Numer. Math. 70: 181–227 (1995)

Numerische Mathematik

Electronic Edition

Look-ahead Levinson and Schur algorithms for non-Hermitian Toeplitz systems

Martin H. Gutknecht[1], **Marlis Hochbruck**[2]

[1] Interdisciplinary Project Center for Supercomputing, ETH Zurich, ETH-Zentrum, CH-8092 Zurich, Switzerland. E-mail: na.gutknecht@na-net.ornl.gov
[2] Mathematisches Institut, Universität Tübingen, Auf der Morgenstelle 10, D-72076 Tübingen, Germany. E-mail: marlis@na.mathematik.uni-tuebingen.de

Received July 26, 1993

Dedicated to Professor J. Stoer on the occasion of his 60th birthday

Summary. We present generalizations of the nonsymmetric Levinson and Schur algorithms for non-Hermitian Toeplitz matrices with some singular or ill-conditioned leading principal submatrices. The underlying recurrences allow us to go from any pair of successive well-conditioned leading principal submatrices to any such pair of larger order. If the look-ahead step size between these pairs is bounded, our generalized Levinson and Schur recurrences require $O(N^2)$ operations, and the Schur recurrences can be combined with recursive doubling so that an $O(N \log^2 N)$ algorithm results. The overhead (in operations and storage) of look-ahead steps is very small. There are various options for applying these algorithms to solving linear systems with Toeplitz matrix.

Mathematics Subject Classification (1991): 65F05; 30E05, 41A21, 42A70, 60G25

1. Introduction

Systems of linear equations with Toeplitz coefficient matrices $\mathbf{T} := \mathbf{T}_{N+1} \in \mathbb{C}^{N+1,N+1}$ arise in many important applications. It is well-known that the Toeplitz structure of $\mathbf{T}$ can be exploited in their solution. There are mainly two types of recursive algorithms of complexity $O(N^2)$: Levinson-type algorithms compute — at least implicitly — an LDU decomposition of the inverse of $\mathbf{T}$; Schur-type algorithms, such as the Bareiss algorithm, compute an LDU decomposition of $\mathbf{T}$ itself. An important difference between these two types is that Levinson's algorithm requires the computation of two inner products of size $n+1$ in step n, while Schur's algorithm works without inner products. Therefore, for parallel architectures, Schur type algorithms are better suited.

In addition to many variants of the classical Levinson [27] and Schur [31] algorithms, a number of superfast algorithms, which are of complexity $O(N \log^2 N)$, have been proposed (Bitmead and Anderson [5], Morf [29], Musicus [30], de Hoog [11], Ammar and Gragg [2, 1, 3]). These algorithms are also variations of the Schur algorithm. However, they do not necessarily yield the complete LDU decomposition

of **T**, but instead they only aim at the last row of L and the last column of U. Then an inversion formula (like the Gohberg-Semencul formula [16]) is applied for representing $\mathbf{T}^{-1}$.

However, a major drawback of all these algorithms is that they require all the leading principal submatrices of **T** to be nonsingular. If only one of them is singular, the algorithms break down. In finite precision arithmetic, such exact breakdowns are rare, but nearly singular or ill-conditioned submatrices can occur frequently. In this case, the algorithms become unstable and may produce results that are far away from the exact solution.

There are also fast and superfast algorithms for Hankel systems, but if they are applied to a reflected Toeplitz matrix, which is Hankel, then the nonsingularity condition for the leading principal Hankel submatrices corresponds to a nonsingularity condition for certain submatrices along the antidiagonal of the Toeplitz matrix. For a Toeplitz matrix that approximates a typical Toeplitz operator, these "antidiagonal" submatrices tend to be very ill conditioned, however. Hence, Hankel solvers are not suitable for solving typical Toeplitz systems. Therefore, they are left out in our discussion.

In the past ten years, the interest in the solution of indefinite or nonsymmetric Toeplitz systems has grown substantially. A number of fast algorithms that can deal with exactly singular principal submatrices were developed; see [9, 14, 19] for references. The more important, but more difficult task of avoiding near-breakdowns, which is a must for any kind of numerical stability, was only approached more recently. Since row or column pivoting would destroy the Toeplitz structure, some kind of block pivoting must be applied, which one may also call a look-ahead strategy. (For a Hankel system, the look-ahead Lanczos algorithm [18, 20] is in fact readily translated into a fast look-ahead Hankel solver [15].) It turns out that there is still much freedom in choosing the details of such block pivoting schemes. Sweet [32] was the first to sketch an algorithm that generalizes the Bareiss version [4] of the Schur algorithm and performs some local pivoting, followed by steps to recover the Toeplitz structure. However, its details, worked out in [33], are very complicated. Chan and Hansen [9, 23, 10] developed an inverse block LDU decomposition that generalizes the Levinson algorithm and preserves the Toeplitz structure. Recently, Freund and Zha [14] came up with another such algorithm, which also determines an inverse block LDU decomposition, but has a much smaller overhead. Their approach is based on an interpretation of the Levinson algorithm in terms of formally biorthogonal polynomials. Both the Chan-Hansen and the Freund-Zha algorithm can handle exact and near-breakdowns.

At the same time, Gutknecht [19] presented new general row and sawtooth recurrences for the Padé table. Making use of the well known connection between the computation of Padé approximants and the solution of Toeplitz systems, he proposed another six new fast and superfast solvers that can handle exact and near-breakdowns. However, none of them is exactly a generalization of either the Schur or the Levinson algorithm in the sense that it reduces to one of these classical algorithms in the absence of look-ahead steps. The underlying idea for the methods in [19] is analogous to the one of Cabay and Meleshko's weakly stable diagonal Padé recurrences and the corresponding look-ahead Hankel solver [8, 28].

Here we now present analogous true generalizations of the Levinson and Schur algorithms, but give pure linear-algebra derivations without using tools and results from Padé approximation. As in the Chan-Hansen and the Freund-Zha algorithm, the LDU decompositions that are implicitly produced by the classical algorithms now become block LDU decompositions. However, our algorithms for finding these

decompositions differ from the other two mentioned. The main difference is that we need not compute the full inverse block LDU decomposition, but only the first row of each block of the L-factor and the first column of each block of the U-factor. This has the effect that look-ahead steps require no additional storage, except for a matrix of order $2k-2$ and a few vectors of size $k-1$. Therefore, when we choose to represent $\mathbf{T}^{-1}$ according to the Gohberg-Semencul formula, the total storage required is just $2N + O(k_{\max}^2)$, where $k_{\max}$ denotes the length of the longest look-ahead step. This compares very favorably with the competing algorithms. The computational overhead depends on what exactly is computed and on the look-ahead strategy applied. For our algorithms we give a choice of three such strategies differing in cost and quality of the condition estimate. If the cheapest is used, and if we just want to apply the Gohberg-Semencul formula, the computational overhead of our look-ahead Levinson algorithm is very small, roughly half of the one in the Freund-Zha algorithm. In any case, all these algorithms are of complexity $O(N^2)$ if we assume that none of the blocks becomes arbitrarily large as $N \to \infty$.

Our approach also leads to a superfast, $O(N \log^2 N)$ Schur-type look-ahead solver for general Toeplitz systems that reduces to a variation of the algorithm of de Hoog [11] if no look-ahead steps are necessary. At the end it makes use of the Gohberg-Semencul inversion formula as well. If applied to Hermitian indefinite systems, all the methods proposed here are easily modified such that they capitalize upon the symmetry.

On the other hand, the algorithms introduced in this paper have the disadvantage that look-ahead steps are terminated only when two successive leading principal submatrices are well conditioned. Hence, the look-ahead step size is in general larger than in the competing algorithms. In fact, a look-ahead step of length k in our algorithms corresponds typically to a normal step plus a look-ahead step of length $k-1$ of Chan and Hansen [9, 23, 10] or Freund and Zha [14], or of Gutknecht's sawtooth algorithms [19]. Much worse, successive look-ahead steps of the other algorithms transform into just one long look-ahead step here. However, in this situation, one could switch temporarily to the analogue sawtooth algorithm based on regular pairs; see [21] for a detailed discussion of a realization of this idea.

We present several numerical examples for the new fast and superfast algorithms. They indicate that the proposed algorithms have good numerical properties. Since the computation of the residual and the application of the inversion formula require only $O(N \log N)$ arithmetic operations when FFT techniques are used, iterative refinement for improving the accuracy can be applied very effectively. In all our examples, one step of iterative refinement was sufficient to reduce the relative error to small multiples of the machine precision.

The paper is organized as follows: We start with a review of the nonsymmetric Levinson and Schur algorithms in Sects. 2–3. In Sects. 4-5 we introduce and discuss column-regular pairs and derive the recurrences that are the basis of our look-ahead Levinson and Schur algorithms, which we summarize in Sect. 7 after the discussion of several look-ahead strategies in Sect. 6. The superfast version of the look-ahead Schur algorithm is sketched in Sect. 8. In Sect. 9 we reinterprete our look-ahead Schur algorithm as matrix transformation generalizing the Bareiss algorithm. Along with this goes an interpretation of our look-ahead Levinson algorithm as a matrix transformation. In Sect. 10 we list several options for applying our algorithms to solving linear systems with Toeplitz matrix. The computational costs, and, in particular, the look-ahead overhead, are specified in Sect. 11 and compared with the costs of the

algorithms of Chan and Hansen [9, 23, 10] and Freund and Zha [14]. Finally, Sect. 12 contains numerical examples for the various algorithms we suggest here.

2. The nonsymmetric Levinson algorithm

Consider a finite or infinite nested sequence $\{\mathbf{T}_n\}_{n=1}^{N+1}$ ($N \leq \infty$) of real or complex Toeplitz matrices

$$\mathbf{T}_n := \begin{bmatrix} \mu_0 & \mu_{-1} & \dots & \mu_{-n+1} \\ \mu_1 & \mu_0 & \dots & \mu_{-n+2} \\ \vdots & \vdots & \dots & \vdots \\ \mu_{n-1} & \mu_{n-2} & \dots & \mu_0 \end{bmatrix}. \tag{2.1}$$

In this preliminary section, where we review the Levinson algorithm, we assume that, for all n, $\mathbf{T}_n$ is nonsingular. Then the two systems

$$\mathbf{T}_n \left[\begin{array}{c|c} \hat{\rho}_{0,n} & \rho_{n-1,n} \\ \vdots & \vdots \\ \hat{\rho}_{n-1,n} & \rho_{0,n} \end{array}\right] = - \left[\begin{array}{c|c} \mu_{-n} & \mu_1 \\ \vdots & \vdots \\ \mu_{-1} & \mu_n \end{array}\right], \tag{2.2}$$

which are called *Yule-Walker equations*, have unique solutions. If we set $\hat{\rho}_{n,n} := 1$ and $\rho_{n,n} := 1$ and let

$$\hat{\varepsilon}_{n,n} := \sum_{k=0}^{n} \mu_k \hat{\rho}_{n-k,n}, \quad \varepsilon_{n,n} := \sum_{k=0}^{n} \mu_{-k} \rho_{n-k,n}, \tag{2.3}$$

then these solutions are seen to solve also the bordered linear systems

$$\mathbf{T}_{n+1} \left[\begin{array}{c|c} \hat{\rho}_{0,n} & \rho_{n,n} \\ \vdots & \rho_{n-1,n} \\ \hat{\rho}_{n-1,n} & \vdots \\ \hat{\rho}_{n,n} & \rho_{0,n} \end{array}\right] = \left[\begin{array}{c|c} 0 & \varepsilon_{n,n} \\ \vdots & 0 \\ 0 & \vdots \\ \hat{\varepsilon}_{n,n} & 0 \end{array}\right]. \tag{2.4}$$

For $n \leq N$, instead of solving (2.2) we can find its solutions from (2.4) by replacing $\hat{\varepsilon}_{n,n}$ and $\varepsilon_{n,n}$ by 1 first, and then renormalizing the so obtained solutions if necessary. In fact, the quantities $\hat{\varepsilon}_{n,n}$ and $\varepsilon_{n,n}$ from (2.3) must be nonzero, since, otherwise, (2.4) would imply that $\hat{\rho}_{0,n} = \dots = \hat{\rho}_{n,n} = 0$ and $\rho_{0,n} = \dots = \rho_{n,n} = 0$, respectively, because $\mathbf{T}_{n+1}$ is nonsingular.

Following Bultheel [7], we let additionally

$$\hat{\pi}_{0,n} := \sum_{k=0}^{n} \mu_{-k-1} \hat{\rho}_{k,n}, \quad \pi_{0,n} := \sum_{k=0}^{n} \mu_{k+1} \rho_{k,n}, \tag{2.5}$$

and conclude further that

$$\mathbf{T}_{n+2} \left[\begin{array}{c|c} 0 & \rho_{n,n} \\ \hat{\rho}_{0,n} & \rho_{n-1,n} \\ \vdots & \vdots \\ \hat{\rho}_{n-1,n} & \rho_{0,n} \\ \hat{\rho}_{n,n} & 0 \end{array}\right] = \left[\begin{array}{c|c} \hat{\pi}_{0,n} & \varepsilon_{n,n} \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ \hat{\varepsilon}_{n,n} & \pi_{0,n} \end{array}\right]. \tag{2.6}$$

Our aim is to construct a solution of (2.4) for n incremented by one:

$$\mathbf{T}_{n+2}\left[\begin{array}{c|c} \hat{\rho}_{0,n+1} & \rho_{n+1,n+1} \\ \vdots & \rho_{n,n+1} \\ \hat{\rho}_{n,n+1} & \vdots \\ \hat{\rho}_{n+1,n+1} & \rho_{0,n+1} \end{array}\right] = \left[\begin{array}{c|c} 0 & \varepsilon_{n+1,n+1} \\ \vdots & 0 \\ 0 & \vdots \\ \hat{\varepsilon}_{n+1,n+1} & 0 \end{array}\right].$$

The comparison with (2.6) suggests to combine the two vectors on the left-hand side of (2.6) linearly in order to cancel either the first or the last component on the right-hand side. To eliminate the first component the multipliers are 1 and $-\hat{\pi}_{0,n}/\varepsilon_{n,n}$, for the last component they are $-\pi_{0,n}/\hat{\varepsilon}_{n,n}$ and 1. In the following, these *Schur parameters* or *reflection coefficients* [7]

$$\hat{\gamma}_0^{(n)} := -\frac{\hat{\pi}_{0,n}}{\varepsilon_{n,n}}, \quad \gamma_0^{(n)} := -\frac{\pi_{0,n}}{\hat{\varepsilon}_{n,n}}. \tag{2.7}$$

will play an important role. We conclude that the recurrences

$$\left[\begin{array}{c|c} \hat{\rho}_{0,n+1} & \rho_{n+1,n+1} \\ \vdots & \vdots \\ \vdots & \vdots \\ \hat{\rho}_{n+1,n+1} & \rho_{0,n+1} \end{array}\right] = \left[\begin{array}{c|c} 0 & \rho_{n,n} \\ \hat{\rho}_{0,n} & \vdots \\ \vdots & \rho_{0,n} \\ \hat{\rho}_{n,n} & 0 \end{array}\right] \left[\begin{array}{cc} 1 & \gamma_0^{(n)} \\ \hat{\gamma}_0^{(n)} & 1 \end{array}\right] \tag{2.8}$$

and

$$\begin{aligned} \hat{\varepsilon}_{n+1,n+1} &= \hat{\varepsilon}_{n,n} + \hat{\gamma}_0^{(n)}\pi_{0,n}, \\ \varepsilon_{n+1,n+1} &= \varepsilon_{n,n} + \gamma_0^{(n)}\hat{\pi}_{0,n} \end{aligned} \tag{2.9}$$

hold true. They remain valid if $\hat{\pi}_{0,n} = 0$ or $\pi_{0,n} = 0$. Since $\hat{\varepsilon}_{0,0} = \varepsilon_{0,0} = \mu_0$, it follows by induction that $\hat{\varepsilon}_{n,n} = \varepsilon_{n,n}$ for all $n = 0, 1, \ldots$. Hence,

$$\hat{\varepsilon}_{n+1,n+1} = \varepsilon_{n+1,n+1} = \varepsilon_{n,n}\left(1 - \hat{\gamma}_0^{(n)}\gamma_0^{(n)}\right). \tag{2.10}$$

If $\varepsilon_{n+1,n+1} \neq 0$ we can proceed to the next step. Otherwise, $\mathbf{T}_{n+1}$ is singular. We summarize these well known results as

Algorithm 1. (Nonsymmetric Levinson algorithm without look-ahead)

1. Set $\hat{\rho}_{0,0} = \rho_{0,0} = 1$, $\hat{\varepsilon}_{0,0} = \varepsilon_{0,0} = \mu_0$.

For $n = 0, 1, 2, \ldots, N$:

2. compute $\hat{\pi}_{0,n}$ and $\pi_{0,n}$ from (2.5), as well as $\hat{\gamma}_0^{(n)}$ and $\gamma_0^{(n)}$ from (2.7);
3. compute $\hat{\rho}_{0,n+1}, \ldots, \hat{\rho}_{n+1,n+1}$ and $\rho_{0,n+1}, \ldots, \rho_{n+1,n+1}$ from (2.8);
4. compute $\hat{\varepsilon}_{n+1,n+1} = \varepsilon_{n+1,n+1}$ from (2.10);
   if $\varepsilon_{n+1,n+1} = 0$, stop: $\mathbf{T}_{n+1}$ is singular.

The total amount of work involved is $2N^2 + O(N)$ additions and multiplications each and $2N$ divisions. Storage (including the original data, which must be kept) is at least $4N + O(1)$, but normally one stores all the $\hat{\rho}$s and ρs, which requires $N^2 + O(N)$.

If we define

$$\hat{\mathbf{q}}_n := [\hat{\rho}_{0,n} \cdots \hat{\rho}_{n,n}]^{\mathrm{T}} \quad \text{and} \quad \mathbf{q}_n := [\rho_{n,n} \cdots \rho_{0,n}]^{\mathrm{T}}, \tag{2.11}$$

then, (2.4) is equivalent to

$$\mathbf{T}_{n+1}[\ \hat{\mathbf{q}}_n \mid \mathbf{q}_n\ ] = [\ \mathbf{e}_n\hat{\varepsilon}_{n,n} \mid \mathbf{e}_0\varepsilon_{n,n}\ ]. \tag{2.12}$$

Let $\mathbf{J}_n$ be the antidiagonal unit matrix or reflection matrix of order n. In view of the persymmetry of Toeplitz matrices we have

$$\mathbf{J}_n\mathbf{T}_n\mathbf{J}_n = \mathbf{T}_n^{\mathrm{T}}. \tag{2.13}$$

Hence, the second equation in (2.12) can be written as

$$(\mathbf{J}_{n+1}\mathbf{q}_n)^{\mathrm{T}}\mathbf{T}_{n+1} = \varepsilon_{n,n}\mathbf{e}_n^{\mathrm{T}}. \tag{2.14}$$

Compiling this and the first equation in (2.12) for $n = 1, \ldots, N$, we obtain for $\mathbf{T} := \mathbf{T}_{N+1}$ the matrix decompositions

$$\mathbf{T}\hat{\mathbf{R}} = \hat{\mathbf{E}} \quad \text{and} \quad \mathbf{R}^{\mathrm{T}}\mathbf{T} = \mathbf{E}^{\mathrm{T}}, \tag{2.15}$$

where

$$\begin{aligned} \hat{\mathbf{R}} &:= \begin{bmatrix} \hat{\rho}_{0,0} & \cdots & \hat{\rho}_{0,N} \\ & \ddots & \vdots \\ & & \hat{\rho}_{N,N} \end{bmatrix}, \quad \hat{\mathbf{E}} := \begin{bmatrix} \hat{\varepsilon}_{0,0} & & \\ \vdots & \ddots & \\ \hat{\varepsilon}_{N,0} & \cdots & \hat{\varepsilon}_{N,N} \end{bmatrix}, \\ \mathbf{R} &:= \begin{bmatrix} \rho_{0,0} & \cdots & \rho_{0,N} \\ & \ddots & \vdots \\ & & \rho_{N,N} \end{bmatrix}, \quad \mathbf{E} := \begin{bmatrix} \varepsilon_{0,0} & & \\ \vdots & \ddots & \\ \varepsilon_{N,0} & \cdots & \varepsilon_{N,N} \end{bmatrix}. \end{aligned} \tag{2.16}$$

Hence, $\hat{\mathbf{R}}$ and $\mathbf{R}$ are unit upper triangular, while $\hat{\mathbf{E}}$ and $\mathbf{E}$ are lower triangular.

By (2.15),

$$\mathbf{D} := \mathbf{R}^{\mathrm{T}}\mathbf{T}\hat{\mathbf{R}} = \mathbf{R}^{\mathrm{T}}\hat{\mathbf{E}} = \mathbf{E}^{\mathrm{T}}\hat{\mathbf{R}}. \tag{2.17}$$

Since $\mathbf{R}^{\mathrm{T}}\hat{\mathbf{E}}$ is lower triangular and $\mathbf{E}^{\mathrm{T}}\hat{\mathbf{R}}$ is upper triangular, $\mathbf{D}$ is a diagonal matrix. Moreover, $\hat{\mathbf{R}}$ and $\mathbf{R}$ have unit diagonal, and therefore,

$$\mathbf{D} = \mathrm{diag}\,(\hat{\mathbf{E}}) = \mathrm{diag}\,(\mathbf{E}) \tag{2.18}$$

and

$$\hat{\mathbf{E}} = \mathbf{R}^{-\mathrm{T}}\mathbf{D} \quad \text{and} \quad \mathbf{E}^{\mathrm{T}} = \mathbf{D}\hat{\mathbf{R}}^{-1}. \tag{2.19}$$

Hence,

$$\mathbf{T} = \hat{\mathbf{E}}\mathbf{D}^{-1}\mathbf{E}^{\mathrm{T}} \quad \text{and} \quad \mathbf{T}^{-1} = \hat{\mathbf{R}}\mathbf{D}^{-1}\mathbf{R}^{\mathrm{T}} \tag{2.20}$$

are an LDU factorization of $\mathbf{T}$ and a UDL factorization of $\mathbf{T}^{-1}$, respectively. The decomposition (2.17) is called inverse LDU decomposition of $\mathbf{T}$. Note that $\hat{\varepsilon}_{n,n} = \varepsilon_{n,n}$ is the $(n+1)$st pivot element. It can also be understood as the Schur complement $\mathbf{T}_{n+1}/\mathbf{T}_n$ of $\mathbf{T}_{n+1}$ with respect to $\mathbf{T}_n$, as one can verify easily from (2.2), (2.4), and $\hat{\rho}_{n,n} = \rho_{n,n} = 1$.

3. The nonsymmetric Schur algorithm

To derive the Schur algorithm, we do not only need the triangular matrices defined in (2.16), but also the matrices $\mathbf{P}$ and $\hat{\mathbf{P}}$ defined as follows:

$$\begin{bmatrix} \mathbf{T}_\wedge \\ \mathbf{T} \end{bmatrix} \hat{\mathbf{R}} = \begin{bmatrix} \hat{\mathbf{P}} \\ \hat{\mathbf{E}} \end{bmatrix}, \tag{3.1}$$

$$\mathbf{R}^{\mathrm{T}} [\mathbf{T}_< \ \mathbf{T}] = \left[\mathbf{P}^{\mathrm{T}} \ \mathbf{E}^{\mathrm{T}}\right]. \tag{3.2}$$

Here, $\mathbf{T}_\wedge$ and $\mathbf{T}_<$ are semi-infinite Toeplitz matrices such that $\begin{bmatrix} \mathbf{T}_\wedge \\ \mathbf{T} \end{bmatrix}$ and $[\mathbf{T}_< \ \mathbf{T}]$ again have Toeplitz structure.

Since will only need a finite lower triangular part of the semi-infinite matrices $\hat{\mathbf{P}}$ and $\mathbf{P}$, we define the $(2N - n + 2) \times n$ matrix

$$\tilde{\mathbf{T}}_n := \left[\begin{array}{ccc} \mu_{-N+n-1} & \cdots & \mu_{-N} \\ \vdots & & \vdots \\ \hline \mu_0 & \cdots & \mu_{-n+1} \\ \vdots & & \vdots \\ \mu_{n-1} & \cdots & \mu_0 \\ \hline \vdots & & \vdots \\ \mu_N & \cdots & \mu_{N-n+1} \end{array}\right] \begin{matrix} \left.\vphantom{\begin{array}{c} a \\ a \\ a \end{array}}\right\} N-n+1 \\ \left.\vphantom{\begin{array}{c} a \\ a \\ a \end{array}}\right\} n \\ \left.\vphantom{\begin{array}{c} a \\ a \\ a \end{array}}\right\} N-n+1 \end{matrix} \tag{3.3}$$

Note, that the row dimension of $\tilde{\mathbf{T}}_n$ decreases as n increases. In particular, $\tilde{\mathbf{T}}_{N+1} = \mathbf{T}$. The matrix between the horizontal lines is just $\mathbf{T}_n$. With (3.3), we can extend (2.4) to

$$\tilde{\mathbf{T}}_{n+1} \left[\begin{array}{c|c} \hat{\rho}_{0,n} & \rho_{n,n} \\ \vdots & \rho_{n-1,n} \\ \hat{\rho}_{n-1,n} & \vdots \\ \hat{\rho}_{n,n} & \rho_{0,n} \end{array}\right] = \left[\begin{array}{c|c} \hat{\pi}_{N-n-1,n} & \varepsilon_{N,n} \\ \vdots & \vdots \\ \hat{\pi}_{0,n} & \varepsilon_{n+1,n} \\ \hline 0 & \varepsilon_{n,n} \\ \mathbf{0}_{n-1} & \mathbf{0}_{n-1} \\ \hat{\varepsilon}_{n,n} & 0 \\ \hline \hat{\varepsilon}_{n+1,n} & \pi_{0,n} \\ \vdots & \vdots \\ \hat{\varepsilon}_{N,n} & \pi_{N-n-1,n} \end{array}\right] \begin{matrix} \left.\vphantom{\begin{array}{c} a \\ a \\ a \end{array}}\right\} N-n \\ \left.\vphantom{\begin{array}{c} a \\ a \\ a \end{array}}\right\} n+1 \\ \left.\vphantom{\begin{array}{c} a \\ a \\ a \end{array}}\right\} N-n \end{matrix} \tag{3.4}$$

and (2.6) to

$$\tilde{\mathbf{T}}_{n+2} \left[\begin{array}{c|c} 0 & \rho_{n,n} \\ \hat{\rho}_{0,n} & \rho_{n-1,n} \\ \vdots & \vdots \\ \hat{\rho}_{n-1,n} & \rho_{0,n} \\ \hat{\rho}_{n,n} & 0 \end{array}\right] = \left[\begin{array}{c|c} \hat{\pi}_{N-n-1,n} & \varepsilon_{N-1,n} \\ \vdots & \vdots \\ \hat{\pi}_{0,n} & \varepsilon_{n,n} \\ \mathbf{0}_n & \mathbf{0}_n \\ \hat{\varepsilon}_{n,n} & \pi_{0,n} \\ \hline \vdots & \vdots \\ \hat{\varepsilon}_{N-1,n} & \pi_{N-n-1,n} \end{array}\right] \begin{matrix} \left.\vphantom{\begin{array}{c} a \\ a \\ a \end{array}}\right\} N-n-1 \\ \left.\vphantom{\begin{array}{c} a \\ a \\ a \end{array}}\right\} n+2 \\ \left.\vphantom{\begin{array}{c} a \\ a \\ a \end{array}}\right\} N-n-1 \end{matrix} \tag{3.5}$$

Note that the rows between the horizontal lines in (3.4) and (3.5) correspond to the rows of the Toeplitz matrices $\mathbf{T}_{n+1}$ and $\mathbf{T}_{n+2}$, respectively.

If we postmultiply (3.5) by the 2×2 matrix from (2.8), we get the following recurrences for the elements in the matrices $\hat{\mathbf{P}}$ and $\mathbf{E}$

$$(3.6)\quad \left[\begin{array}{c|c} 0 & \varepsilon_{n+1,n+1} \\ \hat{\pi}_{0,n+1} & \varepsilon_{n+2,n+1} \\ \vdots & \vdots \\ \hat{\pi}_{N-n-2,n+1} & \varepsilon_{N,n+1} \end{array}\right] = \left[\begin{array}{c|c} \hat{\pi}_{0,n} & \varepsilon_{n,n} \\ \hat{\pi}_{1,n} & \varepsilon_{n+1,n} \\ \vdots & \vdots \\ \hat{\pi}_{N-n-1,n} & \varepsilon_{N-1,n} \end{array}\right] \begin{bmatrix} 1 & \gamma_0^{(n)} \\ \hat{\gamma}_0^{(n)} & 1 \end{bmatrix},$$

while for $\hat{\mathbf{E}}$ and $\mathbf{P}$ we get

$$(3.7)\quad \left[\begin{array}{c|c} \hat{\varepsilon}_{n+1,n+1} & 0 \\ \hat{\varepsilon}_{n+2,n+1} & \pi_{0,n+1} \\ \vdots & \vdots \\ \hat{\varepsilon}_{N,n+1} & \pi_{N-n-2,n+1} \end{array}\right] = \left[\begin{array}{c|c} \hat{\varepsilon}_{n,n} & \pi_{0,n} \\ \hat{\varepsilon}_{n+1,n} & \pi_{1,n} \\ \vdots & \vdots \\ \hat{\varepsilon}_{N-1,n} & \pi_{N-n-1,n} \end{array}\right] \begin{bmatrix} 1 & \gamma_0^{(n)} \\ \hat{\gamma}_0^{(n)} & 1 \end{bmatrix}.$$

This concludes the derivation of the nonsymmetric Schur algorithm for computing the LDU decomposition of $\mathbf{T}$:

Algorithm 2. (Nonsymmetric Schur algorithm without look-ahead)

1. For $j = 0, 1, \ldots, N$: set $\hat{\varepsilon}_{j,0} = \mu_j$, and $\varepsilon_{j,0} = \mu_{-j}$;
   for $j = 0, 1, \ldots, N-1$: set $\hat{\pi}_{j,0} = \mu_{-j-1}$, $\pi_{j,0} = \mu_{j+1}$.

For $n = 0, 1, 2, \ldots, N$:

2. compute $\hat{\gamma}_0^{(n)}$ and $\gamma_0^{(n)}$ from (2.7);
   compute the $(n+1)$th column of $\mathbf{E}$ and of the lower triangular part of $\hat{\mathbf{P}}$ from (3.6) and the $(n+1)$th column of $\hat{\mathbf{E}}$ and of the lower triangular part of $\mathbf{P}$ from (3.7);
   if $\varepsilon_{n+1,n+1} = 0$, stop: $\mathbf{T}_{n+1}$ is singular.

The total amount of work involved, the minimum storage (one column of $\hat{\mathbf{E}}$, $\hat{\mathbf{P}}$, $\mathbf{E}$, $\mathbf{P}$ each), as well as the storage for the full decomposition are the same as for the Levinson algorithm.

4. Column-regular pairs

As mentioned before, the classical algorithms break down whenever the coefficients $\hat{\varepsilon}_{n,n} = \varepsilon_{n,n}$ vanish for some n. In this case, LDU decompositions (without pivoting) of $\mathbf{T}$ and $\mathbf{T}^{-1}$ no longer exist. However, block LDU decompositions still exist, and it is natural to try to find fast algorithms in this way [9, 23, 14]. Then, in (2.20), $\mathbf{D}$ is block diagonal and $\hat{\mathbf{E}}$ and $\mathbf{E}$ are block lower triangular, but we can still choose $\hat{\mathbf{R}}$ and $\mathbf{R}$ as upper triangular. While standard block Gauss elimination with unit submatrices as diagonal blocks of $\hat{\mathbf{R}}$ and $\mathbf{R}$ would destroy the Toeplitz structure, suitable choice of these diagonal blocks preserves the structure. When we refer to blocks of $\hat{\mathbf{R}}$ and $\mathbf{R}$ we always mean the (symmetric) block structure imposed by $\mathbf{D}$ and the product $\hat{\mathbf{R}}\mathbf{D}^{-1}\mathbf{R}^{\mathrm{T}}$ in (2.20).

We will see that because of the underlying Toeplitz structure we can choose to compute only the first columns of all blocks in $\hat{\mathbf{R}}$ and $\mathbf{R}$, or in $\hat{\mathbf{E}}$, $\mathbf{E}$, $\hat{\mathbf{P}}$, and $\mathbf{P}$. Such

columns of $\hat{\mathbf{R}}$ and $\mathbf{R}$ will be called a column-regular pair. At the end, the Gohberg-Semencul [16] formula allows us to compute the inverse of $\mathbf{T} = \mathbf{T}_{N+1}$ from the last columns of the matrices $\hat{\mathbf{R}}$ and $\mathbf{R}$, provided that these columns are a column-regular pair. We will see that this means that $\mathbf{T}_N$ is also nonsingular. However, if $\mathbf{T}_N$ is singular or ill conditioned, we can extend $\mathbf{T}_{N+1}$ to $\mathbf{T}_{N+2}$ such that $\mathbf{T}_{N+2}$ is well conditioned and then apply another Gohberg-Semencul formula (see [16] or (4.3) below). Therefore, the assumption that $\mathbf{T}_N$ be nonsingular or well conditioned is not really restrictive. But we will also show how the other columns of the matrices $\hat{\mathbf{R}}$, $\mathbf{R}$, $\hat{\mathbf{E}}$, $\mathbf{E}$, $\hat{\mathbf{P}}$, and $\mathbf{P}$ can be found. This then makes the full block LDU decompositions of $\mathbf{T}$ and $\mathbf{T}^{-1}$ available. However, the fact that, also for $n < N$, column-regularity requires that $\mathbf{T}_n$ and $\mathbf{T}_{n+1}$ be nonsingular is a disadvantage of this look-ahead algorithm, since the resulting inverse block LDU decomposition may have larger blocks than necessary. We will see that each block of $\mathbf{D}$ is itself a 2×2 block diagonal matrix with a block of order 1 in the upper left corner. But the second block may still have some isolated nonsingular (even well-conditioned) principal submatrices, in which case the competing algorithms of Chan and Hansen [9, 23, 10], Freund and Zha [14], and Gutknecht [19] perform several shorter look-ahead steps instead of a single long one.

We start with the definition of column-regular pairs, which are closely related to column-regular Padé approximants, as defined in [19].

Definition. The nonnegative index n is *column-regular* if the two Yule-Walker equations (2.2) have a unique pair of solutions, and these solutions satisfy $\hat{\varepsilon}_{n,n} \neq 0$ and $\varepsilon_{n,n} \neq 0$, respectively. If the index n is column-regular, the pair $\hat{\mathbf{q}}_n$, $\mathbf{q}_n \in \mathbb{C}^{n+1}$ defined in (2.11) with $\hat{\rho}_{n,n} := 1$, $\rho_{n,n} := 1$ is called a *column-regular pair*.

A number of equivalent conditions for column regularity are given in the following lemma. Probably, each of these equivalences has appeared somewhere in the Toeplitz matrix literature. In particular, most of them can be found or easily deduced from the treatment in Chapter 1 of Heinig and Rost [24]. Note the subtle difference between conditions (i) and (v). While one direction of this equivalence is trivial, the other one is the hardest part of the proof of the lemma. The lemma can also be reformulated in terms of quantities from Padé approximation; in that context the equivalences follow easily from the well-known block structure theorem [17], see [22] and [19].

Lemma 4.1. *The following statements are equivalent:*

(i) *the two Yule-Walker equations* (2.2) *have a unique pair of solutions, and these solutions satisfy* $\hat{\varepsilon}_{n,n} \neq 0$ *and* $\varepsilon_{n,n} \neq 0$, *respectively; i.e.,* n *is column-regular;*

(ii) $\mathbf{T}_n$ *and* $\mathbf{T}_{n+1}$ *are nonsingular;*

(iii) *every pair of nontrivial solutions of* (2.3)–(2.4) *satisfies* $\hat{\rho}_{n,n} \neq 0$ *and* $\varepsilon_{n,n} \neq 0$*;*

(iv) *every pair of nontrivial solutions of* (2.3)–(2.4) *satisfies* $\rho_{n,n} \neq 0$ *and* $\hat{\varepsilon}_{n,n} \neq 0$*;*

(v) *the two Yule-Walker equations* (2.2) *have a pair of solutions which satisfy* $\hat{\varepsilon}_{n,n} \neq 0$ *and* $\varepsilon_{n,n} \neq 0$, *respectively;*

(vi) *the systems* (2.4) *are solvable for* $\hat{\varepsilon}_{n,n} = 1$, $\varepsilon_{n,n} = 1$, *and the solutions satisfy* $\hat{\rho}_{n,n} \neq 0$ *and* $\rho_{n,n} \neq 0$, *respectively;*

(vii) *the polynomials*

$$\hat{q}_n(\zeta) := \sum_{j=0}^{n} \hat{\rho}_{j,n}\zeta^j \quad \textit{and} \quad q_n(\zeta) := \sum_{j=0}^{n} \rho_{n-j,n}\zeta^j \tag{4.1}$$

are relatively prime, and $\mu_0 \neq 0$ *if* $n = 0$ *or* $\rho_{0,n} = \ldots = \rho_{n-1,n} = 0$*;*

(viii) *the Gohberg-Semencul formulas are valid:*

$$\mathbf{T}_{n+1}^{-1} = \frac{1}{\varepsilon_{n,n}} \left\{ \begin{bmatrix} \rho_{n,n} & & & \mathbf{O} \\ \rho_{n-1,n} & \rho_{n,n} & & \\ \vdots & \ddots & \ddots & \\ \rho_{0,n} & \cdots & \rho_{n-1,n} & \rho_{n,n} \end{bmatrix} \begin{bmatrix} \hat{\rho}_{n,n} & \hat{\rho}_{n-1,n} & \cdots & \hat{\rho}_{0,n} \\ & \ddots & \ddots & \vdots \\ & & \hat{\rho}_{n,n} & \hat{\rho}_{n-1,n} \\ \mathbf{O} & & & \hat{\rho}_{n,n} \end{bmatrix} \right.$$

$$\left. - \begin{bmatrix} 0 & & & \mathbf{O} \\ \hat{\rho}_{0,n} & 0 & & \\ \vdots & \ddots & \ddots & \\ \hat{\rho}_{n-1,n} & \cdots & \hat{\rho}_{0,n} & 0 \end{bmatrix} \begin{bmatrix} 0 & \rho_{0,n} & \cdots & \rho_{n-1,n} \\ & \ddots & \ddots & \vdots \\ & & 0 & \rho_{0,n} \\ \mathbf{O} & & & 0 \end{bmatrix} \right\}, \tag{4.2}$$

$$\mathbf{T}_n^{-1} = \frac{1}{\varepsilon_{n,n}} \left\{ \begin{bmatrix} \rho_{n,n} & & \mathbf{O} \\ \vdots & \ddots & \\ \rho_{1,n} & \cdots & \rho_{n,n} \end{bmatrix} \begin{bmatrix} \hat{\rho}_{n,n} & \cdots & \hat{\rho}_{1,n} \\ & \ddots & \vdots \\ \mathbf{O} & & \hat{\rho}_{n,n} \end{bmatrix} \right.$$

$$\left. - \begin{bmatrix} \hat{\rho}_{0,n} & & \mathbf{O} \\ \vdots & \ddots & \\ \hat{\rho}_{n-1,n} & \cdots & \hat{\rho}_{0,n} \end{bmatrix} \begin{bmatrix} \rho_{0,n} & \cdots & \rho_{n-1,n} \\ & \ddots & \vdots \\ \mathbf{O} & & \rho_{0,n} \end{bmatrix} \right\}. \tag{4.3}$$

(*When* $n = 0$*, the matrix* $\mathbf{T}_n$ *and the systems* (2.2) *are empty and should be considered as nonsingular.*)

Proof. (i)⇔(ii): to prove the equivalence of (i) and (ii), note that the uniqueness in (i) is equivalent to $\mathbf{T}_n$ being nonsingular. If $\varepsilon_{n,n} = 0$ or $\hat{\varepsilon}_{n,n} = 0$, the corresponding system in (2.4) is homogeneous and has a nontrivial solution (since $\hat{\rho}_{n,n} = \rho_{n,n} = 1$); hence, $\mathbf{T}_{n+1}$ is singular. Conversely, if $\mathbf{T}_{n+1}$ is singular, its last row is a linear combination of the first n when $\mathbf{T}_n$ is nonsingular; hence, the solution $\hat{\mathbf{q}}_n$ from (2.2), which solves the first n (homogeneous) equations in (2.4), solves the last equation only when $\hat{\varepsilon}_{n,n} = 0$. Any other solution of the first system in (2.4) is obtained by scaling. For $\mathbf{q}_n$ and $\varepsilon_{n,n}$ an analogous argument applies.

(ii)⇔(iii), (ii)⇔(iv): clearly, the nonsingularity of $\mathbf{T}_{n+1}$ is necessary for (iii) and (iv): if $\mathbf{T}_{n+1}$ is singular, the homogeneous system $\mathbf{T}_{n+1}\mathbf{q}_n = \mathbf{0}$ obtained from (2.4) when $\varepsilon_{n,n} = 0$ has a nontrivial solution. Moreover, when $\mathbf{T}_{n+1}$ is nonsingular, then by Cramer's rule $\hat{\rho}_{n,n} = \hat{\varepsilon}_{n,n}\det\mathbf{T}_n/\det\mathbf{T}_{n+1}$ and $\rho_{n,n} = \varepsilon_{n,n}\det\mathbf{T}_n/\det\mathbf{T}_{n+1}$; hence, $\det\mathbf{T}_n \neq 0$ is also necessary for (iii) and (iv). Conversely, if $\mathbf{T}_n$ and $\mathbf{T}_{n+1}$ are nonsingular, then $\hat{\varepsilon}_{n,n} \neq 0$ and $\varepsilon_{n,n} \neq 0$ for every nontrivial pair of solutions; moreover, $\hat{\rho}_{n,n} \neq 0$ and $\rho_{n,n} \neq 0$ by Cramer's rule. Hence, (ii) is equivalent to (iii) and (iv). It follows that, if (iii) or (iv) hold, then both are true, and Cramer's rule yields

$$\varepsilon_{n,n}\hat{\rho}_{n,n} = \hat{\varepsilon}_{n,n}\rho_{n,n}. \tag{4.4}$$

(iii)⇔(iv): follows from the two equivalences just proved, but also directly from (4.4). Here is another simple proof of this identity: let $\hat{\mathbf{q}}_n$, $\mathbf{q}_n$ be any pair of solutions of (2.3)–(2.4). Then, using the first equations in (2.12), we obtain

$$(\mathbf{J}_{n+1}\mathbf{q}_n)^{\mathrm{T}}\mathbf{T}_{n+1}\hat{\mathbf{q}}_n = (\mathbf{J}_{n+1}\mathbf{q}_n)^{\mathrm{T}}\mathbf{e}_n\hat{\varepsilon}_{n,n} = \rho_{n,n}\hat{\varepsilon}_{n,n},$$

and, from (2.14),

$$(\mathbf{J}_{n+1}\mathbf{q}_n)^{\mathrm{T}}\mathbf{T}_{n+1}\hat{\mathbf{q}}_n = \varepsilon_{n,n}\mathbf{e}_n^{\mathrm{T}}\hat{\mathbf{q}}_n = \varepsilon_{n,n}\hat{\rho}_{n,n}.$$

Therefore, any pair of solutions of (2.3)–(2.4) satisfies (4.4).

(i)⇒(v): is trivial.

(v)⇔(vi): as long as we do not prescribe $\hat{\varepsilon}_{n,n}$ or $\varepsilon_{n,n}$, (2.3)–(2.4) are of course solvable. Therefore, (vi) follows from (v) by renormalizing the solutions. The converse is obvious.

(vi)⇒(ii): this part of the proof is taken from Theorems 1.2 and 1.3 in [24].

(ii)⇔(vii): the equivalence of (ii) and (vii) follows readily from the Lemmas 2.4 and 2.5 in [19]; see also [22].

(vi)⇔(viii): for a derivation of the Gohberg-Semencul formulas see, e.g., Heinig and Rost [24, Theorems 1.2 and 1.3]. Clearly, they require that $\mathbf{T}_{n+1}$ is nonsingular and that $\varepsilon_{n,n} \neq 0$ and $\hat{\rho}_{n,n} \neq 0$, which, as we have seen above, implies that $\mathbf{T}_n$ is nonsingular.

Note that (4.4) implies that

$$\varepsilon_{n,n} = \hat{\varepsilon}_{n,n} \tag{4.5}$$

for (normalized) column-regular pairs.

While column-regular pairs can be used to set up algorithms that are reliable in exact arithmetic, the finite accuracy of the floating-point arithmetic requires to avoid not only singular systems but any ill-conditioned intermediate results; therefore, one has to further restrict column-regular pairs in this situation.

Definition. The (normalized) column-regular pair $\hat{\mathbf{q}}_n$, $\mathbf{q}_n \in \mathbb{C}^{n+1}$ The (normalized) column-regular pair $\hat{\mathbf{q}}_n$, $\mathbf{q}_n \in \mathbb{C}^{n+1}$ is *well-column-regular* if $||\hat{\mathbf{q}}_n||$, $||\mathbf{q}_n|| < \mathrm{Tol}(n)$ and ($|\hat{\varepsilon}_{n,n}|$ =) $|\varepsilon_{n,n}| > \mathrm{tol}(n)$. The index n is then also called well-column-regular. Here, $\mathrm{tol}(n) > 0$ and $\mathrm{Tol}(n) > 1$ denote given tolerances that are monotone increasing functions of n.

Well-column-regularity means that $\mathbf{T}_n$ and $\mathbf{T}_{n+1}$ are well conditioned. As usual, we let $\sigma_{\min}(\mathbf{T}_n)$ denote the smallest singular value of $\mathbf{T}_n$. Then $\sigma_{\min}^{-1}(\mathbf{T}_n) = ||\mathbf{T}_n^{-1}||$, and the following quantitative connections hold:

Lemma 4.2. *If n is well-column-regular, then*

$$\max\left\{||\mathbf{T}_n^{-1}||,\ ||\mathbf{T}_{n+1}^{-1}||\right\} < 2n\frac{[\mathrm{Tol}(n)]^2}{\mathrm{tol}(n)}.$$

Conversely, if $||\mathbf{T}|| < \tau$, $||\mathbf{T}_n^{-1}|| < \tau'$, and $||\mathbf{T}_{n+1}^{-1}|| < \tau''$, then

$$\max\{||\hat{\mathbf{q}}_n||,\ ||\mathbf{q}_n||\} < \sqrt{1 + (\tau\tau')^2}$$

and

$$\begin{aligned} |\hat{\varepsilon}_{n,n}| &= |\varepsilon_{n,n}| \geq \sigma_{\min}(\mathbf{T}_{n+1})\ \max\{||\hat{\mathbf{q}}_n||,\ ||\mathbf{q}_n||\} \\ &> \frac{1}{\tau''}\ \max\{||\hat{\mathbf{q}}_n||,\ ||\mathbf{q}_n||\} \geq \frac{1}{\tau''}. \end{aligned} \tag{4.6}$$

Proof. From the Gohberg-Semencul formula (4.2), we conclude

$$\|\mathbf{T}_{n+1}^{-1}\| \le \|\mathbf{T}_{n+1}^{-1}\|_F \le \frac{2n}{\varepsilon_{n,n}} \|\hat{\mathbf{q}}_n\| \cdot \|\mathbf{q}_n\| \le \frac{2n[\mathrm{Tol}(n)]^2}{\mathrm{tol}(n)}.$$

By using (4.3) instead of (4.2), the same technique can be applied to $\|\mathbf{T}_n^{-1}\|$.

For the second part of the Lemma we can use (2.2):

$$\|[\rho_{n-1,n} \ \cdots \ \rho_{0,n}]\| \le \|\mathbf{T}_n^{-1}\| \cdot \|[\mu_1 \ \cdots \ \mu_n]\| \le \|\mathbf{T}_n^{-1}\| \cdot \|\mathbf{T}\| < \tau\tau'.$$

Hence, with $\rho_{n,n} = 1$, we obtain $\|\mathbf{q}_n\| \le \sqrt{1 + (\tau\tau')^2}$. The same argument holds for $\hat{\mathbf{q}}_n$.

From (2.12), we have $\mathbf{q}_n = \mathbf{T}_{n+1}^{-1}\mathbf{e}_0\varepsilon_{n,n}$. Taking norms on both sides of this equation, and using $\rho_{n,n} = 1$ again, we conclude that

$$1 \le \|\mathbf{q}_n\| \le |\varepsilon_{n,n}| \, \|\mathbf{T}_{n+1}^{-1}\|. \tag{4.7}$$

The analogue argument for $\hat{\mathbf{q}}_n$ completes the proof.

From Lemma 4.1 it follows that in the limit $\mathrm{tol}(n) \to 0$, $\mathrm{Tol}(n) \to \infty$ "well-column-regular" becomes "column-regular". Of course, column-regular pairs or even well-column-regular pairs need not exist for every $n \ge 0$, but there always exists a maximal subset

$$\{0 \le n_1 < n_2 < \ldots < n_J \le N+1\} \subseteq \{0, 1, \ldots, N+1\} \tag{4.8}$$

such that $\hat{\mathbf{q}}_{n_j}$ and $\mathbf{q}_{n_j}$ are a well-column-regular pair for $j = 1, 2, \ldots, J$. In the following we assume that $N-1$ or N is a well-column-regular index, so that, in any case, $\mathbf{T}_N$ is well conditioned and $\mathbf{T}_N^{-1}\mathbf{c}$ can be computed stably with one of the two Gohberg-Semencul formulas.

5. Look-ahead Levinson and Schur algorithms: derivation

Our first new look-ahead algorithm, which is an extension of the Levinson algorithm, is based on the following idea. We assume that $\hat{\mathbf{q}}_n$ and $\mathbf{q}_n$ are a well-column-regular pair, and we try to compute a well-column-regular pair $\hat{\mathbf{q}}_{n+k}$, $\mathbf{q}_{n+k}$ of order $n+k$ such that these new vectors are linear combinations of shifted copies of $\hat{\mathbf{q}}_n$ and $\mathbf{q}_n$. Specifically, we set

$$[\ \hat{\mathbf{q}}_{n+k} \mid \mathbf{q}_{n+k}\ ] := \left[\begin{array}{ccc|ccc} 0 & \cdots & 0 & \rho_{n,n} & & 0 \\ \hat{\rho}_{0,n} & \ddots & \vdots & \vdots & \ddots & \\ \vdots & \ddots & 0 & \vdots & & \rho_{n,n} \\ \vdots & & \hat{\rho}_{0,n} & \rho_{0,n} & & \vdots \\ \hat{\rho}_{n,n} & & \vdots & 0 & \ddots & \vdots \\ & \ddots & \vdots & \vdots & \ddots & \rho_{0,n} \\ 0 & & \hat{\rho}_{n,n} & 0 & \cdots & 0 \end{array}\right] \left[\begin{array}{c|c} \hat{\alpha}_0^{(n)} & \beta_0^{(n)} \\ \vdots & \vdots \\ \hat{\alpha}_{k-1}^{(n)} & \beta_{k-1}^{(n)} \\ \hline \hat{\beta}_{k-1}^{(n)} & \alpha_{k-1}^{(n)} \\ \vdots & \vdots \\ \hat{\beta}_0^{(n)} & \alpha_0^{(n)} \end{array}\right]. \tag{5.1}$$

Note that this means that even in the look-ahead case we need to store only the two vectors $\hat{\mathbf{q}}_{n+k}$ and $\mathbf{q}_{n+k}$. Due to the normalization $\hat{\rho}_{n+k,n+k} = \rho_{n+k,n+k} = \hat{\rho}_{n,n} = \rho_{n,n} = 1$ we have

$$\hat{\alpha}_{k-1}^{(n)} = \alpha_{k-1}^{(n)} = 1. \tag{5.2}$$

We have to show how the coefficients $\hat{\alpha}_j^{(n)}$, $\alpha_j^{(n)}$ $(j = 0, \ldots, k-2)$ and $\hat{\beta}_j^{(n)}$, $\beta_j^{(n)}$ $(j = 0, \ldots, k-1)$ can be chosen such that $\hat{\mathbf{q}}_{n+k}$ and $\mathbf{q}_{n+k}$ satisfy

$$\mathbf{T}_{n+k+1}[\ \hat{\mathbf{q}}_{n+k} \mid \mathbf{q}_{n+k}\ ] = [\ \mathbf{e}_{n+k}\hat{\varepsilon}_{n+k,n+k} \mid \mathbf{e}_0\varepsilon_{n+k,n+k}\ ], \tag{5.3}$$

$$||\hat{\mathbf{q}}_{n+k}||,\ ||\mathbf{q}_{n+k}|| < \mathrm{Tol}(n+k), \tag{5.4}$$

$$(|\hat{\varepsilon}_{n+k,n+k}| =)\ |\varepsilon_{n+k,n+k}| > \mathrm{tol}(n+k). \tag{5.5}$$

The next step consists in inserting (5.1) into (5.3). In order to rewrite the resulting equation in compact form, let us define extended versions of $\hat{\mathbf{q}}_n$ and $\mathbf{q}_n$ from (2.11):

$$\hat{\mathbf{q}}_{n,e} := [\hat{\rho}_{0,n} \cdots \hat{\rho}_{n,n}\, 0 \cdots 0]^{\mathrm{T}}, \quad \mathbf{q}_{n,e} := [\rho_{n,n} \cdots \rho_{0,n}\, 0 \cdots 0]^{\mathrm{T}} \in \mathbb{C}^{n+k+1}. \tag{5.6}$$

Moreover, the shift matrix (of adaptable size) is denoted by

$$\mathcal{S} := \begin{bmatrix} 0 & & & & \\ 1 & 0 & & & \\ & 1 & 0 & & \\ & & \ddots & \ddots & \\ & & & 1 & 0 \end{bmatrix}.$$

The $(n+k+1) \times 2k$ matrix on the right-hand side of (5.1) can then be written as

$$\left[\ \mathcal{S}\hat{\mathbf{q}}_{n,e} \quad \mathcal{S}^2\hat{\mathbf{q}}_{n,e} \quad \cdots \quad \mathcal{S}^k\hat{\mathbf{q}}_{n,e} \mid \mathbf{q}_{n,e} \quad \mathcal{S}\mathbf{q}_{n,e} \quad \cdots \quad \mathcal{S}^{k-1}\mathbf{q}_{n,e}\ \right]. \tag{5.7}$$

If $k = n+1$, its determinant is called the resultant of the two polynomials $\zeta\hat{q}(\zeta)$ and $q(\zeta)$ defined in (4.1). It is well known that the resultant is nonzero if and only if the two polynomials are relatively prime; see, e. g., [34, pp. 103–108]. Therefore, statement (vii) of Lemma 4.1 yields the following result:

Lemma 5.1. *For $k = n+1$, the matrix* (5.7) *is nonsingular if and only if the pair $\hat{\mathbf{q}}_n$, $\mathbf{q}_n$ is column-regular or $n = 0$ or $\rho_{0,n} = \ldots = \rho_{n-1,n} = 0$. In particular, if the index n is column-regular and $k \le n+1$, then the matrix* (5.7) *has full column-rank.*

Alternatively, this result can be proved by using results about the kernel structure of Toeplitz matrices; for details see Chapter 5 of Heinig and Rost [24]. A simple proof is obtained from Theorem 5.1 there. The result also follows immediately from Theorem 5.5 and Corollary 2.5 in [24].

Equation (3.5) implies that

$$\tilde{\mathbf{T}}_{n+k+1}\left[\begin{array}{cccc|ccc} \mathcal{S}\hat{\mathbf{q}}_{n,e} & \mathcal{S}^2\hat{\mathbf{q}}_{n,e} & \cdots & \mathcal{S}^k\hat{\mathbf{q}}_{n,e} & \mathbf{q}_{n,e} & \mathcal{S}\mathbf{q}_{n,e} & \cdots & \mathcal{S}^{k-1}\mathbf{q}_{n,e} \end{array}\right]$$

$$= \left[\begin{array}{ccc|ccc} \hat{\pi}_{N-n-k,n} & \cdots & \hat{\pi}_{N-n-1,n} & \varepsilon_{N-k,n} & \cdots & \varepsilon_{N-1,n} \\ \vdots & & \vdots & \vdots & & \vdots \\ \hat{\pi}_{1,n} & \cdots & \hat{\pi}_{k,n} & \varepsilon_{n+1,n} & \cdots & \varepsilon_{n+k,n} \\ \hline \hat{\pi}_{0,n} & \cdots & \hat{\pi}_{k-1,n} & \varepsilon_{n,n} & \cdots & \varepsilon_{n+k-1,n} \\ & \ddots & \vdots & & \ddots & \vdots \\ \mathbf{0}_n & & \hat{\pi}_{0,n} & \mathbf{0}_n & & \varepsilon_{n,n} \\ & \ddots & & & \ddots & \\ \hat{\varepsilon}_{n,n} & & \mathbf{0}_n & \pi_{0,n} & & \mathbf{0}_n \\ \vdots & \ddots & & \vdots & \ddots & \\ \hat{\varepsilon}_{n+k-1,n} & \cdots & \hat{\varepsilon}_{n,n} & \pi_{k-1,n} & \cdots & \pi_{0,n} \\ \hline \hat{\varepsilon}_{n+k,n} & \cdots & \hat{\varepsilon}_{n+1,n} & \pi_{k,n} & \cdots & \pi_{1,n} \\ \vdots & & \vdots & \vdots & & \vdots \\ \hat{\varepsilon}_{N-1,n} & \cdots & \hat{\varepsilon}_{N-k,n} & \pi_{N-n-1,n} & \cdots & \pi_{N-n-k,n} \end{array}\right] \cdot \begin{array}{l} \left.\vphantom{\begin{array}{c}a\\a\\a\end{array}}\right\} N-n-k \\ \left.\vphantom{\begin{array}{c}a\\a\\a\\a\\a\\a\\a\end{array}}\right\} n+k+1 \\ \left.\vphantom{\begin{array}{c}a\\a\\a\end{array}}\right\} N-n-k \end{array}$$

(5.8)

We denote the $(n+k+1)\times 2k$ matrix consisting of the two middle Toeplitz blocks on the right-hand side by $\mathbf{S}^{\circ}_{k,n}$. If $k \le n+1$, there are $n-k+1$ zero rows in the middle of $\mathbf{S}^{\circ}_{k,n}$. (The symbol $^{\circ}$ is supposed to remind the reader of these excess zeros.) If $k > n+1$, the upper triangular and the lower triangular subblocks overlap. In any case, the entries $\hat{\pi}_{0,n}$ and $\hat{\varepsilon}_{n,n}$ on the left as well as $\varepsilon_{n,n}$ and $\pi_{0,n}$ on the right are separated by n zero diagonals. When we extract the $n+k+1$ equations that belong to $\mathbf{S}^{\circ}_{k,n}$, we have to replace $\tilde{\mathbf{T}}_{n+k+1}$ by $\mathbf{T}_{n+k+1}$ on the left-hand side:

$$\mathbf{T}_{n+k+1}\left[\begin{array}{cccc|ccc} \mathcal{S}\hat{\mathbf{q}}_{n,e} & \mathcal{S}^2\hat{\mathbf{q}}_{n,e} & \cdots & \mathcal{S}^k\hat{\mathbf{q}}_{n,e} & \mathbf{q}_{n,e} & \mathcal{S}\mathbf{q}_{n,e} & \cdots & \mathcal{S}^{k-1}\mathbf{q}_{n,e} \end{array}\right] = \mathbf{S}^{\circ}_{k,n}.$$

(5.9)

In addition to $\mathbf{S}^{\circ}_{k,n}$ we consider the $2k \times 2k$ matrix

$$\mathbf{S}^{\square}_{k,n} := \left[\begin{array}{ccc|ccc} \hat{\pi}_{0,n} & \cdots & \hat{\pi}_{k-1,n} & \varepsilon_{n,n} & \cdots & \varepsilon_{n+k-1,n} \\ & \ddots & \vdots & & \ddots & \vdots \\ \mathbf{O} & & \hat{\pi}_{0,n} & \mathbf{O} & & \varepsilon_{n,n} \\ \hline \hat{\varepsilon}_{n,n} & & \mathbf{O} & \pi_{0,n} & & \mathbf{O} \\ \vdots & \ddots & & \vdots & \ddots & \\ \hat{\varepsilon}_{n+k-1,n} & \cdots & \hat{\varepsilon}_{n,n} & \pi_{k-1,n} & \cdots & \pi_{0,n} \end{array}\right]. \tag{5.10}$$

(Here, the symbol $^{\square}$ indicates that the matrix is square.) If $k \le n+1$, it consists of the nonzero rows of $\mathbf{S}^{\circ}_{k,n}$. In general, the two matrices are related by

$$\mathbf{S}^{\circ}_{k,n} = \left[\begin{array}{cc} \mathbf{I}_k & \mathbf{O}_{(n+1)\times k} \\ \mathbf{O}_{(n+1)\times k} & \mathbf{I}_k \end{array}\right] \mathbf{S}^{\square}_{k,n}, \tag{5.11}$$

where $\mathbf{I}_k$ is the $k \times k$ unit matrix and $\mathbf{O}_{m\times k}$ is the $m \times k$ zero matrix. Since $\hat{\mathbf{q}}_n$ and $\mathbf{q}_n$ are well-column-regular, the two off-diagonal blocks of $\mathbf{S}^{\square}_{k,n}$ are nonsingular, and, in view of (5.9), their columns are bounded by

$$||\mathbf{T}_{n+k+1}||\ \max\{||\hat{\mathbf{q}}_n||,\ ||\mathbf{q}_n||\}\,. \tag{5.12}$$

Hence, their condition cannot be very bad. Later, we will use these off-diagonal blocks as coefficient matrices of linear systems for determining "inner" vectors $\hat{\mathbf{q}}_{n+k}$, $\mathbf{q}_{n+k}$ that do not form a column-regular pair.

Introducing the notation

$$\left[\begin{array}{c|c} \hat{\mathbf{a}} & \mathbf{b} \\ \hline \hat{\mathbf{b}} & \mathbf{a} \end{array}\right] := \left[\begin{array}{c|c} \hat{\alpha}_0^{(n)} & \beta_0^{(n)} \\ \vdots & \vdots \\ \hat{\alpha}_{k-1}^{(n)} & \beta_{k-1}^{(n)} \\ \hline \hat{\beta}_{k-1}^{(n)} & \alpha_{k-1}^{(n)} \\ \vdots & \vdots \\ \hat{\beta}_0^{(n)} & \alpha_0^{(n)} \end{array}\right], \tag{5.13}$$

multiplying (5.1) from the left by $\tilde{\mathbf{T}}_{n+k+1}$, and using (5.2), (5.8), we see that (5.3) holds if and only if, for $\hat{\alpha}_{k-1}^{(n)} = \alpha_{k-1}^{(n)} = 1$,

$$\mathbf{S}_{k,n}^{\circ}\left[\begin{array}{c|c} \hat{\mathbf{a}} & \mathbf{b} \\ \hline \hat{\mathbf{b}} & \mathbf{a} \end{array}\right] = [\ \mathbf{e}_{n+k}\hat{\varepsilon}_{n+k,n+k}\ |\ \mathbf{e}_0\varepsilon_{n+k,n+k}\ ] \in \mathbb{C}^{(n+k+1)\times 2}, \tag{5.14}$$

Additionally, (5.4) and (5.5) will be required to hold, but we will return to them later. In view of (5.11) and

$$\left[\begin{array}{c|c} 0 & \varepsilon_{n+k,n+k} \\ \mathbf{0}_{n+k-1} & \mathbf{0}_{n+k-1} \\ \hat{\varepsilon}_{n+k,n+k} & 0 \end{array}\right] = \left[\begin{array}{cc} \mathbf{I}_k & \mathbf{O}_{(n+1)\times k} \\ \mathbf{O}_{(n+1)\times k} & \mathbf{I}_k \end{array}\right] \left[\begin{array}{c|c} 0 & \varepsilon_{n+k,n+k} \\ \mathbf{0}_{2(k-1)} & \mathbf{0}_{2(k-1)} \\ \hat{\varepsilon}_{n+k,n+k} & 0 \end{array}\right], \tag{5.15}$$

every solution of

$$\mathbf{S}_{k,n}^{\square}\left[\begin{array}{c|c} \hat{\mathbf{a}} & \mathbf{b} \\ \hline \hat{\mathbf{b}} & \mathbf{a} \end{array}\right] = [\ \mathbf{e}_{2k-1}\hat{\varepsilon}_{n+k,n+k}\ |\ \mathbf{e}_0\varepsilon_{n+k,n+k}\ ] \in \mathbb{C}^{2k\times 2}, \tag{5.16}$$

is also a solution of (5.14). In fact, the systems (5.14) and (5.16) are equivalent if $k \le n+1$; but, in general, this is not true when $k > n+1$.

From (3.4), (5.8), and (5.16) we also obtain general Schur recurrences:

$$
\left[\begin{array}{c|c}
\hat\pi_{N-n-k-1,n+k} & \varepsilon_{N,n+k} \\
\vdots & \vdots \\
\hat\pi_{0,n+k} & \varepsilon_{n+k+1,n+k} \\
\hline
0 & \varepsilon_{n+k,n+k} \\
\hline\hline
\hat\varepsilon_{n+k,n+k} & 0 \\
\hline
\hat\varepsilon_{n+k+1,n+k} & \pi_{0,n+k} \\
\vdots & \vdots \\
\hat\varepsilon_{N,n+k} & \pi_{N-n-k-1,n+k}
\end{array}\right] =
$$

$$
\left[\begin{array}{ccc|ccc}
\hat\pi_{N-n-k,n} & \cdots & \hat\pi_{N-n-1,n} & \varepsilon_{N-k,n} & \cdots & \varepsilon_{N-1,n} \\
\vdots & & \vdots & \vdots & & \vdots \\
\hat\pi_{1,n} & \cdots & \hat\pi_{k,n} & \varepsilon_{n+1,n} & \cdots & \varepsilon_{n+k,n} \\
\hline
\hat\pi_{0,n} & \cdots & \hat\pi_{k-1,n} & \varepsilon_{n,n} & \cdots & \varepsilon_{n+k-1,n} \\
\hline\hline
\hat\varepsilon_{n+k-1,n} & \cdots & \hat\varepsilon_{n,n} & \pi_{k-1,n} & \cdots & \pi_{0,n} \\
\hline
\hat\varepsilon_{n+k,n} & \cdots & \hat\varepsilon_{n+1,n} & \pi_{k,n} & \cdots & \pi_{1,n} \\
\vdots & & \vdots & \vdots & & \vdots \\
\hat\varepsilon_{N-1,n} & \cdots & \hat\varepsilon_{N-k,n} & \pi_{N-n-1,n} & \cdots & \pi_{N-n-k,n}
\end{array}\right]
\left[\begin{array}{c|c}
\hat\alpha_0^{(n)} & \beta_0^{(n)} \\
\vdots & \vdots \\
\hat\alpha_{k-1}^{(n)} & \beta_{k-1}^{(n)} \\
\hline
\hat\beta_{k-1}^{(n)} & \alpha_{k-1}^{(n)} \\
\vdots & \vdots \\
\hat\beta_0^{(n)} & \alpha_0^{(n)}
\end{array}\right].
$$

(5.17)

Between the double lines we have deleted the $2(k-1)$ homogeneous equations from (5.16). These recurrences allow us to compute the $(n+k)$th column of $\hat{\mathbf{P}}$, $\hat{\mathbf{E}}$, $\mathbf{P}$, and $\mathbf{E}$ without evaluating inner products.

Relation (5.9) leads now easily to the following result:

Lemma 5.2. *Assume n is a column-regular index and $\hat{\mathbf{q}}_n$, $\mathbf{q}_n$ is the corresponding column-regular pair. Assume further that $k \le n+1$ and that $n+k$ is also a column-regular index. Then the matrices $\mathbf{S}^{\circ}_{k-1,n}$ and $\mathbf{S}^{\circ}_{k,n}$ have full column rank, and the matrices $\mathbf{S}^{\square}_{k-1,n}$ and $\mathbf{S}^{\square}_{k,n}$ are nonsingular.*

Proof. By Lemma 5.1, since n is a column-regular index, the matrix (5.7) has full column rank as long as $k \le n+1$. Moreover, since $n+k$ is also a column-regular index, the matrices $\mathbf{T}_{n+k}$ and $\mathbf{T}_{n+k+1}$ are nonsingular; see Lemma 4.1 (iii). Consequently, by (5.9), $\mathbf{S}^{\circ}_{k-1,n}$ and $\mathbf{S}^{\circ}_{k,n}$ have also full column rank, and this rank does not change when we delete zero rows.

When $k > n+1$, the matrix $\mathbf{S}^{\circ}_{k,n}$ cannot have full column-rank, since it has less rows than columns. However, the last statement of the lemma still holds.

Lemma 5.3. *Assume n is a column-regular index and $\hat{\mathbf{q}}_n$, $\mathbf{q}_n$ is the corresponding column-regular pair. Then the matrices $\mathbf{S}^{\square}_{k-1,n}$ and $\mathbf{S}^{\square}_{k,n}$ are nonsingular if $n+k$ is a column-regular index.*

Proof. Assume $n+k$ to be a column-regular index and $\mathbf{S}^{\square}_{k,n} \in \mathbb{C}^{2k,2k}$ to be singular. Then there exists a nontrivial solution $[\beta_0 \cdots \beta_{k-1}\, \alpha_{k-1} \cdots \alpha_0]^{\mathrm{T}}$ of (5.16) for $\varepsilon_{n+k,n+k} = 0$. If $\alpha_{k-1} \neq 0$ or $\beta_{k-1} \neq 0$, then (5.1) provides us with a nontrivial solution of $\mathbf{T}_{n+k+1}\mathbf{q} = \mathbf{0}$. But this implies that $\mathbf{T}_{n+k+1}$ is singular which is a contradiction to the column-regularity of $n+k$, cf. Lemma 4.1 (ii).

If $\alpha_{k-1} = \beta_{k-1} = 0$ for every nontrivial solution of (5.16), then the kth and the $(k+1)$st equation of (5.16) lead to $\alpha_0 = \beta_0 = 0$ (recall that $\varepsilon_{n,n} = \hat{\varepsilon}_{n,n} \neq 0$ since n is a column-regular index). Thus, in this case, the nontrivial solution is of the form $[0\,\beta_1 \cdots \beta_{k-2}\,0\,0\,\alpha_{k-2} \cdots \alpha_1\,0]^{\mathrm{T}}$. Now we proceed in the same way: if $\alpha_{k-2} \neq 0$ or $\beta_{k-2} \neq 0$, then (5.1) again provides a nontrivial solution of $\mathbf{T}_{n+k+1}\mathbf{q} = \mathbf{0}$. If $\alpha_{k-2} = \beta_{k-2} = 0$ for any nontrivial solution of (5.16), then the $(k-1)$st and the $(k+2)$nd equation of (5.16) lead to $\alpha_1 = \beta_1 = 0$, and so on.

Therefore, either $\mathbf{T}_{n+k+1}$ is singular, which contradicts the column regularity of $n+k$, or $\alpha_j = \beta_j = 0$ for $j = 0, \ldots, k-1$, which contradicts the assumption of $\mathbf{S}^{\square}_{k,n}$ being singular. Hence, column-regularity of $n+k$ implies that $\mathbf{S}^{\square}_{k,n}$ is nonsingular.

Obviously, the same proof applies if k is replaced by $k-1$, which shows that the singularity of $\mathbf{S}^{\square}_{k-1,n}$ implies the singularity of $\mathbf{T}_{n+k}$. In view of Lemma 4.1, this completes the proof.

Let us return to the two systems (5.16). Since we do not know $\varepsilon_{n+k,n+k}$ $(= \hat{\varepsilon}_{n+k,n+k})$ beforehand, we remove the last and the first equation, respectively, and determine normalized solutions of the homogeneous linear systems. Then we compute

$$\hat{\varepsilon}_{n+k,n+k} = \sum_{j=0}^{k-1} \left(\hat{\varepsilon}_{n+j,n} \hat{\alpha}^{(n)}_{k-1-j} + \pi_{j,n} \hat{\beta}^{(n)}_{j} \right) \tag{5.18a}$$

or

$$\varepsilon_{n+k,n+k} = \sum_{j=0}^{k-1} \left(\hat{\pi}_{j,n} \beta^{(n)}_{j} + \varepsilon_{n+j,n} \alpha^{(n)}_{k-1-j} \right) \tag{5.18b}$$

and check if (5.4) and (5.5) are satisfied. This is analogous to replacing (2.4) by the Yule-Walker equations (2.2). Taking a closer look at the linear systems (5.16), we easily verify that, for any k, the normalization (5.2) and the kth and the $(k+1)$th equation lead to

$$\hat{\beta}^{(n)}_0 = -\frac{\hat{\pi}_{0,n}}{\varepsilon_{n,n}} = \hat{\gamma}^{(n)}_0 \quad \text{and} \quad \beta^{(n)}_0 = -\frac{\pi_{0,n}}{\hat{\varepsilon}_{n,n}} = \gamma^{(n)}_0. \tag{5.19}$$

We conclude in particular that, if $\hat{\mathbf{q}}_n$, $\mathbf{q}_n$ are a well-column-regular pair and $k = 1$, the two systems (5.16) always have normalized solutions, namely those that were used in Sects. 2–3. When $k > 1$, by capitalizing upon the normalization (5.2) and (5.19), we can move the columns for $\hat{\alpha}^{(n)}_{k-1}$, $\hat{\beta}^{(n)}_0$, $\beta^{(n)}_0$, and $\alpha^{(n)}_{k-1}$ to the right-hand side of the systems (5.16). Thus, to compute $\hat{\alpha}^{(n)}_j$ and $\hat{\beta}^{(n)}_j$, we can eliminate rows and columns k and $2k$ of (5.16) and to compute $\beta^{(n)}_j$ and $\alpha^{(n)}_j$, we can eliminate rows and columns 1 and $k+1$. Due to the block Toeplitz structure of $\mathbf{S}^{\square}_{k,n}$, in both cases we end up with the same coefficient matrix, which is just $\mathbf{S}^{\square}_{k-1,n}$:

$$\mathbf{S}^{\square}_{k-1,n} \begin{bmatrix} \hat{\alpha}^{(n)}_0 \\ \vdots \\ \hat{\alpha}^{(n)}_{k-2} \\ \hat{\beta}^{(n)}_{k-1} \\ \vdots \\ \hat{\beta}^{(n)}_1 \end{bmatrix} = - \begin{bmatrix} \hat{\pi}_{k-1,n} \\ \vdots \\ \hat{\pi}_{1,n} \\ 0 \\ \vdots \\ 0 \end{bmatrix} - \hat{\beta}^{(n)}_0 \begin{bmatrix} \varepsilon_{n+k-1,n} \\ \vdots \\ \varepsilon_{n+1,n} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{5.20}$$

and

$$\text{(5.21)}\qquad \mathbf{S}^{\square}_{k-1,n}\begin{bmatrix}\beta^{(n)}_1\\ \vdots\\ \beta^{(n)}_{k-1}\\ \alpha^{(n)}_{k-2}\\ \vdots\\ \alpha^{(n)}_0\end{bmatrix} = -\begin{bmatrix}0\\ \vdots\\ 0\\ \pi_{1,n}\\ \vdots\\ \pi_{k-1,n}\end{bmatrix} - \beta^{(n)}_0\begin{bmatrix}0\\ \vdots\\ 0\\ \hat{\varepsilon}_{n+1,n}\\ \vdots\\ \hat{\varepsilon}_{n+k-1,n}\end{bmatrix}.$$

In summary, we see that we can detect and construct a column-regular pair $\hat{\mathbf{q}}_{n+k}$, $\mathbf{q}_{n+k}$ in this way if and only if (5.20) and (5.21) have a solution and ($\hat{\varepsilon}_{n+k,n+k} =$) $\varepsilon_{n+k,n+k} \neq 0$. Alternatively, one could solve (5.16) for $\hat{\varepsilon}_{n+k,n+k} := \varepsilon_{n+k,n+k} := 1$, check if ($\hat{\alpha}^{(n)}_{k-1} =$) $\alpha^{(n)}_{k-1} \neq 0$, and, if this holds, renormalize the solution. We still have to show that every column-regular pair can be found in this way. Moreover, in order that such a pair is well-column-regular we will need that (5.4) and (5.5) hold. We will come back to these stability conditions in Sect. 6. First, we want to clarify the relationship between the nonsingularity of the matrices $\mathbf{S}^{\square}_{k,n}$ and $\mathbf{S}^{\square}_{k-1,n}$ and the existence and uniqueness of normalized solutions of (5.16) and (5.18)–(5.19). We introduce column-regular index pairs $(k;n)$ associated to $\mathbf{S}^{\square}_{k,n}$, and prove a partial analogue of Lemma 4.1.

Definition. The pair of nonnegative indices $(k;n)$ is *column-regular* if the index n is column-regular and if the two systems (5.20) and (5.21) have a unique pair of solutions, and these solutions satisfy $\hat{\varepsilon}_{n+k,n+k} \neq 0$ and $\varepsilon_{n+k,n+k} \neq 0$, respectively.

Lemma 5.4. *For $\hat{\varepsilon}_{n,n} \neq 0$ and $\varepsilon_{n,n} \neq 0$, the following statements are equivalent:*

(i) *the two systems* (5.20) *and* (5.21) *have a unique pair of solutions, and these solutions satisfy $\hat{\varepsilon}_{n+k,n+k} \neq 0$ and $\varepsilon_{n+k,n+k} \neq 0$, respectively;*

(ii) *$\mathbf{S}^{\square}_{k-1,n}$ and $\mathbf{S}^{\square}_{k,n}$ are nonsingular;*

(iii) *every pair of nontrivial solutions of* (5.16) *and (5.17)–(5.18) satisfies $\hat{\alpha}^{(n)}_{k-1} \neq 0$ and $\varepsilon_{n+k,n+k} \neq 0$;*

(iv) *every pair of nontrivial solutions of (5.16) and (5.17)–(5.18) satisfies $\alpha^{(n)}_{k-1} \neq 0$ and $\hat{\varepsilon}_{n+k,n+k} \neq 0$;*

Proof. (ii)⇔(iii), (ii)⇔(iv): obviously, the nonsingularity of $\mathbf{S}^{\square}_{k,n}$ is necessary for (iii) and (iv): if $\mathbf{S}^{\square}_{k,n}$ is singular, the homogeneous systems (5.16) obtained when $\varepsilon_{n+k,n+k} = \hat{\varepsilon}_{n+k,n+k} = 0$ have nontrivial solutions. On the other hand, if $\mathbf{S}^{\square}_{k,n}$ is nonsingular, $\hat{\varepsilon}_{n+k,n+k} \neq 0$ and $\varepsilon_{n+k,n+k} \neq 0$ for any nontrivial solution; moreover, we can apply Cramer's rule to verify that

$$\text{(5.22)}\quad \hat{\alpha}^{(n)}_{k-1} = \hat{\varepsilon}_{n+k,n+k}\varepsilon_{n,n}\frac{\det \mathbf{S}^{\square}_{k-1,n}}{\det \mathbf{S}^{\square}_{k,n}} \quad\text{and}\quad \alpha^{(n)}_{k-1} = \varepsilon_{n+k,n+k}\hat{\varepsilon}_{n,n}\frac{\det \mathbf{S}^{\square}_{k-1,n}}{\det \mathbf{S}^{\square}_{k,n}}.$$

Hence, $\hat{\alpha}^{(n)}_{k-1} \neq 0$ and $\alpha^{(n)}_{k-1} \neq 0$ if and only if $\mathbf{S}^{\square}_{k-1,n}$ is nonsingular. Therefore, (ii) is equivalent to (iii) and (iv). Moreover, (5.22) yields the relation

$$\text{(5.23)}\qquad \varepsilon_{n+k,n+k}\hat{\varepsilon}_{n,n}\hat{\alpha}^{(n)}_{k-1} = \alpha^{(n)}_{k-1}\varepsilon_{n,n}\hat{\varepsilon}_{n+k,n+k},$$

which is analogous to (4.4).

(i)⇔(ii): to prove the equivalence of (i) and (ii), note that the uniqueness in (i) is equivalent to $\mathbf{S}^{\square}_{k-1,n}$ being nonsingular. If $\mathbf{S}^{\square}_{k,n}$ is also nonsingular, (5.22) implies that $\hat{\varepsilon}_{n+k,n+k} \neq 0$ or $\varepsilon_{n+k,n+k} \neq 0$ (since $\hat{\alpha}^{(n)}_{k-1} = \alpha^{(n)}_{k-1} = 1$). Conversely, because $\hat{\varepsilon}_{n+k,n+k}$, $\varepsilon_{n+k,n+k}$, and $\det \mathbf{S}^{\square}_{k,n}$ depend continuously on the additional elements $\hat{\pi}_{k-1,n}$, $\pi_{k-1,n}$, $\hat{\varepsilon}_{n+k-1,n}$, and $\varepsilon_{n+k-1,n}$ of $\mathbf{S}^{\square}_{k,n}$, we can conclude from (5.22) that $\det \mathbf{S}^{\square}_{k,n} = 0$ implies $\hat{\varepsilon}_{n+k,n+k} = 0$ and $\varepsilon_{n+k,n+k} = 0$.

(iii)⇔(iv): follows from what is already proved, in particular from (5.23).

From the Lemmas 4.1, 5.2, 5.3 and 5.4, we finally obtain

Theorem 5.5. *Let the index n be column-regular. Then, the index $n + k$ is column-regular if and only if the pair of indices $(k; n)$ is column-regular.*

Proof. If n and $n + k$ are column-regular, then, by Lemma 5.3, $\mathbf{S}^{\square}_{k,n}$ and $\mathbf{S}^{\square}_{k-1,n}$ are nonsingular, and thus, by Lemma 5.4, $(k; n)$ is column-regular. Conversely, if n and $(k; n)$ are column-regular, then, by Lemma 5.4, there exist normalized (and unique) solutions of (5.16) and (5.17)–(5.18) with $\hat{\varepsilon}_{n+k,n+k} \neq 0$, $\varepsilon_{n+k,n+k} \neq 0$. Our construction based on (5.1) delivers then a pair $\hat{\mathbf{q}}_{n+k}$, $\mathbf{q}_{n+k}$ of solutions of the Yule-Walker equations with these values of $\hat{\varepsilon}_{n+k,n+k}$ and $\varepsilon_{n+k,n+k}$, and by Lemma 4.1 we can conclude that $n + k$ is column-regular.

The same arguments lead additionally to an equivalence that is analogous to one in Lemma 4.1, but was left out in Lemma 5.4. Note the minor change in the assumptions.

Lemma 5.6. *Assume n is a column-regular index. Then the following statements are equivalent:*

(i) *$n + k$ is column-regular;*
(ii) *the two systems* (5.20) *and* (5.21) *have a unique pair of solutions, and these solutions satisfy $\hat{\varepsilon}_{n+k,n+k} \neq 0$ and $\varepsilon_{n+k,n+k} \neq 0$, respectively;*
(iii) *the two systems* (5.20) *and* (5.21) *have a pair of solutions satisfying $\hat{\varepsilon}_{n+k,n+k} \neq 0$ and $\varepsilon_{n+k,n+k} \neq 0$, respectively;*

Proof. (i)⇒(ii): If n and $n + k$ are column-regular, then, by Lemma 5.3, $\mathbf{S}^{\square}_{k,n}$ and $\mathbf{S}^{\square}_{k-1,n}$ are nonsingular, and by Lemma 5.4 the condition (ii) holds.

(ii)⇒(iii): trivial.

(iii)⇒(iv): from the solutions of (5.20) and (5.21) we can construct a pair of solutions of the Yule-Walker equations with the same values of $\hat{\varepsilon}_{n+k,n+k}$ and $\varepsilon_{n+k,n+k}$. Therefore, by Lemma 4.1, $n + k$ is column-regular.

If $\mathrm{tol}(n) \equiv 0$ and $\mathrm{Tol}(n) \equiv \infty$, the construction described in this section always performs steps of minimal look-ahead step size to the next column-regular index. In particular, if $\mathbf{T}$ is strongly regular, we always have $k = 1$, i.e., the construction works without look-ahead. Then the two systems (5.20) and (5.21) are vacuous; the coefficients are already fully determined by (5.2) and (5.19), in accordance with (2.8). The recurrences thus reduce to those in the classical algorithms of Levinson and Schur. If $\mathbf{T}$ is not strongly regular, or if it is, but $\mathrm{tol}(n) > 0$ or $\mathrm{Tol}(n) < \infty$, look-ahead steps of size $k > 1$ may occur. Then we no longer get the complete block LDU decomposition of $\mathbf{T}$ or $\mathbf{T}^{-1}$, but only those columns of the unit upper triangular matrices $\hat{\mathbf{R}}$ and $\mathbf{R}$ or the lower triangular matrices $\hat{\mathbf{E}}$ and $\mathbf{E}$ that belong to the well-column-regular pairs. If we want to find the missing columns, we need to add *inner* vectors $\hat{\mathbf{q}}_{n+k}$, $\mathbf{q}_{n+k}$ for $n = n_j < n + k < n_{j+1}$ such that, instead of (5.3),

$$\mathbf{T}_{n+k+1}[\hat{\mathbf{q}}_{n+k}|\mathbf{q}_{n+k}] = \left[\begin{array}{c|c} \mathbf{0}_{n+1} & \varepsilon_{n+k,n+k} \\ \hat{\varepsilon}_{n+1,n+k} & \vdots \\ \vdots & \varepsilon_{n+1,n+k} \\ \hat{\varepsilon}_{n+k,n+k} & \mathbf{0}_{n+1} \end{array}\right]. \tag{5.24}$$

Then, in (5.16), $\hat{\mathbf{a}}$ and $\hat{\mathbf{b}}$ only need to fulfill the first k equations, and $\mathbf{a}$ and $\mathbf{b}$ have to satisfy the last k equations. Note that (5.19) remains valid for inner vectors, too; in other words, the first pair of inner vectors (for which $k = 1$) is obtained from the Schur parameters as when no look-ahead is done.

Since only k conditions have to be satisfied besides the normalization (5.2), it turns out that we can set

$$\hat{\alpha}_j^{(n)} = \alpha_j^{(n)} = 0, \quad j = 0, \ldots, k-2. \tag{5.25}$$

To distinguish the coefficients in the recurrences for the inner vectors from those for the computation of a new well-column-regular pair, we rename them according to

$$\hat{\gamma}_j^{(n)} := \hat{\beta}_j^{(n)}, \quad \gamma_j^{(n)} := \beta_j^{(n)}, \quad j = 0, \ldots, k-1. \tag{5.26}$$

Taking (5.19) into account, we just have to solve the $(k-1) \times (k-1)$ nonsingular triangular Toeplitz systems

$$\left[\begin{array}{ccc} \varepsilon_{n,n} & \cdots & \varepsilon_{n+k-2,n} \\ & \ddots & \vdots \\ \mathbf{O} & & \varepsilon_{n,n} \end{array}\right] \left[\begin{array}{c} \hat{\gamma}_{k-1}^{(n)} \\ \vdots \\ \hat{\gamma}_1^{(n)} \end{array}\right] = - \left[\begin{array}{c} \hat{\pi}_{k-1,n} \\ \vdots \\ \hat{\pi}_{1,n} \end{array}\right] - \hat{\gamma}_0^{(n)} \left[\begin{array}{c} \varepsilon_{n+k-1,n} \\ \vdots \\ \varepsilon_{n+1,n} \end{array}\right] \tag{5.27}$$

and

$$\left[\begin{array}{ccc} \hat{\varepsilon}_{n,n} & & \mathbf{O} \\ \vdots & \ddots & \\ \hat{\varepsilon}_{n+k-2,n} & \cdots & \hat{\varepsilon}_{n,n} \end{array}\right] \left[\begin{array}{c} \gamma_1^{(n)} \\ \vdots \\ \gamma_{k-1}^{(n)} \end{array}\right] = - \left[\begin{array}{c} \pi_{1,n} \\ \vdots \\ \pi_{k-1,n} \end{array}\right] - \gamma_0^{(n)} \left[\begin{array}{c} \hat{\varepsilon}_{n+1,n} \\ \vdots \\ \hat{\varepsilon}_{n+k-1,n} \end{array}\right]. \tag{5.28}$$

Note that these systems are nested with respect to k; if $k-1$ is replaced by k, only $\hat{\gamma}_{k-1}^{(n)}$ and $\gamma_{k-1}^{(n)}$ need to be computed additionally:

$$\begin{aligned} \hat{\gamma}_{k-1}^{(n)} &:= -\frac{1}{\varepsilon_{n,n}} \left(\hat{\pi}_{k-1,n} + \sum_{j=0}^{k-2} \varepsilon_{n+k-1-j,n} \hat{\gamma}_j^{(n)} \right), \\ \gamma_{k-1}^{(n)} &:= -\frac{1}{\hat{\varepsilon}_{n,n}} \left(\pi_{k-1,n} + \sum_{j=0}^{k-2} \hat{\varepsilon}_{n+k-1-j,n} \gamma_j^{(n)} \right). \end{aligned} \tag{5.29}$$

This has the consequence that the components of the inner vectors $\hat{\mathbf{q}}_{n+k}$ and $\mathbf{q}_{n+k}$ are obtained according to the following simple recurrences:

$$
\begin{aligned}
\hat{\rho}_{j,n+k} &:= \begin{cases} \hat{\gamma}_{k-1}^{(n)} \rho_{n-j,n} & (j = 0), \\ \hat{\rho}_{j-1,n+k-1} + \hat{\gamma}_{k-1}^{(n)} \rho_{n-j,n} & (j = 1, \ldots, n), \\ \hat{\rho}_{j-1,n+k-1} & (j = n+1, \ldots, n+k-1), \end{cases} \\
\rho_{j,n+k} &:= \begin{cases} \gamma_{k-1}^{(n)} \hat{\rho}_{n-j,n} & (j = 0), \\ \rho_{j-1,n+k-1} + \gamma_{k-1}^{(n)} \hat{\rho}_{n-j,n} & (j = 1, \ldots, n), \\ \rho_{j-1,n+k-1} & (j = n+1, \ldots, n+k-1) \, . \end{cases}
\end{aligned} \tag{5.30}
$$

Recall that $\hat{\rho}_{n+k,n+k} = \rho_{n+k,n+k} = 1$, and note that the diagonal blocks of $\hat{\mathbf{R}}$ and $\mathbf{R}$ have Toeplitz structure. These diagonal blocks correspond to the case $n+1 \le j \le n+k$ in (5.30).

We will show later, in Sect. 7 and Sect. 10 how the block diagonal matrix $\mathbf{D}$ in the decomposition (2.20) can be computed efficiently in our look-ahead Levinson and Schur algorithms, respectively. In Sect. 7 we will also show how to generate $\hat{\varepsilon}_{n+j,n}$ and $\varepsilon_{n+j,n}$ $(j = 1, \ldots, k)$ recursively, without evaluating inner products.

The formulas of this section are valid for $n = 0$ if and only if the pair $\hat{\mathbf{q}}_0 = \mathbf{q}_0 = [1]$ is column-regular, i.e. $n_1 = 0$, which holds if and only if $(\hat{\varepsilon}_{0,0} = \varepsilon_{0,0} =) \ \mu_0 \neq 0$. In floating-point arithmetic one needs that the pair is well-column-regular, which requires that $|\mu_0| > \text{tol}(0)$. Note that $\hat{\mathbf{q}}_0 = \mathbf{q}_0 = [1]$ implies that

$$
\begin{aligned}
\hat{\varepsilon}_{i,0} &= \mu_i \ \ (i \ge 0), & \hat{\pi}_{i-1,0} &= \mu_{-i} \ \ (i \ge 1), \\
\varepsilon_{i,0} &= \mu_{-i} \ \ (i \ge 0), & \pi_{i-1,0} &= \mu_i \ \ (i \ge 1).
\end{aligned} \tag{5.31}
$$

In particular, $[\hat{\varepsilon}_{0,0}] = [\varepsilon_{0,0}] = [\mu_0] = \mathbf{T}_1$. If $n = 0$ is not a well-column-regular index, we have to determine n_1 by solving the Yule-Walker equations (2.2) for $n = 1, 2, \ldots$ until we find a well-column-regular pair. Alternatively, one could use another method to determine the first pair $\mathbf{T}_n$, $\mathbf{T}_{n+1}$ of successive well-conditioned leading principal submatrices of $\mathbf{T}$. The inner vectors in this initial block can be chosen as the unit vectors

$$
\hat{\mathbf{q}}_k := \mathbf{e}_k \in \mathbb{R}^k, \quad \mathbf{q}_k := \mathbf{e}_0 \in \mathbb{R}^k, \quad 0 \le k < n_1, \tag{5.32}
$$

which satisfy (5.24) for $n = n_0 := -1$.

6. Stability

In an LDU decomposition, and likewise in an inverse LDU decomposition, stability is normally achieved by pivoting, which makes pivots large and keeps the growth of the off-diagonal elements under control. In a block LDU decomposition without pivoting, and also in an inverse block LDU decomposition, one avoids small pivots by choosing *block pivots* that are far from singular. If, for each reduced system, the smallest singular value of the block pivot is larger than a fixed fraction $1/\gamma$ of the norm of every off-diagonal block in the same block column, then the norm of any block in the jth reduced system is at most $(1+\gamma)^j$ times the norm of the original block in the same position, i.e., one has a growth factor of at most $1+\gamma$ for each elimination step.[1] But since γ can be much larger than 1, the growth can be considerably larger than when column-pivoting is applied. It is therefore important to keep the growth of the off-diagonal elements somehow under control.

[1] We are indebted to Leslie V. Foster for quickly providing us with a proof of this fact at the Householder XII Symposium

In our situation, unlike in ordinary Gauss elimination, the inversion formulas for Toeplitz matrices make it possible to estimate the norm of the inverses of the principal submatrices $\mathbf{T}_{n_j}$. From the Gohberg-Semencul formulas we conclude that the Frobenius norm of the inverses of $\mathbf{T}_{n_j}$ and $\mathbf{T}_{n_j+1}$ cannot be large if the norms $||\hat{\mathbf{q}}_{n_j}||$ and $||\mathbf{q}_{n_j}||$ of the regular pairs do not become large and $|\varepsilon_{n_j,n_j}|$ is bounded away from zero. This was, of course, the basis of our definition of well-column-regular pairs. This definition specifies a first look-ahead strategy.

However, using only this definition for the determination of the look-ahead step size requires that, for every n, the tentative regular pair and the norm of its two vectors be computed. If the pair fails to be well-column-regular, this computation is wasted. Also in the look-ahead Schur algorithm we may not want to compute these pairs and their norm (at least not for all j). Fortunately, we can estimate the norm. This leads to the requirement to bound the recurrence coefficients $\hat{\alpha}_i^{(n)}$, $\hat{\beta}_i^{(n)}$, $\alpha_i^{(n)}$, $\beta_i^{(n)}$.

Definition. The column-regular index pair $(n;k)$ is *well-column-regular* if the corresponding vectors $[\ \hat{\mathbf{a}}^{\mathrm{T}} \mid \hat{\mathbf{b}}^{\mathrm{T}}\ ]^{\mathrm{T}}$ and $[\ \mathbf{a}^{\mathrm{T}} \mid \mathbf{b}^{\mathrm{T}}\ ]^{\mathrm{T}}$, which are defined by (5.13) and obtained from (5.2) and (5.19)–(5.21), satisfy

$$\left\| \begin{bmatrix} \hat{\mathbf{a}} \\ \hline \hat{\mathbf{b}} \end{bmatrix} \right\|, \ \left\| \begin{bmatrix} \mathbf{a} \\ \hline \mathbf{b} \end{bmatrix} \right\| < \mathrm{Tol}(n;k), \tag{6.1}$$

$$(|\hat{\varepsilon}_{n+k,n+k}| =)\ |\varepsilon_{n+k,n+k}| > \mathrm{tol}(n+k). \tag{6.2}$$

The index pair $(n;k)$ is then also called well-column-regular.

The new tolerance function $\mathrm{Tol}(n;k) > 1$ in (6.1) should be compatible with $\mathrm{Tol}(n)$ in the sense that (6.1) implies that $\hat{\mathbf{q}}_{n+k}$ and $\hat{\mathbf{q}}_{n+k}$, defined by (5.1), satisfy (5.4). The following lemma shows how to fulfill this condition.

Lemma 6.1. *Assume the functions* $\mathrm{Tol}(n)$ (>1) *and* $\mathrm{Tol}(n;k)$ (>1) *are chosen such that*

$$\mathrm{Tol}(n;k) \le \frac{\mathrm{Tol}(n+k)}{\sqrt{n+1}\ \mathrm{Tol}(n)} \quad (n < n+k \le N), \tag{6.3}$$

which requires that

$$\mathrm{Tol}(n+k) > \sqrt{n+1}\mathrm{Tol}(n) \quad (n < n+k \le N). \tag{6.4}$$

Then, if n *and* $(n;k)$ *are well-column-regular,* $n+k$ *also is well-column-regular. Inequality* (6.4) *is fulfilled if, e.g.,* $\mathrm{Tol}(n) := n^{n/2}$.

Proof. We rewrite (5.1) as

$$\hat{\mathbf{q}}_{n+k} = \left[\begin{array}{ccc|ccc} 0 & \cdots & 0 & \hat{\beta}_{k-1}^{(n)} & & 0 \\ \hat{\alpha}_0^{(n)} & \ddots & \vdots & \hat{\beta}_{k-2}^{(n)} & \ddots & \\ \hat{\alpha}_1^{(n)} & \ddots & 0 & \vdots & \ddots & \hat{\beta}_{k-1}^{(n)} \\ \vdots & \ddots & \hat{\alpha}_0^{(n)} & \hat{\beta}_0^{(n)} & & \hat{\beta}_{k-2}^{(n)} \\ \hat{\alpha}_{k-1}^{(n)} & & \hat{\alpha}_1^{(n)} & 0 & \ddots & \vdots \\ & \ddots & \vdots & \vdots & \ddots & \hat{\beta}_0^{(n)} \\ 0 & & \hat{\alpha}_{k-1}^{(n)} & 0 & \cdots & 0 \end{array}\right] \left[\begin{array}{c} \hat{\rho}_{0,n} \\ \vdots \\ \hat{\rho}_{n,n} \\ \hline \rho_{n,n} \\ \vdots \\ \rho_{0,n} \end{array}\right] \tag{6.5}$$

and an analogous relation for $\mathbf{q}_{n+k}$. The matrix is of size $(n + k + 1) \times 2(n + 1)$. For the Euclidean norm of $\hat{\mathbf{q}}_{n+k}$ we obtain the estimate

$$\begin{aligned} ||\hat{\mathbf{q}}_{n+k}|| &\leq ||\hat{\mathbf{a}}|| \sum_{i=0}^{n} |\hat{\rho}_{i,n}| + ||\hat{\mathbf{b}}|| \sum_{i=0}^{n} |\rho_{i,n}| \\ &\leq \sqrt{n+1}\left(||\hat{\mathbf{a}}||\ ||\hat{\mathbf{q}}_n|| + ||\hat{\mathbf{b}}||\ ||\mathbf{q}_n||\right) \\ &< \sqrt{n+1}\ \mathrm{Tol}(n)\mathrm{Tol}(n;k), \end{aligned} \tag{6.6}$$

and an analogous one is found for $||\mathbf{q}_{n+k}||$. Consequently, (6.3) implies that $||\hat{\mathbf{q}}_{n+k}|| \leq \mathrm{Tol}(n + k)$ and $||\mathbf{q}_{n+k}|| \leq \mathrm{Tol}(n + k)$, which is in accordance with (5.4).

Due to $\sqrt{n+1}\ n^{n/2} \leq (n + 1)^{(n+1)/2} \leq (n + k)^{(n+k)/2}$, the inequality (6.4) holds for $\mathrm{Tol}(n) = n^{n/2}$.

Unfortunately, the tolerance function $\mathrm{Tol}(n) = n^{n/2}$ grows far too fast to be useful in practice. The problem here is that the estimate based on the Frobenius norm of the Toeplitz matrices in (6.5), which is responsible for the factor $\sqrt{n+1}$ in (6.6), is very weak. Although it could be sharpened, it seems hard to obtain in this way practical compatible tolerance functions $\mathrm{Tol}(n)$ and $\mathrm{Tol}(n; k)$. Nevertheless, in practice one can still heuristically choose tolerance functions and use (6.1)–(6.2) as look-ahead strategy. The submatrices $\mathbf{T}_n$ that will be considered as well conditioned according to this strategy can then become rather ill conditioned, but there still exists a (possibly very large) a priori upper bound for their condition number, and chances that the actual condition number is anywhere close to the bound may be very small.

A third look-ahead strategy consists in requiring that (6.2) holds and, if $k > 1$, the smallest singular value of $\mathbf{S}^{\square}_{k-1,n}$ be larger than a certain bound tol_σ, which may also depend on n and $k - 1$. First, note that (2.2)–(2.4) imply that $\varepsilon_{n,n} = \hat{\varepsilon}_{n,n}$ is the Schur complement of $\mathbf{T}_n$ in $\mathbf{T}_{n+1}$. This statement only requires that $\mathbf{T}_n$ is nonsingular. Therefore, if $\mathbf{T}_{n+k}$ is nonsingular, then (6.2) guarantees that $\mathbf{T}_{n+k+1}$ is nonsingular. One can also argue as in Chan and Hansen [9] that $|\varepsilon_{n+k,n+k}|$ is an estimate of $\sigma_{\min}(\mathbf{T}_{n+k+1})$ if $\mathbf{T}_{n+k}$ is well conditioned and $|\varepsilon_{n+k,n+k}|$ is small. More exactly, from Lemma 4.2 we see that a well-conditioned $\mathbf{T}_{n+k}$ implies that $||\hat{\mathbf{q}}_{n+k}||$ and $||\mathbf{q}_{n+k}||$ are bounded and that therefore, if (6.2) holds, $||\mathbf{T}^{-1}_{n+k+1}|| = \sigma^{-1}_{\min}(\mathbf{T}_{n+k+1})$ is also bounded.

If $k = 1$, we know from the previous step that $\mathbf{T}_{n+k}$ is well conditioned, and thus it suffices to check (6.2). If $k > 1$, we must make sure that $\mathbf{T}_{n+k}$ is well conditioned. If n is well-column-regular and if the smallest singular value of $\mathbf{S}^{\square}_{k-1,n}$ is larger than tol_σ, one can show that the right-hand sides of (5.20) and (5.21) are bounded. The vectors $\hat{\mathbf{a}}$, $\hat{\mathbf{b}}$, $\mathbf{a}$, and $\mathbf{b}$ of the recurrence coefficients are at most by a factor $1/\mathrm{tol}_\sigma$ larger. From (6.6) and Lemma 4.2 it follows finally that $\mathbf{T}_{n+k}$ is well conditioned.

At first sight, it seems surprising that the aforementioned check contains only the smallest singular value of $\mathbf{S}^{\square}_{k-1,n}$ but not the condition number of this matrix. However, since the elements of $\mathbf{S}^{\square}_{k-1,n}$ are just products of the matrix $\mathbf{T}_{n+k}$ (whose norm is bounded by $||\mathbf{T}||$) and the well-column-regular pair $\hat{\mathbf{q}}_n$, $\mathbf{q}_n$, these elements cannot be very large. Therefore, roundoff will not be a serious problem when we are solving (5.20)-(5.21).

Unfortunately, the exact quantitative connections between $||\hat{\mathbf{q}}_n||$, $||\mathbf{q}_n||$, $\varepsilon_{n,n}$, $\sigma_{\min}(\mathbf{S}^{\square}_{k-1,n})$, $||\hat{\mathbf{q}}_{n+k}||$, and $||\mathbf{q}_{n+k}||$ seem to be very complicated. So, again, it is impossible to specify really useful tolerance functions $\mathrm{tol}(n)$, $\mathrm{Tol}(n)$, and $\mathrm{tol}_\sigma(n; k)$ for which an analogue of Lemma 6.1 holds. Nevertheless, in practice this strategy

has proven to be very successful. It is the one chosen for our numerical examples in Sect. 12. Regarding the discrepancy between actual condition number and guaranteed condition number the same remark as for the second strategy applies. As we know from Gaussian elimination with partial pivoting, a strategy may be very good in nearly all cases, although the theoretical bounds suggest that it is far too weak.

Note that, as in (4.6), we can conclude from (5.2) and (5.16) that

$$|\hat{\varepsilon}_{n+k,n+k}| = |\varepsilon_{n+k,n+k}| \geq \sigma_{\min}(\mathbf{S}^{\square}_{k,n}). \tag{6.7}$$

Hence, there is at least one simple relation between $\sigma_{\min}(\mathbf{S}^{\square}_{k,n})$ and the condition (6.2) (which is the same as (5.5)).

When $k = 1$, the condition estimator used by Chan and Hansen [9] is comparable to our estimate of Lemma 4.2 based on the Gohberg-Semencul formula. They use the infinity-norm of the vectors $\hat{\mathbf{q}}_n$ and $\mathbf{q}_n$, which is rather expensive to compute, while we use the Euclidean norm, but have an additional factor $2n$ in the estimate. When $k > 1$, Chan and Hansen need the $2k$ solution vectors of the "block Yule-Walker equations" for their condition estimate. This seems unnecessarily complicated since the Gohberg-Semencul formula shows that the inverse only depends on one pair of vectors. The look-ahead strategy of Freund and Zha [14] involves the computation of the condition number of matrices of order k that are the products of the corresponding three diagonal blocks of the factors of the block LDU decomposition of $\mathbf{T}$. When $k = 1$ this is just a check of $\varepsilon_{n+1,n+1} \not\approx 0$. Additionally, the size of the recurrence coefficient vectors is checked: their 1-norm must be below a certain bound that is enlarged when k exceeds a specified value. Hence, the Freund and Zha look-ahead strategy has some similarity with our second strategy, which is based on well-column-regular index pairs $(n; k)$.

7. Look-ahead Levinson and Schur algorithms: summary

Summarizing our derivation from the previous sections, we obtain the following look-ahead versions of the Levinson and the Schur algorithm, respectively. We have chosen here the look-ahead strategy that checks the condition (5.5) and, for look-ahead step size $k > 1$, the smallest singular value of $\mathbf{S}^{\square}_{k-1,n}$.

Algorithm 3. (Nonsymmetric Levinson algorithm with look-ahead)

1. (Initialization)
   Set $n := 0$, $n_0 := -1$;
   while not ($\mathbf{T}_n$ and $\mathbf{T}_{n+1}$ are well conditioned):
   set $\hat{\rho}_{n,n} := \rho_{n,n} := 1$, $\hat{\rho}_{j,n} := \rho_{j,n} := 0$ ($j = 0, \ldots, n-1$);
   $n := n + 1$
   end while;
   solve (2.2) (e.g. with Gauss elimination with pivoting) and obtain a well-column-regular pair $\hat{\mathbf{q}}_n$, $\mathbf{q}_n$, as well as $\hat{\varepsilon}_{n,n} = \varepsilon_{n,n}$;
   set $l := 1$; $n_l := n$;
   optional: set $\mathbf{D}_0 := \mathbf{T}_n$, $\mathbf{D}_1 := [\varepsilon_{n,n}]$.
2. (Compute inner products and Schur parameters)
   Compute

$\hat{\pi}_{0,n} := [\mu_{-1} \cdots \mu_{-n-1}]\hat{\mathbf{q}}_n$, $\pi_{0,n} := [\mu_{n+1} \cdots \mu_1]\mathbf{q}_n$;
$\hat{\beta}_0^{(n)} := \hat{\gamma}_0^{(n)} := -\hat{\pi}_{0,n}/\varepsilon_{n,n}$, $\beta_0^{(n)} := \gamma_0^{(n)} := -\pi_{0,n}/\hat{\varepsilon}_{n,n}$;
$\hat{\varepsilon}_{n+1,n+1} := \varepsilon_{n+1,n+1} := \varepsilon_{n,n} - \hat{\pi}_{0,n}\beta_0^{(n)}$.

3. (Check well-column-regularity of $\hat{\mathbf{q}}_{n+1}$, $\mathbf{q}_{n+1}$)
Set $k := 1$, $column\_regular := (|\varepsilon_{n+1,n+1}| \geq \text{tol}(n+1))$.
4. (Look-ahead step)
While not $column\_regular$:
if $k > 1$, compute $\hat{\gamma}_{k-1}^{(n)}$ and $\gamma_{k-1}^{(n)}$ according to (5.29);
compute $\hat{\rho}_{n,n+k}$, $\rho_{n,n+k}$, $\hat{\rho}_{n+1,n+k}$, $\rho_{n+1,n+k}$ according to (5.30);
optional: compute the full inner vectors $\hat{\mathbf{q}}_{n+k}$, $\mathbf{q}_{n+k}$ according to (5.30);
optional: update $\mathbf{D}_l$ according to (7.4)–(7.6) below;
compute $\hat{\pi}_{k,n} := [\mu_{-k-1} \cdots \mu_{-n-k-1}]\hat{\mathbf{q}}_n$, $\pi_{k,n} := [\mu_{n+k+1} \cdots \mu_{k+1}]\mathbf{q}_n$;
compute $\hat{\varepsilon}_{n+k,n}$ and $\varepsilon_{n+k,n}$ according to (7.9) and (7.10) below;
set $k := k+1$, fill in $\mathbf{S}_{k-1,n}^{\square}$;
if $\sigma_{\min}(\mathbf{S}_{k-1,n}^{\square}) \geq \text{tol}_\sigma(n; k-1)$:
solve (5.20) to obtain $\hat{\alpha}_0^{(n)}, \ldots, \hat{\alpha}_{k-2}^{(n)}$, $\hat{\beta}_1^{(n)}, \ldots, \hat{\beta}_{k-1}^{(n)}$, set $\hat{\alpha}_{k-1}^{(n)} := 1$;
solve (5.21) to obtain $\beta_1^{(n)}, \ldots, \beta_{k-1}^{(n)}$, $\alpha_0^{(n)}, \ldots, \alpha_{k-2}^{(n)}$, set $\alpha_{k-1}^{(n)} := 1$;
compute $\hat{\varepsilon}_{n+k,n+k} := \varepsilon_{n+k,n+k}$ from (5.18) or (5.19);
$column\_regular := (|\varepsilon_{n+k,n+k}| \geq \text{tol}(n+k)$
else
$column\_regular := false$
end if
end while.
5. (Compute a well-column-regular pair)
Compute $\hat{\mathbf{q}}_{n+k}$ and $\mathbf{q}_{n+k}$ from (5.1);
optional: set $\mathbf{D}_l := [\varepsilon_{n,n}]$.
6. Set $l := l+1$, $n_l := n := n+k$; goto 2.

The look-ahead Schur algorithm differs from the look-ahead Levinson algorithm mainly by the replacement of the inner products by the SAXPYs of (5.17), which yield directly the elements of $\mathbf{S}_{k-1,n}^{\square}$ for all $k \leq N-n$. The computation of the pairs $\hat{\mathbf{q}}_n$ and $\mathbf{q}_n$ is optional.

Algorithm 4. (Nonsymmetric Schur algorithm with look-ahead)

1. (Initialization)
Set $n := 0$, $n_0 := -1$;
while not ($\mathbf{T}_n$ and $\mathbf{T}_{n+1}$ are well conditioned):
for $j = 0, \ldots, n-1$:
set $\hat{\varepsilon}_{j,n} := \mu_{j-n}$, $\varepsilon_{j,n} := \mu_{-j+n}$ $(j = 0, \ldots, N)$;
set $\hat{\pi}_{j,n} := \mu_{-n-j-1}$, $\pi_{j,n} := \mu_{n+j+1}$ $(j = 0, \ldots, N-n-1)$;
$n := n+1$
end while;
solve (2.2) (e.g. with Gauss elimination with pivoting) and obtain a well-column-regular pair $\hat{\mathbf{q}}_n$, $\mathbf{q}_n$;
compute $\hat{\varepsilon}_{j,n}$, $\varepsilon_{j,n}$ $(j = n, \ldots, N)$ and $\hat{\pi}_{j,n}$, $\pi_{j,n}$ $(j = 0, \ldots, N-n-1)$ from (3.4);
set $l := 1$; $n_l := n$;
optional: set $\mathbf{D}_0 := \mathbf{T}_n$, $\mathbf{D}_1 := [\varepsilon_{n,n}]$.

2. (Compute Schur parameters and check well-column-regularity of $\mathbf{q}_{n+1}$, $\hat{\mathbf{q}}_{n+1}$)
   Compute
   $\beta_0^{(n)} := \gamma_0^{(n)} := -\pi_{0,n}/\hat{\varepsilon}_{n,n}$, $\hat{\beta}_0^{(n)} := \hat{\gamma}_0^{(n)} := -\hat{\pi}_{0,n}/\varepsilon_{n,n}$;
   $\hat{\varepsilon}_{n+1,n+1} := \varepsilon_{n+1,n+1} := \varepsilon_{n,n} - \hat{\pi}_{0,n}\beta_0^{(n)}$;
   set $k := 1$, $column\_regular := (|\varepsilon_{n+1,n+1}| \geq \mathrm{tol}(n+1))$.
3. (Look-ahead step)
   While not $column\_regular$:
   optional: if $k > 1$: compute $\hat{\gamma}_{k-1}^{(n)}$ and $\gamma_{k-1}^{(n)}$ from (5.29) and

$$(7.1)\qquad \begin{aligned} \hat{\varepsilon}_{j,n+k} &:= \begin{cases} \hat{\gamma}_{k-1}^{(n)}\pi_{0,n} & (j = n+1), \\ \hat{\varepsilon}_{j-1,n+k-1} + \hat{\gamma}_{k-1}^{(n)}\pi_{j-n-1,n} & (j = n+2,\ldots,N), \end{cases} \\ \varepsilon_{j,n+k} &:= \begin{cases} \gamma_{k-1}^{(n)}\hat{\pi}_{0,n} & (j = n+1), \\ \varepsilon_{j-1,n+k-1} + \gamma_{k-1}^{(n)}\hat{\pi}_{j-n-1,n} & (j = n+2,\ldots,N) \end{cases} \end{aligned}$$

   else

$$(7.2)\qquad \begin{aligned} \hat{\varepsilon}_{j,n+1} &:= \hat{\varepsilon}_{j-1,n} + \hat{\gamma}_0^{(n)}\pi_{j-n-1,n} \qquad (j = n+1,\ldots,N) \\ \varepsilon_{j,n+1} &:= \varepsilon_{j-1,n} + \gamma_0^{(n)}\hat{\pi}_{j-n-1,n} \end{aligned}$$

   end if;
   optional: compute inner vectors $\hat{\mathbf{q}}_{n+k}$, $\mathbf{q}_{n+k}$ according to (5.30);
   optional: update $\mathbf{D}_l$ according to (7.4)–(7.7), where $\rho_{n,n+k}$ is computed via (7.9);
   set $k := k+1$, fill in $\mathbf{S}_{k-1,n}^{\square}$;
   if $\sigma_{\min}(\mathbf{S}_{k-1,n}^{\square}) \geq \mathrm{tol}_\sigma(n; k-1)$:
      solve (5.20) to obtain $\hat{\alpha}_0^{(n)},\ldots,\hat{\alpha}_{k-2}^{(n)}$, $\hat{\beta}_1^{(n)},\ldots,\hat{\beta}_{k-1}^{(n)}$, set $\hat{\alpha}_{k-1}^{(n)} := 1$;
      solve (5.21) to obtain $\beta_1^{(n)},\ldots,\beta_{k-1}^{(n)}$, $\alpha_0^{(n)},\ldots,\alpha_{k-2}^{(n)}$, set $\alpha_{k-1}^{(n)} := 1$;
      compute $\hat{\varepsilon}_{n+k,n+k} = \varepsilon_{n+k,n+k}$ from (5.18) or (5.19);
      $column\_regular := (|\varepsilon_{n+k,n+k}| \geq \mathrm{tol}(n+1))$
   else
      $column\_regular := false$
   end if
   end while.
4. (Compute regular vectors)
   Compute $\hat{\varepsilon}_{j,n+k}$, $\varepsilon_{j,n+k}$ $(j = n+k+1,\ldots,N)$ and
   $\hat{\pi}_{j,n+k}$, $\pi_{j,n+k}$ $(j = 0,\ldots,N-n-k-1)$ from (5.17);
   optional: compute $\hat{\mathbf{q}}_{n+k}$ and $\mathbf{q}_{n+k}$ from (5.1);
   optional: set $\mathbf{D}_l := [\varepsilon_{n,n}]$.
5. Set $l := l+1$, $n_l := n := n+k$; goto 2.

It remains to show how to update $\mathbf{D}$, which is defined in (2.17), and how to compute $\hat{\varepsilon}_{n+k,n}, \varepsilon_{n+k,n}$ in Step 4 of the look-ahead Levinson algorithm.

First we consider the update of $\mathbf{D}$. Since $\mathbf{R}^{\mathrm{T}}\hat{\mathbf{E}}$ is block lower triangular and $\mathbf{E}^{\mathrm{T}}\hat{\mathbf{R}}$ is block upper triangular, the matrix $\mathbf{D}$ is block diagonal. We can easily compute its blocks from (2.17) if $\hat{\mathbf{R}}$ and $\mathbf{R}$ are available. We let $\mathbf{D} =: (\delta_{i,j})_{i,j=0,\ldots,N} =: \mathrm{diag}\,(\mathbf{D}_0,\ldots,\mathbf{D}_J)$. If $n = n_l$ is the last well-column-regular index and k is such that $n_l < n+k < n_{l+1}$, then, for $j = 1,\ldots,k$,

$$\begin{aligned} \delta_{n+j,n+k} &= \sum_{i=1}^{j} \rho_{n+i,n+j}\hat{\varepsilon}_{n+i,n+k} = \sum_{i=1}^{j} \rho_{n,n+j-i}\hat{\hat{\varepsilon}}_{n+i,n+k}\ , \\ \delta_{n+k,n+j} &= \sum_{i=1}^{j} \hat{\rho}_{n+i,n+j}\varepsilon_{n+i,n+k} = \sum_{i=1}^{j} \hat{\rho}_{n,n+j-i}\varepsilon_{n+i,n+k}. \end{aligned} \tag{7.3}$$

For the first pair of vectors in each block we have, using (2.17) and (5.24),

$$\delta_{n,n} = \hat{\varepsilon}_{n,n} = \varepsilon_{n,n}, \qquad \delta_{n,j} = \delta_{j,n} = 0, \quad j \neq n, \tag{7.4}$$

i.e., the block $\mathbf{D}_l$ is itself block diagonal with a 1×1 block in its upper left corner. To update $\mathbf{D}_l$ efficiently we define, for $j = 1, \ldots, k$,

$$\hat{\psi}_{j,n} = \sum_{i=0}^{j} \hat{\rho}_{n,n+j-i}\hat{\pi}_{i,n}, \qquad \psi_{j,n} = \sum_{i=0}^{j} \rho_{n,n+j-i}\pi_{i,n}. \tag{7.5}$$

Then the following recurrences for the elements of $\mathbf{D}_l$ hold:

$$\begin{aligned} \delta_{n+j,n+k} &= \begin{cases} \hat{\gamma}_{k-1}^{(n)}\pi_{0,n} & (j = 1), \\ \hat{\gamma}_{k-1}^{(n)}\psi_{j-1,n} + \delta_{n+j-1,n+k-1} & (j = 2, \ldots, k), \end{cases} \\ \delta_{n+k,n+j} &= \begin{cases} \gamma_{k-1}^{(n)}\hat{\pi}_{0,n} & (j = 1), \\ \gamma_{k-1}^{(n)}\hat{\psi}_{j-1,n} + \delta_{n+k-1,n+j-1} & (j = 2, \ldots, k). \end{cases} \end{aligned} \tag{7.6}$$

They are derived from (7.3) using (5.30) and (7.1):

$$\begin{aligned} \delta_{n+j,n+k} &= \hat{\gamma}_{k-1}^{(n)} \sum_{i=1}^{j} \rho_{n,n+j-i}\pi_{i-1,n} + \sum_{i=2}^{j} \rho_{n,n+j-i}\hat{\hat{\varepsilon}}_{n+i-1,n+k-1} \\ &= \hat{\gamma}_{k-1}^{(n)}\psi_{j-1,n} + \sum_{i=1}^{j-1} \rho_{n,n+j-i}\hat{\hat{\varepsilon}}_{n+i,n+k} \\ &= \hat{\gamma}_{k-1}^{(n)}\psi_{j-1,n} + \delta_{n+j-1,n+k-1}. \end{aligned}$$

The proof for $\delta_{n+k,n+j}$ is analogous.

Since (7.6) provides us with two different formulas for the diagonal elements of $\mathbf{D}_l$, we can eliminate either $\psi_{j,n}$ or $\hat{\psi}_{j,n}$. To be more precise,

$$\hat{\gamma}_{k-1}^{(n)}\psi_{k-1,n} = \gamma_{k-1}^{(n)}\hat{\psi}_{k-1,n}. \tag{7.7}$$

Next we show that one can implement our look-ahead Levinson algorithm with the same number of inner products as the classical algorithm without look-ahead: inner products occur only in Step 4 of the algorithm, when we compute $\hat{\pi}_{k,n}$ and $\pi_{k,n}$, whereas $\hat{\varepsilon}_{n+k,n}$ and $\varepsilon_{n+k,n}$ are found without forming inner products of length $O(n)$. We start from (2.19) and consider rows $n + 1, \ldots, n + k$ and column n of the matrix equation $\mathbf{R}^{\mathrm{T}}\hat{\mathbf{E}} = \mathbf{D}$, where $\mathbf{R}$ is unit upper triangular and $\hat{\mathbf{E}}$ is block lower triangular. Recall from (5.30) that our special choice of inner vectors implies that the diagonal blocks of $\mathbf{R}$ inherit the Toeplitz structure. Hence, using (7.4), we are left with the following $n \times (n + 1)$ system:

$$\left[\begin{array}{c|ccc} \rho_{n,n+1} & \rho_{n,n} & & 0 \\ \vdots & \vdots & \ddots & \\ \rho_{n,n+k} & \rho_{n,n+k-1} & \cdots & \rho_{n,n} \end{array}\right] \left[\begin{array}{c} \hat{\varepsilon}_{n,n} \\ \hline \hat{\varepsilon}_{n+1,n} \\ \vdots \\ \hat{\varepsilon}_{n+k,n} \end{array}\right] = \left[\begin{array}{c} 0 \\ \vdots \\ 0 \end{array}\right].$$

Since $\hat{\varepsilon}_{n,n}$ is known from the previous step, we can move the first column to the right-hand side to obtain the triangular Toeplitz system

$$\left[\begin{array}{ccc} \rho_{n,n} & & 0 \\ \vdots & \ddots & \\ \rho_{n,n+k-1} & \cdots & \rho_{n,n} \end{array}\right] \left[\begin{array}{c} \hat{\varepsilon}_{n+1,n} \\ \vdots \\ \hat{\varepsilon}_{n+k,n} \end{array}\right] = -\hat{\varepsilon}_{n,n} \left[\begin{array}{c} \rho_{n,n+1} \\ \vdots \\ \rho_{n,n+k} \end{array}\right]. \tag{7.8}$$

From this relation, we can compute $\hat{\varepsilon}_{n+j,n}$ $(j = 1, \ldots, k)$ recursively; in particular,

$$\hat{\varepsilon}_{n+k,n} = -\rho_{n,n+k}\hat{\varepsilon}_{n,n} - \sum_{l=1}^{k-1} \rho_{n,n+k-l}\hat{\varepsilon}_{n+l,n}. \tag{7.9}$$

Similarly, $\varepsilon_{n+k,n}$ is obtained from

$$\varepsilon_{n+k,n} = -\hat{\rho}_{n,n+k}\varepsilon_{n,n} - \sum_{l=1}^{k-1} \hat{\rho}_{n,n+k-l}\varepsilon_{n+l,n}. \tag{7.10}$$

In particular, for $k = 1$, we can compute $\hat{\varepsilon}_{n+1,n}$ and $\varepsilon_{n+1,n}$ from $\hat{\varepsilon}_{n,n} = \varepsilon_{n,n}$, $\rho_{n,n+1}$, and $\hat{\rho}_{n,n+1}$. Recall that the last two quantities are found from the Schur parameters and (5.1) or (5.30) as when $n + 1$ is a column-regular index. Once, $\hat{\varepsilon}_{n+1,n}$ and $\varepsilon_{n+1,n}$ are known, we get $\gamma_1^{(n)}$ and $\gamma_1^{(n)}$ from (5.27) and (5.28); see (5.29). Then, $\{\hat{\rho}_{j,n+2}, \rho_{j,n+2}\}_{j=0}^{n+1}$ follow from (5.30). Note that for the next step, where $k = 2$, we only need the four numbers $\{\hat{\rho}_{j,n+2}, \rho_{j,n+2}\}_{j=n}^{n+1}$. In view of the Toeplitz structure in (7.8), also in the kth step only the four new numbers $\{\hat{\rho}_{j,n+k}, \rho_{j,n+k}\}_{j=n}^{n+1}$ need to be calculated from (5.30). Hence, there is no need to compute the full inner vectors and to allocate additional memory space of order n.

In our look-ahead Schur algorithm, the situation is just the opposite to the above. We have $\hat{\varepsilon}_{n+k,n}$ and $\varepsilon_{n+k,n}$ available and need the diagonal blocks of $\hat{\mathbf{R}}$ and $\mathbf{R}$ to compute $\mathbf{D}$. However, solving the equations (7.9) and (7.10) for $\hat{\rho}_{n+k,n}$ and $\rho_{n+k,n}$, respectively, provides us with a recursion for the elements in the diagonal blocks of $\hat{\mathbf{R}}$ and $\mathbf{R}$. With these parameters, we can update $\mathbf{D}_l$ according to (7.4)–(7.6) without computing the full regular vectors $\hat{\mathbf{q}}_{n_l}$ and $\mathbf{q}_{n_l}$.

We postpone the discussion of how to solve the Toeplitz system to Sect. 10 and the specification of the costs and of the look-ahead overhead to Sect. 11.

8. Superfast Toeplitz solvers

In the previous sections we have been concerned with fast, i.e. $O(N^2)$, variants of Toeplitz solvers. Now, we want to present a superfast, i.e. $O(N \log^2 N)$, variant with look-ahead. The main idea of superfast Toeplitz solvers is recursive doubling. It can be applied only to Schur type algorithms, so in the course of the computation elements of $\hat{\mathbf{E}}$, $\mathbf{E}$, and $\mathbf{S}_{n_j}^{\square}$ $(j = 1, \ldots, J-1)$ are constructed. Recall that the latter matrices contain

elements from the matrices $\hat{\mathbf{P}}$ and $\mathbf{P}$ defined in (3.1) and (3.2). Instead of computing all the columns of $\mathbf{E}$ with well-column-regular index n_j and all the corresponding matrices $\mathbf{S}^{\square}_{n_j}$, we now want to roughly double the index in each step. If no look-ahead is necessary, this means that we compute only the columns with index $2^j - 1$ for $j = 0, 1, \ldots$. To show how this can be done efficiently, we start with (5.17) and a solution of (5.16). The aim is to compute the solution of (5.16) for $k + k'$ instead of k from a solution for k. Normally, $k' = k$, so the length of the step is doubled.

$$\mathbf{S}^{\square}_{k+k',n} \left[\begin{array}{ccc|ccc}
0 & \cdots & 0 & \beta_0^{(n)} & & 0 \\
\hat{\alpha}_0^{(n)} & \ddots & \vdots & \vdots & \ddots & \\
\vdots & \ddots & 0 & \vdots & & \beta_0^{(n)} \\
\vdots & & \hat{\alpha}_0^{(n)} & \beta_{k-1}^{(n)} & & \vdots \\
\hat{\alpha}_{k-1}^{(n)} & & \vdots & 0 & \ddots & \vdots \\
 & \ddots & \vdots & \vdots & \ddots & \beta_{k-1}^{(n)} \\
0 & & \hat{\alpha}_{k-1}^{(n)} & 0 & \cdots & 0 \\
\hline
0 & \cdots & 0 & \alpha_{k-1}^{(n)} & & 0 \\
\hat{\beta}_{k-1}^{(n)} & \ddots & \vdots & \vdots & \ddots & \\
\vdots & \ddots & 0 & \vdots & & \alpha_{k-1}^{(n)} \\
\vdots & & \hat{\beta}_{k-1}^{(n)} & \alpha_0^{(n)} & & \vdots \\
\hat{\beta}_0^{(n)} & & \vdots & 0 & \ddots & \vdots \\
 & \ddots & \vdots & \vdots & \ddots & \alpha_0^{(n)} \\
0 & & \hat{\beta}_0^{(n)} & 0 & \cdots & 0
\end{array}\right]$$

$$= \left[\begin{array}{ccc|ccc}
\hat{\pi}_{0,n+k} & \cdots & \hat{\pi}_{k'-1,n+k} & \varepsilon_{n+k,n+k} & \cdots & \varepsilon_{n+k+k'-1,n+k} \\
 & \ddots & \vdots & & \ddots & \vdots \\
\mathbf{O} & & \hat{\pi}_{0,n+k} & \mathbf{O} & & \varepsilon_{n+k,n+k} \\
\mathbf{0}_{2k} & \cdots & \mathbf{0}_{2k} & \mathbf{0}_{2k} & \cdots & \mathbf{0}_{2k} \\
\hat{\varepsilon}_{n+k,n+k} & & \mathbf{O} & \pi_{0,n+k} & & \mathbf{O} \\
\vdots & \ddots & & \vdots & \ddots & \\
\hat{\varepsilon}_{n+k+k'-1,n+k} & \cdots & \hat{\varepsilon}_{n+k,n+k} & \pi_{k'-1,n+k} & \cdots & \pi_{0,n+k}
\end{array}\right]. \tag{8.1}$$

The matrices on the left have sizes $(2k + 2k') \times (2k + 2k')$ and $(2k + 2k') \times 2k'$, respectively. The nonzero rows of the matrix on the right-hand side just form the matrix $\mathbf{S}^{\square}_{k',n+k}$. Post-multilying (8.1) by new coefficient vectors

$$\begin{aligned}
&[\hat{\alpha}_0^{(n;k)} \cdots \hat{\alpha}_{k'-1}^{(n;k)} \mid \hat{\beta}_{k'-1}^{(n;k)} \cdots \hat{\beta}_0^{(n;k)}]^{\mathrm{T}}, \\
&[\beta_0^{(n;k)} \cdots \beta_{k'-1}^{(n;k)} \mid \alpha_{k'-1}^{(n;k)} \cdots \alpha_0^{(n;k)}]^{\mathrm{T}},
\end{aligned} \tag{8.2}$$

we see that a solution of (5.16) for $k + k'$ can computed by solving a linear system with the matrix $\mathbf{S}^{\square}_{k',n+k}$. Again, we want to fulfill the normalization (5.2), so we need to solve two linear systems with the coefficient matrix $\mathbf{S}^{\square}_{k'-1,n+k}$ of size $2(k'-1) \times 2(k'-1)$, similar to (5.20) and (5.21).

The construction of a superfast solver becomes now clear. Instead of computing the $(n + k + k')$th column of $\mathbf{P}$ (partially) and $\mathbf{E}$ by solving two linear systems with coefficient matrix $\mathbf{S}^{\square}_{k+k'-1,n}$ of size $2(k + k' - 1) \times 2(k + k' - 1)$, we solve four smaller linear systems, two of size $2(k-1) \times 2(k-1)$ and the other two of size $2(k'-1) \times 2(k'-1)$. If this procedure is applied recursively to perform bigger and bigger steps, we obtain an algorithm of complexity $O(N \log^2 N)$, as can be seen by a standard argument; see the discussion of the similar superfast look-ahead Toeplitz solvers in [19].

The implementation of the superfast solver can be done with two recursive procedure, COLDAC and COLDAC2, outlined below, of which the first makes usually one call to itself and one call to COLDAC2, while the second makes two calls to itself. In these calls the order of the problem is roughly halved. The superfast algorithm for $\mathbf{T}_{\overline{N}}$ is started by a call to COLDAC(*true*, $\overline{N}, \overline{N}, \mathbf{T}_{\overline{N}}$).

The procedures COLDAC and COLDAC2 are variations of the procedures SAWDAC and SAWDAC2 from [19]. The aim of COLDAC is to find the minimum well-column-regular index N in the range $\underline{N} \le N \le \overline{N}$. If it does not succeed, the *flag* is set *true*. In this case it may still have found a well-column-regular pair, but only one with index $N < \underline{N}$. The Boolean variable *dac* indicates when the divide and conquer option should be shut off since it is known already that there is no well-column-regular index in the target range. In an analogous way, the aim of COLDAC2 is to find a $(0; N)$ well-column-regular index pair. (Note the change of notation from $(n; k)$ to $(0; N)$.) *flag* and *dac* have the same meaning as in COLDAC. The integer n_{tot} indicates how far we have at this point computed information for evaluating $\hat{\mathbf{q}}_N$ and $\mathbf{q}_N$. In particular, after COLDAC2 has been called from COLDAC, $n_{\text{tot}} = n + k$. The evaluation of $\hat{\mathbf{a}}_N, \hat{\mathbf{b}}_N, \mathbf{a}_N, \mathbf{b}_N$ in step C5 is done by multiplying the second matrix on the left in (8.1) by the vectors (8.2), which in COLDAC2 appear as $\hat{\mathbf{a}}^{(n)}_k, \hat{\mathbf{b}}^{(n)}_k, \mathbf{b}^{(n)}_k, \mathbf{a}^{(n)}_k$.

Another variant of a superfast solver is to apply COLDAC2 alone. Then, the procedure provides us with a $(0; N)$ well-column-regular index pair and vectors $\hat{\mathbf{a}}^{(0)}_N, \hat{\mathbf{b}}^{(0)}_N, \mathbf{b}^{(0)}_N, \mathbf{a}^{(0)}_N$. From these vectors, $\hat{\mathbf{q}}_N$ and $\mathbf{q}_N$ can be computed by means of (5.1) (with $n = 0$ and $k = N$). In the generic case, i.e., in absence of true look-ahead steps, this variant is equivalent to the superfast algorithm proposed by de Hoog [11]. It is known that de Hoog's algorithm can be implemented with only $5N \log^2 N + O(N \log N)$ multiplications and $10N \log^2 N + O(N \log N)$ additions. Our superfast algorithm can be expected to have roughly the same complexity. We will investigate this topic further in a forthcoming paper.

Procedure COLDAC:
$[N, \hat{\mathbf{q}}_N, \mathbf{q}_N, \hat{\mathbf{p}}_N, \mathbf{p}_N, \hat{\mathbf{e}}_N, \mathbf{e}_N, \mathit{flag}]$ = COLDAC(*dac*, $\underline{N}, \overline{N}, \mathbf{T}_{\overline{N}}$);
if (*dac* and $\underline{N} \ge 1$)

A1) $[n, \hat{\mathbf{q}}_n, \mathbf{q}_n, \hat{\mathbf{p}}_n, \mathbf{p}_n, \hat{\mathbf{e}}_n, \mathbf{e}_n, \mathit{flag}]$ = COLDAC(*true*, $\lfloor \underline{N}/2 \rfloor, \overline{N} - 1, \mathbf{T}_{\overline{N}-1}$)
A2) if *flag* and $n = -1$
$[N, \hat{\mathbf{q}}_N, \mathbf{q}_N, \hat{\mathbf{p}}_N, \mathbf{p}_N, \hat{\mathbf{e}}_N, \mathbf{e}_N, \mathit{flag}]$ = COLDAC(*false*, $\underline{N}, \overline{N}, \mathbf{T}_{\overline{N}}$);
return
end if;

A3) $[k, \hat{\mathbf{a}}_k^{(n)}, \hat{\mathbf{b}}_k^{(n)}, \mathbf{a}_k^{(n)}, \mathbf{b}_k^{(n)}, n_{\text{tot}}, \textit{flag}]$
= COLDAC2(*true*, $\underline{N} - n, \overline{N} - n, \hat{\mathbf{p}}_n, \mathbf{p}_n, \hat{\mathbf{e}}_n, \mathbf{e}_n, n$);
$N = n_{\text{tot}}$ (= $n + k$);
if *flag* and $k = 0$
return
end if;

A4) update $\hat{\mathbf{q}}_n, \mathbf{q}_n, \hat{\mathbf{p}}_n, \mathbf{p}_n, \hat{\mathbf{e}}_n, \mathbf{e}_n$ to get $\hat{\mathbf{q}}_N, \mathbf{q}_N, \hat{\mathbf{p}}_N, \mathbf{p}_N, \hat{\mathbf{e}}_N, \mathbf{e}_N$;
return

else

B1) $n = \underline{N}$;
while not ($\mathbf{T}_n$ and $\mathbf{T}_{n+1}$ are well conditioned) and $n < \overline{N}$
$n = n + 1$
end while;
if $column\_regular$ (i.e., $\mathbf{T}_n$ and $\mathbf{T}_{n+1}$ are well conditioned)
$N = n$, *flag* = *false*;
solve (2.2) to obtain a well-column-regular pair $\hat{\mathbf{q}}_N$, $\mathbf{q}_N$;
compute $\hat{\mathbf{p}}_N, \hat{\mathbf{e}}_N, \mathbf{p}_N, \mathbf{e}_N$ from (3.4)
else
$N = -1$, *flag* = *true*;
end if;
return

end if

Procedure COLDAC2:
$[N, \hat{\mathbf{a}}_N, \hat{\mathbf{b}}_N, \mathbf{a}_N, \mathbf{b}_N, n_{\text{tot}}, \textit{flag}]$ = COLDAC2(*dac*, $\underline{N}, \overline{N}, \hat{\mathbf{p}}, \hat{\mathbf{e}}, \mathbf{p}, \mathbf{e}, n_{\text{tot}}$);
if *dac* and $\underline{N} \geq 2$

C1) $[n, \hat{\mathbf{a}}_n, \hat{\mathbf{b}}_n, \mathbf{a}_n, \mathbf{b}_n, n_{\text{tot}}, \textit{flag}]$ = COLDAC2(*true*, $\lfloor \underline{N}/2 \rfloor, \overline{N} - 1, \hat{\mathbf{p}}, \hat{\mathbf{e}}, \mathbf{p}, \mathbf{e}, n_{\text{tot}}$);

C2) if *flag* and $n = 0$
$[N, \hat{\mathbf{a}}_N, \hat{\mathbf{b}}_N, \mathbf{a}_N, \mathbf{b}_N, n_{\text{tot}}, \textit{flag}]$ = COLDAC2(*false*, $\underline{N}, \overline{N}, \hat{\mathbf{p}}, \hat{\mathbf{e}}, \mathbf{p}, \mathbf{e}, n_{\text{tot}}$);
return
end if;

C3) evaluate right-hand side of (8.1) [for $k \leftarrow n$, $k' \leftarrow \overline{N} - n$, $n \leftarrow 0$, $n + k \leftarrow n$]
to get $\hat{\mathbf{p}}^{(n)}, \mathbf{p}^{(n)}, \hat{\mathbf{e}}^{(n)}, \mathbf{e}^{(n)}$;

C4) $[k, \hat{\mathbf{a}}_k^{(n)}, \hat{\mathbf{b}}_k^{(n)}, \mathbf{a}_k^{(n)}, \mathbf{b}_k^{(n)}, n_{\text{tot}}, \textit{flag}]$
= COLDAC2(*true*, $\underline{N} - n, \overline{N} - n, \hat{\mathbf{p}}^{(n)}, \hat{\mathbf{e}}^{(n)}, \mathbf{p}^{(n)}, \mathbf{e}^{(n)}, n_{\text{tot}}$);
if *flag* and $k = 0$
$N = n_{\text{tot}}$;
return
end if;

C5) $N = n + k$, *flag* = *false*;
evaluate $\hat{\mathbf{a}}_N, \hat{\mathbf{b}}_N, \mathbf{a}_N, \mathbf{b}_N$;
return

else

D1) $k = \underline{N}$;
    while not $column\_regular$ and $k < \overline{N}$
        $k = k + 1$
    end while;
    if $column\_regular$
        flag = false, $N = k$, $n_{\text{tot}} = n_{\text{tot}} + k$;
        solve (5.17) to obtain $\hat{\mathbf{a}}_N, \hat{\mathbf{b}}_N, \mathbf{a}_N, \mathbf{b}_N$
    else
        $N = 0$, *flag = true*
    end if;
    return

end if

---

9. The Bareiss-Delosme-Ipsen formulation of the Schur algorithm, and its look-ahead extension

Bareiss [4] rediscovered the Schur algorithm, but presented it in different form. Starting from two copies of the Toeplitz matrix $\mathbf{T} = \mathbf{T}_{N+1}$ he applied a series of two-dimensional transformations which bring one copy of the matrix in upper triangular form and the other in lower triangular form. Delosme and Ipsen [13, 25] analyzed the Bareiss algorithm further and generalized it to arbitrary matrices with the property that all their contiguous principal submatrices are nonsingular. They showed in particular that the resulting triangular matrices are the second factors in the LU and the UL decomposition of $\mathbf{T}$, and that the first factors can be retrieved from the mentioned two-dimensional transformations. Here, we extend this result to the look-ahead case. In contrast to [4, 13, 25] we work with columns instead of rows. In fact, in its original form, the Bareiss algorithm for $\mathbf{T}$ is the same as the Schur algorithm for $\mathbf{T}^{\mathrm{T}}$. We discuss here the Bareiss-Delosme-Ipsen formulation of the Schur algorithm presented in Sect. 3 and extend it to the look-ahead case. This formulation of our look-ahead Schur algorithm is obtained readily, by just recycling formulas of the foregoing sections. It also provides us with a new matrix formulation of our look-ahead Levinson algorithm.

Let $\hat{\mathbf{E}}_n$, $\mathbf{E}_n$, $\hat{\mathbf{R}}_n$ and $\mathbf{R}_n$ be the $(N+1) \times (n+1)$ matrices consisting of the first $n+1$ columns of the respective matrices $\hat{\mathbf{E}}$, $\mathbf{E}$, $\hat{\mathbf{R}}$ and $\mathbf{R}$ defined by (2.16). Clearly, (2.15) holds for these submatrices also:

$$\mathbf{T}\hat{\mathbf{R}}_n = \hat{\mathbf{E}}_n \quad \text{and} \quad \mathbf{R}_n^{\mathrm{T}}\mathbf{T} = \mathbf{E}_n^{\mathrm{T}}, \tag{9.1}$$

Additionally, we introduce the notation

$$\mathbf{E}_n^{\mathrm{J}} := \mathbf{J}_{N+1}\mathbf{E}_n\mathbf{J}_{n+1}, \quad \mathbf{R}_n^{\mathrm{J}} := \mathbf{J}_{N+1}\mathbf{R}_n\mathbf{J}_{n+1}, \tag{9.2}$$

where $\mathbf{J}_n$ still denotes the antidiagonal unit matrix or reflection matrix of order n. One obtains $\mathbf{E}_n^{\mathrm{J}}$ by rotating $\mathbf{E}_n$ by 180°. Hence, $\mathbf{E}_n^{\mathrm{J}}$ is upper triangular, and $\mathbf{R}_n^{\mathrm{J}}$ is lower triangular. The upper index $^{\mathrm{J}}$ should not be confused with the index bound J.

Consider (5.9) in the case $N = n + k$, where $\mathbf{T}_{n+k+1} = \mathbf{T}_{N+1} = \mathbf{T}$ and where the $(N+1) \times (2N - 2n)$ matrix on the right-hand side is

$$\text{(9.3)}\quad \mathbf{S}_n^\circ := \mathbf{S}_{N-n,n}^\circ := \left[\begin{array}{ccc|ccc} \hat\pi_{0,n} & \cdots & \hat\pi_{N-n-1,n} & \varepsilon_{n,n} & \cdots & \varepsilon_{N-1,n} \\ & \ddots & \vdots & & \ddots & \vdots \\ \mathbf{0}_n & & \hat\pi_{0,n} & \mathbf{0}_n & & \varepsilon_{n,n} \\ & \ddots & & & \ddots & \\ \hat\varepsilon_{n,n} & & \mathbf{0}_n & \pi_{0,n} & & \mathbf{0}_n \\ \vdots & \ddots & & \vdots & \ddots & \\ \hat\varepsilon_{N-1,n} & \cdots & \hat\varepsilon_{n,n} & \pi_{N-n-1,n} & \cdots & \pi_{0,n} \end{array}\right].$$

Equation (5.9) becomes

$$\text{(9.4)}\qquad \mathbf{T}\left[\begin{array}{ccc|ccc} \mathcal{S}\hat{\mathbf{q}}_{n,e} & \cdots & \mathcal{S}^k\hat{\mathbf{q}}_{n,e} & \mathbf{q}_{n,e} & \cdots & \mathcal{S}^{k-1}\mathbf{q}_{n,e} \end{array}\right] = \mathbf{S}_n^\circ.$$

This can be combined with (9.1):

$$\text{(9.5)}\qquad \mathbf{T}\left[\begin{array}{c|ccc|ccc|c} \hat{\mathbf{R}}_n & \mathcal{S}\hat{\mathbf{q}}_{n,e} & \cdots & \mathcal{S}^k\hat{\mathbf{q}}_{n,e} & \mathbf{q}_{n,e} & \cdots & \mathcal{S}^{k-1}\mathbf{q}_{n,e} & \mathbf{R}_n^{\mathrm{J}} \end{array}\right] = \left[\begin{array}{c|c|c} \hat{\mathbf{E}}_n & \mathbf{S}_n^\circ & \mathbf{E}_n^{\mathrm{J}} \end{array}\right].$$

Note that on the left-hand side $\mathbf{T}$ is multiplied by a matrix whose left half is a square upper triangular block and whose right half is a square lower triangular block. On the right-hand side, the left half of the matrix is a square block with n zero codiagonals above its main diagonal, while the right half is a square block with n zero codiagonals below its main diagonal. If $\mathbf{I} = \mathbf{I}_{N+1}$ denotes the unit matrix of order $N+1$, then for $n = 0$ this equation reduces to

$$\text{(9.6)}\qquad \mathbf{T}\left[\begin{array}{c|c} \mathbf{I} & \mathbf{I} \end{array}\right] = \left[\begin{array}{c|c} \mathbf{T} & \mathbf{T} \end{array}\right],$$

while for $n = N$ and $\mathbf{J} := \mathbf{J}_{N+1}$ we get again

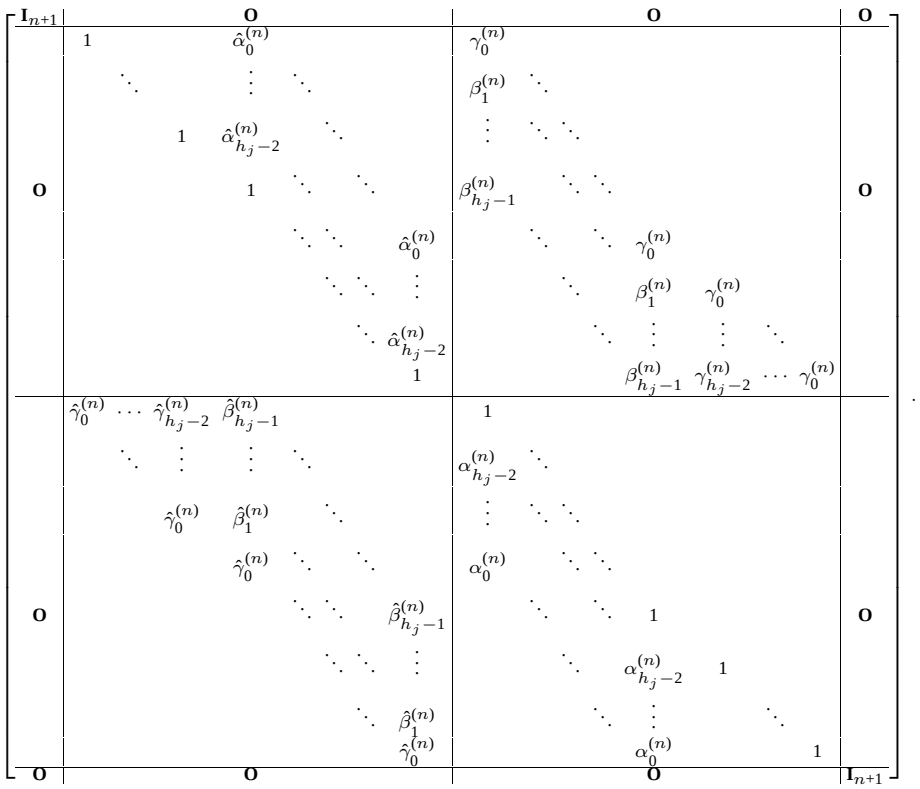
$$\text{(9.7)}\qquad \mathbf{T}\left[\begin{array}{c|c} \hat{\mathbf{R}} & \mathbf{R}^{\mathrm{J}} \end{array}\right] = \left[\begin{array}{c|c} \hat{\mathbf{E}} & \mathbf{E}^{\mathrm{J}} \end{array}\right].$$

Recall that by applying two-dimensional transformations (from the right) the Schur algorithm transforms the right-hand side of (9.6) into the one of (9.7). In contrast, the Levinson algorithm applies the same transformations to the matrices on the left-hand side. The two-dimensional transformations that are applied at the nth step may be bundled in the $(2N+2) \times (2N+2)$ matrix

$$\text{(9.8)}\qquad \mathbf{C}_n := \left[\begin{array}{c|c|c|c} \mathbf{I}_{n+1} & \mathbf{O} & \mathbf{O} & \mathbf{O} \\ \mathbf{O} & \mathbf{I}_{N-n} & \gamma_0^{(n)}\mathbf{I}_{N-n} & \mathbf{O} \\ \mathbf{O} & \hat\gamma_0^{(n)}\mathbf{I}_{N-n} & \mathbf{I}_{N-n} & \mathbf{O} \\ \mathbf{O} & \mathbf{O} & \mathbf{O} & \mathbf{I}_{n+1} \end{array}\right],$$

with the Schur parameters $\hat\gamma_0^{(n)}$ and $\gamma_0^{(n)}$ defined in (2.7). In the case of look-ahead, this matrix becomes more complicated. From (5.1), (5.2), and (5.25) we can conclude that the step from $n = n_j$ to $n_{j+1} = n_j + h_j$ is represented by

(9.9) $\mathbf{C}_{n_j} :=$



This is also correct for $n = n_1 = 0$ if 0 is a well-column-regular index. Recall that for $n = 0$ the matrices $\hat{\mathbf{R}}_{n_1}$ and $\mathbf{R}_{n_1}$ just contain the first unit vector, and $\hat{\mathbf{E}}_{n_1}$ and $\mathbf{E}^{\mathrm{J}}_{n_1}$ consist of the first and last column of $\mathbf{T}$, respectively, so that (9.5) reduces to (9.6). In contrast, if $n_1 > 0$, the first n_1 columns of $\hat{\mathbf{R}}_{n_1}$ and $\mathbf{R}_{n_1}$ are the the first n_1 unit vector, since these are the inner vectors of the 0th block, see (4.2). The last columns contain the first well-column-regular pair. The matrices $\hat{\mathbf{E}}_{n_1}$ and $\mathbf{E}^{\mathrm{J}}_{n_1}$ coincide in their first and last n_1 columns, respectively, with $\mathbf{T}$.

Therefore, when $n_1 > 0$ we still need a transformation matrix $\mathbf{C}_{-1}$ such that

$$[\ \mathbf{T}\ |\ \mathbf{T}\ ]\,\mathbf{C}_{-1} = [\ \hat{\mathbf{E}}_{n_1}\ |\ \mathbf{S}^{\circ}_{n_1}\ |\ \mathbf{E}^{\mathrm{J}}_{n_1}\ ]. \tag{9.10}$$

It follows that $\mathbf{C}_{-1}$ is of the form (9.9) with $j = 0$, $n = n_0 := -1$, $h_0 := n_1 + 1$, and

$$\hat{\alpha}_i^{(-1)} = \hat{\rho}_{i,n_1}, \quad \alpha_i^{(-1)} = \rho_{i,n_1}, \quad i = 0, \ldots, n_1, \tag{9.11}$$

$$\hat{\beta}_i^{(-1)} = \beta_i^{(-1)} = 0, \quad i = 0, \ldots, n_1. \tag{9.12}$$

If $n_1 = 0$, we may set $n_0 := -1$, $\mathbf{C}_{-1} := \mathbf{I}_{2N+2}$.

With this notation the transformations implemented in the look-ahead Schur algorithm can be summarized in matrix form as

$$[\ \mathbf{T}\ |\ \mathbf{T}\ ] \prod_{j=0}^{J-1} \mathbf{C}_{n_j} = [\ \hat{\mathbf{E}}\ |\ \mathbf{E}^{\mathrm{J}}\ ]. \tag{9.13}$$

Likewise, the look-ahead Levinson algorithm becomes

$$[\ \mathbf{I}\ |\ \mathbf{I}\ ] \prod_{j=0}^{J-1} \mathbf{C}_{n_j} = [\ \hat{\mathbf{R}}\ |\ \mathbf{R}^{\mathrm{J}}\ ]. \tag{9.14}$$

At the intermediate stages we have, with $n = n_l$,

$$[\ \mathbf{T}\ |\ \mathbf{T}\ ] \prod_{j=0}^{l-1} \mathbf{C}_{n_j} = \Big[\ \underbrace{\hat{\mathbf{E}}_n}_{n+1}\ \Big|\ \underbrace{\mathbf{S}_n^{\circ}}_{2N-2n}\ \Big|\ \underbrace{\mathbf{E}_n^{\mathrm{J}}}_{n+1}\ \Big] \tag{9.15}$$

and

$$\begin{aligned}&[\ \mathbf{I}\ |\ \mathbf{I}\ ] \prod_{j=0}^{l-1} \mathbf{C}_{n_j}\\ &= \Big[\ \underbrace{\hat{\mathbf{R}}_n}_{n+1}\ \Big|\ \underbrace{\mathcal{S}\hat{\mathbf{q}}_{n,e}\ \cdots\ \mathcal{S}^{N-n}\hat{\mathbf{q}}_{n,e}}_{N-n}\ \Big|\ \underbrace{\mathbf{q}_{n,e}\ \cdots\ \mathcal{S}^{N-n-1}\mathbf{q}_{n,e}}_{N-n}\ \Big|\ \underbrace{\mathbf{R}_n^{\mathrm{J}}}_{n+1}\ \Big].\end{aligned} \tag{9.16}$$

While in normal steps $\hat{\mathbf{E}}$, $\mathbf{E}$, $\hat{\mathbf{R}}$, and $\mathbf{R}$ grow by one column and thus $\mathbf{S}_n^{\circ}$ and $[\mathcal{S}\hat{\mathbf{q}}_{n,e}\ \cdots\ \mathcal{S}^{N-n}\hat{\mathbf{q}}_{n,e}\ |\ \mathbf{q}_{n,e}\ \cdots\ \mathcal{S}^{N-n-1}\mathbf{q}_{n,e}]$ shrink by two columns, in the look-ahead steps several columns are added to $\hat{\mathbf{E}}$, $\mathbf{E}$, $\hat{\mathbf{R}}$, and $\mathbf{R}$ at once.

Let us consider the product $\prod \mathbf{C}_{n_j}$ more closely. According to (9.9) the nontrivial "inner part" of each factor consists of two upper triangular blocks on the left and two lower triangular blocks on the right. Each of these blocks can be divided into a unit matrix or square triangular matrix of order $(h_j - 1)$, an $(N - n_j - h_j + 1) \times (h_j - 1)$ zero matrix, and an $(N - n_j) \times (N - n_j - h_j + 1)$ Toeplitz matrix. In the generic case, where $h_j = 1$, the first two subblocks are vacuous and the Toeplitz subblocks become diagonal matrices of order $N - n_j$, see (9.8). Hence, the unit or triangular subblock is solely due to the existence of inner vectors. By choosing a particular set of these not uniquely defined inner vectors, namely a set generated by recurrences satisfying (5.25), we obtained the unit subblocks. Using another set of inner vectors would amount to multiplying $\mathbf{C}_{n_j}$ from the right by a block diagonal matrix with unit diagonal, whose only nontrivial blocks are a unit upper triangular one of order $h_j - 1$ and a unit lower triangular block of the same order. These two blocks are positioned so that they react with the unit subblocks and the triangular subblocks of this order in $\mathbf{C}_{n_j}$. The unit subblocks then also become triangular. By this block diagonal "scaling", the diagonal blocks in $\mathbf{C}_{n_j}$ become also triangular.

Multiplying several successive matrices $\mathbf{C}_{n_j}$ together creates a matrix of the same type, but with h_j equal to the sum of the individual ones and with the unit subblocks of size $h_j - 1$ again replaced by triangular subblocks. This could be verified algebraically, but it follows directly from the fact that we did not assume in our theory that the step size h_j is chosen as small as possible: we can leave out several column-regular pairs, and indeed we eventually do that when searching for a well-column-regular pair. Hence, it is no surprise that the structure of a product of matrices $\mathbf{C}_{n_j}$ is essentially

the same as the one of $\mathbf{C}_{n_j}$ with large h_j. Moreover, the full product $\prod \mathbf{C}_{n_j}$ appearing in (9.13)-(9.16) has the form

$$\prod_{j=0}^{J-1} \mathbf{C}_{n_j} = \left[\begin{array}{c|c} \hat{\mathbf{A}} & \mathbf{B}^{\mathrm{J}} \\ \hline \hat{\mathbf{B}} & \mathbf{A}^{\mathrm{J}} \end{array}\right], \tag{9.17}$$

where $\hat{\mathbf{A}}$ and $\mathbf{A}$ are unit upper triangular, and $\hat{\mathbf{B}}$ and $\mathbf{B}$ are upper triangular, respectively, with diagonal elements $\hat{\gamma}_0^{(n)}$ and $\gamma_0^{(n)}$. (For the generic case, the form of such a matrix product was also investigated by Delosme and Ipsen [13], but in their treatment the factors come in reverse order and the two blocks $\mathbf{I}_{n+1}$ in (9.8) appear in the center of the matrix.) If $n_1 = 0$, one can show that the first column and row of (9.17) contain just the first unit vector and its transposed, while the last column and row contain the last unit vector and its transposed.

The columns of the matrices $\hat{\mathbf{A}}$, $\hat{\mathbf{B}}$, $\mathbf{A}$, and $\mathbf{B}$ can be seen to contain the coefficients of what Ammar and Gragg [1] call Schur polynomials, here generalized to the non-Hermitian case, where there are four sequences of these polynomials instead of two. From (9.14) and (9.17) we see that these four sets of Schur polynomials are closely related to the two sets of Szegö polynomials, whose coefficients are stored in the columns of $\hat{\mathbf{R}}$ and $\mathbf{R}$. Essentially, the addition of two Schur polynomials yields a Szegö polynomial. More exactly, one of the two polynomials must be divided by its variable ζ before adding, and the complex conjugate of the sum is then the Szegö polynomial.

There is also an obvious equivalence between the product $\prod \mathbf{C}_{n_j}$ and the product of 2×2 matrices with polynomial entries which appears in the treatment of Padé recurrences given in [22].

The superfast look-ahead Schur algorithm that we discussed in Sect. 8 can be understood as an evaluation of $\prod_{j=0}^{J-1} \mathbf{C}_{n_j}$ by building up a binary tree.

We can inhance our understanding of the Schur-Bareiss algorithm and its relationship to the nonsymmetric generalization of the original Schur algorithm involving the Schur polynomials [31, 1] by making a minor modification in the derivation given above The right-hand side of (9.5) can be written as

$$[\ \hat{\mathbf{E}}_{n_1} \mid \mathbf{S}^{\circ}_{n_1} \mid \mathbf{E}^{\mathrm{J}}_{n_1}\ ] = [\ \mathbf{I} \mid \mathbf{I}\ ] \left[\begin{array}{c|c|c} \mathbf{O} & \mathbf{S}^{\circ\circ}_{n_1} & \mathbf{E}^{\mathrm{J}}_{n_1} \\ \hat{\mathbf{E}}_{n_1} & & \mathbf{O} \end{array}\right], \tag{9.18}$$

where, more generally,

$$\mathbf{S}^{\circ\circ}_n := \left[\begin{array}{ccc|ccc} \hat{\pi}_{0,n} & \cdots & \hat{\pi}_{N-n-1,n} & \varepsilon_{n,n} & \cdots & \varepsilon_{N-1,n} \\ & \ddots & \vdots & & \ddots & \vdots \\ \mathbf{O} & & \hat{\pi}_{0,n} & \mathbf{O} & & \varepsilon_{n,n} \\ \hline \mathbf{0}_{2n+2} & \cdots & \mathbf{0}_{2n+2} & \mathbf{0}_{2n+2} & \cdots & \mathbf{0}_{2n+2} \\ \hline \hat{\varepsilon}_{n,n} & & \mathbf{O} & \pi_{0,n} & & \mathbf{O} \\ \vdots & \ddots & & \vdots & \ddots & \\ \hat{\varepsilon}_{N-1,n} & \cdots & \hat{\varepsilon}_{n,n} & \pi_{N-n-1,n} & \cdots & \pi_{0,n} \end{array}\right] \tag{9.19}$$

contains the same elements as $\mathbf{S}^{\square}_{N-n,n}$, but has an additional $(2n + 2) \times (2N - 2n)$ zero block in the middle. If $n_1 = 0$, the new matrix on the right-hand side of (9.18) just contains the elements of $\mathbf{T}$, except that the upper and the lower triangular parts

are separated, the diagonal being included in the lower triangular part at left and in the upper triangular part at right:

$$(9.20)\quad \left[\begin{array}{c|c|c} \mathbf{0}_{N+1} \\ \hat{\mathbf{E}}_0 \end{array}\middle|\ \mathbf{S}_0^{\circ\circ}\ \middle|\begin{array}{c} \mathbf{E}_0^{\mathrm{J}} \\ \mathbf{0}_{N+1} \end{array}\right] = \left[\begin{array}{c|ccc|ccc|c} 0 & \mu_{-1} & \cdots & \mu_{-N} & \mu_0 & \cdots & \mu_{-N+1} & \mu_{-N} \\ \vdots & & \ddots & \vdots & & \ddots & \vdots & \vdots \\ 0 & \mathbf{O} & & \mu_{-1} & \mathbf{O} & & \mu_0 & \mu_{-1} \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & \mu_0 \\ \hline \mu_0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 \\ \mu_1 & \mu_0 & & \mathbf{O} & \mu_1 & & \mathbf{O} & 0 \\ \vdots & \vdots & \ddots & & \vdots & \ddots & \vdots & \\ \mu_N & \mu_{N-1} & \cdots & \mu_0 & \mu_N & \cdots & \mu_1 & 0 \end{array}\right].$$

It is easy to see from our derivation that we may understand our look-ahead Schur algorithm as successive multiplication of this matrix by $\mathbf{C}_{n_1} \cdots \mathbf{C}_{n_{J-1}}$. The step from index n_j to n_{j+1} is given by

$$(9.21)\qquad \left[\begin{array}{c|c|c} \mathbf{O} \\ \hat{\mathbf{E}}_{n_j} \end{array}\middle|\ \mathbf{S}_{n_j}^{\circ\circ}\ \middle|\begin{array}{c} \mathbf{E}_{n_j}^{\mathrm{J}} \\ \mathbf{O} \end{array}\right] \mathbf{C}_{n_j} = \left[\begin{array}{c|c|c} \mathbf{O} \\ \hat{\mathbf{E}}_{n_{j+1}} \end{array}\middle|\ \mathbf{S}_{n_{j+1}}^{\circ\circ}\ \middle|\begin{array}{c} \mathbf{E}_{n_{j+1}}^{\mathrm{J}} \\ \mathbf{O} \end{array}\right],$$

and the whole process can be summarized as

$$(9.22)\qquad \left[\begin{array}{c|c|c} \mathbf{O} \\ \hat{\mathbf{E}}_{n_1} \end{array}\middle|\ \mathbf{S}_{n_1}^{\circ\circ}\ \middle|\begin{array}{c} \mathbf{E}_{n_1}^{\mathrm{J}} \\ \mathbf{O} \end{array}\right] \prod_{j=1}^{J-1} \mathbf{C}_{n_j} = \left[\begin{array}{c|c} \mathbf{O} & \hat{\mathbf{E}} \\ \mathbf{E}^{\mathrm{J}} & \mathbf{O} \end{array}\right].$$

Recall that $\hat{\mathbf{E}}$ and $\mathbf{E}^{\mathrm{J}}$ are lower triangular and nonsingular. Therefore, the matrix on the right-hand side is nonsingular, and hence, all matrices on the left-hand side too. For stability, they all need to be well conditioned.

If $n_1 = 0$, we can return from (9.22) to (9.13) by multiplying (9.22) from the left by $[\ \mathbf{I}\ |\ \mathbf{I}\ ]$. If $n_1 > 0$, we have additionally to insert (9.10) into the resulting relation.

10. Options for solving Toeplitz systems

We have seen that our look-ahead Levinson and Schur algorithms yield a block LDU decomposition of $\mathbf{T}^{-1}$ and $\mathbf{T}$, respectively, at least if we compute additionally the respective inner vectors and the block diagonal matrix $\mathbf{D}$ and its inverse. Clearly, when one of these LDU decomposition is known, it is easy to solve a Toeplitz system

$$(10.1)\qquad \mathbf{Tx} = \mathbf{c}$$

in roughly N^2 additions and multiplications. However, this is just one way of applying the outcome of our algorithms to this task. There are several other options, which we want to compare here with these two.

10.1. Inverse block LDU decomposition

The well-column-regular columns $\hat{\mathbf{q}}_{n_j}$ and $\mathbf{q}_{n_j}$ of $\hat{\mathbf{R}}$ and $\mathbf{R}$ in the LDU decomposition

$$\mathbf{T}^{-1} = \hat{\mathbf{R}}\mathbf{D}^{-1}\mathbf{R}^{\mathrm{T}} \tag{10.2}$$

of $\mathbf{T}^{-1}$ are directly generated by our look-ahead Levinson algorithm. They can also be generated as a byproduct of the look-ahead Schur algorithm, but, unless these columns are used anyway to control the look-ahead step size, this increases the costs by about 50%, namely for applying the Levinson recursion (without computation of the inner products), which is equivalent to applying (9.14). In any case, for the full decomposition we need the inner vectors (i.e., the columns of $\hat{\mathbf{R}}$ and $\mathbf{R}$ whose index is not well-column-regular) and the block diagonal matrix $\mathbf{D}$, whose computation has been discussed in Sect. 7. The multiplication $\mathbf{T}^{-1}\mathbf{c} = \hat{\mathbf{R}}\mathbf{D}^{-1}\mathbf{R}^{\mathrm{T}}\mathbf{c}$ finally requires $N^2 + O(N)$ additions and multiplications each. (Again, we neglect work of order $O(k^2)$ or $O(k^3)$.) Note that it is not necessary to store the whole matrices $\hat{\mathbf{R}}$ and $\mathbf{R}$, since it is possible to update the solutions from step to step. This requires storage for the vectors of one block only. As a byproduct of this updating procedure, we obtain the solutions of all the subsystems

$$\mathbf{T}_n\mathbf{x}_n = \mathbf{c}_n \in \mathbb{C}^n, \quad n = n_j \quad \text{or} \quad n = n_j + 1 \quad (j = 1, \ldots, J-1), \tag{10.3}$$

where $\mathbf{c}_n$ contains the first n components of $\mathbf{c}$; see [14] for details.

10.2. Block LDU decomposition

The well-column-regular columns of $\hat{\mathbf{E}}$ and $\mathbf{E}$ in the LDU decomposition

$$\mathbf{T} = \hat{\mathbf{E}}\mathbf{D}^{-1}\mathbf{E}^{\mathrm{T}} \tag{10.4}$$

of $\mathbf{T}$ are the primary output of our $O(N^2)$ look-ahead Schur algorithm. Again, the full decomposition requires to compute any inner columns and the block diagonal matrix $\mathbf{D}$. The inverse of the latter is not needed for solving (10.1).

When using the LDU decomposition (10.4) for computing the solution of the linear system (10.1) one can perform the forward elimination $\hat{\mathbf{E}}\mathbf{y} = \mathbf{c}$ with $O(N)$ storage, see Sect. 10.5 below. But one still has to store the complete triangular part of $\mathbf{E}$ for the backsubstitution. However, for solving (10.1) with the Schur algorithm there are also more efficient ways that we will discuss next.

10.3. Gohberg-Semencul formulas

Here we just need the last well-column-regular pair, i.e., $\hat{\mathbf{q}}_N$, $\mathbf{q}_N$, and $\varepsilon_{N,N}$ if $\mathbf{T}_{N-1}$ and $\mathbf{T}_N$ are well conditioned, and $\hat{\mathbf{q}}_{N+1}$, $\mathbf{q}_{N+1}$, $\varepsilon_{N+1,N+1}$ if $\mathbf{T}_N$ and $\mathbf{T}_{N+1}$ are well conditioned, so that we can apply the second Gohberg-Semencul formula. The look-ahead Levinson algorithm produces this last pair directly, while, as mentioned above, in the look-ahead Schur algorithm some extra work is required. However, in the superfast version we only need to make at most $\lceil \log_2(N+1) \rceil$ updates (5.1) of well-column-regular pairs, and the FFT can be used to do them. (In the generic case with $N = 2^L - 1$, one computes only $\hat{\mathbf{q}}_{2^\ell-1}$, $\mathbf{q}_{2^\ell-1}$, $\ell = 1, \ldots, L$; in general, at most as many pairs are computed.) Hence, the extra cost to produce these pairs is very small.

In the notation of Sect. 9 the computation of the last columns $\hat{\mathbf{q}}_N$ and $\mathbf{q}_N$ of $\hat{\mathbf{R}}$ and $\mathbf{R}$, respectively, can be expressed as 5pt3pt

$$\begin{aligned} [\ \hat{\mathbf{q}}_N \mid \mathbf{J}_{N+1}\mathbf{q}_N\ ] &= [\ \hat{\mathbf{R}} \mid \mathbf{R}^{\mathrm{J}}\ ] \left[\begin{array}{c|c} \mathbf{e}_N & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{e}_0 \end{array}\right] \\ &= [\ \mathbf{I} \mid \mathbf{I}\ ] \prod_{j=1}^{J-1} \mathbf{C}_{n_j} \left[\begin{array}{c|c} \mathbf{e}_N & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{e}_0 \end{array}\right]. \end{aligned} \tag{10.5}$$

At the end of the superfast Schur algorithm, the product is available in the form of a similar product with only at most $\lceil \log_2(N+1) \rceil$ terms.

Alternatively, one could produce the last columns of the upper triangular matrices $\hat{\mathbf{A}}$, $\mathbf{A}$, $\hat{\mathbf{B}}$, and $\mathbf{B}$ that appear in (9.17) (i.e., the last Schur polynomials) as in [1], but since these are four $(N+1)$-vectors instead of two, this costs twice as much.

10.4. Transformation of the right-hand side according to Bareiss

We pointed out in Sect. 9 that the original Bareiss algorithm is equivalent to the Schur algorithm (without look-ahead) applied to $\mathbf{T}^{\mathrm{T}}$. If we apply our look-ahead Schur algorithm to $\mathbf{T}^{\mathrm{T}}$, both sides of (9.13) have to be transposed, so that the product in (9.13) is on the left of the two Toeplitz matrices $\mathbf{T}$ that are then stacked on top of each other. Therefore, as observed by Bareiss already, this product can also be applied to a vector consisting of two copies of the right-hand side $\mathbf{c}$ of (10.1) stacked on top of each other. This is analogous to transforming the right-hand side in the Gauss algorithm, but costs (without look-ahead) $N^2 + O(N)$ multiplications and additions (each), while Gauss eliminations costs only $N/2 + O(N)$. Here, however, since one copy of $\mathbf{c}$ gets implicitly multiplied by the inverse of the L factor and the other by the inverse of the U factor, the backsubstitution can be applied to two systems of size $\frac{1}{2}(N+1)$ instead of one system of size $(N+1)$; this reduces the costs of backsubstitution by a factor of 4; see [4, 13, 25]. We could capitalize upon this idea by applying our look-ahead algorithm to $\mathbf{T}^{\mathrm{T}}$ and simultaneously transforming the right-hand side of

$$\mathbf{x}^{\mathrm{T}}[\ \mathbf{T}^{\mathrm{T}} \mid \mathbf{T}^{\mathrm{T}}\ ] = [\ \mathbf{c}^{\mathrm{T}} \mid \mathbf{c}^{\mathrm{T}}\ ] \tag{10.6}$$

by post-multiplication with the factors $\mathbf{C}_{n_j}$.

To reduce storage to $O(N)$, Brent and Luk [6] replaced the backsubstitution by a "reverse Bareiss algorithm" applied to the right-hand side. It is based on the fact that the inverses of the matrices $\mathbf{C}_n$ of (9.8) are known and have the same structure. But this applies only to the case of a strongly regular Toeplitz matrix, not to look-ahead.

10.5. Another approach complementing the Schur algorithm

Since $\mathbf{T} = \hat{\mathbf{E}}\hat{\mathbf{R}}^{-1}$, we can write $\mathbf{Tx} = \mathbf{c}$ as

$$\hat{\mathbf{E}}\mathbf{y} = \mathbf{c}, \quad \text{where } \mathbf{x} = \hat{\mathbf{R}}\mathbf{y}. \tag{10.7}$$

Using SAXPYs, the first, lower triangular system for $\mathbf{y}$ can be solved by forward substitution without storing $\hat{\mathbf{E}}$. Moreover, it is sufficient to compute the inner vectors

of $\hat{\mathbf{E}}$; there is no need for computing the inner vectors of $\mathbf{E}$. We assume that $\hat{\mathbf{R}}$ has not been computed. Then, using (9.14), we find $\mathbf{x}$ according to

$$\mathbf{x} = \left[\, \hat{\mathbf{R}} \;\middle|\; \mathbf{R}^{\mathrm{J}} \,\right] \left[\begin{array}{c} \mathbf{y} \\ \hline \mathbf{0} \end{array}\right] = \left[\, \mathbf{I} \;\middle|\; \mathbf{I} \,\right] \prod_{j=1}^{J-1} \mathbf{C}_{n_j} \left[\begin{array}{c} \mathbf{y} \\ \hline \mathbf{0} \end{array}\right] . \tag{10.8}$$

Note the similarity between (10.8) and (10.5). There, however, the vectors $\mathbf{e}_N$ and $\mathbf{e}_0$ simplify the multiplication: only the Toeplitz parts of the matrices $\mathbf{C}_{n_j}$ come into action, even if some of these matrices are multiplied together, as is the case in the superfast Schur algorithm. Therefore fast convolutions can be applied, which, for large N, make the evaluation of (10.5) much cheaper than the one of (10.8).

Recall that here we work with the Schur recursions applied to $\mathbf{T}$, not $\mathbf{T}^{\mathrm{T}}$. For the Bareiss algorithm applied to a symmetric positive definite $\mathbf{T}$, Delosme and Ipsen [12, 25] derived a formula that is similar to (10.8). They made use of the fact that in this case the matrices $\mathbf{C}_n$ are hyperbolic rotations, and thus the inverse of their product is equal to a diagonally scaled transposed of the product. Thus, both the forward elimination and the backsubstitution can be performed by applying a product of hyperbolic rotations.

11. Operation counts

Before we give a detailed analysis of the work and overhead of our fast look-ahead algorithms, recall that in step n, the classical Levinson algorithm requires two inner products and two SAXPYs of length n, while the classical Schur algorithm requires four SAXPYs of length $N - n$. For the solution of the linear system, further work is necessary. Since we have several options for solving the linear system, we will not include the operation count for the solution step in our tables.

Compared to the classical algorithms, the look-ahead algorithms involve some overhead if blocks of size $k > 1$ occur. However, in the following, we assume k to be bounded (independent of N) and $k \ll N$, which justifies to neglect any contributions of at most $O(k^3)$ operations.

In Table 1 we summarize the operation counts for performing a block step of size $k \geq 1$ of our fast look-ahead algorithms (Algorithms 3 and 4). Note that inner pairs do not occur if $k = 1$, and that in this case the algorithms reduce to the classical versions, except for the tests of well-column-regularity (i.e., N comparisons). In the presence of look-ahead steps, the computation of inner pairs is optional.

Table 1. Operation counts for our look-ahead Levinson and our look-ahead Schur algorithm

Operations in a block of size k	look-ahead Levinson		look-ahead Schur	
	regular pair	inner pair	regular pair	inner pair
inner products/step	$2k$	–	–	–
SAXPYs/step	$2(2k-1)$	2	$4(2k-1)$	2

Next, in Table 2, we list the overhead of our look-ahead algorithms and compare it to that of the look-ahead Levinson algorithms proposed by Chan and Hansen [9] and by Freund and Zha [14]. The overhead of the Chan-Hansen algorithm was extracted from (37) and (38) (for $p > 1$) in [9] by subtracting the work for the solution step, and the overhead of the Freund-Zha algorithm was taken from Table 1 in [14]. Recall that

Table 2. Overhead of different look-ahead algorithms

Overhead in a block of size $k \geq 2$	Chan/Hansen (Levinson)	Freund/Zha (Levinson)	Gutkn./Hochb. (Levinson)	Gutkn./Hochb. (Schur)
Block size	k	k	$k+1$	$k+1$
inner products	$\frac{1}{2}k^2 + \frac{1}{2}k$	–	–	–
SAXPYs	$4(k-1)$	$4k$	$2k$	$4k$

our algorithms are based on column-regular pairs, and therefore, a step of size $k+1$ in our algorithms corresponds normally to a step of size one and a step of size $k \geq 1$ of the other two algorithms. To give a fair comparison, we took this into account by inserting $k+1$ as the step size of our algorithms. Nevertheless, the overhead of our look-ahead Levinson algorithm is only half of the overhead in the Freund-Zha algorithm if we choose not to compute inner vectors, in which case the only option to obtain the solution of the Toeplitz system is to apply the Gohberg-Semencul formula. In contrast, the Freund-Zha algorithm requires to compute the inner vectors too, the benefit being that it always yields an inverse block LDU decomposition. If we compute this decomposition too, then the overhead of both algorithms is the same, while the overhead of the algorithm proposed by Chan and Hansen is considerably bigger. The overhead of the latter algorithm reduces to N^2 comparisons if no look-ahead steps are needed, the overhead of the other algorithms amounts to only N comparisons.

For Levinson's algorithm, the two options for solving the linear system are applying an inversion formula or using the inverse block LDU decomposition as outlined in Sect. 10.1. The total work adds up to $2N^2 + O(N \log N) + O(N)$ additions and multiplications with the Gohberg-Semencul formula and to $3N^2 + O(N)$ with the inverse block LDU decomposition. In both cases, iterative refinement can be used to improve the accuracy.

For Schur's algorithm, we have several options for solving the linear system, and they have an influence on what needs to be computed before. We will briefly discuss the amount of work necessary for the different choices. First, a look-ahead step to proceed from a well-column-regular index n to a well-column-regular index $n+k$, costs

A) $4(2k-1)$ SAXPYs of length $N-n$ to compute the $(n+k)$th column of $\hat{\mathbf{E}}, \mathbf{E}, \hat{\mathbf{P}}, \mathbf{P}$;
B) $2(2k-1)$ SAXPYs of length n to compute the column-regular pair $\hat{\mathbf{q}}_{n+k}, \mathbf{q}_{n+k}$;
C) $2(k-1)$ SAXPYs of length $N-n$ to compute the inner columns of this block in $\hat{\mathbf{E}}$ and $\mathbf{E}$;
D) $k-1$ SAXPYs of length $N-n$ to compute the inner columns of this block in $\hat{\mathbf{E}}$ *or* $\mathbf{E}$;
E) $k-1$ SAXPYs of length n to compute the inner vectors $\hat{\mathbf{q}}_{n+j}$ *or* $\mathbf{q}_{n+j}$, $j = 1, \ldots, k-1$.

In the classical Schur algorithm, only A) is needed to find the LDU decomposition of $\mathbf{T}$. Then, k steps starting from step n require the computation of $4k$ SAXPYs of length $N-n$. One may choose to add B), and then k steps require additional $2k$ SAXPYs of length $n+k$. From these figures the overhead for the different options is easily calculated.

For solving the linear system, the following are the most efficient choices complementing the Schur algorithm:

A)+B) Apply the Gohberg-Semencul inversion formula. Then the solution step costs $O(N \log N)$ operations and $O(N)$ storage. The total work for the system is therefore $3N^2 + O(N \log N) + O(N)$ additions and multiplications.

A)+C) Use the LDU factorization as described in Sect. 10.2. This requires $N^2 + O(N)$ additions and multiplications, but $O(N^2)$ storage. In total, the whole solver costs $3N^2 + O(N)$ additions and multiplications.

A)+D) Apply the procedure proposed in Sect. 10.5. For this procedure, in D), it is sufficient to compute only inner columns of $\hat{\mathbf{E}}$. Here, the solution step costs $3/2N^2 + O(N)$ additions and multiplications and $O(N)$ storage. The total work is $7/2N^2 + O(N)$ additions and multiplications.

To summarize, if $O(N^2)$ storage is available, it is cheapest (in terms of computational work) to perform A)+C). The advantages of A)+B) are that only $O(N)$ storage is required and that one can apply iterative refinement (at the cost of only $O(N \log N)$ operations) to improve the accuracy of the solution. This combination is particularly recommended if we have to solve several linear systems with the same Toeplitz matrix but different right-hand sides. In case we wish to solve only a single system, A)+D) usually gives the best accuracy, but since the last regular pair $\hat{\mathbf{q}}_N$, $\mathbf{q}_N$ is not available, further improvement of the solution is expensive (see Sect. 10.3 for a possibility to compute the last pair).

12. Numerical examples

In all our examples we chose the right-hand side of the Toeplitz systems such that $x_{\text{exact}} = [1 \ \cdots \ 1]^{\mathrm{T}}$ is the exact solution. We compared the various Toeplitz solvers on the basis of relative errors

$$\frac{\|x_{\text{exact}} - x_{\text{computed}}\|}{\|x_{\text{exact}}\|}.$$

Recall that the size of a look-ahead step depends on the two inequalities (5.4) and (5.5). For the latter, we chose the constant tolerance function $\text{tol}(n) = 10^{-5}$. The first condition was checked implicitly by keeping the smallest singular value of the matrices $\mathbf{S}^{\square}_{k-1,n}$ under control. To be more precise, after running a large number of examples, it turned out that checking the condition $\sigma_{\min}(\mathbf{S}^{\square}_{k-1,n}) \geq 0.1$ or 0.01 $(= \text{tol}_\sigma)$ was sufficient for obtaining at the same time small look-ahead steps and good accuracy. In all our examples, we used 0.1. However, since in practice one must normally provide storage before starting the algorithm, one has to choose a maximum look-ahead steps size $k_{\max}$, which was equal to 4 in our examples. Then, it may happen that one of the conditions is not fulfilled for all $k = 1, \ldots, k_{\max}$. In most of these cases, this is not caused by all the corresponding principal submatrices being severely ill conditioned, but instead means that the tolerance is too big. Therefore, we implemented an adaptive procedure that generates a monotonically decreasing sequence of lower bounds for the smallest singular value of $\mathbf{S}^{\square}_{k-1,n}$. A similar strategy was originally proposed in [9] and also used in [14].

All our examples have been computed with MATLAB, with a machine precision of order 10^{-16}. In particular, MATLAB allowed us to implement the recursive procedures COLDAC and COLDAC2 of Sect. 8. Both sets of examples have also been used by Chan and Hansen [9] and by Freund and Zha [14].

For the first test, we constructed a set of 100 non-Hermitian matrices of size 64×64 with random off-diagonal entries in $[-1, 1]$. The diagonal was computed

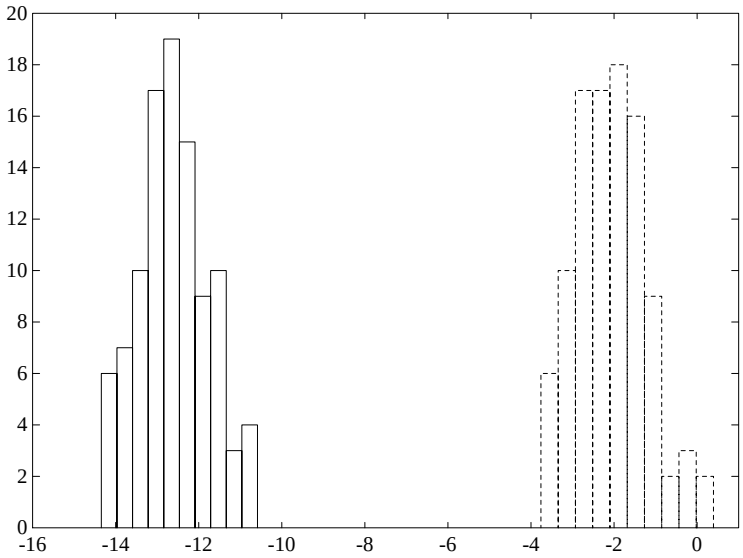


Fig. 1. Histogram of relative errors for the classical (dashed line) and our look-ahead Levinson algorithm

such that at least one principal submatrix was ill conditioned. In Fig. 1 we present the relative errors of the classical (dashed line) and the look-ahead Levinson algorithm (solid line) in the form of a histogram. For both variants, we computed the solution by using the LDU decomposition of $\mathbf{T}^{-1}$, as described in Sect. 10.1. The performance of the classical algorithm is very poor, while the look-ahead variant gives nearly full accuracy. Thus, look-ahead is essential in the presence of ill-conditioned submatrices.

Next, we compare different options for solving these Toeplitz systems with the Levinson, the Schur, and the superfast algorithm with look-ahead. Figure 2 shows the relative error of our look-ahead Levinson algorithm for each of the 100 test matrices. The dotted line represents the relative errors of the solutions computed with a Gohberg-Semencul formula, and the solid line shows the errors when the solutions are updated from step to step using the inverse LDU decomposition (10.2). The figure indicates that the latter yields better accuracy. To obtain the dashed line, we applied one step of iterative refinement with a Gohberg-Semencul formula to the solutions represented by the solid line.

Figure 3 shows the results for our look-ahead Schur algorithm. Again, the dotted line represents the relative errors obtained when applying a Gohberg-Semencul formula. Recall that this requires the computation of the column-regular pairs $\hat{\mathbf{q}}_{n_j}$, $\mathbf{q}_{n_j}$, see the discussion in Sect. 10.1. For the solid line we computed the solutions with the approach described in Sect. 10.5. Again, this leads to slightly more accurate solutions than the application of the inversion formula. However, when using this approach we cannot apply iterative refinement, since the last column-regular pair is not available. Therefore, here, the dashed line shows the relative errors when one step of iterative refinement was applied to the solution computed with a Gohberg-Semencul formula, and not to the one obtained according to Sect. 10.5. Note that from (10.5), we could still compute the last column-regular pair at extra cost.

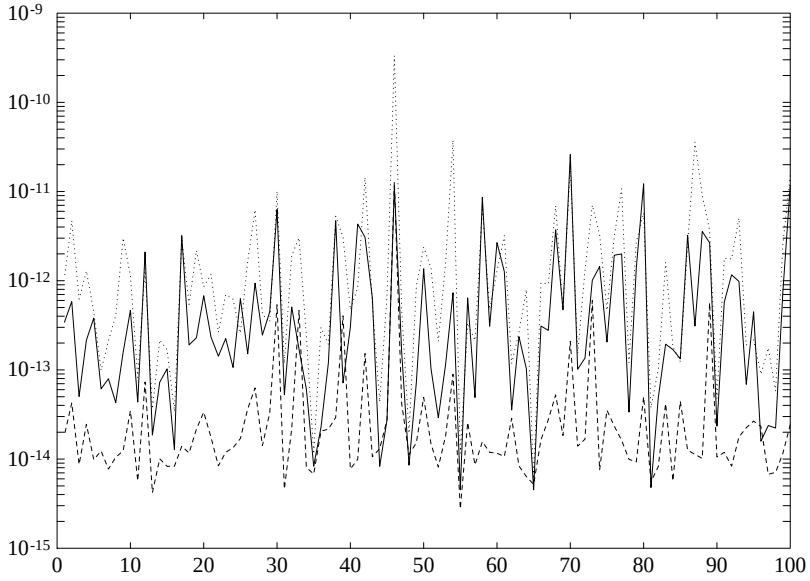


Fig. 2. Relative errors obtained for 100 test matrices with the look-ahead Levinson algorithm and by updating the solution (solid), or by applying a Gohberg-Semencul formula (dotted), or by updating the solution and applying one step of iterative refinement (dashed)

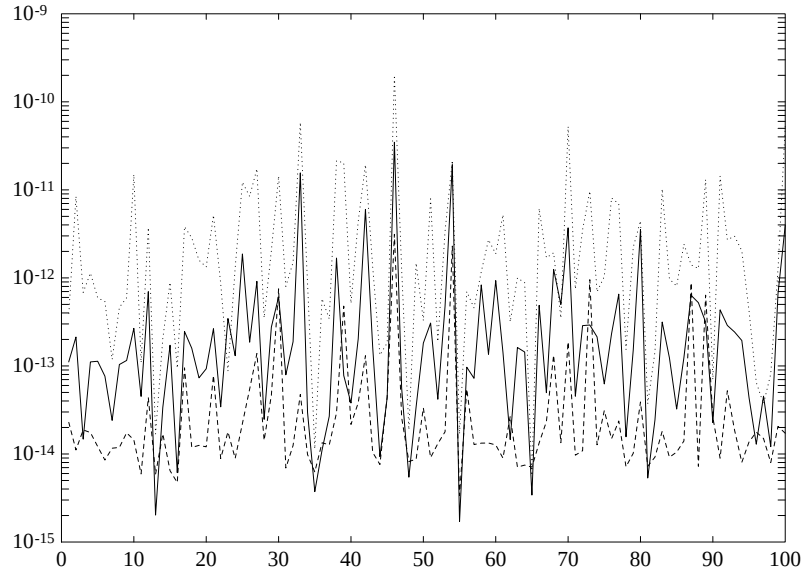


Fig. 3. Relative errors obtained for 100 test matrices with the look-ahead Schur algorithm and the formula of Sect. 10.5 (solid) or a Gohberg-Semencul formula with (dashed) or without (dotted) one step of iterative refinement

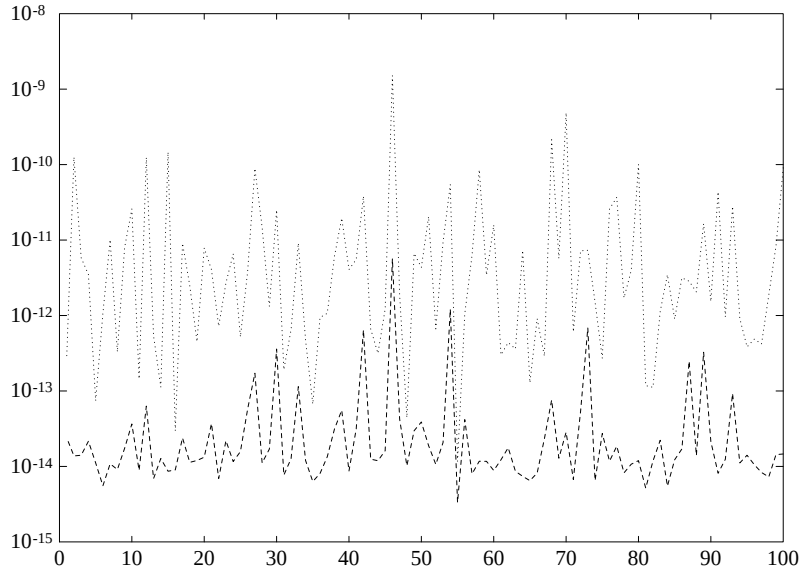


Fig. 4. Relative errors obtained for 100 test matrices with the superfast look-ahead algorithm and a Gohberg-Semencul formula with (dashed) or without (dotted) one step of iterative refinement

Table 3. Relative errors for KMS test matrices of order $N + 1$

N	superfast with G-S formula	Schur (updated solutions)	Schur with G-S formula	Levinson (updated solutions)	Levinson with G-S formula
15	2.59e-15	7.75e-16	1.71e-15	8.99e-16	2.28e-15
30	6.61e-15	1.46e-15	4.06e-15	1.49e-15	1.20e-14
60	1.43e-14	4.13e-15	9.87e-15	4.45e-15	1.57e-14
120	1.76e-13	6.95e-15	2.69e-14	4.30e-15	3.39e-14
240	7.78e-13	1.81e-14	8.14e-14	6.72e-15	9.29e-15
480	1.46e-12	2.24e-14	1.41e-13	1.10e-14	2.31e-13
960	2.94e-12	7.02e-14	4.42e-13	3.17e-14	4.99e-13

Finally, Fig. 4 shows the relative errors for the superfast solver. Here, we neither have an LDU nor an inverse LDU decomposition. Therefore, in the nonsymmetric case, the only option left is to apply a Gohberg-Semencul formula (dotted line). The dashed line again represents the relative errors after one step of iterative refinement.

As a second set of test matrices, we used a special case of the Kac-Murdock-Szegö (KMS) matrices [26] with $\rho = 1/2$, i.e. symmetric matrices with moments

$$\mu_0 = \delta, \quad \mu_j = \mu_{-j} = (1/2)^j, \quad j = 1, \ldots, N.$$

Recall that all our algorithm can exploit symmetry, which means that work and storage requirements are only half as big as those necessary for the non-Hermitian case. For $\delta = 0$, every third principal submatrix, i.e. $\mathbf{T}_{3m+1}$, $m = 0, 1, \ldots$, is singular while the remaining submatrices are well conditioned. If δ is a small number different from zero, the matrices $\mathbf{T}_{3m+1}$ are ill conditioned. We set $\delta = 10^{-14}$, as in [9] and [14]. All the three different types of look-ahead algorithms discovered the structure correctly: in

Table 4. Relative errors for KMS test matrices of order $N + 1$ after one step of iterative refinement

N	superfast	Schur	Levinson (updated solutions)	Levinson with G-S formula
15	1.29e-15	5.02e-16	9.57e-16	4.29e-16
30	1.38e-15	1.25e-15	1.12e-15	7.49e-16
60	1.74e-15	2.08e-15	2.87e-15	1.65e-15
120	6.97e-15	2.84e-15	2.57e-15	2.08e-15
240	4.85e-15	3.60e-15	3.08e-15	4.81e-15
480	2.58e-14	8.45e-15	9.68e-15	1.37e-15
960	1.12e-14	1.93e-14	6.43e-15	1.16e-14

all our algorithms, the look-ahead step sizes were minimal. Table 3 shows the results for $N = 15, 30, 60, 120, 240, 480, 960$ and the different variants that we described for the previous set of examples. For comparison, the errors of the classical Levinson algorithm are of order 10^{-2}, for all values of N. Table 4 gives the results improved by one step of iterative refinement with a Gohberg-Semencul formula.

Further numerical examples with non-Hermitian matrices will be presented in [21] and compared with the results obtained with two variations of our approach.

Acknowledgements. Most of this work was done while the second author was visiting the Interdisciplinary Project Center for Supercomputing (IPS) at ETH Zurich. She would like to thank the first author for his warm hospitality and for introducing her into the topic.

Both authors would like to thank the referee for his detailed report. His suggestions of comparing the operation overhead of the different look-ahead Levinson algorithms made us add Sect. 11. While working out the details, we found a considerable reduction of the overhead of our look-ahead Levinson algorithm.

References

1. Ammar G.S., Gragg W.B. (1987): The generalized Schur algorithm for the superfast solution of Toeplitz systems. In: J. Gilewicz, M. Pindor, W. Siemaszko, eds., Rational Approximation and its Applications in Mathematics and Physics, Lecture Notes in Mathematics **1237**, Springer, Berlin Heidelberg New York, pp. 315–330
2. Ammar G.S., Gragg W.B. (1986): The implementation and use of the generalized Schur algorithm. In: C. Byrnes, A. Lindquist, eds., Computational and Combinatorial Methods in System Theory, Elsevier Science Publishers B.V., 1986, pp. 265–279
3. Ammar G.S., Gragg W.B. (1988): Superfast solution of real positive definite Toeplitz systems, SIAM J. Matrix Anal. Appl., **9**, pp. 61–76
4. Bareiss E.H. (1969): Numerical solution of linear equations with Toeplitz and vector Toeplitz matrices, Numer. Math. **13**, pp. 404–424
5. Bitmead R.R., Anderson B.D.O. (1980): Asymptotically fast solution of Toeplitz and related systems of linear equations, Linear Algebra Appl. **34**, pp. 103–116
6. Brent R.P., Luk F.T. (1983): A systolic array for the linear-time solution of Toeplitz systems of equations, J. VLSI Computer Systems **1**, pp. 1–22
7. Bultheel A. (1987): Laurent Series and their Padé Approximations, Birkhäuser, Basel/Boston.
8. Cabay S., Meleshko R. (1993): A weakly stable algorithm for Padé approximants and the inversion of Hankel matrices, SIAM J. Matrix Anal. Appl. **14**, pp. 735–765
9. Chan T.F., Hansen P.C. (1992): A look-ahead Levinson algorithm for general Toeplitz systems, IEEE Trans. Signal Processing **40**, pp. 1079–1090
10. Chan T.F., Hansen P.C. (1993): Erratum: a look-ahead Levinson algorithm for indefinite Toeplitz systems, SIAM J. Matrix Anal. Appl. **14**, p. 1191.

11. de Hoog F. (1987): A new algorithm for solving Toeplitz systems of equations, Linear Algebra Appl. **88/89**, pp. 123–138
12. Delosme J., Ipsen I.C.F. (1986): Parallel solution of symmetric positive definite systems with hyperbolic rotations, Linear Algebra Appl. **77**, pp. 75–111
13. Delosme J., Ipsen I.C.F. (1989): From Bareiss' algorithm to the stable computation of partial correlations, J. Comput. Appl. Math. **27**, pp. 53–91
14. Freund R., Zha H. (1993): Formally biorthogonal polynomials and a look-ahead Levinson algorithm for general Toeplitz systems, Linear Algebra Appl. **188/189**, pp. 255–304
15. Freund R., Zha H. (1993): A look-ahead algorithm for the solution of general Hankel systems, Numer. Math. **64**, pp. 295–321
16. Gohberg I., Semencul A. (1972): On the inversion of finite Toeplitz matrices and their continuous analogs (in Russian), Mat. Issled. **2**, pp. 201–233
17. Gragg W.B. (1972): The Padé table and its relation to certain algorithms of numerical analysis, SIAM Rev. **14**, pp. 1–62
18. Gutknecht M.H. (1992): A completed theory of the unsymmetric Lanczos process and related algorithms, Part I, SIAM J. Matrix Anal. Appl. **13**, pp. 594–639
19. Gutknecht M.H. (1993): Stable row recurrences in the Padé table and generically superfast lookahead solvers for non-Hermitian Toeplitz systems, Linear Algebra Appl. **188/189**, pp. 351–421
20. Gutknecht M.H. (1994): A completed theory of the unsymmetric Lanczos process and related algorithms, Part II, SIAM J. Matrix Anal. Appl. **15**, pp. 15–58
21. Gutknecht M.H., Hochbruck M.: Optimized look-ahead row recurrences in the Padé table. In preparation
22. Gutknecht M.H., Hochbruck M. (1994): Look-ahead Levinson- and Schur-type recurrences in the Padé table. Electronic Trans. Numer. Anal. **2**, 104–129
23. Hansen P.C., Chan T.F. (1992): FORTRAN subroutines for general Toeplitz systems, ACM Trans. Math. Softw. **18**, pp. 256–273
24. Heinig G., Rost K. (1984): Algebraic Methods for Toeplitz-like Matrices and Operators, Akademie-Verlag, Berlin, DDR, and Birkhäuser, Basel/Stuttgart.
25. Ipsen I. (1990): Some remarks on the generalised Bareiss and Levinson algorithms. In: G. H. Golub, P. van Dooren, eds., Numerical Linear Algebra, Digital Signal Processing and Parallel Algorithms, Springer, Berlin, pp. 189–214
26. Kac M., Murdock W.L., Szegö G. (1953): On the eigen-values of certain Hermitian forms, J. Rat. Mech. Anal. **2**, pp. 767–800
27. Levinson N. (1947): The Wiener rms (root-mean-square) error criterion in filter design and prediction, J. Math. Phys. **25** pp. 261–278
28. Meleshko R.J. (1990): A stable algorithm for the computation of Padé approximants, PhD thesis, Department of Computing Science, University of Alberta (Canada).
29. Morf M. (1980): Doubling algorithms for Toeplitz and related equations, in Proc. IEEE Internat. Conf. on Acoustics, Speech and Signal Processing, Denver, CO, pp. 954–959
30. Musicus B.R. (1984): Levinson and fast Choleski algorithms for Toeplitz and almost Toeplitz matrices, tech. report, Res. Lab. of Electronics, M.I.T.
31. Schur I. (1917): Ueber Potenzreihen, die im Innern des Einheitskreises beschränkt sind, I, Crelle's J. (J. Reine Angew. Math.) **147**, pp. 205–232
32. Sweet D.R. (1986): The use of pivoting to improve the numerical performance of Toeplitz solvers. In: J. M. Speiser, ed., Advanced Algorithms and Architectures for Signal Processing, of Proc. SPIE **696**, pp. 8–18
33. Sweet D.R. (1993): The use of pivoting to improve the numerical performance of algorithms for Toeplitz matrices, SIAM J. Matrix Anal. Appl. **14**, pp. 468–493
34. van der Waerden B.L. (1966): Algebra I, Springer, Berlin.

This article was processed by the author using the LaTeX style file *pljour1* from Springer-Verlag.