Fraction-Free Factorization of a Toeplitz Matrix

YUVAL BISTRITZ
Tel Aviv University
School of Electrical Engineering
ISRAEL
bistritz@tau.ac.il

Abstract: The paper presents a fast and efficient integer algorithm for the fraction-free triangular factorization of a strongly regular Hermitian Toeplitz matrix. The algorithm enhances the ordinary fast Schur algorithm for this factorization with the property that when it is applied to a matrix with (Gaussian or real) integer entries, the algorithm is completed over the respective integral domain with integers of minimal length and an overall low binary complexity.

Key–Words: Schur algorithm, Integer algorithms, Covariance matrix, LDU factorization.

1 Introduction

Consider a Hermitian Toeplitz matrix

$$\mathbf{T}_n = \begin{bmatrix} c_0 & c_1 & \cdots & c_n \\ c_1^\star & c_0 & \cdots & c_{n-1} \\ \vdots & & \ddots & \\ c_n^\star & c_{n-1}^\star & \cdots & c_0 \end{bmatrix} \tag{1}$$

defined by $n+1$ complex numbers $c_i \in \mathbb{C}$ (where * denotes complex conjugate). We assume that the matrix is strongly regular, namely all $\mathbf{T}_m$, $m = 0, \ldots, n$ are nonsingular. Toeplitz matrices arise as the covariance matrix in statistical modeling of signals for linear prediction problem, in modelling layered media modeling, in realization of lattices filters by *reflection coefficients*, in modeling and coding of speech (by LPC, linear prediction coding), and more.

Fast algorithms that exploit the structure of the Toeplitz matrix to solve a set Toeplitz equations and/or obtain the inverse of the matrix in just order n^2 (rather than n^3) flops (count of floating point arithmetic operations) are collectively called Levinson algorithms [1]. Among them, the most widely used version in signal processing is an algorithm that finds $\mathbf{T}_n^{-1}$ in a lower-diagonal-upper (LDU) factorized form. This version is also known as the Levinson-Durbin algorithm, referring to its independent derivation in this context by Durbin in [2]. Some years after the wide acceptance of the Levinson algorithm, it has become realized, mostly through work of Kailath and his associates, that another fast algorithm that has been traced back to an earlier work by Schur [3] can equally produce the same reflection coefficients and hence the same lattice filter realization, see the narrative [4] [5]. The latter papers appeared in a special volume dedicated to the impact of the work of Schur on signal processing and operator theory. This volume also contains translation to English of the pertinent couple of paper of Schur that are too referenced in [3].

The paper introduces an integer version for the Schur algorithm. By Schur algorithm we refer here to a fast algorithm that obtains a triangular (LDU) factorization $\mathbf{T}_n$ and by its corresponding Levinson algorithm we refer to the algorithm that obtains the triangular factorization of $\mathbf{T}_n^{-1}$. The two algorithms are intimately related. For example they both produce the same set of the aforementioned reflection coefficients that give rise to same form lattice filter realizations and they can be used to complement each other in additional ways. The Schur algorithm has been noted to have a relative advantage in parallel computation [4] [6]. The two algorithms were subsequently extended also to certain close-to-Toeplitz matrices called Quasi-Toeplitz (QT) and to non-symmetric Toeplitz and QT matrices, see [8] and references therein. This reference also proposed an interesting combination of the Levinson and Schur algorithms to invert and factorize QT matrices beyond the restriction to Toeplitz and at most to a subclass of admissible QT matrices [7] that Levinson algorithm can handle alone.

Integer preserving (IP) Levinson algorithms for Toeplitz and admissible QT matrices were obtained in [9] [10] [11]. The IP property means that for an integer matrix the algorithm can be carried out over the integers. The integers may be the ordinary integers, denoted by $\mathbb{Z}$, or Gaussian integers defined by $\mathbb{Z}_\mathbb{G} = \{a + jb | a, b \in \mathbb{Z}\}$ ($j = \sqrt{-1}$). The terms IP and fraction-free (FF) are often used interchangeably to describe integer algorithms. In principle, an ordi-

nary algorithm (designed for floating point numbers) can be arranged to stay over integers simply by avoiding division. An example is the forthcoming division-free (DF) version of the Schur recursions brought below (to serve as an intermediate stage toward reaching the sought final form of the algorithm). The term fraction-free provides a more descriptive distinction between an algorithm that is over integers just because it is DF (but produces integers of length growing at an exponential rate) and an integer algorithm that actually uses divisions but nevertheless remains fraction-free because the divisions act to recursively remove common integer factors as soon as possible. In other words, our FF Schur algorithm will involve cancellation type divisions that act to reduce the length of the involved integers. Efficiency of an integer algorithm is determined by a measure that counts not just the number of flops but also the length of the integers involved (multiplication of longer integers costs more). It is called *computing time* or *binary complexity* [12] [13]. The cancelation of common integer factors can be shown to reduce the length of the integers involved from an exponential rate of growth (from step to step) in a DF algorithm to just a linear rate of growth in the FF SChur algorithm. We skip here details on the integer aspects of the proposed algorithm but they can be shown to be comparable to the characterization of the corresponding FF Levinson algorithm in [9] [10] [11].

2 The Ordinary Schur Algorithm

The triangular factorization of a strongly regular Hermitian matrix (1) is usually expressed by

$$\mathbf{T}_n = \mathbf{P}_n^{\star}\mathbf{D}_n\mathbf{P}_n^t \tag{2}$$

where $\mathbf{P}_n$ is a unit-triangular matrix of size $n+1$ (that for concreteness is assumed to be lower-triangular) and ^{t} denotes transpose. The adjective "unit" means that the triangular matrix has 1's on its main diagonal. $\mathbf{D}_n$ is a diagonal matrix with entries denoted by

$$\mathbf{D}_n = \text{diag}[D_0, D_1, \ldots, D_n]\,. \tag{3}$$

The assumption that the matrix is strongly regular (i.e. all its principal submatrices are non-singular) is equivalent to the condition that all the entries of $\mathbf{D}_n$ are non-zero. Indeed, the principal minors of the matrix are given by

$$\det \mathbf{T}_m = \prod_{i=0}^{m} D_i \quad , \quad m = 0, \ldots, n \tag{4}$$

In order to have a factorization over integers for an integer $\mathbf{T}_n$, we must abandon the above normalization to a unit-triangular matrix. This introduces infinitely many LDU presentations for the matrix because the diagonal matrix can compensate on any arbitrarily rescaling of each of the columns of the triangular matrix. In [8] we pursued uniformly looking Levinson and Schur algorithms with common transmission lines parameterized by the celebrated reflection coefficients, see also [7] [14]. To this end, we used there a scaled version of the Schur algorithm that produces columns for the following not unit-triangular factorization

$$\mathbf{T}_n = \mathbf{U}_n^{\star}\mathbf{D}_n^{-1}\,\mathbf{U}_n^t \tag{5}$$

The diagonal matrices $\mathbf{D}_n$ here and in (2) are the same. This means that the lower-triangular matrix $\mathbf{U}_n$ is related to the unit-triangular matrix by $\mathbf{U}_n = \mathbf{P}_n\mathbf{D}_n$. Another convenience that we must too abandon is the "no loss of generality" normalization $c_0 = 1$ that is made in virtually all previous reports on the LDU factorization of Toeplitz matrices and their inverse, including [8]. It is evidently restrictive for an integer matrix. Fortunately, the only further change in the form of the algorithm in [8] for a Hermitian Toeplitz matrix is changing the initialization there into $D_0 = c_0$. This modification ha been already included in the forthcoming reference algorithm, Algorithm 1. By this, we have reached a conceptually pleasing separation between changes that stem from abandoning traditional conveniences in the ordinary that are no longer appropriate for an integer algorithm from forthcoming changes that aim to turn Algorithm 1 that is still a "floating point" algorithm, into an efficient integer algorithm.

The ordinary (i.e. not IP) Schur algorithm below obtains the columns $\mathbf{u}_m = [u_{m,0}, \ldots, u_{m,n}]^t$ for the lower-triangular matrix $\mathbf{U}_n$

$$\mathbf{U}_n = \begin{bmatrix} u_{0,0} & 0 & \cdots & 0 \\ u_{0,1} & u_{1,1} & \cdots & 0 \\ \vdots & & \ddots & \\ u_{0,n} & u_{1,n} & \cdots & u_{n,n} \end{bmatrix} \tag{6}$$

and the entries D_m for the diagonal matrix (3) for the LDU factorization (5) in a recursive manner. The algorithm here and in the sequel use polynomial recursion (that if desirable, can be easily converted into a recursive update expressions for the involved entries) with the following convention. The recursion propagates a pair of polynomials

$$\begin{aligned} u_m(z) &= \mathbf{z}_n[0, \ldots, 0, u_{m,m}, \ldots, u_{m,n}, \ldots]^t \\ v_m(z) &= \mathbf{z}_n[0, \ldots, 0, 0, v_{m,m+1}, \ldots, v_{m,n}, \ldots]^t \end{aligned} \tag{7}$$

where $\mathbf{z} = [1, z, z^2, \ldots]$ (of length as needed). The recursion keeps nullifying the least power coefficients

such that at step m the first m coefficients of $u_m(z)$ and the first $m+1$ coefficients of the auxiliary variable $v_m(z)$ are zeros as shown in (7). Each recursion step gets as input the pair $u_{(m-1)}(z), v_{(m-1)}(z)$ both of degree n and produces as output a pair $u_m(z), v_m(z)$ of degree increased by one. Since we need only the first $n+1$ coefficients for all m, computation of coefficients of powers higher than necessary can be avoided by recursive truncation of the output polynomials beyond the power z^n, for all m. We denote the truncated polynomials by

$$\begin{aligned} u_{(m)}(z) &= \mathbf{z}_n[0,\ldots,0,u_{m,m},\ldots,u_{m,n}]^t \\ v_{(m)}(z) &= \mathbf{z}_n[0,\ldots,0,0,v_{m,m+1},\ldots,v_{m,n}]^t \end{aligned} \tag{7}$$

where $\mathbf{z}_n = [1, z, z^2, \ldots, z^n]$. In the implementation of the algorithm, in order to avoid more more computation than necessary, it is also important not to compute tthe first m and $m+1$ coefficients of $u_{(m)}(z), v_{(m)}(z)$ that are zero by structure. So our notation convention is not the most foolproof possible to ensure that only the minimal number of required coefficients is calculated. It however adheres to the notation we used in [7] in order to attain a look-alike Levinson and the Schur algorithms. As a consequence, our forthcoming FF Schur algorithm will too bear a formal resemblance to the FF Levinson algorithm for a Toeplitz matrix in [9] [10] [11].

Algorithm 1. The ordinary Schur algorithm

Given $\mathbf{T}_n$ in (1) with $c_i \in \mathbb{C}$, set $u_{(0)}(z) = \mathbf{z}_n[c_0, c_1, \ldots, c_n]^t$, $v_{(0)}(z) = u_{(0)}(z) - c_0$, $d_0 = c_0$. For $m = 1, \ldots, n$ do:

$$k_m = \frac{v_{m-1,m}}{D_{m-1}} \tag{8a}$$

$$\begin{bmatrix} u_m(z) \\ v_m(z) \end{bmatrix} = \begin{bmatrix} 1 & -k_m^\star \\ -k_m & 1 \end{bmatrix} \begin{bmatrix} zu_{(m-1)}(z) \\ v_{(m-1)}(z) \end{bmatrix} \tag{8b}$$

$$\begin{aligned} u_{(m)}(z) &= \mathbf{z}_n[0,\ldots,0,u_{m,m},\ldots,u_{m,n}]^t \\ v_{(m)}(z) &= \mathbf{z}_n[0,\ldots,0,0,v_{m,m+1},\ldots,v_{m,n}]^t \end{aligned}$$

$$D_m = u_{m,m} \tag{8c}$$

As was shown in [8], taking the coefficient vector of $u_{(m)}(z)$ to be the $(m+1)$-th column of $\mathbf{U}_n$ (6) and setting D_m $m = 0, \ldots, n$ into the diagonal matrix $\mathbf{D}_n$ (3) produces the LDU factorization (5) for $\mathbf{T}_n$.

3 FF Schur algorithms

We want to convert Algorithm 1 into an efficient integer algorithm for (1) with $c_i \in \mathbb{Z}_\mathbb{G}$ or $c_i \in \mathbb{Z}$. An obvious way to get an integers algorithm is to avoid divisions. The resulting DF recursion can be shown to create integers of length that grows at an exponential rate (due to a mechanism that will become apparent through the material in the Appendix). We begin with the DF recursions and then will use them to derive of the sought efficient FF integer algorithm.
Suppose we take the Algorithm 1 and carry it our without admitting divisions (they are introduced via the reflection coefficients, k_m, (8a)). All the variables of the DF recursions are tagged with $\hat{}$ (to be removed later in the FF version). The main replacements are $u_{(m)}(z) \to \hat{x}_{(m)}(z)$ and $v_{(m)}(z) \to \hat{y}_{(m)}(z)$. The outcome is as follows.

Division-free recursions (an intermediate result). Set $\epsilon_{-1} = 1$, $\epsilon_0 = c_0$ $\hat{x}_0(z) = \mathbf{z}_n[c_0, c_1, \ldots, c_n]^t$, $\hat{y}_0(z) = \hat{x}_0(z) - c_0$.

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

$$\hat{\delta}_m = \hat{y}_{m-1,m} \tag{9a}$$

$$\begin{bmatrix} \hat{x}_m(z) \\ \hat{y}_m(z) \end{bmatrix} = \begin{bmatrix} \hat{\epsilon}_{m-1} & -\hat{\delta}_m^\star \\ -\hat{\delta}_m & \hat{\epsilon}_{m-1} \end{bmatrix} \begin{bmatrix} z\hat{x}_{(m-1)}(z) \\ \hat{y}_{(m-1)}(z) \end{bmatrix} \tag{9b}$$

$$\begin{aligned} \hat{x}_{(m)}(z) &= \mathbf{z}_n[0,\ldots,0,\hat{x}_{m,m},\ldots,\hat{x}_{m,n}]^t \\ \hat{y}_{(m)}(z) &= \mathbf{z}_n[0,\ldots,0,0,\hat{y}_{m,m+1},\ldots,\hat{y}_{m,n}]^t \end{aligned}$$

$$\hat{\epsilon}_m = \hat{x}_{m,m} \tag{9c}$$

Let $a|b$ denote that integer a divides integer b without remainder, and $a|x_\ell(z)$ to mean that a is a common integer divisor for all the coefficients of the polynomial $x_\ell(z)$. Then the next lemma is proved in the appendix.

Lemma 1. *For an integer $\mathbf{T}_n$ in (1) (i.e. c_i in $\mathbb{Z}_\mathbb{G}$ or in $\mathbb{Z}$) the above DF recursions produce $\hat{\epsilon}_m$ in $\mathbb{Z}$ such that $\hat{\epsilon}_m$ is a common factor of all the integer coefficients of the polynomials from $\hat{x}_{(m+2)}(z)$ $\hat{y}_{(m+2)}(z)$ and on, i.e.*

$$\hat{\epsilon}_m | \hat{x}_{(m+2+i)}(z), \hat{y}_{(m+2+i)}(z)\,,\ i = 0, 1, ..., n-m-2$$

Lemma 1 is proved in the appendix where it is further explained that it implies that Algorithm 2 is indeed fraction-free as stated in the next theorem.

Theorem 2 (Algorithm 2 is FF). *For $\mathbf{T}_n$ in (1) with $c_i \in \mathbb{Z}_\mathbb{G}$ or $c_i \in \mathbb{Z}$, all the coefficients that Algorithm 2 produces are in $\mathbb{Z}_\mathbb{G}$ or $\mathbb{Z}$, respectively, and all the computation can be completed over the respective integral domain.*

Algorithm 2. FF Schur algorithm for $\mathbf{T}_n$

Set $\epsilon_{-1} = 1$, $\epsilon_0 = c_0$, $x_0(z) = \mathbf{z}_n[c_0, c_1 \ldots, c_n]^t$, $y_0(z)=x_0(z) - c_0$
For $m = 1, \ldots, n$ do:

$$\delta_m = y_{m-1,m} \tag{10a}$$

$$\begin{bmatrix} x_m(z) \\ y_m(z) \end{bmatrix} = \frac{1}{\epsilon_{m-2}} \begin{bmatrix} \epsilon_{m-1} & -\delta_m^\star \\ -\delta_m & \epsilon_{m-1} \end{bmatrix} \begin{bmatrix} z x_{(m-1)}(z) \\ y_{(m-1)}(z) \end{bmatrix} \tag{10b}$$

$$x_{(m)}(z) = \mathbf{z}_n[0, \ldots, 0, x_{m,m}, \ldots, x_{m,n}]^t$$
$$y_{(m)}(z) = \mathbf{z}_n[0, \ldots, 0, 0, y_{m,m+1}, \ldots, y_{m,n}]^t$$

$$\epsilon_m = x_{m,m} \tag{10c}$$

The next proposition draws the relationship between the new and the classical Schur algorithms.

Proposition 3. *The following relations between the FF Schur algorithm, Algorithm 2, and the ordinary Schur algorithm in Algorithm 1 hold*

$$\begin{bmatrix} x_{(m)}(z) \\ y_{(m)}(z) \end{bmatrix} = \epsilon_{m-1} \begin{bmatrix} u_{(m)}(z) \\ v_{(m)}(z) \end{bmatrix}, \; m = 0, \ldots, n \tag{11a}$$

and by consequence,

$$\epsilon_m = \epsilon_{m-1} D_m \,, \quad m = 1, \ldots, n \tag{11b}$$

$$\delta_m = \epsilon_{m-1} k_m \,, \quad m = 1, \ldots, n \tag{11c}$$

Proof. First note that if (11a) is true then it implies (11b) because $\epsilon_m = x_{m,m} = \epsilon_{m-1} u_{m,m} \overset{(8c)}{=} \epsilon_{m-1} D_m$. Eq. (11a) implies also (11c) because $\delta_m \overset{(10a)}{=} y_{m-1,m} = \epsilon_{m-2} v_{m-1,m} \overset{(8a)}{=} \epsilon_{m-2} D_{m-1} k_m$ that with (11b) gives (11c). The main relation (11a) can be proved by induction as follows. By definition, $x_0(z) = \epsilon_{-1} u_0(z)$ and $y_0(z) = \epsilon_{-1} v_0(z)$. Next, $x_{(1)}(z) = \epsilon_0 z x_0(z) - \delta_1^\star y_0(z) = c_0\{z u_0(z) - k_1^\star v_0(z)\} = \epsilon_0 u_{(1)}(z)$ and $y_{(1)}(z) = -\delta_1 z x_0(z) - \epsilon_0 y_0(z) = c_0\{-k_1 z u_0(z) + v_0(z)\} = \epsilon_0 v_{(1)}(z)$. Assume that (11a) holds for $m = 0, 1, \ldots \ell - 1$. Then for $m = \ell$,

$$\epsilon_{\ell-1} u_{(\ell)}(z) = z u_{(\ell-1)}(z) - k_\ell^\star v_{(\ell-1)}(z) =$$
$$\epsilon_{\ell-1} \left\{ z \frac{x_{(\ell-1)}(z)}{\epsilon_{\ell-2}} - \frac{y_{\ell-1,\ell}^\star(z)}{\epsilon_{\ell-1}} \frac{y_{(\ell-1)}(z)}{\epsilon_{\ell-2}} \right\} =$$
$$\frac{1}{\epsilon_{\ell-2}} \left\{ \epsilon_{\ell-1} z x_{(\ell-1)}(z) - \delta_\ell^\star y_{(\ell-1)}(z) \right\} = x_{(\ell)}(z)$$

Similarly,

$$\epsilon_{\ell-1} v_{(\ell)}(z) = -k_\ell z u_{(\ell-1)}(z) + v_{(\ell-1)}(z) =$$
$$\epsilon_{\ell-1} \left\{ -\frac{y_{\ell-1,\ell}(z)}{\epsilon_{\ell-1}} z \frac{x_{(\ell-1)}(z)}{\epsilon_{\ell-2}} + \frac{y_{(\ell-1)}(z)}{\epsilon_{\ell-2}} \right\} =$$
$$\frac{1}{\epsilon_{\ell-2}} \left\{ -\delta_\ell z x_{(\ell-1)}(z) + \epsilon_{\ell-1} y_{(\ell-1)}(z) \right\} = x_{(\ell)}(z)$$

This completes the proof. □

Using (11b) repeatedly gives $\epsilon_m = \prod_{\ell=0}^m D_\ell$ that can be compared to (4) to conclude the important fact that the algorithm produces directly

$$\epsilon_m = \det \mathbf{T}_m \,, \quad m = 0, \ldots, n \tag{12}$$

Thus the ϵ_m's are same as those in the FF Levinson algorithm for a Toeplitz matrix [9] [10]. Same is true also for the δ_m's here and there. Upon comparing the coefficient of the power z^m in the upper line of equation (10b) we get

$$\epsilon_m = \frac{\epsilon_{m-1}^2 - |\delta_m|^2}{\epsilon_{m-2}} \tag{13}$$

In the FF Levinson algorithm, the computation of the δ_m involves an inner product between two vectors and (13) saves there an alternative possible computation of ϵ_m by a second inner product [9] [10]. In the current FF Schur algorithm both δ_m and ϵ_m are obtained directly (without "side computation" of inner products). This is the FF dressing for a similar advantage of the ordinary Schur algorithm over the ordinary Levinson algorithm for concurrent computation [4] [6]. The FF Schur algorithm provides a tool to compute the reflection coefficients k_m in an error-free manner via (11c). As is well known, all $|k_m| < 1$ is a necessary and sufficient condition for the matrix to be positive definite and they are not likely to take integer values also in more general situations. However, the FF algorithm offers a way to reach their value expressed by the ratio of two integers computed in an error-free manner.

Theorem 4. *Algorithm 2 produces the next LDU factorization*

$$\mathbf{T}_n = \mathbf{X}_n^\star \, \mathbf{E}_n^{-1} \, \mathbf{X}_n^t \tag{14}$$

where $\mathbf{X}_n$ is a lower-triangular matrix whose columns are the coefficient vectors of $x_{(m)}(z)$, $m = 0, \ldots, n$, viz.

$$\mathbf{X}_n = \begin{bmatrix} x_{0,0} & 0 & \cdots & 0 \\ x_{0,1} & x_{1,1} & \cdots & 0 \\ \vdots & & \ddots & \\ x_{0,n} & x_{1,n} & \cdots & x_{n,n} \end{bmatrix} \tag{15}$$

and $\mathbf{E}_n$ *is a diagonal matrix given by the product* $\mathbf{E}_n = \grave{\mathbf{E}}_n \dot{\mathbf{E}}_n$

$$\grave{\mathbf{E}}_n = \text{diag}\,[\epsilon_{-1}, \epsilon_0, \cdots, \epsilon_{n-1}] \tag{16}$$

$$\dot{\mathbf{E}}_n = \text{diag}\,[\epsilon_0, \epsilon_1, \cdots, \epsilon_n] \tag{17}$$

Proof. Proposition 3 implies the matrix relations

$$\mathbf{X}_n = \mathbf{U}_n \grave{\mathbf{E}}_n\,, \quad \dot{\mathbf{E}}_n = \grave{\mathbf{E}}_n \mathbf{D}_n\,.$$

The IP factorization (14) follows by setting the above relations into the factorization (5). □

Thus, Algorithm 2 produces a FF Gaussian/real integer LDU factorization for a Gaussian/real integer Toeplitz matrix. Note that the diagonal matrices are real also in the complex case, see (13). Writing the diagonal as the product of the indicated two diagonal matrices is useful to express the factorization in terms of the minimal possible length of integers. The algorithm can be shown to have a restrained growth of the length of integers similar to the FF Levinson algorithm in [9] [10] [11].

4 A Numerical Examples

Algorithm 2 is easy to program and running it with some decent size matrices is the way to learn its effectiveness. Here is a toy numerical example (useful to test your programming). The application of Algorithm 2 to the next integer Hermitian Toeplitz matrix

$$\mathbf{T}_3 = \begin{bmatrix} 7 & 3+j & 1+2j & 1+j \\ 3-j & 7 & 3+j & 1+2j \\ 1-2j & 3-j & 7 & 3+j \\ 1-j & 1-2j & 3-j & 7 \end{bmatrix}$$

runs as follows. The initialization is: $x_{(0)}(z) = 7 + (3+j)z + (1+2j)z^2 + (1+j)z^3$, $y_{(0)}(z) = (3+j)z + (1+2j)z^2 + (1+j)z^3$ and $\epsilon_0 = 7$. Step $m = 1$: $\delta_1 = 3+j$, $x_{(1)}(z) = 39z + (16+2j)z^2 + (3+12j)z^3$, $y_{(1)}(z) = (-1+8j)z^2 + 6z^3$, $\epsilon_1 = 39$. Step $m = 2$: $\delta_2 = -1+8j$, $x_{(2)}(z) = 208z^2 + (90+18j)z^3$, $y_{(2)}(z) = (38-18j)z^3$, $\epsilon_2 = 208$. Step $m = 3$: $\delta_3 = 38-18j$, $x_{(3)}(z) = 1064z^3$, $y_{(3)}(z) = 0$, $\epsilon_3 = 1064$. Thus the algorithm produces the factorization $\mathbf{T}_3 = \mathbf{X}_3^\star \mathbf{E}_3^{-1} \mathbf{X}_3^t$ (14) with

$$\mathbf{X}_3 = \begin{bmatrix} 7 & 0 & 0 & 0 \\ 3+j & 39 & 0 & 0 \\ 1+2j & 16+2j & 208 & 0 \\ 1+j & 3+12j & 90+18j & 1064 \end{bmatrix}$$

and $\mathbf{E}_3$ given by the product of $\grave{\mathbf{E}} = \text{diag}\,[1, 7, 39, 208]$ and $\dot{\mathbf{E}} = \text{diag}\,[7, 39, 208, 1064]$.

5 concluding remarks

The paper presented an efficient fraction-free algorithm for the triangular (LDU) factorization of a strongly regular Hermitian Toeplitz matrix. The presented FF Schur algorithm is usable also for not integer matrices but is designed to provide an efficient fraction-free algorithm for an integer (Gaussian or ordinary) that involves integers of minimal possible length (in general). This is a sought property for computer algebra systems (and for computation with symbols). It also provides error-free computation for integer matrices and can improve the accuracy of computation also for not-integer matrices. A matrix with decimal entries of acceptable accuracy can always be scaled up into an integer matrix. Then the reflection coefficients can be obtained with no further loss of accuracy (they are not affected by the scaling) and after a similarly accurate integer LDU factorization has been reached it can be inversely scaled to an LDU with floating point values of improved accuracy. Extension of the current algorithm to FF Schur algorithms for nonsymmetric and to close to Toeplitz matrices as in [8] is planned for another publication.

APPENDIX: PROVING THE FRACTION-FREE PROPERTY

We begin by preparing some properties of the DF recursions needed for the proof of Lemma 1.

Lemma 5. *The recursions in (9) have the following (a)-(c) properties.*

$$(a)\ \hat{\epsilon}_m = \hat{\epsilon}_{m-1}^2 - \hat{\delta}_m \hat{\delta}_m^\star \tag{18}$$

$$(b)\ \hat{x}_{(m)}(z)\hat{x}_{(m)}^\star(w) - \hat{y}_{(m)}(z)\hat{y}_{(m)}^\star(w) = \tag{19}$$
$$\hat{\epsilon}_m\{zw\hat{x}_{(m-1)}(z)\hat{x}_{(m-1)}^\star(w) - \hat{y}_{(m-1)}(z)\hat{y}_{(m-1)}(w)\}$$

$$(c)\ \hat{x}_{(m)}(z)\hat{y}_{(m)}(w) - \hat{y}_{(m)}(z)\hat{x}_{(m)}(w) = \tag{20}$$
$$\hat{\epsilon}_m\{zw\hat{x}_{(m-1)}(z)\hat{y}_{(m-1)}(w) - \hat{y}_{(m-1)}(z)\hat{x}_{(m-1)}(w)\}$$

Proof. Property (a) is obtained by comparing the coefficient of z^m at the two sides of the upper part of (9b), $\hat{x}_{m,m} = \hat{\epsilon}_{m-1}x_{m-1,m-1} - \hat{\delta}_m^\star y_{m-1,m}$ and using the definitions of $\hat{\epsilon}_m$ and $\hat{\delta}_m$ there. To obtain (b), use $K = \text{diag}[1, -1]$ and (9b) to rewrite (b) as follows

$$[\hat{x}_{(m)}(z), \hat{y}_{(m)}(z)]K[\hat{x}_{(m)}(w), \hat{y}_{(m)}(w)]^t = \tag{21}$$
$$[z\hat{x}_{(m-1)}(z), \hat{y}_{(m-1)}(z)]\Theta_m^t K \Theta_m^\star \cdot$$
$$[w\hat{x}_{(m-1)}(w), \hat{y}_{(m-1)}(w)]^t$$

Then evaluate for it

$$\Theta_m^t K \Theta_m^\star = (\hat{\epsilon}_{m-1}^2 - \hat{\delta}_m \hat{\delta}_m^\star) K \overset{(18)}{=} \hat{\epsilon}_m K \qquad (22)$$

This proves property (b). To prove (c), evaluate the l.h.s. of property (c), using $J = \left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right]$,

$$[\hat{x}_{(m)}(z), \hat{y}_{(m)}(z)] K [\hat{y}_{(m)}(w), \hat{x}_{(m)}(w)]^t =$$
$$[z\hat{x}_{(m-1)}(z), \hat{y}_{(m-1)}(z)] \Theta_m^t K J \Theta_m J \cdot$$
$$[\hat{y}_{(m-1)}(w), w\hat{x}_{(m-1)}(w)]^t$$

But $J\Theta_m J = \Theta_m^\star$ and then at the center we have again, using (22), $\Theta_m^t K \Theta_m^\star = \hat{\epsilon}_m K$. This completes the proof of property (c). □

Proof of Lemma 1. For $\mathbf{T}_n$ with entries in $\mathbb{Z}_\mathbb{G}$ or $\mathbb{Z}$ the DF recursions remain over the respective integral domain because it involves no division. The fact that $\hat{\epsilon}_m \in \mathbb{Z}$ (also for input in $\mathbb{Z}_\mathbb{G}$) becomes apparent from (18). In the following we say that a is congruent to b if $a = b$ modulus $\hat{\epsilon}_m$ and denote this by $a \cong b$. We can then conclude that $\hat{\epsilon}_m | \hat{x}_{(m+2)}(z), \hat{y}_{(m+2)}(z)$ if we prove that $\hat{x}_{(m+2)}(z) \cong 0$ and $\hat{y}_{(m+2)}(z) \cong 0$. The required successive substitutions is simplified by noticing that the structure of the recursion allows to drop from a sum of terms any term that is already recognized to be congruent to 0, because a term that contain the factor $\hat{\epsilon}_m$ inherits it to all forthcoming summands in which it participates. We used this technique also to establish the FF property for the Levinson algorithms [9] [10] [11]. It was first introduced as a tool to derive efficient two-dimensional discrete stability tests [15] and afterward was used also to derive FF stability tests for discrete and continuous linear systems [16] [17] [18].

From step m to step $m+1$ we obtain

$$\hat{x}_{(m+1)}(z) = \hat{\epsilon}_m z \hat{x}_{(m)}(z) - \hat{\delta}_{m+1}^\star \hat{y}_{(m)}(z)$$
$$\cong -\hat{\delta}_{m+1}^\star \hat{y}_{(m)}(z) \qquad (23)$$
$$\hat{y}_{(m+1)}(z) = -\hat{\delta}_{m+1} z \hat{x}_{(m)}(z) + \hat{\epsilon}_m \hat{y}_{(m)}(z)$$
$$\cong -\hat{\delta}_{m+1} z \hat{x}_{(m)}(z) \qquad (24)$$

from where we also obtain

$$\hat{\epsilon}_{m+1} = \hat{x}_{m+1,m+1} \overset{(23)}{\cong} -\hat{\delta}_{m+1}^\star \hat{\delta}_{m+1} \qquad (25)$$

At the next step, from $m+1$ to $m+2$, we get

$$\hat{\delta}_{m+2} = y_{m+1,m+2} \overset{(24)}{\cong} -\hat{\delta}_{m+1} \hat{x}_{m,m+1} \qquad (26)$$

and

$$\hat{x}_{(m+2)}(z) = \hat{\epsilon}_{m+1} z \hat{x}_{(m+1)}(z) - \hat{\delta}_{m+2}^\star \hat{y}_{(m+1)}(z)$$
$$\overset{(25)(23)(26)(24)}{\cong} \left(-\hat{\delta}_{m+1}^\star \hat{\delta}_{m+1}\right) \left(-z\hat{\delta}_{m+1}^\star \hat{y}_{(m)}(z)\right) -$$
$$\left(-\hat{\delta}_{m+1}^\star \hat{x}_{m,m+1}^\star\right) \left(-\hat{\delta}_{m+1} z \hat{x}_{(m)}(z)\right) =$$
$$z\hat{\delta}_{m+1}\hat{\delta}_{m+1}^\star \left\{\hat{y}_{m,m+1}^\star \hat{y}_{(m)}(z) - \hat{x}_{m,m+1}^\star \hat{x}_{(m)}(z)\right\} \cong 0$$

The congruence to 0 follows because the term in the curly bracket is equal to the coefficient of w^m in (19). Similarly,

$$\hat{y}_{(m+2)}(z) = -\hat{\delta}_{m+2}^\star z \hat{x}_{(m+1)}(z) + \hat{\epsilon}_{m+1} \hat{y}_{(m+1)}(z) \cong$$
$$\left(-\hat{\delta}_{m+1} \hat{x}_{m,m+1}\right) \left(-z\hat{\delta}_{m+1}^\star \hat{y}_{(m)}(z)\right) -$$
$$\left(-\hat{\delta}_{m+1}^\star \hat{\delta}_{m+1}\right) \left(-\hat{\delta}_{m+1} z \hat{x}_{(m)}(z)\right) =$$
$$z\hat{\delta}_{m+1}\hat{\delta}_{m+1}^\star \left\{-\hat{x}_{m,m+1}^\star \hat{y}_{(m)}(z) + \hat{y}_{m,m+1} \hat{x}_{(m)}(z)\right\} \cong 0$$

This time the congruence to 0 follows because the term in the curly bracket is recognized as the coefficient of w^m in (20). This completes the proof of Lemma 1 that $\hat{\epsilon}_m | \hat{x}_{(m+2)}(z), \hat{y}_{(m+2)}(z)$ □

It can be concluded that this factor propagates and multiplies in the DF recursions such that $\hat{\epsilon}_m^{1+i} | \hat{x}_{(m+2+i)}(z), \hat{y}_{(m+2+1)}(z)$, $i = 0, 1, 2, \ldots$. Consequently it causes a severe exponential rate of growth of the integers in the DF recursions.

Finally, the claimed IP property of Algorithm 2 stated in Theorem 2 follows from Lemma 1 after realizing that the common factor $\hat{\epsilon}_m$ exposed at step $m+2$ can (and should) be removed as soon as it was formed (i.e. at step $m+2$). Removing $\hat{\epsilon}_m$ at step $m+2$ does not interfere with the a similar cycle that makes $\hat{\epsilon}_{m+1}$ a removable common integer factor at step $m+3$ and so forth.

Acknowledgements: This research was supported by the Israel Science Foundation (grant No. 1698/16).

References:

[1] N. Levinson, "The Wiener RMS error criterion in filter design and prediction", *J. Math. Phys.*, vol. 25, pp. 261-278, 1947.

[2] J. Durbin, "The Fitting of Time-Series Models", *Rev. the Int. Stat. Inst.*, vol. 28(3), pp. 233-244, 1960.

[3] I. Schur, "Über potenzreihen, die in innern des einheitskreises beschränkt sind" *Journal für die Reine und Angewandte Mathematik*, vol. pp. 147

205-232, 1917 and vol. 148 pp. 122-145,1918. [English translation in *I Schur Methods in Operator Theory and Signal Processing, Operator Theory: Advances and Applications,* I. Gohberg (ed.) vol. 18, pp. 31-88, Basel, Switzerland: Birkhaüer Verlag, 1986.

[4] T. Kailath, "A Theorem of I. Schur and its Impact on Modern Signal Processing," in *I. Schur Methods in Operator Theory and Signal Processing, Operator Theory: Advances and Applications*, I. Gohberg (ed.), Vol. 18, pp. 9-30, Birkhaüser Verlag, Basel, 1986.

[5] H. Lev-Ari and T. Kailath, "Triangular factorization of structured Hermitian matrices", in *I. Schur Methods in Operator Theory and Signal Processing, Operator Theory: Advances and Applications*, I. Gohberg (ed.), Vol. 18, pp. 301-324, Birkhaüser Verlag, Basel, 1986.

[6] S-Y Kung and Y-H Hu, "A highly concurrent algorithm and pipelined architecture for solving Toeplitz systems", *IEEE Trans. Acoustics, Speech, and Signal Processing*, ASSP-31 (1), pp. 83-96, 1983.

[7] H. Lev-Ari and T. Kailath, "Lattice Filter Parametrization and Modeling of Nonstationary Processes" IEEE Trans. on Information Theory, vol. IT - 30 (1), 1984.

[8] Y. Bistritz and T. Kailath, "Inversion and factorization of non-Hermitian quasi-Toeplitz matrices" *Linear Algebra and its Applications,* vol. 98, pp. 77-121, 1988.

[9] Y. Segalov and Y. Bistritz, "Levinson algorithm over integers for strongly regular hermitian Toeplitz matrices" *Proc. of the 2008 IEEE Int. Conference on Acoustics, Speech, and Signal Processing,* ICASSP-2008, Las Vegas, USA.

[10] Y. Bistritz and Y. Segalov, "Fraction-free inversion of a Toeplitz matrix" *Proc. of the 2010 IEEE Int. Symposium on Circuits and Systems*, ISCAS-2010, Paris, France.

[11] Y. Bistritz and Y. Segalov, "Integer Levinson algorithms for quasi-Toeplitz matrices" *Proc. of the 2010 American Control Conference* ACC-2010, Baltimore, USA.

[12] P. G. Anderson, M.R. Garey and L.E. Heindel, "Computational aspects of deciding if all roots of a polynomial lie within the unit circle", *Computing* vol. 16, pp. 293-304, 1976.

[13] S. Basu, R. Pollack and M. F. Roy, *Algorithms in Real Algebraic Geometry*, 2nd Ed., Springer-Verlag, 2008.

[14] T. Kailath, F. Bruckstein and D. Morgan "Fast matrix factorization via discrete transmission line", *Linear Algebra and its Applications,* vol. 75, pp. 1-25, 1986.

[15] Y. Bistritz, "Stability testing of two-dimensional discrete linear system polynomials by a two-dimensional tabular form," *IEEE Trans. on Circuits Syst. I:Fundam. Theory Appl.*, vol. 46 (June), pp. 666-676, 1999.

[16] Y. Bistritz, "An efficient integer-preserving stability test for discrete-time systems", *Circuits Syst. Signal Process.*, vol. 23 (3), pp. 195-213, 2004.

[17] Y. Bistritz, "Fraction-free unit-circle stability tests", *Circuits Syst Signal Process*, vol. 33, pp. 3783–3807, 2014

[18] Y. Bistritz, "Optimal fraction-free Routh tests for complex and real integer polynomials *IEEE Trans. on Circuits and Systems - I*, vol. 60 (9), pp. 2453-2464, 2013.