^t. \tag{2.3d}$$

A transmission-line realization of this Schur algorithm is depicted by the pair of lattices in Figure 3. The algorithm presents the situation in which the four vectors (1.2) are applied as inputs to the two lattices in the way

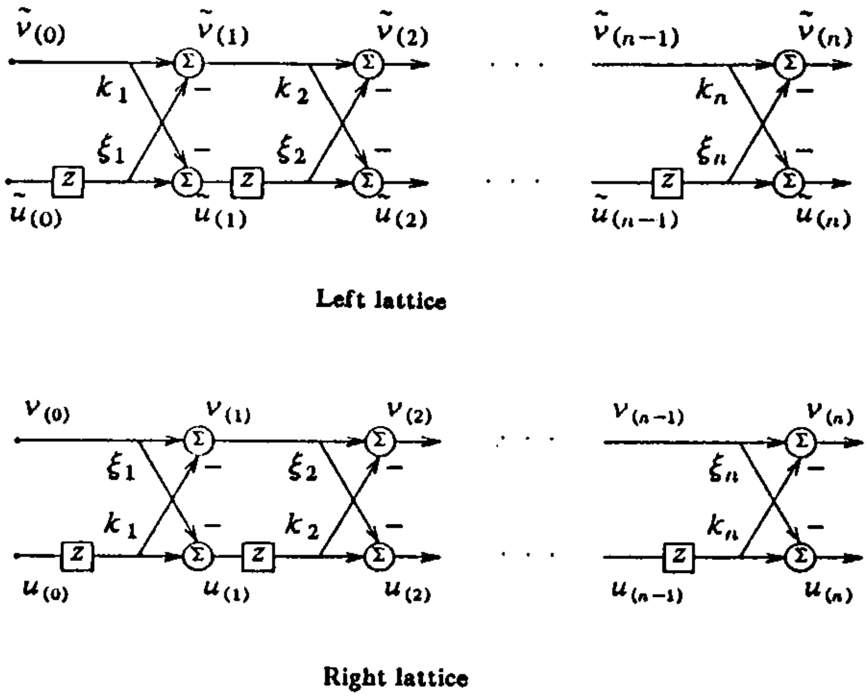


FIG. 3. Scattering-domain transmission lines for the non-Hermitian Schur algorithm.

illustrated by Figure 3. The Schur algorithm produces the *LDU* factorization of $\mathbf{R}_n$ [6], viz. $\mathbf{R}_n = \tilde{\mathbf{U}}_n \mathbf{D}_n^{-1} \mathbf{U}_n^t$, where the columns of $\tilde{\mathbf{U}}_n$ and $\mathbf{U}_n$ are given by the Schur variables $\tilde{\mathbf{u}}_{(m)}$ and $\mathbf{u}_{(m)}$. More significant to our current interest is that the Schur algorithm provides the most direct way to obtain the set of reflection coefficients $\{k_m, \xi_m\}$. If z blocks are interpreted as unit time delays, the "reflection coefficient" ξ_m (or k_m) is the ratio of the upper-line to the lower-line input signals to section m at "time" m; that is, the Schur algorithm offers an "on-line" construction of the two transmission lines. The importance of the Schur algorithm as a means to "construct" the pair of transmission lines that is common to all fast algorithms for QT matrices with a common "hidden" Toeplitz matrix (1.8) is illuminated by the extended QT algorithm [6], which can produce all the inversion and factorization algorithms, including the cases where the Levinson algorithms are not applicable. We shall see that the immittance Schur algorithms provide similarly a means to construct the pair of immittance lattices which again admit, via the immittance version of the extended QT algorithm, the solution of the fundamental equations (1.5) and the generation of the GS inversion formula (1.9), even when the matrix is not Toeplitz or admissible QT.

We have already described the scattering Levinson algorithm in the introduction section [see (1.6)]; therefore it need not be repeated here. It applies for QT matrices that satisfy the admissibility constraints (1.4). These cases have an important significance in modeling the propagation of waves in layered media by extending the model to allow partial surface reflectance. It also presents the "fairest" generalization of the Levinson algorithm that leaves its simple form essentially unaffected. The solution of the normal set of equations (1.5) for a general QT matrix is possible by the extended QT algorithm, which is composed of the Schur algorithm followed by a recursive convolution algorithm. We shall give a description of this algorithm in a later section (Section 4.3), side by side with its immittance version.

2.2. *Three-Term Recursions*

It is always possible to replace a two-term recursion in two variables (polynomials or vectors) by a three-term recursion that involves only one of the variables in the pair [7, 8]. Suppose $\{f_n(z), g_n(z)\}$ is a sequence of polynomials satisfying the two-term recursion

$$\begin{bmatrix} f_m(z) \\ \phi_m(z) \end{bmatrix} = M_m(z) \begin{bmatrix} f_{m-1}(z) \\ \phi_{m-1}(z) \end{bmatrix}, \qquad M_m(z) = \begin{bmatrix} \alpha_m(z) & \beta_m(z) \\ \gamma_m(z) & \delta_m(z) \end{bmatrix}. \quad (2.4a,b)$$

Following the general technique for converting two-term recursions into three-term recursions, eliminating, say, $\phi_m(z)$ (see [9]), we obtain

$$f_{m+1}(z) = \left\{ \alpha_{m+1}(z) + \frac{\beta_{m+1}(z)\delta_m(z)}{\beta_m(z)} \right\} f_m(z) - \frac{\beta_{m+1}(z)\Delta_m(z)}{\beta_m(z)} f_{m-1}(z), \quad (2.5a)$$

where $\Delta_n(z) = \det M_n(z)$. The second variable that is eliminated can, when desirable, be retrieved from last two primary variables by an auxiliary equation

$$\phi_n(z) = \frac{\delta_n(z) f_n(z) - \Delta_n(z) f_{n-1}(z)}{\beta_n(z)}. \quad (2.5b)$$

A useful demonstration of the above technique would be its application to the Levinson algorithm for non-Hermitian admissible QT matrices (1.6). The demonstration will be constructive in two ways. First, since the algorithm (1.6) is considered to be new in [6], the following will also present an algorithm not presented before. Second, in our line of exposition, it serves to demonstrate that the technique of this section that was found to reduce the arithmetic counts in the Hermitian immittance algorithms in [7, 8] does not in itself necessarily achieve such an effect, not even in the Hermitian case for the scattering variables.

We choose to replace the two two-term Levinson recursions (1.6a, b) by two three-term recursions in the variables $a_n(z)$ and $b_n(z)$, which are the only variables actually needed as solutions for (1.5) or in the *LDU* decomposition of $\mathbf{R}_n^{-1}$ [6]. The resulting Levinson algorithm, as a simple exercise of the technique depicted by (2.4,5) would verify, is as follows:

$$a_{m+1}(z)=\left(z-\frac{k_{m+1}}{k_m}\right)a_m(z)-\frac{k_{m+1}}{k_m}(1-\xi_m k_m)za_{m-1}(z), \quad (2.6a)$$

$$b_{m+1}(z)=\left(z-\frac{\xi_{m+1}}{\xi_m}\right)b_m(z)-\frac{\xi_{m+1}}{\xi_m}(1-\xi_m k_m)zb_{m-1}(z), \quad (2.6b)$$

$$k_{m+1}=\frac{[v_{01},\dots,v_{0,m+1}]\mathbf{a}_m}{D_m}, \qquad \xi_{m+1}=\frac{[\tilde{v}_{01},\dots,\tilde{v}_{0,m+1}]\mathbf{b}_m}{D_m}, \quad (2.6c)$$

$$D_m=(1-\xi_m k_m)D_{m-1}, \qquad D_0=1. \quad (2.6d)$$

The algorithm is initiated by

$$a_{-1}(z)=b_{-1}(z)=0, \qquad a_0(z)=b_0(z)=1, \qquad k_0=\frac{1}{\alpha_0}, \qquad \xi_0=\frac{1}{\beta_0}. \quad (2.6e)$$

A similar three-term version for the Schur algorithm, taking $u_{(m)}(z)$ and $\tilde{u}_{(m)}(z)$ as the primary variables is also possible.

The above also illustrates the usefulness of the technique, in certain cases, for singling out the truly primary variables while eliminating variables that are redundant. In these cases, the variables $v_m(z), \tilde{v}_m(z)$ and $\alpha_m(z), \beta_m(z)$ are not required, for example, in *LDU* factorization of $\mathbf{R}_n$ or its inverse, respectively. It may also be observed, though, that the three-term versions offer no computational advantage over their two-term counterpart, since in both cases four polynomial multiplications are required per recursion step. Identical counts characterize also the Hermitian case, where, as mentioned in the introduction (see [6] for details) $\xi = k_m^*$ and the second recursion and inner product can be dropped in both (1.6) and (2.6) here. Finally, we mention that for Hermitian matrices the relation of the polynomials in the Levinson algorithm for Toeplitz matrices to polynomials orthogonal on the unit circle is well known, and three-term recursions for polynomials orthogonal on the circle do appear already in some early classical works [15, 28].

3. IMMITTANCE TRANSFORMATIONS

Suppose we transform each of the two two-term recursions into new variables according to

$$\begin{bmatrix} f_m(z) \\ \phi_m(z) \end{bmatrix} = T_m \begin{bmatrix} a_m(z) \\ \alpha_m(z) \end{bmatrix}, \qquad \begin{bmatrix} g_m(z) \\ \gamma_m(z) \end{bmatrix} = \tilde{T}_m \begin{bmatrix} b_m(z) \\ \beta_m(z) \end{bmatrix} \tag{3.1a}$$

or

$$\begin{bmatrix} \tilde{x}_{(m)}(z) \\ \tilde{y}_{(m)}(z) \end{bmatrix} = T_m \begin{bmatrix} \tilde{u}_{(m)}(z) \\ \tilde{v}_{(m)}(z) \end{bmatrix}, \qquad \begin{bmatrix} x_{(m)}(z) \\ y_{(m)}(z) \end{bmatrix} = \tilde{T}_m \begin{bmatrix} u_{(m)}(z) \\ v_{(m)}(z) \end{bmatrix}. \tag{3.1b}$$

It has been shown in [8, 7] that a most general transformation that yields a uniform transmission line structure and computationally efficient recursions has to be of the form

$$T_m = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} \psi_m & 0 \\ 0 & \upsilon_m \end{bmatrix}, \qquad \tilde{T}_m = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} \tilde{\psi}_m & 0 \\ 0 & \tilde{\upsilon}_m \end{bmatrix}. \tag{3.1c}$$

A transformation of this type has the effect of replacing a pair of polynomials $\{a_n(z), \alpha_n(z)\}$, whose ratio is a bounded function, by a pair of polynomials whose ratio is a positive real function. Indeed, let

$$\begin{bmatrix} f_m(z) \\ \phi_m(z) \end{bmatrix} = T_m \begin{bmatrix} a_m(z) \\ \alpha_m(z) \end{bmatrix},$$

$$S_m(z) := \frac{\alpha_m(z)}{a_m(z)}, \qquad Z_m(z) := \frac{\phi_m(z)}{f_m(z)}, \qquad \eta_m := \frac{\upsilon_m}{\psi_m}. \tag{3.2}$$

Then it can be seen that if $S_m(z)$ is analytic in $|z| \geqslant 1$, and $|S_n(z)| \leqslant 1$ there, and $|\eta_m| \leqslant 1$, then $Z_m(z)$ is analytic and $\operatorname{Re} Z_m(z) \geqslant (1-|\eta_m|)/(1+|\eta_m|) \geqslant 0$ in $|z| \geqslant 1$. In network theory, bounded functions like $S_n(z)$ and positive real functions like $Z_n(z)$ describe, respectively, ratios of forward to scattered waves and the impedance or admittance ratios of pairs of wave variables. For this reason we called these transformed variables immittance variables. Thus, depending on the physical wave propagation phenomenon that is being modeled, the immittance variables may present, for example, voltage and current, seismological or acoustical pressure and velocity, etc.

Setting the new variables into the two two-term recursions, we obtain two two-term recursions in the new variables:

$$\begin{bmatrix} f_m(z) \\ \phi_m(z) \end{bmatrix} = M_m(z) \begin{bmatrix} f_{m-1}(z) \\ \phi_{m-1}(z) \end{bmatrix}, \qquad \begin{bmatrix} g_m(z) \\ \gamma_m(z) \end{bmatrix} = \tilde{M}_m(z) \begin{bmatrix} g_{m-1}(z) \\ \gamma_{m-1}(z) \end{bmatrix} \tag{3.3a,b}$$

with matrices $M_m(z) := T_m K_m(z) T_{m-1}^{-1}$ and $\tilde{M}_m(z) := \tilde{T}_m \tilde{K}_m(z) \tilde{T}_{m-1}^{-1}$.

The two two-term recursions (3.2a, b) can be transformed into a pair of three-term recursions

$$f_{m+1}(z) = \frac{\psi_{m+1} - v_{m+1}\xi_{m+1}}{\psi_m - v_m\xi_m} \times \left[\left(z + \frac{v_m \psi_{m-1}}{v_{m-1}\psi_m} \right) f_m(z) - \frac{v_m}{v_{m-1}} (1 - k_m \xi_m) z f_{m-1}(z) \right], \tag{3.4a}$$

$$g_{m+1}(z) = \frac{\tilde{\psi}_{m+1} - \tilde{v}_{m+1} k_{m+1}}{\tilde{\psi}_m - \tilde{v}_m k_n} \times \left[\left(z + \frac{\tilde{v}_m \tilde{\psi}_{m-1}}{\tilde{v}_{m-1}\tilde{\psi}_m} \right) g_m(z) - \frac{\tilde{v}_m}{\tilde{v}_{m-1}} (1 - \xi_m k_m) z g_{m-1}(z) \right]. \tag{3.4b}$$

This follows from the procedure (24.5) incorporating in each recursion an implied condition shown in [8] (since each of the two two-term recursions here has structure already considered in [8]) that has to be satisfied for the polynomials in the three-term recursions to have polynomial (rather than rational function) coefficients. The implied conditions are

$$\frac{(v_m - \psi_m k_m)\psi_{m-1}}{(\psi_m - v_m \xi_m)v_{m-1}} = c, \qquad \frac{(\tilde{v}_m - \tilde{\psi}_m \xi_m)\tilde{\psi}_{m-1}}{(\tilde{\psi}_m - \tilde{v}_m k_m)\tilde{v}_{m-1}} = \tilde{c},$$

where c and $\tilde{c}$ are arbitrary complex numbers not dependent on m. As a first step toward reducing the number of different coefficients in the pair of recursions (3.4), it is observed that $c = \tilde{c}$ is desirable. Setting them both equal to 1 makes the current non-Hermitian extension consistent with the Hermitian case [8] and also maintains the "scattering"-to-"immittance" interpretation of the variable transformation. The choice $c = \tilde{c} = 1$ implies the simple relations $\tilde{v}_m = \psi_m$, $\tilde{\psi}_m = v_m$ and reduces the number of different recursion coefficients in (3.4) from six to three. We therefore set the following *fundamental constraints*:

$$\frac{(v_m - \psi_m k_m)\psi_{m-1}}{(\psi_m - v_m \xi_m)v_{m-1}} = 1, \qquad \tilde{v}_m = \psi_m, \qquad \tilde{\psi}_m = v_m. \tag{3.5}$$

The fundamental constraints imply the relations

$$\eta_{m-1} = \frac{\eta_m - k_m}{1 - \eta_m \xi_m}, \quad \eta_m = \frac{\eta_{m-1} + k_m}{1 + \eta_{m-1}\xi_m} \qquad (\eta_0 = 1), \tag{3.6a,b}$$

where we denote, as in (3.2),

$$\eta_m := \frac{v_m}{\psi_m}, \qquad \tilde{\eta}_m := \frac{\tilde{v}_m}{\tilde{\psi}_m} = \eta_m^{-1}. \tag{3.6c,d}$$

These relations demonstrate that the fundamental constraints leave little further freedom in choosing the scaling factors ψ_m, v_m and $\tilde{\psi}_m, \tilde{v}_m$ of the immittance transformation. In fact it is possible to impose exactly one more constraint, which we shall choose so as to reduce the number of different nontrivial coefficients in the pair of recursions from three to two.

In the sequel we shall mostly be concerned with immittance algorithms based on three-term recursions, which were found to be more efficient in our previous studies [7, 8]. Nevertheless we want to comment beforehand that (3.2) give rise also to two-term immittance recursions that can also be made efficient. Indeed, the fundamental constraints of (3.5) also simplify the matrices $M_m(z)$ and $\tilde{M}_m(z)$ of (3.3). Thus, the entries of $M_m(z)$ [with notation as in (2.4b)] become

$$\alpha_m(z) = \frac{\psi_m - v_m\xi_m}{2\psi_{m-1}}(z+1), \tag{3.7a}$$

$$\beta_m(z) = \frac{\psi_m - v_m\xi_m}{2\psi_{m-1}}(z-1), \tag{3.7b}$$

$$\gamma_m(z) = \frac{\psi_m - v_m\xi_m}{2\psi_{m-1}}(\kappa_m z - \tilde{\kappa}_m), \tag{3.7c}$$

$$\delta_m(z) = \frac{\psi_m - v_m\xi_m}{2\psi_{m-1}}(\kappa_m z + \tilde{\kappa}_m), \tag{3.7d}$$

where

$$\kappa_m := \frac{\psi_m + v_m\xi_m}{\psi_m - v_m\xi_m}, \qquad \tilde{\kappa}_m := \frac{v_m + \psi_m k_m}{v_m - \psi_m k_m}. \tag{3.7e}$$

The corresponding entries $\tilde{\alpha}_m(z), \tilde{\beta}_m(z), \tilde{\gamma}_m(z), \tilde{\delta}_m(z)$ of $\tilde{M}_m(z)$ go through a similar simplification and are obtained by exchanging $\psi_m \leftrightarrow v_m$, $k_m \leftrightarrow \xi_m$ in the right-hand sides of (3.7a, d). Let us also note that the best choice for low arithmetic-operation counts in the recursions is the constraint that makes the upper row entries in the two transmission matrices monic. This can be the basis of the Levinson or Schur algorithms, the Hermitian versions of which were discussed in [7, 8]. Such algorithms become useful when for some reason one wants the complete sequence of immittance pairs of variables.

We next examine all the possible ways to set the additional constraints. It turns out that there are five different choices, which result in five possible pairs of recursions. These choices have the effect of reducing the number of coefficients in the pair of recursions from three or two.

(1) *Balanced pair.* These recursions result from imposing the constraint

$$\frac{\left(\psi_{m+1}^B - v_{m+1}^B\xi_{m+1}\right)v_m^B}{\left(\psi_m^B - v_m^B\xi_m\right)v_{m-1}^B}(1 - k_m\xi_m) = 1 \tag{3.8}$$

to obtain the two recursions

$$f_{m+1}(z) = (\delta_m z + \zeta_m) f_m(z) - z f_{m-1}(z), \tag{3.9a}$$

$$g_{m+1}(z) = (\zeta_m z + \delta_m) g_m(z) - z g_{m-1}(z) \tag{3.9b}$$

with

$$\delta_m = \frac{v_{m-1}}{v_m(1 - k_m \xi_m)}, \qquad \zeta_m = \frac{\psi_{m-1}}{\psi_m(1 - k_m \xi_m)}. \tag{3.10a,b}$$

The constraint (3.8) clearly originates from an attempt to simplify in (3.4a) the multiplier of $z f_{m-1}(z)$. It is then seen via the fundamental constraint (3.5) that

$$\frac{\left(\tilde{\psi}^B_{m+1} - \tilde{v}^B_{m+1} k_{m+1}\right) \tilde{v}^B_m}{\left(\tilde{\psi}^B_m - \tilde{v}^B_m k_m\right) \tilde{v}^B_{m-1}} (1 - k_m \xi_m) = 1,$$

which in turn has an identical simplifying effect also on (3.4b), yielding a (3.9a) lookalike recursion with $\tilde{\delta}_m$ and $\tilde{\zeta}_m$ rather than δ_m and ζ_m. Finally, it follows from (3.5) that $\tilde{\delta}_m = \zeta_m$ and $\tilde{\zeta}_m = \delta_m$, which completes the proof for the validity of (3.9a, b). In the Hermitian case these recursions reduce to what has been called the *balanced recursion* in [7, 8], the name "balanced" pertaining to the fact that the two recursions look alike and are similar in ascending and in descending indices.

(2) *Left-monic pair.* The constraint here originates from requiring the recursion for $f_m(z)$ to give monic polynomials (if appropriately initiated by monic polynomials). The constraint is

$$\psi^M_m (1 - \eta_m \xi_m) = 1. \tag{3.11}$$

It results in the pair of recursions

$$f_{m+1}(z) = (z + \mu_m) f_m(z) - \lambda_m z f_{m-1}(z), \tag{3.12a}$$

$$g_{m+1}(z) = (\mu_m z + 1) g_m(z) - \lambda_m z g_{m-1}(z) \tag{3.12b}$$

with

$$\mu_m = \frac{\psi_{m-1}^M}{\psi_m^M} \frac{\upsilon_m^M}{\upsilon_{m-1}^M}, \qquad \lambda_m = \frac{\upsilon_m^M}{\upsilon_{m-1}^M}(1 - k_m\xi_m). \tag{3.13a,b}$$

Indeed, (3.11) is equivalent to

$$\frac{\psi_{m+1}^M - \upsilon_{m+1}^M \xi_{m+1}}{\psi_m^M - \upsilon_m^M \xi_m} = 1,$$

which leaves $zf_m(z)$ free and admits monic polynomials. This constraint, in combination with the fundamental constraint (3.4), implies

$$\frac{\tilde{\psi}_{m+1}^M - \tilde{\upsilon}_{m+1}^M k_{m+1}}{\tilde{\psi}_m^M - \tilde{\upsilon}_m^M k_m} = \frac{\tilde{\psi}_m^M}{\tilde{\psi}_{m-1}^M} \frac{\tilde{\upsilon}_{m-1}^M}{\tilde{\upsilon}_m^M},$$

from which (3.12) and (3.13) follow. It also becomes apparent now that either the first or the second recursion, but not both simultaneously, can be made monic.

(3) *Right-monic pair.* Imposing on (3.4) the constraint

$$\psi_m^{\hat{M}}(\eta_m - k_m) = 1, \tag{3.14}$$

one obtains

$$f_{m+1}(z) = (\hat{\mu}_m z + 1) f_m(z) - \hat{\lambda}_m z f_{m-1}(z), \tag{3.15a}$$

$$g_{m+1}(z) = (z + \hat{\mu}_m) g_m(z) - \hat{\lambda}_m z g_{m-1}(z), \tag{3.15b}$$

where

$$\hat{\mu}_m = \frac{\psi_m^{\hat{M}}}{\psi_{m-1}^{\hat{M}}} \frac{\upsilon_{m-1}^{\hat{M}}}{\upsilon_m^{\hat{M}}}, \qquad \hat{\lambda}_m = \frac{\psi_m^{\hat{M}}}{\psi_{m-1}^{\hat{M}}}(1 - k_m\xi_m). \tag{3.16a,b}$$

(4) *Left-dual pair.* A dual pair to the right-monic pair of recursions follows from choosing the constraint to be

$$\frac{v_m^D}{v_{m-1}^D}(1-k_m\xi_m)=1. \tag{3.17}$$

The resulting recursions are

$$\theta_{m+1}f_{m+1}(z)=(z+\rho_m)f_m(z)-zf_{m-1}(z), \tag{3.18a}$$

$$\theta_{m+1}g_{m+1}(z)=(\rho_m z+1)g_m(z)-zg_{m-1}(z) \tag{3.18b}$$

with

$$\theta_m=\frac{\psi_m^D-v_m^D\xi_m}{\psi_{m+1}^D-v_{m+1}^D\xi_{m+1}}, \qquad \rho_m=\frac{\psi_{m-1}^D}{\psi_m^D}\frac{v_m^D}{v_{m-1}^D}. \tag{3.19a,b}$$

(5) *Right-dual pair.* The constraint this time is taken to be

$$\frac{\psi_m^{\hat{D}}}{\psi_{m-1}^{\hat{D}}}(1-k_m\xi_m)=1, \tag{3.20}$$

and the resulting recursions are

$$\hat{\theta}_{m+1}f_{m+1}(z)=(\hat{\rho}_m z+1)f_m(z)-zf_{m-1}(z), \tag{3.21a}$$

$$\hat{\theta}_{m+1}g_{m+1}(z)=(z+\hat{\rho}_m)g_m(z)-zg_{m-1}(z) \tag{3.21b}$$

with

$$\hat{\rho}_m=\frac{\psi_m^{\hat{D}}}{\psi_{m-1}^{\hat{D}}}\frac{v_{m-1}^{\hat{D}}}{v_m^{\hat{D}}}, \qquad \hat{\theta}_m=\frac{v_m^{\hat{D}}-\psi_m^{\hat{D}}k_m}{v_{m+1}^{\hat{D}}-\psi_{m+1}^{\hat{D}}k_{m+1}}. \tag{3.22}$$

3.1. *Interrelations among Coefficients*

As might be expected, there exist close connections among the coefficients in the five pairs of recursions [because the different constraints on ψ_m

and v_m act only by weighting and scaling linear combinations of scattering variables; cf. (3.1)]. Suppose we pick the left-monic recursion coefficients (3.13) as reference parameters and check their relations to the coefficients in the remaining pairs of recursions. We first observe that the coefficients in the left-monic recursions are also given by

$$\mu_m = \frac{\eta_m}{\eta_{m-1}}, \qquad \lambda_m = \left(\eta_{m-1}^{-1} - \xi_{m-1}\right)\left(\eta_{m-1} + k_m\right). \tag{3.23}$$

Examination of the left-dual recursions reveals that their coefficients are identical with the left-monic coefficients (hence the name dual), viz.,

$$\rho_m = \mu_m, \qquad \theta_{m+1} = \lambda_m. \tag{3.24}$$

The balanced recursion coefficients, in turn, are related to the μ_m and λ_m by

$$\zeta_m = \mu_m \delta_m, \quad \delta_m \delta_{m-1} = \frac{1}{\lambda_m} \qquad (\delta_0 = 1). \tag{3.25}$$

Switching from "left" to "right" recursions, we similarly find that

$$\hat{\rho}_m = \hat{\mu}_m = \frac{\eta_{m-1}}{\eta_m}, \qquad \hat{\theta}_{m+1} = \hat{\lambda}_m = \left(\eta_{m-1} - k_{m-1}\right)\left(\eta_{m-1}^{-1} + \xi_m\right). \tag{3.26}$$

Finally, the connections between left and right recursion coefficients are

$$\hat{\mu}_m = \mu_m^{-1}, \qquad \hat{\lambda}_m = \frac{\eta_{m-2}}{\eta_m}\lambda_m. \tag{3.27}$$

We note that with the above, we have also established the connections between the various immittance coefficients and the scattering coefficients $\{k_m, \xi_m\}$. They follow through (3.23), or (3.26) and the recursive definition (3.6b) for the sequence $\{\eta_m\}$ starting with $\eta_0 = 1$ [the initiation that corresponds to our choices in (4.3,5) below].

We choose not to incorporate the relations shown above in the various pairs of recursions, but retain for each the original individual notation. This will add to the clarity of the completion of these recursions into Schur and Levinson algorithms that we shall carry out in the sequel. As we shall see, the derivation of the inner products in the various Levinson algorithms employs the relations between the recursion coefficients and their corresponding constraints, to which identical parameters in different recursions relate differently.

4. IMMITTANCE SCHUR AND LEVINSON ALGORITHMS

In this section we bring the details of five Schur and Levinson algorithms into correspondence with the five forms of efficient recursion pairs that were evaluated in the previous section. The Schur recursions propagate

$$\tilde{x}_{(m)}(z) = \psi_n \tilde{u}_{(m)}(z) + \upsilon_n \tilde{v}_{(m)}(z), \qquad x_{(m)}(z) = \upsilon_m u_{(m)}(z) + \psi_m v_{(m)}(z) \tag{4.1a,b}$$

[see (3.1)], and the Levinson algorithms propagate corresponding linear combinations of the scattering Levinson polynomials, that is,

$$f_m(z) = \psi_m a_m(z) + \upsilon_m \alpha_m(z), \qquad g_m(z) = \upsilon_m b_m(z) + \psi_m \beta_m(z). \tag{4.2a,b}$$

It is possible to arrange initial conditions that are common to all the five versions of the algorithms by making some careful choices of available indeterminate scalars such as ψ_0, ψ_{-1}, υ_0, υ_{-1}, k_0, and ξ_0. Such a possible set of initial conditions for all subsequent algorithms is as follows. Take

$$\delta_0 = 1, \quad \zeta_0 = 1, \quad \lambda_0 = 1, \quad \hat{\lambda}_0 = 1, \quad \rho_0 = \mu_0 = 1, \quad \hat{\rho}_0 = \hat{\mu}_0 = 1. \tag{4.3}$$

Initiate the Schur algorithms with

$$z\tilde{x}_{(-1)} = \tfrac{1}{2}(1 - z)\{\tilde{u}_{(0)}(z) - \tilde{v}_{(0)}(z)\}, \qquad \tilde{x}_{(0)}(z) = \tfrac{1}{2}\{\tilde{u}_{(0)}(z) + \tilde{v}_{(0)}(z)\} \tag{4.4a}$$

$$zx_{(-1)} = \tfrac{1}{2}(1 - z)\{u_{(0)}(z) - v_{(0)}(z)\}, \qquad x_{(0)}(z) = \tfrac{1}{2}\{u_{(0)}(z) + v_{(0)}(z)\}. \tag{4.4b}$$

Initiate the Levinson algorithms with

$$zf_{-1}(z)=\tfrac{1}{2}(1-z)(1-\alpha_0), \qquad f_0(z)=\tfrac{1}{2}(1+\alpha_0), \tag{4.5a}$$

$$zg_{-1}(z)=\tfrac{1}{2}(1-z)(1-\beta_0), \qquad g_0(z)=\tfrac{1}{2}(1+\beta_0). \tag{4.5b}$$

The above initial conditions are chosen so as to be consistent with [7, 8] and to become "nice" for the Toeplitz ($\alpha_0=\beta_0=1$) case.

4.1. *Immittance Schur Algorithm*

We list in the sequel the five possible forms of the Schur recursions. All subsequent formulae for the computation of the recursion coefficients emerge easily from the observation that the first m coefficients of $x_{(m)}(z)$ and $\tilde{x}_{(m)}(z)$ vanish, i.e.,

$$\begin{aligned}\tilde{x}_{(m)}(z)&=[0,\dots,\tilde{x}_{m,m},\dots,\tilde{x}_{m,n}][1,z,\dots,z^n]^t,\\ x_{(m)}(z)&=[0,\dots,x_{m,m},\dots,x_{m,n}][1,z,\dots,z^n]^t.\end{aligned} \tag{4.6}$$

This fact follows from (2.4) and (4.1).

(1) *Balanced algorithm.* The balanced Schur algorithm consists of the recursions

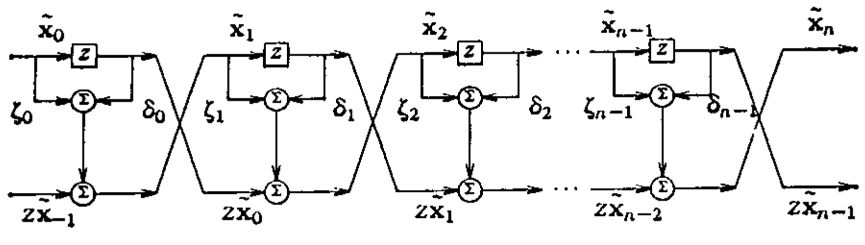
$$\tilde{x}^B_{(m+1)}(z)=(\delta_m z+\zeta_m)\tilde{x}^B_{(m)}(z)-z\tilde{x}^B_{(m-1)}(z), \tag{4.7a}$$

$$x^B_{(m+1)}(z)=(\zeta_m z+\delta_m)x^B_{(m)}(z)-zx^B_{(m-1)}(z) \tag{4.7b}$$

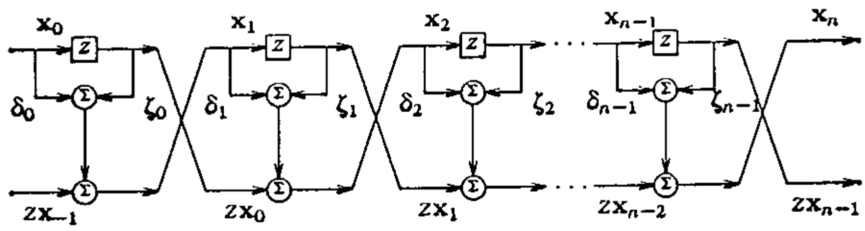
[cf. (3.9)] with coefficients determined by

$$\zeta_m=\tilde{x}^B_{m-1,m-1}/\tilde{x}^B_{m,m}, \qquad \delta_m=x^B_{m-1,m-1}/x^B_{m,m}. \tag{4.8a,b}$$

We describe the Schur algorithm by the transmission line in Figure 4. Similar descriptions are possible also for the four subsequent algorithms. The transmission lines will have the same form, differing only in the position of the multipliers at each section. As mentioned for the scattering Schur algorithms with the transmission lines of Figure 2 (and as discussed in more detail in [6]), interpretation of the z blocks as delays permits a flow-graph interpreta-



Left lattice



Right lattice

FIG. 4. Immittance-domain transmission lines for the non-Hermitian Schur algorithm (balanced form).

tion of the algorithm in which ζ_m (δ_m) is formed by the ratio of the signals at the two inputs to section m of the left (right) lattice at "time" m.

(2) *Left-monic algorithm.* The left-monic Schur algorithm is given by the recursions (3.12):

$$\tilde{x}^M_{(m+1)}(z) = (z+\mu_m)\tilde{x}^M_{(m)}(z) - \lambda_m z\tilde{x}^M_{(m-1)}(z), \tag{4.9a}$$

$$x^M_{(m+1)}(z) = (\mu_m z+1)x^M_{(m)}(z) - \lambda_m z x^M_{(m-1)}(z) \tag{4.9b}$$

with coefficients determined by the formulae

$$\lambda_m = \frac{x^M_{m,m}}{x^M_{m-1,m-1}}, \qquad \mu_m = \lambda_m \frac{\tilde{x}^M_{m-1,m-1}}{\tilde{x}^M_{m,m}}. \tag{4.10a,b}$$

(3) *Right-monic algorithm.* This Schur algorithm is given by the recursions (3.15):

$$\tilde{x}^{\hat{M}}_{(m+1)}(z) = (\hat{\mu}_m z + 1)\tilde{x}^{\hat{M}}_{(m)}(z) - \hat{\lambda}_m z\tilde{x}^{\hat{M}}_{(m-1)}(z), \tag{4.11a}$$

$$x^{\hat{M}}_{(m+1)}(z) = (z + \hat{\mu}_m)x^{\hat{M}}_{(m)}(z) - \hat{\lambda}_m z x^{\hat{M}}_{(m-1)}(z) \tag{4.11b}$$

and the coefficient formulae

$$\hat{\lambda}_m = \frac{\tilde{x}^{\hat{M}}_{m,m}}{\tilde{x}^{\hat{M}}_{m-1,m-1}}, \qquad \hat{\mu}_m = \hat{\lambda}_m \frac{x^{\hat{M}}_{m-1,m-1}}{x^{\hat{M}}_{m,m}}. \tag{4.12a,b}$$

(4) *Left-dual algorithm.* The dual-form algorithms, as we shall see repeatedly, always require the following ordering of computation: first compute the recursions' right-hand sides, then the coefficients, then the new $(m+1)$th variables. Here, in the left-dual Schur algorithm, the procedure is

$$\tilde{w}^{D}_{(m+1)}(z) := (z + \rho_m)\tilde{x}^{D}_{(m)}(z) - z\tilde{x}^{D}_{(m-1)}(z), \qquad \hat{x}_{(m+1)}(z) = \theta^{-1}_{m+1}\hat{w}^{D}_{(m+1)}(z), \tag{4.13a}$$

$$w_{(m+1)}(z) := (\rho_m z + 1)x^{D}_{(m)}(z) - z x^{D}_{(m-1)}(z), \qquad x_{(m+1)}(z) = \theta^{-1}_{m+1} w_{(m+1)}(z) \tag{4.13b}$$

with formulae for the coefficients

$$\rho_m = \frac{\tilde{x}^{D}_{m-1,m-1}}{\tilde{x}^{D}_{m,m}}, \qquad \theta_{m+1} = 2w^{D}_{m+1,m+1}, \tag{4.14a,b}$$

where $w_{m+1,i}$ are the coefficients of $w_{(m+1)}(z)$ by the regular association of polynomials with their coefficients vectors, that is, $w_{m+1} = \rho_m x_{m,m+1} + x_{m,m} - x_{m-1,m}$.

(5) *Right-dual algorithm.* In this Schur algorithm, we again start with the computation of the right-hand sides of the recursions (3.21),

$$\tilde{w}^{\hat{D}}_{m+1}(z) := (\hat{\rho}_m z + 1)\tilde{x}^{\hat{D}}_{(m)}(z) - z\tilde{x}^{\hat{D}}_{(m-1)}(z), \qquad \tilde{x}^{\hat{D}}_{(m+1)}(z) = \hat{\theta}^{-1}_{m+1}\tilde{w}^{\hat{D}}_{(m+1)}(z), \tag{4.15a}$$

$$w^{\hat{D}}_{(m+1)}(z) := (z + \hat{\rho}_m)x^{\hat{D}}_{(m)}(z) - zx^{\hat{D}}_{(m-1)}(z), \qquad x^{\hat{D}}_{(m+1)}(z) = \hat{\theta}^{-1}_{m+1}w^{\hat{D}}_{(m+1)}(z), \tag{4.15b}$$

where the coefficients are computed by

$$\hat{\rho}_m = \frac{x^{\hat{D}}_{m-1,m-1}}{x^{\hat{D}}_{m,m}}, \qquad \hat{\theta}_{m+1} = 2\tilde{w}^{\hat{D}}_{m+1,m+1}. \tag{4.16a,b}$$

4.2. *Immittance Levinson Algorithm*

In order to complete the five pairs of three-term recursions into immittance-domain Levinson algorithms, we have to provide inner-product formulae for the computation of the recursion coefficients. We introduce the following compact notation:

$$\tau(p_m) := [1, u_{01}, \dots, u_{0m}]\mathbf{p}_m, \qquad \tilde{\tau}(p_m) := [1, \tilde{u}_{01}, \dots, \tilde{u}_{0m}]\mathbf{p}_m \tag{4.17}$$

for inner products of some vector $\mathbf{p}_m := [p_{m0}, \dots, p_{mm}]^t$ with vectors of the first column and row of $\mathbf{R}_m$. Each algorithm involves two inner products of this form per step, usually $\tau(f_m)$ and $\tilde{\tau}(g_m)$. The actual way these inner products yield the coefficients for the recursions varies from one version to the other. It depends on the expressions obtained in Section 3 for the coefficients and on the following key identities:

$$\tau(f_m) = \alpha_0 v_m D_m, \quad \tilde{\tau}(g_m) = \beta_0 \psi_m D_m, \qquad m \geqslant 1; \tag{4.18}$$

we formally define $\tau(f_0) = \frac{1}{2}\alpha_0$ and $\tilde{\tau}(g_0) = \frac{1}{2}\beta_0$. The identities (4.18) are obtained by inserting (4.2) into the identities

$$\tau(a_m) = 0, \qquad \tilde{\tau}(\alpha_m) = \alpha_0 D_m, \qquad \tau(b_m) = 0, \qquad \tilde{\tau}(\beta_m) = \beta_0 D_m, \tag{4.19}$$

which were obtained in [6]. We note that since (4.18) follows from (4.2) and (4.19), it holds for all the five recursion forms.

(1) *Balanced algorithm.* The Levinson algorithm in its balanced form consists of the recursions (3.9) with the formulae for the coefficients, found by considering (4.18) and (3.10),

$$\delta_m = \frac{\tau\left(f_{m-1}^B\right)}{\tau\left(f_m^B\right)}, \quad \zeta_m = \frac{\tilde{\tau}\left(g_{m-1}^B\right)}{\tilde{\tau}\left(g_m^B\right)}, \qquad m \geqslant 1. \qquad (4.20\text{a,b})$$

(2) *Left-monic algorithm.* The left-monic Levinson algorithm consists of the recursion (3.12) with coefficients given by [cf. (4.19) and (3.13)]

$$\lambda_m = \frac{\tau\left(f_m^M\right)}{\tau\left(f_{m-1}^M\right)}, \quad \mu_m = \lambda_m \frac{\tilde{\tau}\left(g_{m-1}^M\right)}{\tilde{\tau}\left(g_m^M\right)}, \qquad m \geqslant 1. \qquad (4.21\text{a,b})$$

(3) *Right-monic algorithm.* The right-monic version of the Levinson algorithm consists of the recursions (3.15) and the following formulae that follow from (4.18) and (3.16):

$$\hat{\lambda}_m = \frac{\tilde{\tau}\left(g_m^{\hat{M}}\right)}{\tilde{\tau}\left(g_{m-1}^{\hat{M}}\right)}, \quad \hat{\mu}_m = \hat{\lambda}_m \frac{\tau\left(f_{m-1}^{\hat{M}}\right)}{\tau\left(f_m^{\hat{M}}\right)}, \qquad m \geqslant 1. \qquad (4.22\text{a,b})$$

(4) *Left-dual algorithm.* As always the case with the dual recursions, in this Levinson algorithm the right-hand sides of (3.18a, b) need to be computed before the second coefficients can be found. The algorithm is:

$$h_{m+1}^f(z) = (z + \rho_m) f_m^D(z) - z f_{m-1}^D(z), \qquad f_{m+1}(z) = \frac{1}{\theta_{m+1}} h_{m+1}^f(z),$$

$$h_{m+1}^g(z) = (\rho_m z + 1) g_m^D(z) - z g_{m-1}^D(z), \qquad g_{m+1}(z) = \frac{1}{\theta_{m+1}} h_{m+1}^g(z).$$

Here we have to proceed somehow differently, because it follows from (4.18) and (3.15) that now

$$\rho_m = \frac{\tilde{\tau}\left(g_{m-1}^D\right)}{\tilde{\tau}\left(g_m^D\right)}, \quad \theta_m = \frac{2}{\alpha_0} \tau\left(h_m^f\right), \qquad m \geqslant 1. \qquad (4.23\text{a,b})$$

The necessary order of computation is: first ρ_m, then $h^f_{m+1}(z)$ and $h^g_{m+1}(z)$, then θ_{m+1}, then $f_{m+1}(z)$ and $g_{m+1}(z)$. We note that by (4.18) and (3.17)

$$\frac{\tau\left(f^D_{m-1}\right)}{\tau\left(f^D_m\right)} = \frac{v_{m-1}}{v_m(1-k_m\xi_m)} = 1,$$

namely, $\tau(f^D_m)$ are the same for all m, and thus equal to $\tau(f^D_0) := \alpha_0/2$. This is the reason for the slight deviation from the order of computation that might be expected.

(5) *Right-dual algorithm.* This Levinson algorithm too proceeds differently than the first three versions and is similar to the previous case with the role of the first and second recursion interchanged. The algorithm is given by

$$h^f_{m+1}(z) = (\hat{\rho}_m z + 1) f^{\hat{D}}_m(z) - z f^{\hat{D}}_{m-1}(z), \qquad f^{\hat{D}}_{m+1}(z) = \frac{1}{\hat{\theta}_{m+1}} h^f_{m+1}(z),$$

$$h^g_{m+1}(z) = (z + \hat{\rho}_m) g^{\hat{D}}_m(z) - z g^{\hat{D}}_{m-1}(z), \qquad g^{\hat{D}}_{m+1}(z) = \frac{1}{\hat{\theta}_{m+1}} h^g_{m+1}(z)$$

with

$$\hat{\rho}_m = \frac{\tau\left(f^{\hat{D}}_{m-1}\right)}{\tau\left(f^{\hat{D}}_m\right)}, \quad \hat{\theta}_m = \frac{2}{\beta_0}\tilde{\tau}(h^g_m), \qquad m \geqslant 1. \qquad (4.24\text{a,b})$$

This time the modifications are required by the fact that now (4.18) with (3.22) imply $\tilde{\tau}(g_m) = \tilde{\tau}(g_0) = \beta_0/2$ for all m.

4.3. *Immittance Extended* QT *Algorithm*

A so-called extended QT algorithm in the scattering variables was suggested in [6] for the inversion of a general non-Hermitian QT algorithm. It is based on the observation that one can use the Schur algorithm to first find the reflection coefficients $\{k_m, \xi_m\}$ for any QT matrix $\mathbf{R}_n$. Then, since these coefficients are common to all $\mathbf{R}_n$ that are congruent to the same "hidden" Toeplitz matrix $\mathbf{T}_n$ by the relation (1.8), it is possible to solve (1.5) by proper interpretation of the similarity relations (1.8) as convolution between the solutions of (1.5) in the Toeplitz case and the "congruency" vectors $\mathbf{h}_{0:n}$ and $\tilde{\mathbf{h}}_{0:n}$. Similar considerations also led us to obtain the GS-type inversion formula (1.9) and show that its generating vectors (1.9b) are recursively produced by the following so-called *recursive convolution algorithm*:

$$\begin{pmatrix} \tilde{d}_{(m)}(z) \\ \tilde{e}_{(m)}(z) \end{pmatrix} = K_m(z) \begin{pmatrix} \tilde{d}_{(m-1)}(z) \\ \tilde{e}_{(m-1)}(z) \end{pmatrix}, \qquad \tilde{g}_{(0)}(z) = \tilde{\Gamma}_{0:n}(z), \quad \tilde{e}_{(0)}(z) = \tilde{\Gamma}_{0:n}(z), \tag{4.25a}$$

$$\begin{pmatrix} d_{(m)}(z) \\ e_{(m)}(z) \end{pmatrix} = \tilde{K}_m(z) \begin{pmatrix} d_{(m-1)}(z) \\ e_{(m-1)}(z) \end{pmatrix}, \qquad e_{(0)}(z) = \Gamma_{0:n}(z), \quad e_{(0)}(z) = \Gamma_{0:m}(z), \tag{4.25b}$$

where the initial conditions are the polynomials associated with the vectors $\Gamma_{0:n}$ and $\tilde{\Gamma}_{0:n}$ defined by the "filter inversion" operation,

$$L(\Gamma_{0:n}) = L^{-1}(\mathbf{h}_{0:n}), \qquad L(\tilde{\Gamma}_{0:n}) = L^{-1}(\tilde{\mathbf{h}}_{0:n}), \tag{4.25c}$$

and by our regular association of vectors with polynomials,

$$\Gamma_{0:n}(z) = [1, z, \dots, z^n]\Gamma_{0:n}, \qquad \tilde{\Gamma}_{0:n}(z) = [1, z, \dots, z^n]\tilde{\Gamma}_{0:n}. \tag{4.25d}$$

In further accordance with the Summary of Notation, we note that the four variables are polynomials of degrees n for all m, say,

$$d_{(m)}(z) = \sum_{i=0}^{n} d_{m,i} z^i, \qquad e_{(m)}(z) = \sum_{i=0}^{n} e_{m,i} z^i, \tag{4.26a}$$

$$\tilde{g}_{(m)}(z) = \sum_{i=0}^{n} \tilde{g}_{m,i} z^i, \qquad \tilde{e}_{(m)}(z) = \sum_{i=0}^{n} \tilde{e}_{m,i} z^i, \tag{4.26b}$$

and the coefficients for the formula (1.9) have to be extracted as follows:

$$\mathbf{e}_m = [1, e_{m,1}, \dots, e_{m,m}]^t, \qquad \downarrow\mathbf{d}_m = [0, d_{m,0}, \dots, d_{m,m-1}]^t, \tag{4.27a}$$

$$\tilde{\mathbf{e}}_m = [1, \tilde{e}_{m,1}, \dots, \tilde{e}_{m,m}]^t, \qquad \downarrow\tilde{\mathbf{d}}_m = [0, \tilde{d}_{m,0}, \dots, \tilde{d}_{m,m-1}]^t. \tag{4.27b}$$

The scattering-domain recursions (4.25) have the same formal structure with which we dealt before in transforming the Schur and the Levinson algorithms. We therefore can proceed in a familiar manner to transform the recursions to immittance variables. Let us define the immittance variables

$$\begin{bmatrix} \tilde{r}_{(m)}(z) \\ \tilde{\rho}_{(m)}(z) \end{bmatrix} = T_m \begin{bmatrix} \tilde{d}_{(m)}(z) \\ \tilde{e}_{(m)}(z) \end{bmatrix}, \qquad \begin{bmatrix} r_{(m)}(z) \\ \rho_{(m)}(z) \end{bmatrix} = \tilde{T}_m \begin{bmatrix} d_{(m)}(z) \\ e_{(m)}(z) \end{bmatrix}. \quad (4.28a,b)$$

Then, choosing $\tilde{r}_{(m)}(z)$ and $r_{(m)}$ as our primary variables, we can immediately write down five different pairs of three-term recursions. For example, the balanced pair is

$$\tilde{r}^B_{(m+1)}(z) = (\delta_m z + \zeta_m)\tilde{r}^B_{(m)}(z) - z\tilde{r}^B_{(m-1)}(z), \qquad (4.29a)$$

$$r^B_{(m+1)}(z) = (\zeta_m z + \delta_m)x^B_{(m)}(z) - zr^B_{(m-1)}(z). \qquad (4.29b)$$

We recall that the convolution algorithm always follows a Schur algorithm that has prepared the recursion coefficients for it. Thus a possible immittance extended QT algorithm would comprise for example the balanced Schur algorithm (4.7, 8) followed by (4.29) above initiated with

$$r_{(0)}(z) = \Gamma_{0:n}(z), \qquad \tilde{r}_{(0)}(z) = \tilde{\Gamma}_{0:n}(z), \qquad r_{(-1)}(z) = \tilde{r}_{(-1)}(z) = 0, \quad (4.30)$$

which would be the initial conditions for all four other immittance recursive algorithms. Since, the extended QT algorithm has two separate stages, it is simpler but not necessary for the Schur and recursive convolution to be of the same recursion type, because the relations between the coefficients in the various recursions can be applied. The choice to illustrate the immittance extended QT algorithm as a balanced Schur algorithm followed by the balanced recursive convolutions is not incidental. While all recursions exhibit the same efficiency in the non-Hermitian case, the balanced recursion is the form that becomes more efficient than the others in the Hermitian case. For a case where the second in the pair of three-term balanced recursions can be dropped and then the two coefficients become related by $\zeta_m = \delta^*_m$, see [8]. It was shown in [6] that the extended QT algorithm can produce the generating vectors for (1.9) or solve (1.5) for a general QT matrix in $O(7n^2)$ for non-Hermitian and $O(3.5n^2)$ for a Hermitian matrix. The immittance algorithm in the balanced form will require $O(7n^2)$ for non-Hermitian but only $O(2n^2)$ for a Hermitian matrix approached by Hermitian balanced Schur and recursive convolution algorithms.[1]

4.4. Recovery of Scattering Variables

The immittance Levinson algorithms have replaced the pairs $\{a_m(z), \alpha_m(z)\}$ and $\{b_m(z), \beta_m(z)\}$ by the pairs $\{f_m(z), \phi_m(z)\}$ and $\{g_m(z), \gamma_m(z)\}$. Subsequently, in the process of passing from two-term to three-term recursions, the second variable in each transformed pair has been eliminated. In this subsection we deal with the problems of recovering the eliminated variables and the reconstruction of the scattering variables when necessary. Such need may arise for example when the final goal is the solution of the normal sets of equations or the inversion formula of the GS type (see [5, 6] for details on generalized GS inversion formulae).

The routes to recovering the eliminated immittance variables and to reconstructing the scattering variables are, in principle, already clear. A variable eliminated in passing to three-term recursion can be recovered by using an auxiliary equation like (2.5b), for which the entries of $M_m(z)$ of (2.4b) are always given by expressions like (3.7) [a valid simplification due to the fundamental constraints (3.5)], where relations of the scalars in the expressions (3.7) to the already available recursion coefficients provide additional simplification. Similarly, to reconstruct the scattering variables one has, in principle, just to invert (3.1).

In the sequel we show some simplified ways, with setting common to all five recursions, to recover the secondary immittance variables and the scattering variables, when desirable. We use here the Levinson polynomial notation. However, unless specifically restricted to Levinson algorithms, the approach applies also to recovery of the Schur variables (or the convolution algorithm variables of Section 4.3).

Consider replacing two-term recursions for the pair $\{f_n(z), \phi_n(z)\}$ by three-term recursions of $\{f_n(z)\}$. Then, the general procedure (2.5) also implies

$$\beta_{n+1}(z)\phi_n(z) = f_{n+1}(z) - \alpha_{n+1} f_n(z). \tag{4.31}$$

Since the immittance transformations are in all cases assumed to satisfy the fundamental constraints, the two-term recursions for the pair $\{f_n(z), \phi_n(z)\}$

[1]We use $O(\alpha n^2)$ to mean a count of $\alpha n^2 +$ (negligible order-n remainder).

are governed by the transition matrix $M_n(z)$ of (3.3) with entries $\alpha_n(z)$ and $\beta_n(z)$ as in (3.7). We obtain

$$(z-1)\phi_n(z) = 2d_n f_{n+1}(z) - (z+1)f_n(z) \tag{4.32a}$$

and similarly

$$(z-1)\gamma_n(z) = 2\tilde{d}_n g_{n+1}(z) - (z+1)g_n(z), \tag{4.32b}$$

where we define the constants $d_n, \tilde{d}_n$ by

$$d_m := \frac{\psi_m}{\psi_m - v_m \xi_m} = \frac{1}{1-\eta_m \xi_m}, \qquad \tilde{d}_m := \frac{v_m}{v_m - \psi_m k_m} = \frac{1}{\eta_m - k_m}. \tag{4.33a,b}$$

d_n and $\tilde{d}_n$ can be related to the recursion coefficients of each specific recursion form through comparison of (4.33a, b) with relations between those coefficients and ψ_m, v_m, k_m, ξ_m. This way of recovering the auxiliary variable is not restricted to the Levinson algorithms, but may be used also for the Schur algorithms or the recursive convolution algorithms. A more convenient way to find d_n and $\tilde{d}_n$ that has a common form for all Levinson algorithms arises from setting $z=1$ in (4.32), viz.,

$$d_n = \frac{f_n(1)}{f_{n+1}(1)}, \qquad \tilde{d}_n = \frac{g_n(1)}{g_{n+1}(1)}. \tag{4.34a,b}$$

In cases where one wishes to have more than just one (the last) scalar in the sequences $\{d_m\}$ and $\{\tilde{d}_m\}$, it may be simpler to compute them iteratively by setting $z=1$ in the relevant recursion. For example, for the balanced recursion, it follows from (3.9a) that the d_m's are related by $d_n^{-1} = (\delta_n + \zeta_n) - d_{n-1}$, and so on.

After the pair of immittance variables have become available, the scattering variables can be recovered using the inverses of the relations (3.1). This may require in general the evaluation of the scalars ψ_n and v_n, or in fact just η_n, using the appropriate relations with the relevant recursion coefficient.

For the reconstruction of the scattering-domain Levinson-algorithm polynomials, a simpler approach can be obtained as follows. By the definition

(3.1) and the fundamental constraints (3.5), we have

$$f_m(z) = \psi_m a_m(z) + v_m \alpha_m(z), \qquad g_m(z) = v_m b_m(z) + \psi_m \beta_m(z), \quad (4.35a)$$

$$\phi_m(z) = \psi_m a_m(z) - v_m \alpha_m(z), \qquad \gamma_m(z) = v_m b_m(z) - \psi_m \beta_m(z). \quad (4.35b)$$

We also observe from the scattering Levinson algorithm that the first and last entries of its vectors exhibit the pattern

$$a_{m,m} = 1, \qquad a_{m,0} = -k_m \alpha_0, \qquad \alpha_{m,m} = -\xi_m, \qquad \alpha_{m,0} = \alpha_0, \quad (4.36a)$$

$$b_{m,m} = 1, \qquad b_{m,0} = -\xi_m \beta_0, \qquad \beta_{m,m} = -k_m, \qquad \beta_{m,0} = \beta_0. \quad (4.36b)$$

They induce on the end entries of the immittance Levinson vectors the following pattern:

$$f_{mm} = \psi_m - v_m \xi_m, \qquad f_{m,0} = -\psi_m \alpha_0 k_m + v_m \alpha_0, \quad (4.37a)$$

$$\phi_{mm} = \psi_m + v_m \xi_m, \qquad \phi_{m,0} = -\psi_m \alpha_0 k_m - v_m \alpha_0, \quad (4.37b)$$

$$g_{mm} = v_m - \psi_m k_m, \qquad g_{m,0} = -v_m \beta_0 \xi_m + \psi_m \beta_0, \quad (4.37c)$$

$$\gamma_{mm} = v_m + \psi_m k_m, \qquad \gamma_{m,0} = -v_m \beta_0 \xi_m - \psi_m \beta_0. \quad (4.37d)$$

The above relations can be used to verify the following procedure for the reconstruction of the scattering Levinson variables.

$$a_n(z) = \frac{1}{f_n(\infty) + \phi_n(\infty)} \{ f_n(z) + \phi_n(z) \}, \quad (4.38a)$$

$$\alpha_n(z) = \frac{\alpha_0}{f_n(0) - \phi_n(0)} \{ f_n(z) - \phi_n(z) \}, \quad (4.38b)$$

$$b_n(z) = \frac{1}{g_n(\infty) + \gamma_n(\infty)} \{ g_n(z) + \gamma_n(z) \}, \quad (4.38c)$$

$$\beta_n(z) = \frac{\beta_0}{g_n(0) - \gamma_n(0)} \{ g_n(z) - \gamma_n(z) \}, \quad (4.38d)$$

where $p_n(\infty)$ is taken to denote p_{nn}, the coefficient of z^n, in a polynomial

$p_n(z)$ in the variable z. It is noted that, since we only used the fundamental constraints in the derivation, the above procedure applies for all versions of the Levinson algorithms.

We emphasize that the recovery and reconstruction of polynomials discussed in this subsection are usually assumed to be carried out only *once* after n steps of recursions of the immittance algorithm. Therefore, such steps contribute a negligible order-n count to the total order-n^2 number of arithmetic operations of the fast algorithms considered.

4.5. *Toeplitz Matrices*

Toeplitz matrices have some distinctive features in the class of QT matrices. They are simpler than all other (nontrivial) matrices in the class, and therefore one might expect algorithms of lower complexity. We showed in [6] that for Toeplitz matrices the variables in the Levinson algorithm (1.6) satisfy the following relations:

$$\overset{\leftrightarrow}{\hat{\beta}}_m(z) = \hat{a}_m(z), \qquad \overset{\leftrightarrow}{\hat{b}}_m(z) = \hat{\alpha}_m(z). \tag{4.39}$$

Therefore, it is sufficient to use one two-term recursion

$$\begin{bmatrix} \hat{a}_m(z) \\ \overset{\leftrightarrow}{\hat{b}}_m(z) \end{bmatrix} = \begin{bmatrix} 1 & -k_m \\ -\xi_m & 1 \end{bmatrix} \begin{bmatrix} z & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{a}_{m-1}(z) \\ \overset{\leftrightarrow}{\hat{b}}_{m-1}(z) \end{bmatrix}, \qquad \hat{a}_0(z) = 1, \quad \overset{\leftrightarrow}{\hat{b}}_0(z) = 1, \tag{4.40}$$

where (1.6c,d) still apply, that is, there are still two inner products per recursion. This is a slightly misleading "anomalous" situation of a single two-term recursion in the realm of non-Hermitian QT matrices, which are always characterized by pairs of recursions, transmission lines, etc. But it is not too surprising once it is properly put in perspective, as we discussed in [6]. It has the effect of reducing the computation count to $O(2n^2)$, compared to $O(3n^3)$ had the redundancy (4.39) not been utilized (see Table 2).

Now, we want to show that the immittance Levinson algorithm has a comparable simplification for Toeplitz matrices (which justifies the lower count we put in the Toeplitz entry in Table 1). Indeed, it is not difficult to prove that the relations (4.39) admit the dropping of one of the recursions in the balanced pair. Thus, choosing to keep, say, the first recursion results in the following Levinson algorithm:

$$\hat{f}_{m+1}(z) = (\delta_m z + \zeta_m)\hat{f}_m(z) - z\hat{f}_{m-1}(z), \tag{4.41a}$$

and the two coefficients can still be computed as follows:

$$\tau_m = [1, u_{01}, \ldots, u_{0,m}]\hat{\mathbf{f}}_m, \qquad \tilde{\tau}_m = [1, \tilde{u}_{01}, \ldots, \tilde{u}_{0,m}]\overset{\leftrightarrow}{\hat{\mathbf{f}}}_m \tag{4.41b}$$

$$\delta_m = \frac{\tau_{m-1}}{\tau_m}, \qquad \zeta_m = \frac{\tilde{\tau}_{m-1}}{\tilde{\tau}_m}. \tag{4.41c}$$

The recovery process for $\hat{a}_n(z)$ and $\overset{\leftrightarrow}{\hat{b}}_n(z)$ will be

$$(z-1)\hat{\phi}_n(z) = \frac{2\hat{f}_n(1)}{\hat{f}_{n+1}(1)}\hat{f}_{n+1}(z) - (z+1)\hat{f}_n(z), \tag{4.42a}$$

$$\hat{a}_n(z) = \frac{\hat{f}_n(z) + \hat{\phi}_n(z)}{\hat{f}_n(\infty) + \hat{\phi}_n(\infty)}, \tag{4.42b}$$

$$\overset{\leftrightarrow}{\hat{b}}_n(z) = \frac{\hat{f}_n(z) - \hat{\phi}_n(z)}{\hat{f}_n(0) - \hat{\phi}_n(0)}. \tag{4.42c}$$

4.6. Computation Counts

The number of arithmetic operations for all immittance algorithms is summarized in Table 1. It includes the counts for the non-Hermitian algorithms developed in this paper side by side with the counts of computation for Hermitian matrices, based on [8] and assuming the balanced recursions there. The counts for the extended QT algorithm, not mentioned in [8], are found, as always by adding up the counts for the Schur algorithm, the convolution algorithm, and the "filter inversion" step (4.25c), needed twice for non-Hermitian and once for Hermitian matrices. We have included the extended QT algorithm counts even for the cases where the Levinson algorithm is applicable, although there is no reason to use it in those cases. We also reproduce Table 2 from [6] with the counts for the scattering algorithms. The counts in Table 1 refer to multiplications only. The number of additions in the immittance algorithms is always the same as for the corresponding scattering algorithm. The counts are of real and complex multiplications and additions for real and complex recursions, respectively. For complex cases, the equivalences (1 complex multiplication) = (4 real multiplications) + (2 real additions) and (1 complex addition) = (2 real additions) are assumed.

5. CONCLUDING REMARKS

This paper has presented a complete set of efficient Schur and Levinson algorithms associated with non-Hermitian Toeplitz and quasi-Toeplitz matrices, covering all the combinations of two- and three-term recursions in the scattering and the immittance variables. It completes our previous systematic studies on the effect of moving between the two type of recursions and variables, and it contains several new algorithms for the non-Hermitian matrix case. Earlier we found that the choices of transformation of the scattering variables to some new variables that are favorable from the point of view of efficiency also represent essentially the only choices of alternative variables that have no less physical significance than the original variables in typical problems modeled by the fundamental sets of equations and the transmission lines depicted in the figures. If the sets of equations with QT matrix coefficients model wave propagation in layered media, the scattering pairs of variables correspond to forward- and backward-moving waves (and may be preferred when it is more intuitive to think of "particles" moving to the "right" and to the "left"), whereas the immittance variables describe the same phenomena in terms of wave variable pairs (voltage and current, electric and magnetic fields, etc.—a description that may be used when a description in terms of the volume velocity and pressure of the "particles" is found more appealing).

Previous studies of immittance algorithms always treated the Hermitian case and repeatedly showed that transformation from scattering to immittance variables followed by moving from two- to three-term recursions reduces (roughly by half) the amount of computation. In some of the problems studied before, such as stability testing and the Levinson algorithm for Hermitian Toeplitz matrices, the saving can be explained by the fact that in these cases the immittance algorithms propagate symmetric (or antisymmetric) polynomials formed by splitting the original polynomials into their symmetric and antisymmetric components. However, we were not satisfied with this explanation, because it does not point to the source of the surprising new gain in the economy of computation. Furthermore, this factor two of computational saving has been observed also in cases like the extension of the Levinson algorithms to quasi-Toeplitz matrices where the immittance variables no longer possess any internal symmetry (the transformation of the Schur or prediction error to immittance variables does not have any particular structure even in the strictly Toeplitz case); nevertheless, the immittance algorithms are more efficient (again roughly by a factor of two in the number of multiplications).

This paper has associated the "mysterious" factor two of improved efficiency with the Hermitian structure of the underlying matrice. The constructive proof applied the technique of [7, 8] to show that the best achievable choices of recursions by a transformation to new variables and/or by moving from two- to three-term recursions of the non-Hermitian versions of corresponding scattering algorithms (as suggested recently in [6]) can only match, but not exceed, their scattering counterparts. The improved efficiency of the immittance algorithms stems from their ability to exploit the symmetry of the underlying matrix when there is such symmetry. The computation counts in Tables 1 and 2 show the exact relation between the amount of structure in the given matrix and the computation required for its inversion or factorization. The anticipated trend of lower achievable counts for more structure is clearly noticed in the computation of $\mathbf{R}_n^{-1}$ and to less extent in the factorization of $\mathbf{R}_n$ itself (the Schur algorithms do not distinguish between Toeplitz, admissible QT, and QT matrices). The fact that all algorithms are of order n^2 (rather than the order-n^3 counts expected for the inversion of general matrices with n^2 arbitrary entries) can be attributed to the QT structure and the fact that $\mathbf{R}_n$ is determined by order-n parameters (the entries of the generating vectors). The leading coefficients of n^2 in more precise counts are seen to vary in accordance with the "refined structure" (more computation is required when the minimal number of parameters that determine the matrix is higher).

The contribution of the immittance-domain algorithm is twofold. First, as mentioned above, it has offered new variables of physical significance for modeling various signal propagation phenomena. Second, it has revealed and removed redundancy in the computation of classical algorithms like the Schur and the Levinson algorithms and the Schur-Cohn stability test. The immittance version of each algorithm requires, in the general non-Hermitian matrix case, as much computation as its scattering counterpart. However, unlike the scattering algorithms, the immittance algorithms can respond also to the structure imposed by the symmetry of the matrix and produce in the Hermitian cases algorithms of improved efficiency. The immittance approach is expected to be effective also in improving other recursive signal-processing algorithms that are associated with symmetric matrices but do not use this symmetry to advantage.

REFERENCES

1 Y. Bistritz, Zero location with respect to the unit circle of discrete-time linear system polynomials, *Proc. IEEE* 72:1131–1142 (Sept. 1984).

2 Y. Bistritz, A new unit circle stability criterion, in *Proceedings of the 1983 International Symposium on the Mathematical Theory of Networks and Systems*, Beer-Sheva, Israel, 1983, Lecture Notes in Control and Inform. Sci., Vol. 58, Springer-Verlag, 1984, pp. 69–87.

3 Y. Bistritz, Z-domain continued-fraction expansions for stable discrete systems polynomials, *IEEE Trans. Circuits and Systems* CAS-32:1162–1166 (Nov. 1985).

4 Y. Bistritz, A circular stability test for general polynomials, *Systems Control Lett.* 7:89–97 (Apr. 1986).

5 Y. Bistritz, A Gohberg-Semencul inversion formula for admissible quasi-Toeplitz matrices and its generation by Levinson algorithms, submitted for publication.

6 Y. Bistritz and T. Kailath, Inversion and factorization of non-Hermitian quasi-Toeplitz matrices, *Linear Algebra Appl.* 98:77–121 (Jan. 1988).

7 Y. Bistritz, H. Lev-Ari, and T. Kailath, Immittance domain Levinson algorithms, *IEEE Trans. Inform. Theory*, to appear; conference version in *Proceedings of the 1986 IEEE International Conference on Acoustics, Speech and Signal Processing*, Tokyo, Apr. 1986, pp. 253–256.

8 Y. Bistritz, H. Lev-Ari, and T. Kailath, Immittance domain three-term Schur and Levinson recursions for quasi-Toeplitz complex Hermitian matrices, submitted for publication.

9 Y. Bistritz, H. Lev-Ari, and T. Kailath, Complexity reduced lattice filters for digital speech processing, in *Proceedings of the 1987 IEEE International Conference on Acoustics, Speech and Signal Processing*, Dallas, Apr. 1987.

10 H. W. Bode, *Network Analysis and Feedback Amplifier Design*, Van Nostrand, New York, 1945.

11 P. Delsarte and Y. Genin, The split Levinson algorithm, *IEEE Trans. Acoust. Speech Signal Process.* ASSP-34:470–478 (June 1986).

12 P. Delsarte and Y. Genin, On the splitting of classical algorithms in linear prediction theory, *IEEE Trans. Acoust. Speech Signal Process.* ASSP-35:645–653 (May 1987).

13 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 Monica, Calif., Aug. 1981, pp. 40–45.

14 P. Delsarte, Y. Genin, and Y. Kamp, Application of the index theory of pseudo-lossless functions to the Bistritz stability test, *Philips J. Res.* 39:226–241 (1984).

15 Ya. L. Geronimus, *Orthogonal Polynomials*, Consultant's Bureau, New York, 1961.

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

17 G. Heinig and K. Rost, *Algebraic Methods for Toeplitz-Like Matrices and Operators*, Akademie-Verlag, Berlin, 1984.

18 E. I. Jury, *Theory and Applications of the Z-Transform Method*, Wiley, New York, 1964.

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

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

21 H. Krishna and S. D. Morgera, The Levinson recurrence and fast algorithms for solving Toeplitz systems of linear equations, *IEEE Trans. Acoust. Speech Signal Process.* ASSP-35 (July 1987).

22 H. Lev-Ari, Nonstationary Lattice Filter Modeling, Ph.D. Thesis, Stanford Univ., 1983.

23 H. Lev-Ari, Y. Bistritz, and T. Kailath, Generalized Bezoutians: The key to efficient root location procedures, *IEEE Trans. Circuits and Systems*, to appear.

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

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

26 J. D. Markel and A. H. Gray, Jr., *Linear Prediction of Speech*, Springer-Verlag, New York, 1978.

27 I. Schur, Über Potenzreihen, die in Innern des Einheitskreises Beschrankt sind, *J. Reine Angew. Math.* 147:205–232 (1917).

28 G. Szegö, *Orthogonal Polynomials*, Colloquium Publications, No. 23, Amer. Math. Soc., Providence, R.I., 2nd ed., 1958; 3rd ed., 1967.

29 W. F. Trench, An algorithm for the inversion of finite Toeplitz matrices, *SIAM J. Appl. Math.* 12:515–522 (1964).

30 S. Zohar, Toeplitz matrix inversion: The algorithm of W. F. Trench, *J. Assoc. Comput. Mach.* 16:591–601 (1969).

Received 30 October 1988; final manuscript accepted 26 January 1989