Linear Algebra and its Applications 435 (2011) 1–59

Contents lists available at ScienceDirect

Linear Algebra and its Applications

journal homepage: www.elsevier.com/locate/laa

Fast algorithms for Toeplitz and Hankel matrices

Georg Heinig[1], Karla Rost *

Dept. of Mathematics, University of Chemnitz, D-09126 Chemnitz, Reichenhainer Str. 39, Germany

ARTICLE INFO

Article history:
Received 19 November 2010
Accepted 6 December 2010
Available online 18 February 2011

Submitted by A. Böttcher

AMS classification:
65F05
15B05
15A06
15A23

Keywords:
Toeplitz matrix
Hankel matrix
Levinson algorithm
Schur algorithm
LU-factorization
ZW-factorization

ABSTRACT

The paper gives a self-contained survey of fast algorithms for solving linear systems of equations with Toeplitz or Hankel coefficient matrices. It is written in the style of a textbook. Algorithms of Levinson-type and Schur-type are discussed. Their connections with triangular factorizations, Padè recursions and Lanczos methods are demonstrated. In the case in which the matrices possess additional symmetry properties, split algorithms are designed and their relations to butterfly factorizations are developed.

© 2010 Elsevier Inc. All rights reserved.

Contents

* Corresponding author.
E-mail address: krost@mathematik.tu-chemnitz.de (K. Rost).
[1] Died on May 10, 2005.

0024-3795/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.laa.2010.12.001

1. Introduction

Engineering or scientific computational problems are typically reduced to matrix computations – eventually, a linear system of equations has to be solved. The structure of the original problem often leads to a structured coefficient matrix in the resulting linear system. Hence, designing special algorithms that exploit these structures in order to be faster than standard algorithms is desirable.

We will present principles how to manage this issue. Moreover, if the coefficient matrix possesses additional symmetry properties we will discuss how to take advantage from both, the structures and the symmetries. Therefore, we offer not only classical material, but also new results which have been subject of recent research and discussion.

The presentation is largely elementary. We only assume basic knowledge in linear algebra. This should make the paper accessible to a wide readership, including graduate students and also researchers who want to enter the field of structured matrices. The exercises aim at gaining deeper understanding, some of them may be challenging.

The paper at hand is written self-contained and in the style of a textbook. Thus, it is suitable for student's self-study. Beyond that the text could serve as an elaborated manuscript for lectures on the topic and could be integrated into courses on structured matrices and on numerical linear algebra as well.

Now, let us describe the content in more detail. The present paper is dedicated to so-called fast algorithms for structured matrices. In particular, we consider Toeplitz matrices

$$T_n = [a_{i-j}]_{i,j=1}^n = \begin{bmatrix} a_0 & a_{-1} & \dots & a_{-n+1} \\ a_1 & a_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{-1} \\ a_{n-1} & \dots & a_1 & a_0 \end{bmatrix}$$

and Hankel matrices

$$H_n = [h_{i+j-1}]_{i,j=1}^n = \begin{bmatrix} h_1 & h_2 & \dots & h_n \\ h_2 & h_3 & \iddots & h_{n-1} \\ \vdots & \iddots & \iddots & \vdots \\ h_n & h_{n-1} & \dots & h_{2n-1} \end{bmatrix}.$$

The entries of the matrices are taken from a given field $\mathbb{F}$. The attribute "fast" indicates that the complexity of the algorithm is $O(n^2)$ compared with $O(n^3)$ complexity for the corresponding standard algorithms for unstructured matrices. Algorithms with a complexity less than $O(n^2)$ are often called "superfast". They are based on divide-and-conquer strategies, which, however, are beyond the scope of our paper. We will consider two kinds of fast algorithms: Levinson-type and Schur-type algorithms.

A *Levinson-type algorithm* recursively computes solutions of special equations for submatrices. In the classical Levinson algorithm these solutions are the last columns of the inverses of the leading principal submatrices (or multiples of them). These solutions can be used in different ways. Firstly, with the help of such algorithms Toeplitz and Hankel systems with general right-hand sides can be solved recursively

using the bordering principle. Secondly, they produce a factorization of the inverse matrix. In the case of the classical Levinson algorithm this is the UL-factorization of the inverse. The factorization can also be used to solve a linear system, but its importance goes far beyond that. For example, it plays a crucial role in the theory of orthogonal polynomials. Thirdly, the vectors eventually computed by the Levinson-type algorithm provide the parameters needed in Bezoutian formulas for the inverse matrix. These Bezoutian formulas represent in particular a basic tool for in the construction of superfast algorithms.

In the same way a Levinson-type algorithm produces a factorization of the inverse matrix, a *Schur-type algorithm* produces a factorization of the matrix itself. The quantities which are computed in the latter case can be interpreted as residuals for the solutions computed by the corresponding Levinson-type algorithm. A Schur-type algorithm can be combined with a corresponding Levinson-type in order to avoid inner product calculations. Let us point out that the importance of the Schur algorithm, like that for the Levinson algorithm, goes far beyond solving linear systems. It was originally designed to solve a problem in complex function theory.

Since a symmetric Toeplitz matrix is also centrosymmetric we can exploit these symmetry properties to reduce the number of operations. The resulting algorithms are referred to as *split algorithms*. There exist split algorithms of Levinson-type and Schur-type. Like the classical Levinson and Schur algorithms are related to triangular factorization, the corresponding split algorithms are related to a different kind of factorization, called *butterfly factorization*. The split Levinson algorithm computes such a factorization of the inverse matrix whereas the split Schur algorithm computes a factorization of the matrix itself.

In most of the sections of this paper we assume that the Toeplitz or Hankel matrix under consideration is strongly nonsingular. An $n \times n$ matrix $A = [a_{ij}]_{i,j=1}^{n}$ is called *strongly nonsingular* if all its leading principal sections $A_k = [a_{ij}]_{i,j=1}^{k}$ $(k = 1, \ldots, n)$ are nonsingular. In particular, positive definite matrices are strongly nonsingular. Toeplitz and Hankel matrices without this property can be treated as well. However, this requires nontrivial generalizations of ideas presented here. Occasionally the condition of strong nonsingularity is replaced by the condition that all central submatrices $[a_{ij}]_{i,j=l}^{n+1-l}$ $\left(1 \le l \le \frac{n+1}{2}\right)$ are nonsingular. A matrix with this property is called *centro-nonsingular*. Since for a Toeplitz matrix the central submatrices are equal to leading principal submatrices, centro-nonsingularity means that every second leading principal submatrix is nonsingular.

At the end of the paper references on the history and the genesis of the results under consideration can be found. We restrict ourselves to the original work of the inventors, but also to books and survey papers. Moreover, we refer interested readers who want to apply the knowledge of this course or who want to learn more about further and adjacent research to papers which are understandable on the basis of the present text. This is a reason for citing a number of papers written by the authors.

We apologize for any omission, which seems to be unavoidable because of the huge and rapidly growing number of publications on structured matrices.

2. Preliminaries

In this section, we discuss some general topics that will be used afterwards. Throughout the paper, $\mathbb{F}$ will denote an arbitrary field. In some sections we restrict ourselves to the case that $\mathbb{F}$ is of characteristic different from 2 or to the case that $\mathbb{F} = \mathbb{C}$ or $\mathbb{R}$, the fields of complex or real numbers, respectively. By $\{ \mathbf{e}_1, \ldots, \mathbf{e}_n \}$ the standard basis of $\mathbb{F}^n$ is denoted. Furthermore, $\mathbf{0}_k$ will stand for the zero vector of length k. If there is no danger of misunderstanding we will omit the subscript k.

Sometimes we use "polynomial language". For $\mathbf{x} = (x_i)_{i=1}^{n} \in \mathbb{F}^n$, we consider the polynomial

$$\mathbf{x}(t) = \sum_{k=1}^{n} x_k t^{k-1}$$

and call it the *generating polynomial of* $\mathbf{x}$. Polynomial language for matrices means that we introduce the *generating polynomial of an* $m \times n$ *matrix* $A = [a_{ij}]_{i=1,j=1}^{m\ \ n} \in \mathbb{F}^{m\times n}$ as the bivariate polynomial

$$A(t, s) = \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}\, t^{i-1} s^{j-1}.$$

2.1. Symmetries

At several places we will exploit symmetry properties of matrices. Besides symmetry, skewsymmetry and Hermitian symmetry in the usual sense we deal with persymmetry and centrosymmetry.

We introduce some notations. Let J_n be the matrix of the flip operator in $\mathbb{F}^n$ mapping $(x_1, x_2, \ldots, x_n)$ to $(x_n, x_{n-1}, \ldots, x_1)$,

$$J_n = \begin{bmatrix} 0 & & 1 \\ & \iddots & \\ 1 & & 0 \end{bmatrix}. \tag{2.1}$$

For a vector $\mathbf{x} \in \mathbb{F}^n$ we denote by $\mathbf{x}^J$ the vector $J_n\mathbf{x}$ and, in case $\mathbb{F} = \mathbb{C}$, by $\mathbf{x}^\#$ the vector $J_n\overline{\mathbf{x}}$, where $\overline{\mathbf{x}}$ is the vector with the conjugate complex entries,

$$\mathbf{x}^J = J_n\mathbf{x} \quad \text{and} \quad \mathbf{x}^\# = J_n\overline{\mathbf{x}}.$$

In polynomial language the latter looks like

$$\mathbf{x}^J(t) = \mathbf{x}(t^{-1})t^{n-1}, \quad \mathbf{x}^\#(t) = \overline{\mathbf{x}}(t^{-1})t^{n-1}.$$

A vector $\mathbf{x}$ is called *symmetric* if $\mathbf{x}^J = \mathbf{x}$, *skewsymmetric* if $\mathbf{x}^J = -\mathbf{x}$, and *conjugate symmetric* if $\mathbf{x}^\# = \mathbf{x}$.

For an $n \times n$ matrix A, we denote

$$A^J = J_nAJ_n \quad \text{and} \quad A^\# = J_n\overline{A}J_n,$$

where $\overline{A}$ is the matrix with the conjugate complex entries.

An $n \times n$ matrix A is called *persymmetric* if $A^J = A^T$. The matrix A is called *centrosymmetric* if $A^J = A$. It is called *centro-skewsymmetric* if $A^J = -A$ and *centro-Hermitian* if $A^\# = A$.

The following facts for square matrices are easy to verify:

1. Any Hankel matrix is symmetric.
2. Any Toeplitz matrix is persymmetric.
3. A Toeplitz matrix is centrosymmetric if and only if it is symmetric.
4. A Toeplitz matrix is centro-skewsymmetric if and only if it is skewsymmetric.
5. A Toeplitz matrix is centro-Hermitian if and only if it is Hermitian.

2.2. Complexity

In this paper, we will estimate the quality of an algorithm according to its computational complexity. The reader should be aware that complexity is not the only criterion to judge about an algorithm. Another important criterion is stability. However, the issue of stability is much more difficult to handle and is beyond the scope of the present paper. To some extend we will also discuss the parallel complexity of the algorithms. Our approach to parallel processing is a naive one. We assume that we have as many processors as we wish and we do not take into consideration the amount that is needed for the information exchange between the processors.

By computational complexity we mean the number of arithmetic operations. We do not count, for example, permutations, multiplication by -1 and, in the complex case, forming the conjugate complex number and multiplication by the imaginary unit. (A) stands for additions or subtractions of elements of $\mathbb{F}$, and (M) stands for multiplications or divisions.

Our complexity estimations will be of asymptotic nature, which means that we are not interested in the exact number of operations but in its dependence on the size of the problem, which here is the length of a vector or the order of a matrix. For example, "the algorithm A has complexity $O(n^2)$" means that the complexity in dependence on n rises like a quadratic function. We will always neglect lower order terms. For example, the statement "algorithm A has complexity $C(n) = an^2$ " means that the complexity is equal to $C(n) = an^2 + C'(n)$ where $\lim_{n\to\infty} C'(n)n^{-2} = 0$. We always have $C'(n) = O(n)$.

We will mainly have three types of operations:

1. addition of two vectors,
2. multiplication of a vector by a scalar, and
3. inner products.

By "inner product" we mean the multiplication of a row by a column vector of $\mathbb{F}^k$. Clearly, vector addition of vectors of length k requires k (A) and multiplication of a vector by a scalar k (M). These two operations are completely parallelizable. If k processors are available, then we need only 1 (A) or 1 (M), respectively. For an inner product k (A) plus k (M) are needed. But inner product calculation is not completely parallelizable. The most what can be achieved is a parallel complexity of $O(\log k)$. This is one reason why we are looking, among other things, for algorithms that avoid inner product calculations.

The number of operations reduce if the vectors have some symmetry properties. For example, the sum of two vectors of length k which are both symmetric or skewsymmetric requires only $\frac{1}{2}k$ (A). The same reduction appears for the multiplication of such a vector by a scalar and for inner products of such vectors. For an inner product of a general vector and a symmetric or skewsymmetric vector of length k only $\frac{1}{2}k$ (M) but k (A) are needed. In fact, suppose that $\mathbf{f}^T\mathbf{u}$ has to be computed, where $\mathbf{f} = \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{bmatrix}$ and $\mathbf{u} = \begin{bmatrix} \mathbf{v} \\ \mathbf{v}^J \end{bmatrix}$. Then

$$\mathbf{f}^T\mathbf{u} = \left(\mathbf{f}_1 + \mathbf{f}_2^J\right)^T \mathbf{v}.$$

In the case that $\mathbb{F} = \mathbb{C}$ it is reasonable to distinguish between complex multiplication (CM) and addition (CA) and their real counterparts (RM) and (RA). We consider 1 (CA) as equivalent to 2 (RA) and 1 (CM) as equivalent to 4 (RM) plus 2 (RA), although there are versions of complex multiplication with only 3 (RM) but more (RA). Thus the inner product of two complex vectors of length k requires $4k$ (RM) plus $4k$ (RA). Let us estimate the amount for the inner product of a general by a conjugate symmetric vector of $\mathbb{C}^k$. Suppose that $\mathbf{f} = \begin{bmatrix} \mathbf{g}_1 + \mathrm{i}\mathbf{h}_1 \\ \mathbf{g}_2 + \mathrm{i}\mathbf{h}_2 \end{bmatrix}$ and $\mathbf{u} = \begin{bmatrix} \mathbf{v}^J - \mathrm{i}\mathbf{w}^J \\ \mathbf{v} + \mathrm{i}\mathbf{w} \end{bmatrix}$, where the vectors $\mathbf{g}_1$, $\mathbf{g}_2$, $\mathbf{u}$, and $\mathbf{v}$ are real. Then

$$\mathbf{f}^T\mathbf{u} = \left(\mathbf{g}_1^J + \mathbf{g}_2\right)\mathbf{v} + \left(\mathbf{h}_1^J - \mathbf{h}_2\right)\mathbf{w} + \mathrm{i}\left[\left(\mathbf{h}_1^J + \mathbf{h}_2\right)\mathbf{v} - \left(\mathbf{g}_1^J - \mathbf{g}_2\right)\mathbf{w}\right].$$

That means the complexity is $2k$ (RM) plus $3k$ (RA).

The algorithms presented here in this paper are of recursive nature. In the Levinson-type algorithms we will have vectors of length k where k runs from 1 to n. For the Schur-type algorithms we will have vectors of length $n+1-k$ for k running from 1 to n. If the complexity of the operations for vectors of length k is about ak for some positive a, then in both cases the overall complexity will be $\frac{a}{2}n^2$, since $\sum_{k=1}^{n} k$ is about $\frac{1}{2}n^2$.

The following table lists some complexities for quick reference. Here $\mathbf{u},\ \mathbf{v} \in \mathbb{F}^k$ denote arbitrary vectors, α is a scalar, $\mathbf{u}_+,\ \mathbf{v}_+ \in \mathbb{F}^k$ are symmetric or skewsymmetric vectors in the first two rows. The special case $\mathbb{F} = \mathbb{C}$ is considered in the last two rows. Here $\mathbf{u}_+$ and $\mathbf{v}_+$ are conjugate-symmetric vectors.

	$\mathbf{u}+\mathbf{v}$	$\alpha\mathbf{u}$	$\mathbf{u}^T\mathbf{v}$	$\mathbf{u}_+ + \mathbf{v}_+$	$\alpha\mathbf{u}_+$	$\mathbf{u}_+^T\mathbf{v}_+$	$\mathbf{u}_+^T\mathbf{v}$
(M)	0	k	k	0	$\frac{1}{2}k$	$\frac{1}{2}k$	$\frac{1}{2}k$
(A)	k	0	k	$\frac{1}{2}k$	0	$\frac{1}{2}k$	k
(RM)	0	$4k$	$4k$	0	$2k$	$2k$	$2k$
(RA)	$2k$	$2k$	$4k$	k	k	$2k$	$3k$

3. The Levinson algorithm

Throughout this section, let $T_n = [a_{i-j}]_{i,j=1}^n$ be a strongly nonsingular Toeplitz matrix. Besides T_n we consider the leading principal submatrices

$$T_k = [a_{i-j}]_{i,j=1}^k \quad \text{for } k = 1, \dots, n-1.$$

First we show how to solve some special systems with the coefficient matrix T_k by recursion, then we describe how general systems $T_n\mathbf{z} = \mathbf{b}$ can be solved using the bordering method, which will be described in Section 3.4.

All procedures presented in this section will be referred to as *Levinson algorithms*, although the original Levinson algorithm was designed only for the positive definite case, and the recursion for the special systems is often referred to as Durbin algorithm.

3.1. Recursion for columns of the inverses

We consider two families of special systems

$$T_k\mathbf{x}_k^- = \mathbf{e}_1 \quad \text{and} \quad T_k\mathbf{x}_k^+ = \mathbf{e}_k \qquad (k = 1, \dots, n). \tag{3.1}$$

Obviously, the vector $\mathbf{x}_k^-$ is the first and $\mathbf{x}_k^+$ is the last column of T_k^{-1}. Our aim is to find a recursion for the $\mathbf{x}_k^\pm$.

The crucial observation is that, due to the Toeplitz structure, the matrix T_k can be found twice as a submatrix of T_{k+1},

$$T_{k+1} = \begin{bmatrix} T_k & * \\ * & * \end{bmatrix} = \begin{bmatrix} * & * \\ * & T_k \end{bmatrix}.$$

Hence we have

$$T_{k+1}\begin{bmatrix} \mathbf{x}_k^- & 0 \\ 0 & \mathbf{x}_k^+ \end{bmatrix} = \begin{bmatrix} \mathbf{e}_1 & \gamma_k^- \\ \gamma_k^+ & \mathbf{e}_k \end{bmatrix}, \tag{3.2}$$

where

$$\gamma_k^+ = [a_k \dots a_1]\,\mathbf{x}_k^-, \quad \gamma_k^- = [a_{-1} \dots a_{-k}]\,\mathbf{x}_k^+. \tag{3.3}$$

We introduce the 2×2 matrix

$$\Gamma_k = \begin{bmatrix} 1 & \gamma_k^- \\ \gamma_k^+ & 1 \end{bmatrix}. \tag{3.4}$$

Observe that Γ_k is nonsingular. In fact, otherwise $\begin{bmatrix} \mathbf{x}_k^- \\ 0 \end{bmatrix}$ would be a multiple of $\begin{bmatrix} 0 \\ \mathbf{x}_k^+ \end{bmatrix}$. But this is a contradiction to $\mathbf{e}_1^T\mathbf{x}_k^- = \frac{\det T_{k-1}}{\det T_k} \neq 0$. Multiplying (3.2) from the right by Γ_k^{-1} we obtain on the right-hand side $[\mathbf{e}_1 \;\; \mathbf{e}_{k+1}]$, which is the image of $[\mathbf{x}_{k+1}^- \;\; \mathbf{x}_{k+1}^+]$. Thus the following is true.

Theorem 3.1. *For $k = 1, \dots, n-1$, the vectors $\mathbf{x}_k^\pm$ satisfy the recursion*

$$[\mathbf{x}_{k+1}^- \;\; \mathbf{x}_{k+1}^+] = \begin{bmatrix} \mathbf{x}_k^- & 0 \\ 0 & \mathbf{x}_k^+ \end{bmatrix} \Gamma_k^{-1},$$

where Γ_k is defined by (3.4) and (3.3).

The recursion can be started with $\mathbf{x}_1^\pm = \dfrac{1}{a_0}$.

Complexity. In each step of this algorithm one has 2 inner products, 2 vector additions and 4 scalar times vector multiplications. We conclude that the amount for computing the vectors $\mathbf{x}_k^{\pm}$ by the recursion in Theorem 3.1 is $3\,n^2$ (M) plus $2\,n^2$ (A). In parallel processing the complexity is dominated by the inner product calculation, so that the overall complexity is $O(n \log n)$. An algorithm with parallel complexity $O(n)$ will be presented in the next section.

In polynomial language the recursion in Theorem 3.1 can be written as

$$\left[\mathbf{x}_{k+1}^-(t) \;\; \mathbf{x}_{k+1}^+(t)\right] = \left[\mathbf{x}_k^-(t) \;\; \mathbf{x}_k^+(t)\right] \begin{bmatrix} 1 & 0 \\ 0 & t \end{bmatrix} \Gamma_k^{-1}.$$

The numbers $\gamma_k^{\pm}$ are called *reflection coefficients* or *Schur–Szegő parameters*.

3.2. Recursion for Yule–Walker solutions

From the view point of computational complexity it is convenient to consider instead of the vectors $\mathbf{x}_k^{\pm}$ the solutions $\mathbf{u}_k^{\pm}$ of the equations

$$T_k \mathbf{u}_k^- = \rho_k^- \mathbf{e}_1 \quad \text{and} \quad T_k \mathbf{u}_k^+ = \rho_k^+ \mathbf{e}_k, \tag{3.5}$$

where $\rho_k^{\pm} \in \mathbb{F}$ are so that

$$\mathbf{e}_1^T \mathbf{u}_k^- = 1 \quad \text{and} \quad \mathbf{e}_k^T \mathbf{u}_k^+ = 1.$$

In other words $\mathbf{u}_k^+(t)$ is assumed to be monic and $\mathbf{u}_k^-(t)$ *comonic*, which means that $(\mathbf{u}_k^+)^J(t)$ is monic. Due to the strong nonsingularity of T_n, the numbers $\rho_k^{\pm}$ are nonzero, and the vectors $\mathbf{u}_k^{\pm}$ are uniquely determined. The equations (3.5) are called *Yule–Walker equations*.

The vectors $\mathbf{x}_k^{\pm}$ and $\mathbf{u}_k^{\pm}$ are related via

$$\mathbf{x}_k^{\pm} = \frac{1}{\rho_k^{\pm}} \mathbf{u}_k^{\pm} \quad \text{and} \quad \mathbf{u}_k^{\pm} = \frac{1}{\xi_k^{\pm}} \mathbf{x}_k^{\pm},$$

where ξ_k^- is the first component of $\mathbf{x}_k^-$ and ξ_k^+ the last component of $\mathbf{x}_k^+$. Note that, by Cramer's rule, $\xi_k^{\pm} = \frac{\det T_{k-1}}{\det T_k}$. Thus, $\xi_k^+ = \xi_k^- \neq 0$. This implies

$$\rho_k^+ = \rho_k^- =: \rho_k.$$

Theorem 3.1 leads to a recursion formula for the vectors $\mathbf{u}_k^{\pm}(t)$ which can also be deduced immediately from the relation

$$T_{k+1} \begin{bmatrix} \mathbf{u}_k^- & 0 \\ 0 & \mathbf{u}_k^+ \end{bmatrix} = \begin{bmatrix} \rho_k \mathbf{e}_1 & \alpha_k^+ \\ \alpha_k^- & \rho_k \mathbf{e}_k \end{bmatrix}, \tag{3.6}$$

where $\alpha_k^- = [a_k \ldots a_1]\,\mathbf{u}_k^-$, $\alpha_k^+ = [a_{-1} \ldots a_{-k}]\,\mathbf{u}_k^+$. Note that the reflection coefficients $\gamma_k^{\pm}$ introduced in the previous subsection are given by

$$\gamma_k^{\pm} = \frac{\alpha_k^{\mp}}{\rho_k}.$$

We state the emerging recursion in polynomial language.

Theorem 3.2. *For $k = 1, \ldots, n-1$, the polynomials $\mathbf{u}_k^{\pm}(t)$ and the numbers ρ_k satisfy the recursions*

$$[\mathbf{u}_{k+1}^-(t) \;\; \mathbf{u}_{k+1}^+(t)] = [\mathbf{u}_k^-(t) \;\; \mathbf{u}_k^+(t)] \begin{bmatrix} 1 & 0 \\ 0 & t \end{bmatrix} \Phi_k$$

and

$$\rho_{k+1} = \rho_k \left(1 - \gamma_k^+ \gamma_k^-\right),$$

where

$$\Phi_k = \begin{bmatrix} 1 & -\gamma_k^- \\ -\gamma_k^+ & 1 \end{bmatrix}. \tag{3.7}$$

The recursion can be started with $\mathbf{u}_1^+ = \mathbf{u}_1^- = 1$ and $\rho_1 = a_0$.

Complexity. Like for the computation of the vectors $\mathbf{x}_k^\pm$, in each step of this algorithm one has 2 inner products and 2 vector additions. But we have only 2 scalar times vector multiplications, compared with 4 for the $\mathbf{x}_k^\pm$. Hence the amount for computing the vectors $\mathbf{u}_k^\pm$ by the recursion in Theorem 3.2 is $2\,n^2$ (M) plus $2\,n^2$ (A).

It is worth to mention that the reflection coefficients $\gamma_k^\pm$ coincide with some components of the vectors $\mathbf{u}_k^\pm$. In fact, the following can be concluded from Theorem 3.2.

Corollary 3.3. *Let ν_k^- denote the last component of $\mathbf{u}_k^-$ and ν_k^+ the first component of $\mathbf{u}_k^+$. Then, for $k = 1, \dots, n-1$,*

$$\nu_{k+1}^\pm = -\gamma_k^\mp.$$

3.3. Symmetric and Hermitian cases

We discuss the simplifications of the Levinson algorithm that we have if T_n has some symmetry properties.

Let T_n be symmetric. Then T_n is also centrosymmetric. Thus we have $T_k(\mathbf{x}_k^-)^J = \mathbf{e}_1^J = \mathbf{e}_k$, which means that $\mathbf{x}_k^+ = (\mathbf{x}_k^-)^J =: \mathbf{x}_k$. Analogously $\mathbf{u}_k^+ = (\mathbf{u}_k^-)^J =: \mathbf{u}_k$ and $\gamma_k^+ = \gamma_k^- =: \gamma_k$. Hence the recursions in Theorems 3.1 and 3.2 can be written as

$$\mathbf{x}_{k+1}(t) = \frac{1}{1-\gamma_k^2}\left(t\,\mathbf{x}_k(t) - \gamma_k(\mathbf{x}_k^J)(t)\right) \quad \text{and} \quad \mathbf{u}_{k+1}(t) = t\,\mathbf{u}_k(t) - \gamma_k\,(\mathbf{u}_k^J)(t).$$

Complexity. In the case of a symmetric Toeplitz matrix, the amount for computing the vectors $\mathbf{u}_k$ by the recursion in Theorem 3.2 is n^2 (M) plus n^2 (A).

Let now $\mathbb{F} = \mathbb{C}$ and T_n be Hermitian. Then T_n is also centro-Hermitian. Thus we have $T_k(\mathbf{x}_k^-)^\# = \mathbf{e}_1^\# = \mathbf{e}_k$, which means that $\mathbf{x}_k^+ = (\mathbf{x}_k^-)^\# =: \mathbf{x}_k$. Similarly, $\mathbf{u}_k^+ = (\mathbf{u}_k^-)^\# =: \mathbf{u}_k$, $\gamma_k^- = \overline{\gamma_k^+} =: \gamma_k$. Hence the recursions in Theorems 3.1 and 3.2 can be written as

$$\mathbf{x}_{k+1}(t) = \frac{1}{1-|\gamma_k|^2}\left(t\,\mathbf{x}_k(t) - \gamma_k\mathbf{x}_k^\#(t)\right) \quad \text{and} \quad \mathbf{u}_{k+1}(t) = t\,\mathbf{u}_k(t) - \gamma_k\,\mathbf{u}_k^\#(t).$$

Complexity. In the case of a complex Hermitian Toeplitz matrix we count the number of real operations. Thus the amount for computing the vectors $\mathbf{u}_k$ by the recursion in Theorem 3.2 is $4\,n^2$ (RM) plus $4\,n^2$ (RA).

3.4. Bordering method

The solutions computed by a Levinson-type algorithm can be used to solve a general system $T_n\mathbf{z} = \mathbf{b}$, $\mathbf{b} = (b_i)_{i=1}^n$. The corresponding procedure is called *bordering method*. It is not restricted to Toeplitz matrices. Therefore, we explain it for a system $A\mathbf{z} = \mathbf{b}$ with a general strongly nonsingular coefficient matrix $A = [a_{ij}]_{i,j=1}^n$.

Let $\mathbf{x}_k$ be the solution of $A_k\mathbf{x}_k = \mathbf{e}_k$ and $\mathbf{z}_k$ be the solution of $A_k\mathbf{z}_k = \mathbf{b}_k$, where $\mathbf{b}_k = (b_i)_{i=1}^k$, $k = 1, \dots, n$, $A = A_n$. Then

$$A_{k+1}\begin{bmatrix}\mathbf{z}_k\\0\end{bmatrix}=\begin{bmatrix}\mathbf{b}_k\\\beta_k\end{bmatrix},$$

where

$$\beta_k=\begin{bmatrix}a_{k+1,1}&\dots&a_{k+1,k}\end{bmatrix}\mathbf{z}_k.$$

We conclude that

$$\mathbf{z}_{k+1}=\begin{bmatrix}\mathbf{z}_k\\0\end{bmatrix}+(b_{k+1}-\beta_k)\mathbf{x}_{k+1}. \tag{3.8}$$

We start the recursion with $\mathbf{z}_1=\dfrac{b_1}{a_{11}}$.

Similar relations exist that involve the monic solutions $\mathbf{u}_k$ of $A_k\mathbf{u}_k=\rho_k\mathbf{e}_k$.

Complexity. The application of the bordering formula (3.8) requires in each step 1 inner product, 1 vector addition and 1 scalar times vector multiplication. This results in an overall amount for bordering of n^2 (M) plus n^2 (A). Thus the amount for solving a Toeplitz system (using the vectors $\mathbf{u}_k^{\pm}$) is $3\,n^2$ (M) plus $3\,n^2$ (A). This reduces to $2\,n^2$ (M) plus $2\,n^2$ (A) for solving a symmetric Toeplitz system. The cost for an Hermitian Toeplitz system is $8\,n^2$ (RM) plus $8\,n^2$ (RA). In the case of a symmetric or Hermitian Toeplitz matrix the cost for bordering can be reduced utilizing the symmetry properties. This will be explained in Sections 10 and 12.

4. The Schur algorithm

We now present another algorithm, which is named after I. Schur. Originally the Schur algorithm was designed to answer a question in complex function theory. Later it turned out that this algorithm has a wide range of applications. In particular, it can be used to solve Toeplitz systems, since it produces the LU-factorization of the matrix (see Section 5.2). It can also be combined with the Levinson algorithm replacing the inner product calculations there. The resulting method has a slightly higher complexity than the pure Levinson algorithm in sequential but a significantly lower complexity in parallel computing.

4.1. First version of the Schur algorithm

Besides the submatrix T_k of T_n we consider the two $(n-k+1)\times k$ Toeplitz matrices

$$T_k^-=\begin{bmatrix}a_{k-n}&\dots&a_{1-n}\\\vdots&&\vdots\\a_0&\dots&a_{1-k}\end{bmatrix}\quad\text{and}\quad T_k^+=\begin{bmatrix}a_{k-1}&\dots&a_0\\\vdots&&\vdots\\a_{n-1}&\dots&a_{n-k}\end{bmatrix}. \tag{4.1}$$

Note that the last row of T_k^- is the first row of T_k and the first row of T_k^+ is the last row of T_k.

As in Section 3.4, $\mathbf{x}_k^{\pm}$ will denote the first and the last column of T_k^{-1}, respectively. Then, for $k=1,\dots,n$,

$$\begin{bmatrix}T_k^-\\T_k^+\end{bmatrix}\begin{bmatrix}\mathbf{x}_k^-&\mathbf{x}_k^+\end{bmatrix}=\begin{bmatrix}\mathbf{s}_k^{--}&\mathbf{s}_k^{-+}\\\mathbf{s}_k^{+-}&\mathbf{s}_k^{++}\end{bmatrix},$$

where the vectors $\mathbf{s}_k^{\pm\pm}=(s_{i,k}^{\pm\pm})_{i=1}^{n-k+1}\in\mathbb{F}^{n-k+1}$ are given by

$$s_{i,k}^{+\pm}=\begin{bmatrix}a_{k+i-2}&\dots&a_{i-1}\end{bmatrix}\mathbf{x}_k^{\pm},\quad s_{i,k}^{-\pm}=\begin{bmatrix}a_{k+i-1-n}&\dots&a_{i-n}\end{bmatrix}\mathbf{x}_k^{\pm}.$$

In particular,

$$s_{1,k}^{+-}=0,\quad s_{1,k}^{++}=1,\quad s_{n-k+1,k}^{--}=1,\quad s_{n-k+1,k}^{-+}=0,$$

and the reflection coefficients (3.3) are

$$\gamma_k^+ = s_{2,k}^{+-} \quad \text{and} \quad \gamma_k^- = s_{n-k,k}^{-+}. \tag{4.2}$$

The vectors $\mathbf{s}_k^{\pm\pm}$ will be called *residual vectors*.

Let us explain briefly the importance of the residual vectors for an LU-factorization of the matrix T_n. More details are discussed in Section 5. Let V denote the upper triangular matrix the kth column of which is $\begin{bmatrix} \mathbf{x}_k^+ \\ \mathbf{0} \end{bmatrix}$. Then $L = T_n V$ is lower triangular and the kth column of L equals $\begin{bmatrix} \mathbf{0} \\ \mathbf{s}_k^{++} \end{bmatrix}$. Hence $T_n = LV^{-1}$ is a triangular factorization of T_n. That means that the matrix L formed by the residual vectors $\mathbf{s}_k^{++}$ is just the L-factor of an LU-factorization of T_n.

The Schur algorithms computes the residual vectors recursively. To derive it we utilize the Toeplitz structure, as in the derivation of the Levinson algorithm. Let us adapt some notation. For $\mathbf{u} = (u_j)_{j=1}^m \in \mathbb{F}^m$, we denote by $I_+\mathbf{u}$, $I_-\mathbf{u}$ the vectors

$$I_+\mathbf{u} = (u_j)_{j=2}^m, \quad I_-\mathbf{u} = (u_j)_{j=1}^{m-1}, \tag{4.3}$$

respectively. That means that I_+ cuts off the first and I_- cuts off the last component of the vector. Moreover,

$$I_{\pm\pm}\mathbf{u} = I_\pm(I_\pm\mathbf{u}). \tag{4.4}$$

Note that

$$\begin{bmatrix} T_k^- \\ I_\pm T_k \\ T_k^+ \end{bmatrix} \mathbf{x}_k^\pm = \begin{bmatrix} \mathbf{s}_k^{-\pm} \\ \mathbf{0}_{k-2} \\ \mathbf{s}_k^{+\pm} \end{bmatrix} \in \mathbb{F}^{2n-k} \ (k > 1),$$

and the step $k \to k+1$ means to extend the zero vector in the middle of the right hand side by one zero above and below. We have

$$\begin{bmatrix} T_{k+1}^- \\ T_{k+1}^+ \end{bmatrix} \begin{bmatrix} \mathbf{x}_k^- & 0 \\ 0 & \mathbf{x}_k^+ \end{bmatrix} = \begin{bmatrix} I_+\mathbf{s}_k^{--} & I_-\mathbf{s}_k^{-+} \\ I_+\mathbf{s}_k^{+-} & I_-\mathbf{s}_k^{++} \end{bmatrix}.$$

Theorem 3.1 leads to the following.

Theorem 4.1. *For $k = 1, \dots, n-1$, the residual vectors $\mathbf{s}_k^{\pm\pm}$ satisfy the recursion*

$$\begin{bmatrix} \mathbf{s}_{k+1}^{--} & \mathbf{s}_{k+1}^{-+} \\ \mathbf{s}_{k+1}^{+-} & \mathbf{s}_{k+1}^{++} \end{bmatrix} = \begin{bmatrix} I_+\mathbf{s}_k^{--} & I_-\mathbf{s}_k^{-+} \\ I_+\mathbf{s}_k^{+-} & I_-\mathbf{s}_k^{++} \end{bmatrix} \Gamma_k^{-1},$$

where

$$\Gamma_k = \begin{bmatrix} 1 & s_{n-k,k}^{-+} \\ s_{2,k}^{+-} & 1 \end{bmatrix}.$$

The recursion starts with

$$\mathbf{s}_1^{++} = \mathbf{s}_1^{+-} = \frac{1}{a_0}(a_{i-1})_{i=1}^n, \quad \mathbf{s}_1^{-+} = \mathbf{s}_1^{--} = \frac{1}{a_0}(a_{i-n})_{i=1}^n.$$

To write the Schur recursion in polynomial language, we introduce the projection P_m defined for Laurent polynomials by

$$P_m\left(\sum_{k=-M}^N u_k t^{k-1}\right) = \sum_{k=1}^m u_k t^{k-1}. \tag{4.5}$$

That means P_m cuts off all negative powers of t and all powers greater than $m-1$. In particular, for $\mathbf{u} = (u_i)_{i=1}^m$,

$$(I_+\mathbf{u})(t) = (\mathbf{u}(t) - u_1)t^{-1} = P_{m-1}t^{-1}\mathbf{u}(t), \quad (I_-\mathbf{u})(t) = \mathbf{u}(t) - u_m t^{m-1} = P_{m-1}\mathbf{u}(t).$$

Now the recursion of Theorem 4.1 can be written in the form

$$\begin{bmatrix} \mathbf{s}_{k+1}^{--}(t) & \mathbf{s}_{k+1}^{-+}(t) \\ \mathbf{s}_{k+1}^{+-}(t) & \mathbf{s}_{k+1}^{++}(t) \end{bmatrix} = P_{n-k} \begin{bmatrix} \mathbf{s}_k^{--}(t) & \mathbf{s}_k^{-+}(t) \\ \mathbf{s}_k^{+-}(t) & \mathbf{s}_k^{++}(t) \end{bmatrix} \begin{bmatrix} t^{-1} & 0 \\ 0 & 1 \end{bmatrix} \Gamma_k^{-1}.$$

Here the projection P_{n-k} has to be applied entrywise to the matrix polynomial.

In view of (4.2), the recursion in Theorem 4.1 can be used to compute the vectors $\mathbf{x}_k^{\pm}$ without inner product calculations. As for the Levinson algorithm, the number of multiplications can be reduced if another normalization of the residual vectors is used.

4.2. Recursion for the Yule–Walker residuals

Let, for $k = 1, \ldots, n$, the vectors $\mathbf{r}_k^{\pm\pm} = (r_{i,k}^{\pm\pm})_{i=1}^{n-k+1} \in \mathbb{F}^{n-k+1}$ be given by

$$\begin{bmatrix} T_k^- \\ T_k^+ \end{bmatrix} \begin{bmatrix} \mathbf{u}_k^- & \mathbf{u}_k^+ \end{bmatrix} = \begin{bmatrix} \mathbf{r}_k^{--} & \mathbf{r}_k^{-+} \\ \mathbf{r}_k^{+-} & \mathbf{r}_k^{++} \end{bmatrix},$$

where $\mathbf{u}_k^{\pm}$ are the solutions of the Yule–Walker equations (3.5). In particular,

$$r_{1,k}^{+-} = r_{n-k+1,k}^{-+} = 0, \quad r_{1,k}^{++} = r_{n-k+1,k}^{--} = \rho_k, \quad \gamma_k^+ = \frac{r_{2,k}^{+-}}{\rho_k}, \quad \gamma_k^- = \frac{r_{n-k,k}^{-+}}{\rho_k}.$$

We state the theorem that is analogous to Theorem 4.1 in polynomial language.

Theorem 4.2. *The polynomials $\mathbf{r}_k^{\pm\pm}(t)$ satisfy the recursion*

$$\begin{bmatrix} \mathbf{r}_{k+1}^{--}(t) & \mathbf{r}_{k+1}^{-+}(t) \\ \mathbf{r}_{k+1}^{+-}(t) & \mathbf{r}_{k+1}^{++}(t) \end{bmatrix} = P_{n-k} \begin{bmatrix} \mathbf{r}_k^{--}(t) & \mathbf{r}_k^{-+}(t) \\ \mathbf{r}_k^{+-}(t) & \mathbf{r}_k^{++}(t) \end{bmatrix} \begin{bmatrix} t^{-1} & 0 \\ 0 & 1 \end{bmatrix} \Phi_k,$$

where

$$\Phi_k = \begin{bmatrix} 1 & -\gamma_k^- \\ -\gamma_k^+ & 1 \end{bmatrix}, \quad \gamma_k^+ = \frac{r_{2,k}^{+-}}{r_{1,k}^{++}}, \quad \gamma_k^- = \frac{r_{n-k,k}^{-+}}{r_{n-k+1,k}^{--}}.$$

The initialization of the recursion is given by

$$\mathbf{r}_1^{++}(t) = \mathbf{r}_1^{+-}(t) = \sum_{k=0}^{n-1} a_k t^k, \quad \mathbf{r}_1^{-+}(t) = \mathbf{r}_1^{--}(t) = \sum_{k=0}^{n-1} a_{k-n+1} t^k.$$

Complexity. In each step of the recursion we have 4 vector additions and 4 scalar times vector multiplications. The lengths of the vectors are $n-k+1$. Thus the overall complexity is $2n^2$ (M) plus $2n^2$ (A), which is the same as for the Levinson algorithm. If the Schur algorithm is only used to replace the inner product calculations in the Levinson algorithm, then the amount for computing the vectors $\mathbf{u}_n^{\pm}$ will be $3n^2$ (M) plus $3n^2$ (A). In parallel processing we have 2 vector additions and 2 scalar times vector multiplications, so that the parallel complexity is $2n$ (M) plus $2n$ (A).

4.3. Symmetric and Hermitian cases

In the case of a symmetric Toeplitz matrix T_n we have $T_k^- = J_{n-k+1}T_k^+J_k$. Since in this case $\mathbf{u}_k^- = (\mathbf{u}_k^+)^J$, we obtain

$$I_-\mathbf{r}_k^{--} = T_k^-\mathbf{u}_k^- = J_{n-k}T_k^+\mathbf{u}_k^+ = (I_+\mathbf{r}_k^{++})^J.$$

Together with $r_{1,k}^{++} = r_{n-k+1,k}^{--}$ this leads to $\mathbf{r}_k^{--} = (\mathbf{r}_k^{++})^J$. Analogously, $\mathbf{r}_k^{-+} = (\mathbf{r}_k^{+-})^J$.

Similarly, in the case of an Hermitian Toeplitz matrix T_n we have $\mathbf{r}_k^{--} = (\mathbf{r}_k^{++})^\#$ and $\mathbf{r}_k^{-+} = (\mathbf{r}_k^{+-})^\#$.

Thus, in both cases it is sufficient to describe the recursion of the vectors $\mathbf{r}_k^\pm = \mathbf{r}_k^{+\pm}$,

$$[\mathbf{r}_{k+1}^-(t)\ \ \mathbf{r}_{k+1}^+(t)] = P_{n-k}\left[\mathbf{r}_k^-(t)\ \ \mathbf{r}_k^+(t)\right]\begin{bmatrix} t^{-1} & 0\\ 0 & 1\end{bmatrix}\Phi_k.$$

Analogously, the recursion for the residual vectors $\mathbf{s}_k^{\pm\pm}$ collapses to

$$[\mathbf{s}_{k+1}^-(t)\ \ \mathbf{s}_{k+1}^+(t)] = P_{n-k}\left[\mathbf{s}_k^-(t)\ \ \mathbf{s}_k^+(t)\right]\begin{bmatrix} t^{-1} & 0\\ 0 & 1\end{bmatrix}\Gamma_k^{-1},$$

where $\mathbf{s}_k^\pm = \mathbf{s}_k^{+\pm}$. In all cases the amount reduces by 50% compared with the general case in sequential processing. In parallel processing the amount remains the same.

4.4. Schur-type bordering

The bordering method explained in Section 3.4 involves inner product calculations that could be avoided as shown next.

We use the notation of Section 3.4 and introduce the $(n-k)\times k$ matrix $A_k' = [a_{ij}]_{i=k+1,\ j=1}^{n\qquad k}$ and residual vectors $\mathbf{b}_k' = A_k'\mathbf{z}_k$ and $\mathbf{s}_k = A_k'\mathbf{x}_k$. Note that β_k is the first component of $\mathbf{b}_k'$. Then the recursion for $\mathbf{z}_k$ implies a recursion for the residual vector $\mathbf{b}_k'$

$$\mathbf{b}_{k+1}' = I_+\mathbf{b}_k' + (b_{k+1} - \beta_k)\mathbf{s}_{k+1}. \tag{4.6}$$

In the case of a Toeplitz matrix $A = T_n$ the matrix A_k' is obtained after the first row in T_k^+ is deleted and $\mathbf{s}_k = I_+\mathbf{s}_k^{++}$, where $\mathbf{s}_k^{++}$ is defined in Section 4.1.

In the next section, we show how to solve Toeplitz systems exclusively with the Schur algorithm.

5. Triangular factorizations

In this section, we show how the algorithms discussed before can be used to find triangular factorizations of Toeplitz matrices and their inverses.

5.1. General matrices

To begin with we recall some standard material and present it in a form which is convenient for our purposes.

A representation of a nonsingular $n\times n$ matrix A in the form $A = LDU$ in which L is lower triangular, U is upper triangular and D is diagonal is called *LU-factorization*. Clearly, the LU-factorization of a matrix is, if it exists, not unique. We will consider three kinds of normalizations:

- In the *unit LU-factorization* L and U are assumed to be *unit triangular*. A triangular matrix is called unit if it has ones on the main diagonal.

- If $A = LDU$ is the unit LU-factorization, then $A = (LD)D^{-1}(DU)$ will be called *co-unit LU-factorization*. The reason why we consider this factorization is that in some cases the amount to compute the co-unit LU-factorization is less than the amount for the computation of the unit LU-factorization.
- If $\mathbb{F} = \mathbb{C}$ and A is Hermitian positive definite, then there exists an LU-factorization with $D = I_n$ and $U = L^*$. This is the *Cholesky factorization*. Moreover, the middle factor D in the unit factorization is a real diagonal matrix.

Proposition 5.1. *If a matrix A admits an LU-factorization, then it is strongly nonsingular. Conversely, any strongly nonsingular matrix admits a unique unit, a unique co-unit and, in case A is Hermitian positive definite, a unique Cholesky LU-factorization.*

Analogously to LU-factorization a UL-factorization is defined. The $n \times n$ matrix A admits a UL-factorization if and only if the matrix J_nAJ_n is strongly nonsingular. Speaking about *triangular factorization* we mean an LU- or a UL-factorization.

If $A = LDU$ is the unit LU-factorization of A, then $A^T = U^TDL^T$ is the unit LU-factorization of the transpose of A. Furthermore, $A^{-1} = U^{-1}D^{-1}L^{-1}$ is the unit UL-factorization of A^{-1}. Let us also mention the obvious fact that the LU-factorization of A includes an LU-factorization of all its leading principal submatrices A_k, and the UL-factorization of A^{-1} a UL-factorization of all A_k^{-1}.

The following is a straightforward consequence of the uniqueness of the factorizations introduced above.

Proposition 5.2. *Let $A = LDU$ be the unit or co-unit LU-factorization of A. If A is symmetric, then $U = L^T$. If $\mathbb{F} = \mathbb{C}$ and A is Hermitian, then $U = L^*$.*

Now we show how the factors of the unit (or co-unit) LU-factorization of A and the co-unit (unit) UL-factorization of A^{-1} can be characterized. For this let us adapt a notation. Let $(\mathbf{v}_j)_{j=1}^n$ be a sequence of vectors such that $\mathbf{v}_j \in \mathbb{F}^j$. Then $U(\mathbf{v}_j)_{j=1}^n$ denotes the $n \times n$ upper triangular matrix the kth column of which is equal to

$$U(\mathbf{v}_j)_{j=1}^n \mathbf{e}_k = \begin{bmatrix} \mathbf{v}_k \\ \mathbf{0}_{n-k} \end{bmatrix}.$$

For a sequence $(\mathbf{w}_j)_{j=1}^n$ with $\mathbf{w}_j \in \mathbb{F}^{n+1-j}$, by $L(\mathbf{w}_j)_{j=1}^n$ is denoted the lower triangular matrix the kth column of which is equal to

$$L(\mathbf{w}_j)_{j=1}^n \mathbf{e}_k = \begin{bmatrix} \mathbf{0}_{k-1} \\ \mathbf{w}_k \end{bmatrix}.$$

If $(d_j)_{j=1}^n$, then $D(d_j)_{j=1}^n$ will denote the diagonal matrix diag $(d_1, \dots, d_n)$.

Let $A = [a_{ij}]_{i,j=1}^n$ and $A_k = [a_{ij}]_{i,j=1}^k$ $(k = 1, \dots, n)$, and let $\mathbf{x}_k$ and $\widetilde{\mathbf{x}}_k$ be the solutions of

$$A_k\mathbf{x}_k = \mathbf{e}_k \quad \text{and} \quad A_k^T\widetilde{\mathbf{x}}_k = \mathbf{e}_k,$$

$\xi_k = \mathbf{e}_k^T\mathbf{x}_k = \mathbf{e}_k^T\widetilde{\mathbf{x}}_k$. Then

$$A\begin{bmatrix} \mathbf{x}_k \\ \mathbf{0}_{k-1} \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{k-1} \\ \mathbf{s}_k \end{bmatrix} \quad \text{and} \quad A^T\begin{bmatrix} \widetilde{\mathbf{x}}_k \\ \mathbf{0}_{k-1} \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{k-1} \\ \widetilde{\mathbf{s}}_k \end{bmatrix}$$

for some vectors $\mathbf{s}_k, \widetilde{\mathbf{s}}_k \in \mathbb{F}^{n-k+1}$ the first component of which equals 1. Then the following is true.

Proposition 5.3

1. *The factors of the unit LU-factorization of A, $A = LDU$, are given by $L = L(\mathbf{s}_k)_{k=1}^n$, $U^T = L(\widetilde{\mathbf{s}}_k)_{k=1}^n$, and $D = (D(\xi_k)_{k=1}^n)^{-1}$.*

2. *The factors of the co-unit UL-factorization of* A^{-1}, $A^{-1} = U_1 D_1 L_1$, *are given by* $U_1 = U(\mathbf{x}_k)_{k=1}^n$, $L_1^T = U(\widetilde{\mathbf{x}}_k)_{k=1}^n$, *and* $D = (D(\xi_k)_{k=1}^n)^{-1}$.

For the co-unit factorization of A we consider the solutions $\mathbf{u}_k$ and $\widetilde{\mathbf{u}}_k$ of the equations $A_k\mathbf{u}_k = \rho_k\mathbf{e}_k$ and $A_k^T\widetilde{\mathbf{u}}_k = \widetilde{\rho}_k\mathbf{e}_k$ satisfying $\mathbf{e}_k^T\mathbf{u}_k = \mathbf{e}_k^T\widetilde{\mathbf{u}}_k = 1$. It can be checked that $\widetilde{\rho}_k = \rho_k$. Let

$$A\begin{bmatrix}\mathbf{u}_k\\ \mathbf{0}_{k-1}\end{bmatrix} = \begin{bmatrix}\mathbf{0}_{k-1}\\ \mathbf{r}_k\end{bmatrix} \quad \text{and} \quad A^T\begin{bmatrix}\widetilde{\mathbf{u}}_k\\ \mathbf{0}_{k-1}\end{bmatrix} = \begin{bmatrix}\mathbf{0}_{k-1}\\ \widetilde{\mathbf{r}}_k\end{bmatrix}.$$

Then the following is true.

Proposition 5.4

1. *The factors of the co-unit LU-factorization of* A, $A = LDU$, *are given by* $L = L(\mathbf{r}_k)_{k=1}^n$, $U^T = L(\widetilde{\mathbf{r}}_k)_{k=1}^n$, *and* $D = (D(\rho_k)_{k=1}^n)^{-1}$.
2. *The factors of the unit UL-factorization of* A^{-1}, $A^{-1} = U_1 D_1 L_1$ *are given by* $U_1 = U(\mathbf{u}_k)_{k=1}^n$, $L_1^T = U(\widetilde{\mathbf{u}}_k)_{k=1}^n$, *and* $D_1 = (D(\rho_k)_{k=1}^n)^{-1}$.

In the theory of orthogonal polynomials the unit UL-factorization of A^{-1} appears in polynomial language. In this language UL-factorization means the representation of the generating polynomial of A^{-1} in the form

$$A^{-1}(t,s) = \sum_{k=1}^{n} \frac{1}{\rho_k}\mathbf{u}_k(t)\widetilde{\mathbf{u}}_k(s).$$

Analogously, the unit LU-factorization of A means to represent $A(t,s)$ in the form

$$A(t,s) = \sum_{k=1}^{n} \frac{1}{\xi_k}\mathbf{s}_k(t)t^{k-1}\widetilde{\mathbf{s}}_k(s)s^{k-1}.$$

5.2. Persymmetric matrices

Recall that an $n \times n$ matrix A is called *persymmetric* if $A^J := J_n A J_n = A^T$ and that Toeplitz matrices have this property. Obviously, A is persymmetric if and only if AJ_n is symmetric.

Proposition 5.5 *If* A *is strongly nonsingular and persymmetric, and* $A = LDU$ *is its* (*unit or co-unit*) *LU-factorization, then the* (*unit or co-unit*) *UL-factorization of* A *is given by* $A = U_1 D_1 L_1$, *where* $U_1 = (U^T)^J$, $D_1 = D^J$ *and* $L_1 = (L^T)^J$. *Conversely, a UL-factorization can be transformed into an LU-factorization.*

For a symmetric matrix A the upper triangular factor of the unit UL-factorization of A^{-1} (or of the unit LU-factorization of A) can be immediately obtained from the lower triangular factor by transposition, $L = U^T$. This is not the case for a persymmetric matrix. For the construction of the triangular factors we need both the vectors $\mathbf{u}_k$ and $\widetilde{\mathbf{u}}_k$ (or $\mathbf{s}_k$ and $\widetilde{\mathbf{s}}_k$). However, due to the close relationship between symmetric and persymmetric matrices, there should be some hidden relation between these vectors. Let us describe such a relation between the vectors $\mathbf{u}_k$ and $\widetilde{\mathbf{u}}_k$ in the following proposition.

Proposition 5.6 *If* A *is persymmetric, then the vectors* $\widetilde{\mathbf{u}}_k$ *and* $\mathbf{u}_k$ *are related via*

$$J_k\widetilde{\mathbf{u}}_k = \rho_k \sum_{j=1}^{k} \frac{\widetilde{v}_j}{\rho_j}\mathbf{u}_j,$$

where $\widetilde{v}_j$ *denotes the first component of* $\widetilde{\mathbf{u}}_j$.

Proof. Due to the persymmetry we have

$$A_k \widetilde{\mathbf{u}}_k^J = \rho_k \mathbf{e}_1.$$

Hence

$$J_k \widetilde{\mathbf{u}}_k = \rho_k A_k^{-1} \mathbf{e}_1 = \rho_k U(\mathbf{u}_j)_{j=1}^k \left(D(\rho_j)_{j=1}^k\right)^{-1} \left(U(\widetilde{\mathbf{u}}_j)_{j=1}^k\right)^T \mathbf{e}_1.$$

The assertion is now immediate. □

Proposition 5.6 can be interpreted in the following way. Both systems $\{\mathbf{u}_k(t)\}$ and $\{\widetilde{\mathbf{u}}_k^J(t)\}$ are bases of $\mathbb{F}^n(t)$. Proposition 5.6 describes the matrix of basis change. The inverse of this matrix has a similar form. We leave it to the reader to find it. We also leave to reader to find a relation between the vectors $\mathbf{s}_k$ and $\widetilde{\mathbf{s}}_k$ that generate the triangular factors of the LU-factorization of A.

5.3. Toeplitz matrices

Comparing the discussion above with the content of the previous sections we see that in the case of a strongly nonsingular Toeplitz matrix T_n the Levinson algorithm just computes a UL-factorization of T_n^{-1} and the Schur algorithm computes an LU-factorization of T_n. In fact we have with the notations of Sections 3.1 and 3.2

$$\mathbf{x}_k = \mathbf{x}_k^+, \quad \widetilde{\mathbf{x}}_k = (\mathbf{x}_k^-)^J, \quad \mathbf{u}_k = \mathbf{u}_k^+, \quad \widetilde{\mathbf{u}}_k = (\mathbf{u}_k^-)^J$$

and with the notations of Sections 4.1 and 4.2

$$\mathbf{s}_k = \mathbf{s}_k^{++}, \quad \widetilde{\mathbf{s}}_k = (\mathbf{s}_k^{--})^J, \quad \mathbf{r}_k = \mathbf{r}_k^{++}, \quad \widetilde{\mathbf{r}}_k = (\mathbf{r}_k^{--})^J.$$

Recall that, since T_n is persymmetric, the LU-factorization of T_n can be transformed into a UL-factorization of T_n, and the UL-factorization of T_n^{-1} into a LU-factorization of T_n^{-1}. Furthermore, we recall from Corollary 3.3 that the numbers $\widetilde{\nu}_j$ appearing in Proposition 5.6 can be expressed in terms of the reflection coefficients.

Note that apparently there is a close relationship between bordering and UL-factorization of T_n^{-1}.

5.4. Solving Toeplitz systems with the Schur algorithm

We just have shown that the Schur algorithm produces the LU-factorization $T_n = LDU$ of a strongly nonsingular Toeplitz matrix. This factorization can be used to solve a system $T_n\mathbf{z} = \mathbf{b}$ by back substitution. That means we first solve the lower triangular system $LD\mathbf{y} = \mathbf{b}$ and then the upper triangular system $U\mathbf{z} = \mathbf{y}$.

The complexity for solving a triangular system is $\frac{1}{2}\, n^2$ (M) plus $\frac{1}{2}\, n^2$ (A). Thus the overall complexity for solving a Toeplitz system exclusively by applying the Schur algorithm is $3\, n^2$ (M) plus $3\, n^2$ (A) which is the same as for the Levinson algorithm combined with bordering.

5.5. Inertia computation

First of all let us recall from the basic course in Linear Algebra what is meant by the inertia of a matrix. Assume that $\mathbb{F} = \mathbb{C}$, and let A be an Hermitian $n \times n$ matrix. The triple of integers

$$\operatorname{In} A = (p_+, p_-, p_0)$$

in which p_+ is the number of positive, p_- is the number of negative, and p_0 is the number of zero eigenvalues, counting multiplicities, is called the *inertia of A*. Clearly $p_+ + p_- + p_0 = n$. The integer

$$\operatorname{sgn} A = p_+ - p_-$$

is called the *signature of A*. Note that $p_- + p_+$ is the rank of A, so that rank and signature of an Hermitian matrix determine its inertia.

Two Hermitian $n \times n$ matrices A and B are called *congruent* if there is a nonsingular matrix C such that $B = C^*AC$, where C^* denotes the conjugate transpose of C. It follows *Sylvester's inertia law*: Congruent matrices have the same inertia.

Both the Levinson and the Schur algorithm can be used for the computation of the inertia of a Toeplitz matrix. In fact, let $\mathbb{F} = \mathbb{C}$ and T_n be Hermitian, and let $T_n = LDL^*$ be an LU-factorization of T_n. Then, by Sylvester's inertia law we have $\operatorname{sgn} T_n = \operatorname{sgn} D = \operatorname{sgn} D^{-1}$. Hence we have

$$\operatorname{sgn} T_n = \sum_{k=1}^{n} \operatorname{sgn} \rho_k = \sum_{k=1}^{n} \operatorname{sgn} \xi_k. \tag{5.1}$$

The numbers ξ_k and ρ_k are computed by the Levinson algorithm, the numbers ρ_k also by the Schur algorithm.

6. Displacement structure and quasi-Toeplitz matrices

In this section, we present an alternative derivation of the Schur algorithm for the unit or co-unit LU-factorization of a strongly nonsingular Toeplitz matrix not relying on the Levinson recursion. The advantage of this approach is that it can easily be generalized to more general structured matrices.

6.1. Gauss–Schur reduction

To begin with let us recall a version of Gaussian elimination that is called *Gauss–Schur reduction* or *Schur reduction*. The Gauss–Schur reduction produces an LU-factorization of a strongly nonsingular matrix $A = [a_{ij}]_{i,j=1}^n$. We show this for the co-unit LU-factorization. The procedure for the unit LU-factorization is similar. We use the notation of the previous section.

Put $A_1 = A$, and let $A_1 = LDU$ be the co-unit LU-factorization of A_1. The first column $\mathbf{l}_1$ of L, the first row $\mathbf{u}_1^T$ of U, and the first diagonal element d_1 of D are given by

$$\mathbf{l}_1 = A_1\mathbf{e}_1, \quad \mathbf{u}_1^T = \mathbf{e}_1^T A_1, \quad d_1 = a_{11}^{-1}.$$

Furthermore, the matrix $\widetilde{A}_2 = A - d_1\mathbf{l}_1\mathbf{u}_1^T$ is of the form

$$\widetilde{A}_2 = \begin{bmatrix} 0 & \mathbf{0}^T \\ \mathbf{0} & A_2 \end{bmatrix}$$

for some $(n-1)\times(n-1)$ matrix A_2. Note that A_2 is the Schur complement of the element a_{11} in the matrix A_1. Indeed, the representation of A_1 in the form

$$A_1 = \begin{bmatrix} a_{11} & (I_+\mathbf{u}_1)^T \\ I_+\mathbf{l}_1 & A_{11} \end{bmatrix},$$

where I_+ is defined in (4.3), and A_{11} is the matrix in the upper right corner of A_1, makes clear that $A_2 = A_{11} - (I_+\mathbf{l}_1)\frac{1}{a_{11}}(I_+\mathbf{u}_1)^T$.

From the first column and row of A_2 one can get now the second column of L, the second row of U and the second diagonal element of D. Proceeding in this way one obtains the co-unit LU-factorization of A. Let us summarize.

Proposition 6.1 *Let A be a strongly nonsingular matrix of order n. Then the co-unit LU-factorization of A is given by*

$$D = D(d_k)_{k=1}^n, \quad L = L(\mathbf{l}_k)_{k=1}^n, \quad U^T = L(\mathbf{u}_k)_{k=1}^n,$$

where $A_1 = A$, $d_k = (\mathbf{e}_1^T A_k \mathbf{e}_1)^{-1}$, $\mathbf{l}_k = A_k\mathbf{e}_1$, $\mathbf{u}_k = A_k^T\mathbf{e}_1$, and A_{k+1} is the Schur complement of the $(1,1)$-entry in the matrix A_k.

6.2. Quasi-Toeplitz matrices

If $A = T_n$ is a Toeplitz matrix one would like to exploit the structure of the matrix. Unfortunately, the matrix A_2 is not Toeplitz anymore. Nevertheless some structure is preserved, as we are going to show now.

We consider the transformation $\nabla_+(A)$ in the space of $n \times n$ matrices defined by

$$\nabla_+(A) = A - S_n A S_n^T, \tag{6.1}$$

where S_n is the matrix of the forward shift operator,

$$S_n = \begin{bmatrix} 0 & 0 & & 0 \\ 1 & 0 & & 0 \\ & \ddots & \ddots & \\ 0 & & 1 & 0 \end{bmatrix}. \tag{6.2}$$

It can easily be checked that this transformation is one-to-one. The transformation ∇_+ is called *shift displacement operator*. Note that there are modifications of this transformations that will be discussed in the exercises.

For a Toeplitz matrix $T_n = [a_{i-j}]_{i,j=1}^n$ we have, obviously,

$$\nabla_+(T_n) = \begin{bmatrix} a_0 & a_{-1} & \dots & a_{1-n} \\ a_1 & 0 & \dots & 0 \\ \vdots & \vdots & & \vdots \\ a_{n-1} & 0 & \dots & 0 \end{bmatrix} = \begin{bmatrix} a_0' & 1 \\ a_1 & 0 \\ \vdots & \vdots \\ a_{n-1} & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} a_0' & a_{-1} & \dots & a_{1-n} \\ 1 & 0 & \dots & 0 \end{bmatrix},$$

where $a_0' = \frac{1}{2} a_0$. Another rank decomposition of $\nabla_+(T_n)$, which is more convenient for us, is

$$\nabla_+(T_n) = \frac{1}{a_0} \begin{bmatrix} a_0 & 0 \\ a_1 & a_1 \\ \vdots & \vdots \\ a_{n-1} & a_{n-1} \end{bmatrix} \Sigma \begin{bmatrix} a_0 & a_{-1} & \dots & a_{1-n} \\ 0 & a_{-1} & \dots & a_{1-n} \end{bmatrix}, \tag{6.3}$$

where

$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

In particular, the rank of $\nabla_+(T_n)$ equals 2, unless T_n is triangular. In the latter case the rank of $\nabla_+(T_n)$ equals 1, unless $T_n = O$.

Notice that if T_n is Hermitian, then $\nabla_+(T_n)$ is also Hermitian, and the signature of $\nabla_+(T_n)$ equals zero, unless T_n is diagonal. (Obviously, T_n diagonal means $T_n = a_0 I_n$ and $\operatorname{sign}(\nabla_+(T_n))$ equals the signum of a_0.)

Definition An $n \times n$ matrix A is called *quasi-Toeplitz* if $\operatorname{rank} \nabla_+(A) \le 2$.

Clearly, Toeplitz matrices are also quasi-Toeplitz, but not vice versa. The following proposition gives a complete description of quasi-Toeplitz matrices. Since the proof is an elementary calculation, we leave it to the reader.

Proposition 6.2 *Suppose that* $\nabla_+(A) = \mathbf{g}_+\mathbf{g}_-^T - \mathbf{h}_+\mathbf{h}_-^T$, $\mathbf{g}_\pm = (g_i^\pm)_{i=1}^n$, $\mathbf{h}_\pm = (h_i^\pm)_{i=1}^n$. *Then A can be represented as the sum of 2 products of triangular Toeplitz matrices,*

$$A = \begin{bmatrix} g_1^+ & & 0 \\ \vdots & \ddots & \\ g_n^+ & \dots & g_1^+ \end{bmatrix} \begin{bmatrix} g_1^- & \dots & g_n^- \\ & \ddots & \vdots \\ 0 & & g_1^- \end{bmatrix} - \begin{bmatrix} h_1^+ & & 0 \\ \vdots & \ddots & \\ h_n^+ & \dots & h_1^+ \end{bmatrix} \begin{bmatrix} h_1^- & \dots & h_n^- \\ & \ddots & \vdots \\ 0 & & h_1^- \end{bmatrix}. \tag{6.4}$$

Conversely, if A is given by (6.4), then $\nabla_+(A) = \mathbf{g}_+\mathbf{g}_-^T - \mathbf{h}_+\mathbf{h}_-^T$.

For a quasi-Toeplitz matrix A, $\nabla_+(A)$ admits a representation

$$\nabla_+(A) = G_+ \Sigma\, (G_-)^T$$

for $n \times 2$ matrices $G_\pm$. The middle factor Σ is introduced for convenience. In particular, if A is a symmetric Toeplitz matrix, then $G_- = G_+$, and if A is an Hermitian Toeplitz matrix, then $G_- = \overline{G}_+$. The matrices $G_\pm$ are called *generators of A*. The generators are not unique. In fact, if $\widetilde{G}_\pm = G_\pm \Theta_\pm$ for some nonsingular 2×2 matrices $\Theta_\pm$, then $\widetilde{G}_+ \Sigma \widetilde{G}_-^T = G_+ \Sigma G_-^T$ if

$$\Theta_- = \Sigma \Theta_+^{-T} \Sigma.$$

We now show that the property of being a quasi-Toeplitz matrix is preserved during the process of Gauss–Schur reduction. First we consider a special case.

Suppose that $G_\pm = \left[\, \mathbf{g}_\pm \;\; \mathbf{h}_\pm \,\right]$, $\mathbf{g}_\pm = (g_i^\pm)_{i=1}^n$, $\mathbf{h}_\pm = (h_i^\pm)_{i=1}^n$, are the generators of the matrix $A = [a_{ij}]$.

We say that the generators are in *proper form* if $h_1^+ = h_1^- = 0$. For example the generators in (6.3) are in proper form.

If the generators are in proper form, then $a_{11} = g_1^- g_1^+ \neq 0$,

$$A\mathbf{e}_1 = \nabla_+(A)\mathbf{e}_1 = g_1^-\, \mathbf{g}_+, \quad \mathbf{e}_1^T A = \mathbf{e}_1^T \nabla_+(A) = g_1^+\, \mathbf{g}_-^T.$$

Hence

$$\mathbf{l}_1 = g_1^-\, \mathbf{g}_+, \quad \mathbf{u}_1 = g_1^+\, \mathbf{g}_-, \quad d_1 = (g_1^+ g_1^-)^{-1}.$$

Proposition 6.3 *Let A be a quasi-Toeplitz matrix, $G_\pm$ its generators in proper form, and let A_2 be the Schur complement of a_{11} in A. Then A_2 is quasi-Toeplitz and*

$$\nabla_+(A_2) = \left[\, I_- \mathbf{g}_+ \;\; I_+ \mathbf{h}_+ \,\right] \Sigma \left[\, I_- \mathbf{g}_- \;\; I_+ \mathbf{h}_- \,\right]^T$$

where $I_\pm$ are defined in (4.3).

Proof. Let $\widetilde{A}_2$ be defined as above. Then

$$\begin{aligned} \nabla_+(\widetilde{A}_2) &= \nabla_+(A) - \nabla_+(\mathbf{g}_+ \mathbf{g}_-^T) \\ &= \mathbf{g}_+ \mathbf{g}_-^T - \mathbf{h}_+ \mathbf{h}_-^T - \mathbf{g}_+ \mathbf{g}_-^T + S_n \mathbf{g}_+ (S_n \mathbf{g}_-)^T. \end{aligned}$$

This implies

$$\nabla_+(A_2) = (I_- \mathbf{g}_+)(I_- \mathbf{g}_-)^T - (I_+ \mathbf{h}_+)(I_+ \mathbf{h}_-)^T,$$

which is the assertion. □

It remains to show how generators can be transformed into proper form. For this we observe first that, in view of $a_{11} \neq 0$, we have $g_1^+ g_1^- - h_1^+ h_1^- \neq 0$. In particular, $g_1^+ g_1^- \neq 0$ or $h_1^+ h_1^- \neq 0$. Assume, without loss of generality, that we have the first case. We define

$$\Phi = \begin{bmatrix} 1 & -\gamma_- \\ -\gamma_+ & 1 \end{bmatrix}$$

with $\gamma_\pm = \frac{h_1^\pm}{g_1^\pm}$. In view of $g_1^+ g_1^- - h_1^+ h_1^- \neq 0$, this matrix is nonsingular. Furthermore, we set $\widetilde{G}_+ = G_+ \Phi^T$ and $\widetilde{G}_- = G_- \Phi$. Then

$$\widetilde{G}_+ \Sigma \widetilde{G}_-^T = G_+ \Phi^T \Sigma \Phi^T G_-^T = (1 - \gamma_- \gamma_+) \nabla_+(A),$$

and $\widetilde{G}_\pm$ are in proper form.

6.3. Schur algorithm for quasi-Toeplitz matrices

We now describe an algorithm for LU-factorization of a strongly nonsingular quasi-Toeplitz matrix A. As in Proposition 6.1, let $A_1 = A$ and A_{k+1} the Schur complement of the $(1, 1)$-entry in A_k. We

represent $\nabla_+(A_k)$ in the form

$$\nabla_+(A_k) = \frac{1}{\rho_k} G_+^{(k)} \Sigma (G_-^{(k)})^T,$$

where $G_\pm^{(k)} = \left[\mathbf{g}_\pm^{(k)} \ \mathbf{h}_\pm^{(k)} \right]$, $\mathbf{g}_\pm^{(k)} = (g_{ik}^\pm)_{i=1}^{n-k}$, $\mathbf{h}_\pm^{(k)} = (h_{ik}^\pm)_{i=1}^{n-k}$ are in proper form, i.e. $h_{1k}^\pm = 0$. In the Toeplitz case $G_\pm^{(1)}$ are given by (6.3) and $\rho_1 = a_0 \neq 0$. Our discussion yields the following.

Proposition 6.4 *The generators* $G_\pm^{(k+1)} = \left[\mathbf{g}_\pm^{(k+1)} \ \mathbf{h}_\pm^{(k+1)} \right]$ *and the number* ρ_{k+1} *are recursively given by*

$$G_+^{(k+1)} = \left[I_- \mathbf{g}_+^{(k)} \ I_+ \mathbf{h}_+^{(k)} \right] \Phi_k^T, \quad G_-^{(k+1)} = \left[I_- \mathbf{g}_-^{(k)} \ I_+ \mathbf{h}_-^{(k)} \right] \Phi_k$$

and

$$\rho_{k+1} = (1 - \gamma_k^- \gamma_k^+)\rho_k,$$

where

$$\Phi_k = \begin{bmatrix} 1 & -\gamma_k^- \\ -\gamma_k^+ & 1 \end{bmatrix}, \quad \gamma_k^\pm = \frac{h_{2k}^\pm}{g_{1k}^\pm}.$$

The factors of the co-unit factorization $A = LDU$ are then given by

$$L = L(\mathbf{l}_k)_{k=1}^n, \quad U^T = L(\mathbf{u}_k)_{k=1}^n, \quad D = D(d_k)_{k=1}^n,$$

where $\mathbf{l}_k = \mathbf{g}_+^{(k)}$, $\mathbf{u}_k = \mathbf{g}_-^{(k)}$ and $d_k = \rho_k^{-1}$.

6.4. The Toeplitz case

We show that in the case of a Toeplitz matrix $A = T_n$ the algorithm described by Proposition 6.4 coincides with the Schur algorithm described in Section 4.2.

Comparing the initial data of both algorithms we see that

$$\mathbf{g}_+^{(1)} = \mathbf{r}_1^{++} \quad \mathbf{h}_+^{(1)} = \mathbf{r}_1^{+-} - a_0 \mathbf{e}_1 \quad \mathbf{g}_-^{(1)} = J_n \mathbf{r}_1^{--}, \quad \mathbf{h}_-^{(1)} = J_n \left(\mathbf{r}_1^{-+} - a_0 \mathbf{e}_n \right).$$

Hence

$$I_- \mathbf{g}_+^{(1)} = I_- \mathbf{r}_1^{++}, \quad I_+ \mathbf{h}_+^{(1)} = I_+ \mathbf{r}_1^{+-}, \quad I_- \mathbf{g}_-^{(1)} = J_{n-1} I_+ \mathbf{r}_1^{--}, \quad I_+ \mathbf{h}_-^{(1)} = J_{n-1} I_- \mathbf{r}_1^{-+}.$$

That means after the first step we have the same data. Consequently, for $k = 2, \ldots, n$,

$$\mathbf{g}_+^{(k)} = \mathbf{r}_k^{++} \quad \mathbf{h}_+^{(k)} = \mathbf{r}_k^{+-} \quad \mathbf{g}_-^{(k)} = J_{n+1-k} \mathbf{r}_k^{--}, \quad \mathbf{h}_-^{(1)} = J_{n+1-k} \mathbf{r}_k^{-+}.$$

This justifies to say that the algorithm described in Proposition 6.4 is in the Toeplitz case just the Schur algorithm presented in Theorem 4.2.

6.5. Outlook to Toeplitz-like matrices

The algorithm described in Proposition 6.4 can easily be generalized to a wider class of matrices called *Toeplitz-like matrices*. A matrix A is called *Toeplitz-like* if the rank r of $\nabla_+(A)$ is small compared with the order of the matrix. Using a rank decomposition of $\nabla_+(A)$ a representation of the form (6.4) can be derived, where a sum of r products of lower and upper triangular Toeplitz matrices occurs. It is quite clear how the Schur algorithm can be generalized from quasi-Toeplitz to Toeplitz-like matrices. This is however beyond the scope of the considerations here.

7. Algorithms for Hankel matrices

In this section, we consider strongly nonsingular Hankel matrices $H_n = [h_{i+j-1}]_{i,j=1}^n$ and present algorithms for the solution of Hankel systems and for triangular factorizations of H_n and H_n^{-1}. Recall

that a Hankel system can immediately transferred into a Toeplitz system by multiplying the system by J_n. However, the property of strong nonsingularity might be gotten lost after the transformation. Furthermore, the algorithms for Hankel matrices are of independent interest.

As for Toeplitz matrices, there are two types of algorithms. We call them Levinson-type and Schur-type, although in the literature different names are used.

7.1. Levinson-type algorithm

As in the Toeplitz case, the basic tool is a recursion for the solution of special systems. However, in the Hankel case we consider only one family of equations. But similar to the Toeplitz case, we discuss two versions of special systems, namely

$$H_k\mathbf{x}_k = \mathbf{e}_k$$

and

$$H_k\mathbf{u}_k = \rho_k\mathbf{e}_k, \quad \mathbf{e}_k^T\mathbf{u}_k = 1.$$

In contrast to the Toeplitz case, the Hankel algorithms are based on 3-term recursions. To deduce these recursions for the vectors $\mathbf{x}_k$ we observe that

$$H_{k+1}\begin{bmatrix} 0 & \mathbf{x}_k \\ \mathbf{x}_k & 0 \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ 1 & 0 \\ \sigma_k & 1 \\ \sigma_k' & \sigma_k \end{bmatrix}, \quad H_{k+1}\begin{bmatrix} \mathbf{x}_{k-1} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ 1 \\ \sigma_{k-1} \\ \sigma_{k-1}' \end{bmatrix},$$

where

$$\sigma_k = [h_{k+1} \dots h_{2k}]\,\mathbf{x}_k, \quad \sigma_k' = [h_{k+2} \dots h_{2k+1}]\,\mathbf{x}_k.$$

These observations lead directly to the following recursion.

Theorem 7.1 *For $k = 2, \dots, n-1$, the vectors $\mathbf{x}_k$ satisfy the recursion*

$$\mathbf{x}_{k+1} = \frac{1}{\tau_k}\left(\begin{bmatrix} 0 \\ \mathbf{x}_k \end{bmatrix} - (\sigma_k - \sigma_{k-1})\begin{bmatrix} \mathbf{x}_k \\ 0 \end{bmatrix} - \begin{bmatrix} \mathbf{x}_{k-1} \\ 0 \\ 0 \end{bmatrix}\right),$$

where τ_k is the nonzero constant

$$\tau_k = \sigma_k' - \sigma_{k-1}' - (\sigma_k - \sigma_{k-1})\sigma_k.$$

Proof. We check that the vector of the last three components of

$$H_{k+1}\left(\begin{bmatrix} 0 \\ \mathbf{x}_k \end{bmatrix} - (\sigma_k - \sigma_{k-1})\begin{bmatrix} \mathbf{x}_k \\ 0 \end{bmatrix} - \begin{bmatrix} \mathbf{x}_{k-1} \\ 0 \\ 0 \end{bmatrix}\right)$$

is equal to $\tau_k\mathbf{e}_3$. This proves the recursion. □

Obviously, $\mathbf{x}_1 = \frac{1}{h_1}$, $\sigma_1 = \frac{h_2}{h_1}$ and $\sigma_1' = \frac{h_3}{h_1}$. We can start the recursion with $k = 1$ if we set $\mathbf{x}_0$ empty and $\sigma_0 = \sigma_0' = 0$.

In polynomial language the recursion of Theorem 7.1 can be written as

$$\mathbf{x}_{k+1}(t) = \frac{1}{\tau_k}\,(t - \sigma_k + \sigma_{k-1})\mathbf{x}_k(t) - \mathbf{x}_{k-1}(t).$$

For evaluating the monic vectors $\mathbf{u}_k$ recursively, we compute in each step

$$\rho_k' = [h_{k+1} \dots h_{2k}]\,\mathbf{u}_k \quad \text{and} \quad \rho_k'' = [h_{k+2} \dots h_{2k+1}]\,\mathbf{u}_k.$$

Theorem 7.2 *For $k = 2, \dots, n-1$, the vectors $\mathbf{u}_k$ satisfy the recursion*

$$\mathbf{u}_{k+1} = \begin{bmatrix} 0 \\ \mathbf{u}_k \end{bmatrix} - \alpha_k \begin{bmatrix} \mathbf{u}_k \\ 0 \end{bmatrix} - \beta_k \begin{bmatrix} \mathbf{u}_{k-1} \\ 0 \\ 0 \end{bmatrix},$$

where

$$\beta_k = \frac{\rho_k}{\rho_{k-1}}, \quad \alpha_k = \frac{\rho_k'}{\rho_k} - \frac{\rho_{k-1}'}{\rho_{k-1}}.$$

Furthermore,

$$\rho_{k+1} = \rho_k'' - \alpha_k\rho_k' - \beta_k\rho_{k-1}''.$$

Proof. The recursion formula immediately follows from the relation

$$\begin{bmatrix} \rho_k & 0 & \rho_{k-1} \\ \rho_k' & \rho_k & \rho_{k-1}' \\ \rho_k'' & \rho_k' & \rho_{k-1}'' \end{bmatrix} \begin{bmatrix} 1 \\ -\alpha_k \\ -\beta_k \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \rho_{k+1} \end{bmatrix}. \qquad \square$$

In polynomial language the recursion can be written in the form

$$\mathbf{u}_{k+1}(t) = (t - \alpha_k)\mathbf{u}_k(t) - \beta_k\mathbf{u}_{k-1}(t).$$

We put $\mathbf{u}_0$ empty, $\rho_0 = 1$, $\rho_0' = \rho_0'' = 0$ and start with $\mathbf{u}_1 = 1$, $\rho_1 = h_1$, $\rho_1' = h_2$, $\rho_1'' = h_3$.

Complexity Comparing the number of operations in the computation of the vectors $\mathbf{x}_k$ and $\mathbf{u}_k$ we observe that, in contrast to the Toeplitz case, the overall complexity is about the same. In each step we have 2 inner products, 2 vector additions and 2 scalar times vector multiplications. The vectors have length k, and k runs from 2 to n. This leads to an overall complexity of $2\,n^2$ (M) plus $2\,n^2$ (A), which is the same as for the second version of the Levinson algorithm for Toeplitz matrices (see Section 3.2).

Theorem 7.2 can be interpreted in the following way.

Corollary 7.3 *The matrix D of the operator of multiplication by t with respect to the bases $\{\mathbf{u}_k(t)\}_{k=1}^{n-1}$ and $\{\mathbf{u}_k(t)\}_{k=1}^{n}$ is the $n \times (n-1)$ tridiagonal matrix*

$$D = \begin{bmatrix} \alpha_1 & \beta_2 & & \\ 1 & \alpha_2 & \ddots & \\ & 1 & \ddots & \beta_{n-1} \\ & & \ddots & \alpha_{n-1} \\ & & & 1 \end{bmatrix}.$$

7.2. Schur-type algorithm

Besides the submatrices H_k of H_n we consider the $(2(n-k)+1)\times k$ Hankel matrices

$$H_k' = \begin{bmatrix} h_k & \dots & h_{2k-1} \\ \vdots & & \vdots \\ h_{2n-k} & \dots & h_{2n-1} \end{bmatrix}.$$

Let us point out that the first row of H_k' is the last row of H_k. The residual vectors $\mathbf{s}_k = (s_{i,k})_{i=1}^{2(n-k)+1}$ are defined as

$$\mathbf{s}_k = H_k' \mathbf{x}_k.$$

In particular, $s_{1,k} = 1$, $s_{2,k} = \sigma_k$, and $s_{3,k} = \sigma_k'$.

Then we have

$$H_{k+1}' \begin{bmatrix} 0 & \mathbf{x}_k & \mathbf{x}_{k-1} \\ \mathbf{x}_k & 0 & \mathbf{0}_2 \end{bmatrix} = \begin{bmatrix} I_{++}\mathbf{s}_k & I_{+-}\mathbf{s}_k & I_{++}\mathbf{s}_{k-1} \end{bmatrix},$$

where $I_{\pm\pm}$ are defined in (4.4). From Theorem 7.1 we now conclude the following.

Theorem 7.4 *For $k = 2, \dots, n$, the residual vectors $\mathbf{s}_k$ satisfy the recursion*

$$\mathbf{s}_{k+1} = \frac{1}{\tau_k}\left(I_{++}\mathbf{s}_k - (s_{2,k} - s_{2,k-1})I_{+-}\mathbf{s}_k - I_{++}\mathbf{s}_{k-1}\right),$$

where

$$\tau_k = s_{3,k} - s_{3,k-1} - (s_{2,k} - s_{2,k-1})s_{2,k}.$$

The recursion can be started with $\mathbf{s}_1 = \frac{1}{h_1}(h_i)_{i=1}^{2n-1}$ and an empty $\mathbf{s}_0$.

In polynomial language this can be written as

$$\mathbf{s}_{k+1}(t) = \frac{1}{\tau_k} P_{2(n-k)-1}\left((1 - s_{2,k}t - s_{2,k-1}t)\mathbf{s}_k(t) - \mathbf{s}_{k-1}(t)t^{-2}\right).$$

Now we introduce the residual vectors $\mathbf{r}_k = H_k'\mathbf{u}_k = (r_{i,k})_{i=1}^{2(n-k)+1}$ of the monic vectors $\mathbf{u}_k$. From Theorem 7.2 we obtain the following.

Theorem 7.5 *The polynomials $\mathbf{r}_k(t)$ satisfy the recursion*

$$\mathbf{r}_{k+1}(t) = P_{2(n-k)-1}\left((1 - \alpha_k t)\mathbf{r}_k(t) - \beta_k \mathbf{r}_{k-1}(t)t^{-2}\right),$$

where

$$\beta_k = \frac{r_{1,k}}{r_{1,k-1}}, \quad \alpha_k = \frac{r_{2,k}}{r_{1,k}} - \frac{r_{2,k-1}}{r_{1,k-1}}.$$

The recursion can be started with $\mathbf{r}_1 = (h_i)_{i=1}^{2n-1}$ and an empty $\mathbf{r}_0$.

Complexity In each step of the algorithm described in the previous two theorems we have 2 vector additions and 2 scalar times vector multiplications. The lengths of the vectors are $2(n-k)+1$, so

that the overall complexity is $2\,n^2$ (RM) plus $2\,n^2$ (RA), which is the same as for the Schur algorithm for Toeplitz matrices.

As for Toeplitz matrices, the Schur-type algorithm for Hankel matrices can be used in two ways. First it can replace the inner product calculations in the Levinson-type algorithm. In this way we obtain a mixed Levinson–Schur-type algorithm. Secondly, it provides an LU-factorization of a Hankel matrix, which is, due to symmetry, of the form $H_n = LDL^T$.

7.3. Solution of Hankel systems and LU-factorization

For the solution of general Hankel systems we can repeat everything that was said about the solution of Toeplitz systems. There is a pure Levinson-type algorithm based on Theorem 7.2 and the bordering method described in Section 3.4. There is a mixed Levinson–Schur-type algorithm based on Theorems 7.2 and 7.4 and the Schur-type bordering described in Section 4.4. Finally there is a pure Schur-type algorithm based on the (unit or co-unit) LU-factorization of the Hankel matrix and back substitution. In all cases the complexity is the same as for Toeplitz matrices.

8. Padé recursions

8.1. Padé approximation at zero

Let $\mathbf{f}(t) = \sum_{i=1}^{\infty} a_i t^{i-1}$ be a formal power series, $a_i \in \mathbb{F}$. In case $\mathbb{F} = \mathbb{C}$ one may think of $\mathbf{f}(t)$ as a function that is analytic at $t = 0$ and the series is its Maclaurin (Taylor) series expansion. *Padé approximation at* $t_0 = 0$ means the local approximation at 0 of $\mathbf{f}(t)$ by a rational function

$$\mathbf{f}^{[m/n]}(t) = \frac{\mathbf{p}(t)}{\mathbf{u}(t)},$$

where $\mathbf{p} \in \mathbb{F}^m$, $\mathbf{u} \in \mathbb{F}^n$ and m, n are given. Since $\mathbf{u}(0)$ must be different from zero we may assume that $\mathbf{u}(0) = 1$ to make the fraction representation of the rational function unique. Note that this is only one possibility of normalization. Here we will assume for convenience that $\mathbf{u}(t)$ is monic.

Since $\mathbf{f}^{[m/n]}(t)$ has $m + n - 1$ degrees of freedoms we can expect that in the generic case the first $m + n - 1$ coefficients of the Maclaurin series expansion of $\mathbf{f}(t)$ and $\mathbf{f}^{[m/n]}(t)$ coincide, i.e.

$$\mathbf{f}(t) - \mathbf{f}^{[m/n]}(t) = t^{m+n-1}\mathbf{g}(t)$$

for some formal power series $\mathbf{g}(t)$. If this relation holds, then

$$\mathbf{f}(t)\mathbf{u}(t) = \mathbf{p}(t) + t^{m+n-1}\mathbf{h}(t) \tag{8.1}$$

for some formal power series $\mathbf{h}(t)$. This is the linearized form of the Padé approximation problem. Speaking about Padé approximation we always have this problem in mind.

We translate (8.1) into matrix language. For this we introduce the $n \times n$ Toeplitz matrix $T_{m,n} = [a_{i-j+m}]_{i,j=1}^{n}$ and the $m \times n$ Toeplitz matrix $U_{m,n} = [a_{i-j+1}]_{i=1}^{m}{}_{j=1}^{n}$. Here we set $a_i = 0$ for $i \le 0$. Note that the last row of $U_{m,n}$ equals the first row of $T_{m,n}$. Comparing coefficients in (8.1) we see that (8.1) is equivalent to

$$\begin{bmatrix} U_{m,n} \\ T_{m,n} \end{bmatrix} \mathbf{u} = \begin{bmatrix} \mathbf{p} \\ \rho\,\mathbf{e}_1 \end{bmatrix},$$

where ρ is the last component of $\mathbf{p}$. That means, in order to find the Padé approximation $\mathbf{f}^{[m/n]}(t)$ of $\mathbf{f}(t)$ one has to solve first the Toeplitz system

$$T_{m,n}\mathbf{u} = \rho\,\mathbf{e}_1, \quad \mathbf{u}(0) = 1$$

to obtain $\mathbf{u}$. Then the vector $\mathbf{p}$ is obtained via

$$\mathbf{p} = U_{m,n}\mathbf{u}.$$

8.2. Padé approximation at ∞ and partial realization

Padé approximation at $t_0 = \infty$ means the following. Let an infinite series in powers of t^{-1}, $\mathbf{f}(t) = \sum_{i=1}^{\infty} s_i t^{-i}$, be given. The problem is to find, for given n, a proper rational function $\mathbf{f}^{[n]}(t) = \dfrac{\mathbf{p}_{n-1}(t)}{\mathbf{u}_n(t)}$, where $\mathbf{u}_n(t) \in \mathbb{F}^n(t)$ is monic and $\mathbf{p}_{n-1}(t) \in \mathbb{F}^{n-1}(t)$ such that

$$\mathbf{f}(t)\mathbf{u}_n(t) = \mathbf{p}_{n-1}(t) + t^{-n}\mathbf{h}(t^{-1}). \tag{8.2}$$

Here $\mathbf{h}(t)$ is a formal power series. We introduce the $n \times n$ Hankel matrix $H_n = [s_{i+j-1}]_{i,j=1}^n$ and the upper triangular $n \times n$ Toeplitz matrix $T_n = [s_{j-i}]_{i,j=1}^n$, where we set $s_i = 0$ for $i \leq 0$. Comparing the coefficients in (8.2) we find that then

$$H_n\mathbf{u}_n = \rho\, \mathbf{e}_n, \quad \mathbf{e}_n^T\mathbf{u}_n = 1 \quad \text{and} \quad T_n\mathbf{u}_n = \begin{bmatrix} \mathbf{p}_{n-1} \\ 0 \end{bmatrix}.$$

Here $\rho = \mathbf{h}(0)$. Consequently, a solution of the Padé approximation problem at $t_0 = \infty$ is provided by solving the Hankel system to get $\mathbf{u}_n$ and then by multiplying this vector by the triangular Toeplitz matrix T_n to get $\mathbf{p}_{n-1}$.

Let us now discuss the connection with partial realization. The *partial realization problem* in systems theory consists, in its simplest form, in finding a linear time-invariant discrete-time system $\Sigma = (A, B, C)$,

$$\begin{aligned} \mathbf{x}^{k+1} &= A\mathbf{x}^k + Bu^k, \\ y^k &= C\mathbf{x}^k \qquad (k = 0, 1, \dots) \end{aligned}$$

from the first components of the impulse response. Here u^k is the input, y^k is the output and $\mathbf{x}^k$ is the state of the system at the time k, A is an $m \times m$ matrix, B is a column and C is a row matrix. This problem splits into two. First one has to find a proper rational function $\mathbf{f}_m(t)$ (called the *transfer function*) with m as the degree of the (monic) denominator such that

$$\mathbf{f}_m(t) = s_1 t^{-1} + \cdots + s_{2m} t^{-(2m)} + O(t^{-(2m+1)}),$$

where the numbers s_i (called *Markov parameters*) are given by the impulse response. Then one has to find (A, B, C) such that

$$\mathbf{f}_m(t) = C(tI_m - A)^{-1}B.$$

The solution of the first part is, obviously, closely related to Padé approximation at $t_0 = \infty$ presented above for $n = m + 1$.

8.3. The Padé table

Now we are going to show how the algorithms in Sections 7 and 3 can be applied to find Padé approximants $\mathbf{f}^{[m/n]}$ of a power series $\mathbf{f}(t) = \sum_{i=1}^{\infty} a_i t^{i-1}$. We know from (8.1) that the Padé approximation problem (in its linearized form) consists in finding, for given positive integers m and n, polynomials $\mathbf{p}(t) = \mathbf{p}_{m,n}(t) \in \mathbb{F}^m(t)$ and $\mathbf{u}(t) = \mathbf{u}_{m,n}(t) \in \mathbb{F}^n(t)$ such that

$$\mathbf{f}(t)\mathbf{u}_{m,n}(t) = \mathbf{p}_{m,n}(t) + t^{m+n-1}\mathbf{h}(t)$$

for some power series $\mathbf{h}(t)$. Both the pair $(\mathbf{p}_{m,n}(t), \mathbf{u}_{m,n}(t))$ and the rational function

$$\mathbf{f}^{[m/n]}(t) = \frac{\mathbf{p}_{m,n}(t)}{\mathbf{u}_{m,n}(t)}$$

will be called $[m/n]$-*Padé approximant* of f.

The Padé approximants can be arranged in a *Padé table* in which m is the row and n the column index. The first column of the Padé table is given by the partial sums of $\mathbf{f}(t)$ and the first row by the partial sums of $\mathbf{f}(t)^{-1}$. The latter can be computed recursively in an obvious way.

The Padé table is said to be *normal* if all Padé approximants exist. In this section, we assume that the table of $\mathbf{f}$ is normal.

There are many possibilities to describe recursions between the entries of the Padé table. Here we restrict ourselves to two of them, namely those which are directly related to the algorithms for Hankel and Toeplitz matrices presented before. More relations can be found in the literature.

Recall from the end of Section 8.1 that $\mathbf{u}_{m,n}$ is the solution of the Toeplitz system

$$T_{m,n}\mathbf{u}_{m,n} = \rho_{m,n}\mathbf{e}_1,$$

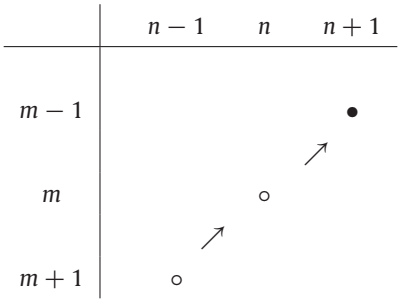
and $\mathbf{p}_{m,n}$ is given by $\mathbf{p}_{m,n} = U_{m,n}\mathbf{u}_{m,n}$, where

$$T_{m,n} = [a_{i-j+m}]_{i,j=1}^{n} \quad \text{and} \quad U_{m,n} = [a_{i-j+1}]_{i=1\ \ j=1}^{m\ \ n},$$

in which we set $a_i = 0$ for $i \leq 0$. In order to apply the algorithms presented earlier in this paper directly we normalize $\mathbf{u}_{m,n}$ by assuming that the last component of $\mathbf{u}_{m,n}$ equals 1, i.e. $\mathbf{u}_{m,n}(t)$ is monic.

8.4. Antidiagonal path

First we show that the algorithms for Hankel matrices presented in Section 7 correspond to a recursion along an antidiagonal $m + n = N$ in the Padé table. Let us illustrate this by a picture, in which empty circles denote elements in the Padé table that are known and full circles elements that will be computed.



For fixed N, we set $\mathbf{u}_n = \mathbf{u}_{m,n}$, $\mathbf{p}_n = \mathbf{p}_{m,n}$ and $\rho_n = \rho_{m,n}$, and we introduce the Hankel matrices $H_n = J_n T_{m,n}$ $(n = 1, \ldots, N)$. It is easy to see that the matrices H_n are the leading principal submatrices of the Hankel matrix $H_N = J_N T_{1,N}$, and $\mathbf{u}_n$ is the monic solution of the Hankel system $H_n\mathbf{u}_n = \rho_n \mathbf{e}_n$. Furthermore, we have $\mathbf{p}_n = J_m\mathbf{r}_n$, where $\mathbf{r}_n$ is the residual vector of $\mathbf{u}_n$ in the sense of Section 7. In particular, ρ_n is the leading coefficient of $\mathbf{p}_n(t)$. That means we can apply Theorems 7.2 and 7.5, which results in the following.

Theorem 8.1 *For $n = 1, \ldots, n-1$ the polynomials $\mathbf{u}_n(t)$ and $\mathbf{p}_n(t)$ satisfy the recursions*

$$\mathbf{u}_{n+1}(t) = (t - \alpha_n)\mathbf{u}_n(t) - \beta_n\mathbf{u}_{n-1}(t),$$
$$\mathbf{p}_{n+1}(t) = (t - \alpha_n)\mathbf{p}_n(t) - \beta_n\mathbf{p}_{n-1}(t),$$

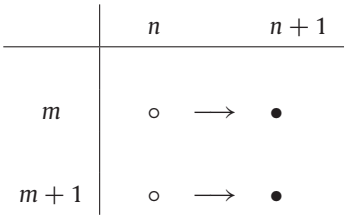
where

$$\beta_n = \frac{\rho_n}{\rho_{n-1}}, \quad \alpha_n = \frac{\rho'_n}{\rho_n} - \frac{\rho'_{n-1}}{\rho_{n-1}},$$

and ρ_n, ρ'_n *are the last two coefficients of* $\mathbf{p}_n(t)$.

8.5. Horizontal path

We show that the Toeplitz algorithms correspond, in principle, to a horizontal path in the Padé table,



The crucial observation is that

$$T_{m+1,n}\mathbf{u}_{m,n} = \alpha_{m,n}\mathbf{e}_n,$$

where

$$\alpha_{m,n} = \begin{bmatrix} a_{m+n} & \dots & a_{m+1} \end{bmatrix} \mathbf{u}_{m,n}.$$

Since

$$T_{m+1,n+1} = \begin{bmatrix} T_{m+1,n} & a_{m+1-n} \\ & \vdots \\ a_{m+1+n} & \dots \quad a_{m+1} \end{bmatrix} = \begin{bmatrix} a_{m+1} & T_{m,n} \\ \vdots & \\ a_{m+1+n} & \dots \quad a_{m+1} \end{bmatrix}$$

we have

$$T_{m+1,n+1} \begin{bmatrix} \mathbf{u}_{m+1,n} & 0 \\ 0 & \mathbf{u}_{m,n} \end{bmatrix} = \begin{bmatrix} \rho_{m+1,n} & \rho_{m,n} \\ \mathbf{0} & \mathbf{0} \\ \alpha_{m+1,n} & \alpha_{m,n} \end{bmatrix}$$

and

$$T_{m,n+1} \begin{bmatrix} \mathbf{u}_{m+1,n} & 0 \\ 0 & \mathbf{u}_{m,n} \end{bmatrix} = \begin{bmatrix} \rho'_{m+1,n} & \rho'_{m,n} \\ \rho_{m+1,n} & \rho_{m,n} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}.$$

This yields the following.

Theorem 8.2 *For fixed* m *and* $n = 1, 2, \dots,$

$$\begin{bmatrix} \mathbf{u}_{m+1,n+1}(t) & \mathbf{u}_{m,n+1}(t) \end{bmatrix} = \begin{bmatrix} \mathbf{u}_{m+1,n}(t) & \mathbf{u}_{m,n}(t) \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & t \end{bmatrix} \Phi_{m,n},$$

$$\begin{bmatrix} \mathbf{p}_{m+1,n+1}(t) & \mathbf{p}_{m,n+1}(t) \end{bmatrix} = \begin{bmatrix} \mathbf{p}_{m+1,n}(t) & \mathbf{p}_{m,n}(t) \end{bmatrix} \Phi_{m,n},$$

where

$$\Phi_{m,n} = \begin{bmatrix} -\dfrac{\alpha_{m,n}}{\alpha_{m+1,n}} & -\dfrac{\rho_{m,n}}{\rho_{m+1,n}} \\ 1 & 1 \end{bmatrix}.$$

Furthermore,

$$\alpha_{m,n+1} = \frac{\alpha_{m,n}\rho_{m+1,n} - \alpha_{m+1,n}\rho_{m,n}}{\rho_{m+1,n}}.$$

9. Hankel recursion and the Lanczos algorithm

9.1. Lanczos method

Let A be a real symmetric $n \times n$ matrix and $\mathbf{b} \in \mathbb{R}^n$. In numerical linear algebra, in particular in connection with iterative methods for solving linear systems, one has to deal with subspaces of the form

$$\mathcal{K}_k = \text{span}\,\{\mathbf{b}, A\mathbf{b}, \ldots, A^{k-1}\mathbf{b}\}.$$

These subspaces are called *Krylov subspaces*. Clearly, $\mathcal{K}_k$ is the range of the matrix

$$K_k = \begin{bmatrix} \mathbf{b} & A\mathbf{b} & \cdots & A^{k-1}\mathbf{b} \end{bmatrix},$$

which is called *Krylov matrix*.

If the vectors $\mathbf{b}, A\mathbf{b}, \ldots, A^{n-1}\mathbf{b}$ are linearly independent, then they form a basis of $\mathcal{K}_n$. However, for increasing k the vectors $A^k\mathbf{b}$ become more and more parallel, so this basis is not convenient for calculations. Therefore one is looking for an orthonormal basis of $\mathcal{K}_n$. The Lanczos algorithm is a procedure for constructing such a basis. In this section, we show that this algorithm is closely related to the Hankel matrix recursion described in Section 7.

To begin with let us state the problem. We want to find numbers q_{ij} such that the vectors

$$\mathbf{w}_j = \sum_{i=1}^{j} q_{ij} A^{i-1}\mathbf{b}$$

form an orthonormal system. Introducing the matrix $U_n = [q_{ij}]_{i,j=1}^n$, $q_{ij} = 0$ for $i > j$, the latter means that $Q_n = K_n U_n$ is a matrix with orthonormal columns $\mathbf{w}_j$ ($j = 1, \ldots, n$), i.e. $Q_n^T Q_n = I_n$. This shows that $K_n = Q_n R_n$ with $R_n = U_n^{-1}$ is the QR-factorization of K_n.

There is the following remarkable interpretation of the left factor of the QR-factorization of K_n.

Proposition 9.1 *If $K_n = Q_n R_n$ is the QR-factorization of K_n, then $M := Q_n^T A Q_n$ is tridiagonal.*

That means the matrix Q_n generates an orthogonal similarity transform that maps A into a tridiagonal matrix.

To prove this proposition we introduce the operator of multiplication by t modulo a monic polynomial

$$\mathbf{a}(t) = \sum_{j=0}^{n} a_j t^j, \quad a_n = 1. \tag{9.1}$$

This operator maps t^{j-1} to t^j for $j = 1, \ldots, n-2$ and t^{n-1} is mapped to $t^n - \mathbf{a}(t)$. Thus the matrix of the operator is given by

$$C(\mathbf{a}) = \begin{bmatrix} 0 & & -a_0 \\ 1 & & \vdots \\ & \ddots & \vdots \\ 0 & 1 & -a_{n-1} \end{bmatrix}$$

and is called the *companion (matrix)* of the polynomial $\mathbf{a}(t)$.

First we observe that $AK_n = K_nC(\mathbf{a})$, where $\mathbf{a}(t)$ is the characteristic polynomial of A. Hence

$$K_n^{-1}AK_n = \left(R_n^{-1}Q_n^T\right)A(Q_nR_n) = C(\mathbf{a}).$$

Consequently,

$$M = Q_n^TAQ_n = R_nC(\mathbf{a})R_n^{-1}. \tag{9.2}$$

Since R_n and R_n^{-1} are triangular, $C(\mathbf{a})$ is upper Hessenberg we conclude that M is also upper Hessenberg. Since M is moreover symmetric, M must be tridiagonal.

Let

$$M = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{n-1} \\ & & \beta_{n-1} & \alpha_n \end{bmatrix}.$$

Taking (9.2) into account we see that β_j is a product of diagonal entries of R_n and R_n^{-1}, thus $\beta_j \neq 0$ for $j = 1, \ldots, n-1$. From $AQ_n = Q_nM$ we see that

$$A\mathbf{w}_j = \beta_{j-1}\mathbf{w}_{j-1} + \alpha_j\mathbf{w}_j + \beta_j\mathbf{w}_{j+1}.$$

From the orthogonality of the vectors $\mathbf{w}_j$, we conclude that $\alpha_j = \mathbf{w}_j^TA\mathbf{w}_j$. Furthermore, we have $\beta_j = \|A\mathbf{w}_j - \beta_{j-1}\mathbf{w}_{j-1} - \alpha_j\mathbf{w}_j\|_2$. That means that the vectors $\mathbf{w}_j$ can be computed via the recursion

$$\mathbf{w}_{j+1} = \frac{1}{\beta_j}(A - \alpha_jI_n)\mathbf{w}_j - \frac{\beta_{j-1}}{\beta_j}\mathbf{w}_{j-1}. \tag{9.3}$$

The corresponding algorithm is named after C. Lanczos.

9.2. Hankel matrix factorization

Now we explain what the Lanczos algorithm has to do with Hankel matrix factorization algorithms. For this we observe that the matrix

$$H_n = K_n^TK_n = [\mathbf{b}^TA^{i+j-2}\mathbf{b}]_{i,j=1}^n$$

is Hankel. Furthermore, $H_n = R_n^TR_n$. Thus R_n is the upper triangular factor of the Cholesky factorization of H_n (cf. Section 5.1). What we actually need to find is $U_n = R_n^{-1}$. The columns $\mathbf{q}_j$ of U_n are the coefficient vectors of orthogonal polynomials that satisfy a 3-term recursion

$$\mathbf{q}_{j+1}(t) = \frac{1}{\beta_j}(t - \alpha_j)\mathbf{q}_j(t) - \frac{\beta_{j-1}}{\beta_j}\mathbf{q}_{j-1}(t) \tag{9.4}$$

which has the same structure as the Lanczos recursion (9.3).

The conclusion is that the Lanczos algorithm computes recursively the Q-factor of the QR-factorization of K_n, and the Levinson-type algorithm for Hankel matrices computes recursively the inverse of the R-factor via the same formulas.

10. Split algorithms for symmetric Toeplitz matrices

This section is dedicated to the special case of symmetric Toeplitz matrices $T_n = [a_{|i-j|}]_{i,j=1}^n$. We assume that the characteristic of the underlying field $\mathbb{F}$ is not equal to 2. The reason for this assumption is that in the case of characteristic 2 we have $1 = -1$, so that symmetric and skewsymmetric vectors cannot be distinguished. This has the consequence that not every vector can be represented as the sum of a symmetric and a skewsymmetric vector.

We also assume throughout the section that the order of the matrix T_n is even, $n = 2m$. This assumption is not essential. It is only to avoid considering different cases and to simplify notation. The case of odd n can be treated analogously.

10.1. Splitting

A natural question is to ask whether the property of T_n to be symmetric can further be exploited to reduce the number of operations. The answer is "yes", but the reduction comes from the centrosymmetry of T_n rather than from the symmetry. Remember from Section 2.1 that T_n is symmetric if and only if it is centrosymmetric.

Let $\mathbb{F}_+^n$ denote the subspace of all symmetric and $\mathbb{F}_-^n$ the subspace of all skewsymmetric vectors in $\mathbb{F}^n$. Obviously, $\mathbb{F}^n$ is the direct sum of $\mathbb{F}_+^n$ and $\mathbb{F}_-^n$, and $P_\pm := \frac{1}{2}(I_n \pm J_n)$ are the projections onto $\mathbb{F}_\pm^n$ along $\mathbb{F}_\mp^n$.

For a centrosymmetric matrix, the subspaces $\mathbb{F}_\pm^n$ are invariant subspaces. Hence a general system $T_n\mathbf{z} = \mathbf{b}$ can be split into the two systems $T_n\mathbf{z}_\pm = P_\pm\mathbf{b}$, where $\mathbf{z}_\pm = P_\pm\mathbf{z}$, i.e. $\mathbf{z} = \mathbf{z}_+ + \mathbf{z}_-$.

10.2. Centrosymmetric bordering

If a system $T_n\mathbf{z}_+ = \mathbf{b}_+$ with a symmetric right-hand side $\mathbf{b}_+ = (b_i)_{i=1}^n$ has to be solved, then it is reasonable to use the following centrosymmetric version of the bordering method.

For $A_n = [a_{ij}]_{i,j=1}^n$, let A_k^c $(k = 1, \dots, m)$ denote the $2k \times 2k$ central submatrix

$$A_k^c = [a_{ij}]_{i,j=m-k+1}^{m+k}.$$

Recall that a matrix is called *centro-nonsingular* if all central submatrices A_k^c $(k = 1, \dots, m)$ are nonsingular.

Note that in the case of a Toeplitz matrix $A = T_n$ we have $A_k^c = T_{2k}$. Hence any strongly nonsingular Toeplitz matrix is also centro-nonsingular.

Assume now that A is centrosymmetric and centro-nonsingular. Suppose that the solutions $\mathbf{w}_{2k}$ of $A_k^c\mathbf{w}_{2k} = 2P_+\mathbf{e}_{2k} = \mathbf{e}_{2k} + \mathbf{e}_1$ are known. Then the solutions $\mathbf{z}_k^c$ of $A_k^c\mathbf{z}_k^c = \mathbf{b}_k^c$, where $\mathbf{b}_k^c = (b_i)_{i=m-k+1}^{m+k}$, can be computed recursively by

$$\mathbf{z}_{k+1}^c = \begin{bmatrix} 0 \\ \mathbf{z}_k^c \\ 0 \end{bmatrix} + (b_{m-k} - \beta_k)\mathbf{w}_{2k+2},$$

where $\beta_k = \mathbf{g}_k^T \begin{bmatrix} 0 \\ \mathbf{z}_k^c \\ 0 \end{bmatrix}$ and $\mathbf{g}_k^T$ is the last row of A_{k+1}^c. (Start with $\mathbf{z}_1^c = b_m\mathbf{w}_2$.)

Analogously, centrosymmetric bordering works for skewsymmetric right-hand sides.

Complexity Let us compare the complexity of centrosymmetric bordering with usual bordering. In each step we have to evaluate 1 inner product of a general vector and a symmetric vector of length $2k$. As mentioned in Section 2.2 such an inner product requires only half of the number of multiplications and the same number of additions. Thus, for the inner product k (M) plus $2k$ (A) are required. Besides

this we have 1 addition of 2 symmetric vectors and 1 multiplication of symmetric vector by a scalar for which k (M) plus k (A) are needed. Since k runs from 1 to $m = n/2$ the overall complexity is $\frac{1}{4}n^2$ (M) plus $\frac{3}{8}n^2$ (A). Recall that the complexity for ordinary bordering is n^2 (M) plus n^2 (A), so that even if we have to run centrosymmetric bordering twice (to compute $\mathbf{z}^{\pm}$) we will save 50% of the multiplications and 25% of the additions.

10.3. The split Levinson algorithm

In view of the splitting idea, it is reasonable to consider, for $k = 1, \dots, n$, the equations

$$T_k \mathbf{w}_k = 2P_+ \mathbf{e}_k = \begin{bmatrix} 1 \\ \mathbf{0} \\ 1 \end{bmatrix} \tag{10.1}$$

the solutions of which are symmetric vectors. We are looking for a recursion of the vectors $\mathbf{w}_k$. We will obtain a 3-term recursion which is more similar to the recursion for Hankel matrices than to the 2-term Levinson recursion for Toeplitz matrices.

We have, for $k = 1, \dots, n-1$,

$$T_{k+1} \begin{bmatrix} \mathbf{w}_k & 0 \\ 0 & \mathbf{w}_k \end{bmatrix} = \begin{bmatrix} 1 & \gamma_k \\ 0 & 1 \\ \mathbf{0} & \mathbf{0} \\ 1 & 0 \\ \gamma_k & 1 \end{bmatrix} \quad \text{and} \quad T_{k+1} \begin{bmatrix} 0 \\ \mathbf{w}_{k-1} \\ 0 \end{bmatrix} = \begin{bmatrix} \gamma_{k-1} \\ 1 \\ \mathbf{0} \\ 1 \\ \gamma_{k-1} \end{bmatrix}$$

with

$$\gamma_k = [a_k \ \dots \ a_1]\mathbf{w}_k.$$

From these relations we conclude that $1+\gamma_k-\gamma_{k-1} \neq 0$, since otherwise T_{k+1} would have a nontrivial nullspace. We obtain the following.

Theorem 10.1 *For $k = 2, \dots, n-1$, the vectors $\mathbf{w}_k$ satisfy the recursion*

$$\mathbf{w}_{k+1} = \frac{1}{\tau_k} \left(\begin{bmatrix} \mathbf{w}_k \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \mathbf{w}_k \end{bmatrix} - \begin{bmatrix} 0 \\ \mathbf{w}_{k-1} \\ 0 \end{bmatrix} \right), \tag{10.2}$$

where $\tau_k = 1 + \gamma_k - \gamma_{k-1}$.

The recursion can be started with $\mathbf{w}_1 = \frac{2}{a_0}$ and $\mathbf{w}_2 = \frac{1}{a_1+a_0} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. The emerging algorithm is called *split Levinson algorithm*.

In polynomial language the recursion in Theorem 10.1 can be written as

$$\mathbf{w}_{k+1}(t) = \frac{1}{\tau_k} \left((1+t)\mathbf{w}_k(t) - t\mathbf{w}_{k-1}(t) \right).$$

Clearly, there is an analogous recursion for the solutions $\mathbf{w}_k^-$ of the equations

$$T_k \mathbf{w}_k^- = 2P_- \mathbf{e}_k = \begin{bmatrix} -1 \\ \mathbf{0} \\ 1 \end{bmatrix}.$$

Instead of the equations (10.1) the symmetric Yule–Walker equations could be considered. However, unlike in the classical Levinson algorithm, we will not get any computational gain from this normalization.

Complexity In the kth step we have to compute 1 inner product of a general vector and a symmetric vector of length k. This requires $\frac{1}{2}\,k$ (M) plus k (A). Then we have 2 vector additions and 1 scalar times vector multiplication of symmetric vectors for which $\frac{1}{2}\,k$ (M) plus k (A) are needed. We have to run the algorithm twice which results in $\frac{1}{2}\,n^2$ (M) plus n^2 (A) compared with n^2 (M) plus n^2 (A) for the classical Levinson algorithm.

10.4. Double-step split Levinson algorithm

Since in centrosymmetric bordering we need only every second vector $\mathbf{w}_k$ it is natural to ask whether we can obtain some computational gain if we consider double steps in the split Levinson algorithm.

We are looking for a recursion of the form

$$\mathbf{w}_{2k+2} = \frac{1}{\sigma_k}\left(\begin{bmatrix}\mathbf{w}_{2k}\\ \mathbf{0}_2\end{bmatrix} + \begin{bmatrix}\mathbf{0}_2\\ \mathbf{w}_{2k}\end{bmatrix} - \begin{bmatrix}\mathbf{0}_2\\ \mathbf{w}_{2k-2}\\ \mathbf{0}_2\end{bmatrix} + \alpha_k\begin{bmatrix}0\\ \mathbf{w}_{2k}\\ 0\end{bmatrix}\right). \tag{10.3}$$

If we multiply the right-hand side by T_{2k+2} from the left we obtain a symmetric vector with all components equal to zero, except for the last 3 and the first 3 components. The vector of the last three components is given by

$$\frac{1}{\sigma_k}\left(\begin{bmatrix}1\\ \gamma_{2k}\\ \gamma'_{2k}\end{bmatrix} + \begin{bmatrix}0\\ 0\\ 1\end{bmatrix} - \begin{bmatrix}1\\ \gamma_{2k-2}\\ \gamma'_{2k-2}\end{bmatrix} + \alpha_k\begin{bmatrix}0\\ 1\\ \gamma_{2k}\end{bmatrix}\right),$$

where γ_{2k} is defined as above and

$$\gamma'_{2k} = \begin{bmatrix} a_{2k+1} & \dots & a_2\end{bmatrix}\mathbf{w}_{2k}.$$

We have to find α_k and σ_k such that this linear combination is equal to $[0\ \ 0\ \ 1]^T$. (Note that $\sigma_k \neq 0$ since otherwise T_{2k+2} would be singular.) An easy calculation leads to the following.

Theorem 10.2 *For $k = 2, \dots, m-1$, the vectors $\mathbf{w}_{2k}$ satisfy the recursion (10.3), where $\alpha_k = \gamma_{2k-2} - \gamma_{2k}$ and*

$$\sigma_k = 1 + \gamma'_{2k} - \gamma'_{2k-2} + \gamma_{2k}(\gamma_{2k-2} - \gamma_{2k}).$$

We can start the recursion with an empty $\mathbf{w}_0$ and $\mathbf{w}_2 = \frac{1}{a_0+a_1}\begin{bmatrix}1\\ 1\end{bmatrix}$.

Complexity Let us compare the complexity of the double-step with the single-step algorithm. For the recursion from k to $k+2$, in the double-step algorithm 2 inner products and 2 scalar times vector multiplications have to be computed, which is the same as for the single-step algorithm. That means the number of multiplications is unchanged. However, we have only 3 vector additions for the double-step algorithm compared with 4 for the single-step algorithm, so the number of additions is slightly smaller. We have $\frac{7}{8}\,n^2$ (A) for the double-step algorithm compared with n^2 (A) for the single-step algorithm.

An advantage of the double-step algorithm in comparison with the single-step algorithm is that instead of strong nonsingularity we need only that every second leading principal submatrix is nonsingular.

10.5. Relations between $\mathbf{w}_k$, $\mathbf{w}_k^-$ *and* $\mathbf{x}_k$

To solve a general Toeplitz system $T_n\mathbf{z} = \mathbf{b}$ we use the splitting idea of Section 10.1. The naive approach is to solve both $T_n\mathbf{z}_+ = P_+\mathbf{b}$ and $T_n\mathbf{z}_- = P_-\mathbf{b}$ by centrosymmetric bordering. However, then we have to run the split Levinson algorithm for both $\mathbf{w}_k$ and $\mathbf{w}_k^-$, and we will have no gain compared with the classical algorithm. Therefore, it is desirable to know some relation between the vectors $\mathbf{w}_k^-$ and $\mathbf{w}_k$.

Proposition 10.3 *For* $k = 2, \ldots, n-1$,

$$\mathbf{w}_k^-(t) = \frac{t\mathbf{w}_{k-1}(t) - c_k\mathbf{w}_{k+1}(t)}{1-t}, \tag{10.4}$$

where $\mathbf{w}_{k+1}(1) \neq 0$ *and* $c_k = \frac{\mathbf{w}_{k-1}(1)}{\mathbf{w}_{k+1}(1)}$.

Proof. We have

$$T_{k+1}\left(\begin{bmatrix}\mathbf{w}_k^-\\0\end{bmatrix} - \begin{bmatrix}0\\\mathbf{w}_k^-\end{bmatrix}\right) = \begin{bmatrix}a\\1\\\mathbf{0}\\1\\a\end{bmatrix} \quad \text{and} \quad T_{k+1}\begin{bmatrix}0\\\mathbf{w}_{k-1}\\0\end{bmatrix} = \begin{bmatrix}b\\1\\\mathbf{0}\\1\\b\end{bmatrix}$$

for some $a, b \in \mathbb{F}$. Hence $(1-t)\mathbf{w}_k^-(t) = t\mathbf{w}_{k-1} - c_k\mathbf{w}_{k+1}(t)$, for some $c_k \in \mathbb{F}$. Taking $t = 1$ we obtain $c_k\mathbf{w}_{k+1}(1) = \mathbf{w}_{k-1}(1)$. We have $\mathbf{w}_{k+1}(1) \neq 0$, since $\mathbf{w}_{k+1}(1) = 0$ would imply $\mathbf{w}_{k-1}(1) = 0$ and so on, which finally leads to $\mathbf{w}_1(1) = 0$ or $\mathbf{w}_2(1) = 0$, which is not true. Hence $\mathbf{w}_{k+1}(1) \neq 0$ and $c_k = \frac{\mathbf{w}_{k-1}(1)}{\mathbf{w}_{k+1}(1)}$. □

Note that polynomial division by a linear factor can be carried out by the Horner scheme and requires k (M) and k (A). In the present case the factor is $1-t$, and hence we have only additions.

Since the solution $\mathbf{x}_k$ of $T_k\mathbf{x}_k = \mathbf{e}_k$ is given by $\mathbf{x}_k = \frac{1}{2}(\mathbf{w}_k + \mathbf{w}_k^-)$, we conclude the following from (10.4).

Corollary 10.4 *For* $k = 2, \ldots, n-1$,

$$\mathbf{x}_k(t) = \frac{1}{2}\left(\mathbf{w}_k(t) + \frac{t\mathbf{w}_{k-1}(t) - c_k\mathbf{w}_{k+1}(t)}{1-t}\right), \tag{10.5}$$

where $c_k = \frac{\mathbf{w}_{k-1}(1)}{\mathbf{w}_{k+1}(1)}$.

If we consider a nonsingular symmetric Toeplitz extension T_{n+1} of T_n (one can show that almost all such extensions are nonsingular), then this corollary is also true for $k = n$, so that the vector $\mathbf{x}_n$ can be computed from $\mathbf{w}_n$, $\mathbf{w}_{n+1}$ and $\mathbf{w}_{n-1}$.

10.6. Solution of systems by classical bordering

The relation between the vectors $\mathbf{w}_k^-$ and $\mathbf{w}_k$ is quite remarkable but it is not convenient to compute the $\mathbf{w}_k^-$ from the $\mathbf{w}_k$ at each step, since this would add another $O(n^2)$ complexity term. We are looking for possibilities to solve a general system using only the vectors $\mathbf{w}_k$.

The first idea is to apply the classical bordering (3.8) with $\mathbf{x}_k$ replaced by $\mathbf{w}_k$. Doing this we obtain vectors $\mathbf{z}'_k$ for which $T_k\mathbf{z}'_k$ equals $\mathbf{b}_k$ except for the first component. Finally we end up with a vector $\mathbf{z}'$ satisfying $T_n\mathbf{z}' = \mathbf{b} + c\,\mathbf{e}_1$ for some c. Now we compute $\mathbf{x}_n$ by (10.5) and obtain $\mathbf{z}$ by

$$\mathbf{z} = \mathbf{z}' - c\,\mathbf{x}_n^J.$$

The complexity for this method is n^2 (M) plus n^2 (A).

10.7. Solution of systems by centrosymmetric bordering – first version

The first method to solve a general system by the split Levinson algorithm and centrosymmetric bordering is based on the following fact.

Lemma 10.5 *Any vector* $\mathbf{b} \in \mathbb{F}^n$ *can be represented in the form*

$$\mathbf{b} = \mathbf{c} + \begin{bmatrix} \mathbf{d} \\ 0 \end{bmatrix}, \tag{10.6}$$

where $\mathbf{c} \in \mathbb{F}_+^n$ *and* $\mathbf{d} \in \mathbb{F}_+^{n-1}$.

We show this lemma for the case $n = 4$. The generalization to arbitrary n is obvious. Let $\mathbf{b} = (b_i)_{i=1}^4$, $\mathbf{c} = (c_i)_{i=1}^4$ and $\mathbf{d} = (d_i)_{i=1}^3$, $c_1 = c_4$, $c_2 = c_3$, $d_1 = d_3$. Then (10.6) is equivalent to the system

$$\begin{bmatrix} 1 & & & \\ 1 & 1 & & \\ & 1 & 1 & \\ & & 1 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ d_1 \\ c_2 \\ d_2 \end{bmatrix} = \begin{bmatrix} b_4 \\ b_1 \\ b_3 \\ b_2 \end{bmatrix}.$$

This system has a unique solution which can be found with n additions.

We solve the systems $T_n\mathbf{y} = \mathbf{c}$ and $T_{n-1}\mathbf{v} = \mathbf{d}$ with symmetric right-hand sides $\mathbf{c}$, $\mathbf{d}$ using centrosymmetric bordering. Then

$$T_n \left(\mathbf{y} + \begin{bmatrix} \mathbf{v} \\ 0 \end{bmatrix} \right) = \mathbf{b} + c\,\mathbf{e}_k.$$

Hence the solution $\mathbf{z}$ is of the form

$$\mathbf{z} = \mathbf{y} + \begin{bmatrix} \mathbf{v} \\ 0 \end{bmatrix} - c\,\mathbf{x}_n,$$

where $\mathbf{x}_n$ is computed by (10.5). In order to compute the coefficient c we multiply this equality by the last row of T_n. From this we obtain

$$c = -b_n + c_n + \begin{bmatrix} a_{n-1} & \dots & a_1 \end{bmatrix} \mathbf{v}.$$

As explained in Section 10.2 this method requires, in addition to the amount for the split Levinson algorithm, only $\frac{1}{2}\,n^2$ (M) plus $\frac{3}{4}\,n^2$ (A) compared with n^2 (M) plus n^2 (A) for the classical bordering.

We will have a problem in solving $T_{n-1}\mathbf{v} = \mathbf{d}$ by centrosymmetric bordering if we run the double-step Levinson algorithm because only every second vector $\mathbf{w}_k$ is computed. In this case we consider the equation $T_n\widetilde{\mathbf{v}} = \widetilde{\mathbf{d}}$, where $\widetilde{\mathbf{d}}(t) = (t+1)\mathbf{d}(t)$, $\widetilde{\mathbf{d}} \in \mathbb{F}_+^n$. We show how $\widetilde{\mathbf{v}}$ is related to $\mathbf{v}$.

We have

$$T_n \left(\begin{bmatrix} \mathbf{v} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \mathbf{v} \end{bmatrix} \right) = \begin{bmatrix} \mathbf{d} \\ * \end{bmatrix} + \begin{bmatrix} * \\ \mathbf{d} \end{bmatrix} = \widetilde{\mathbf{d}} + c(\mathbf{e}_n + \mathbf{e}_1) \tag{10.7}$$

for some $c \in \mathbb{F}$. Hence $(1+t)\mathbf{v}(t) = \widetilde{\mathbf{v}}(t) + c\,\mathbf{w}_n(t)$, so $\mathbf{v}$ is given by

$$\mathbf{v}(t) = \frac{\widetilde{\mathbf{v}}(t) + c\,\mathbf{w}_n(t)}{t+1}.$$

The number c cannot be computed from this, since $\mathbf{w}_n(-1) = 0$ for even n, but it can be found by applying a "test functional", for example by left multiplication with any row of T_{n-1}.

10.8. Solution of systems by centrosymmetric bordering – second version

Now we show that the solution of the system $T_n\mathbf{z} = \mathbf{b}$ can be expressed in terms of the solutions of the two symmetric systems $T_n\mathbf{z}_+ = P_+\mathbf{b}$ and $T_{n-1}\mathbf{z}' = P_+\mathbf{b}'$, where $\mathbf{b}'$ is the vector of the first $n-1$ components of $\mathbf{b}$. This leads to an algorithm with the same complexity as that in Section 10.7. Besides $\mathbf{z}_+$ and $\mathbf{z}'$ we need the solution vector $\mathbf{w}_{n+1}$ for a $(n+1)\times(n+1)$ nonsingular symmetric Toeplitz extension T_{n+1} of T_n.

It is sufficient to find the solution $\mathbf{z}_-$ of $T_n\mathbf{z}_- = P_-\mathbf{b}$, since $\mathbf{z} = \mathbf{z}_+ + \mathbf{z}_-$.

Proposition 10.6 *The solution $\mathbf{z}_-$ can be computed from $\mathbf{z}_+$ and $\mathbf{z}'$ via*

$$\mathbf{z}_-(t) = \frac{(t+1)\mathbf{z}_+(t) - 2t\mathbf{z}'(t) + c\,\mathbf{w}_{n+1}(t)}{t-1}, \tag{10.8}$$

where

$$c = \frac{2}{\mathbf{w}_{n+1}(1)}\,(\mathbf{z}'(1) - \mathbf{z}_+(1)).$$

Proof. Note that $\mathbf{w}_{n+1}(1) \neq 0$ according to Proposition 10.3. We have

$$T_{n+1}\left(\begin{bmatrix}0\\ \mathbf{z}_-\end{bmatrix} - \begin{bmatrix}\mathbf{z}_-\\ 0\end{bmatrix}\right) = \begin{bmatrix}*\\ P_-\mathbf{b}\end{bmatrix} - \begin{bmatrix}P_-\mathbf{b}\\ *\end{bmatrix} =: (c_i^-)_{i=1}^{n+1}$$

and

$$T_{n+1}\left(\begin{bmatrix}0\\ \mathbf{z}_+\end{bmatrix} + \begin{bmatrix}\mathbf{z}_+\\ 0\end{bmatrix} - 2\begin{bmatrix}0\\ \mathbf{z}'\\ 0\end{bmatrix}\right) = \begin{bmatrix}*\\ P_+\mathbf{b}\end{bmatrix} + \begin{bmatrix}P_+\mathbf{b}\\ *\end{bmatrix} - 2\begin{bmatrix}*\\ P_+\mathbf{b}'\\ *\end{bmatrix} =: (c_i^+)_{i=1}^{n+1}.$$

Now, for $i = 2, \dots, n$,

$$c_i^- = b_i - b_{n+1-i} - b_{i-1} + b_{n+2-i}$$

and

$$c_i^+ = b_i + b_{n+1-i} + b_{i-1} + b_{n+2-i} - 2(b_{i-1} + b_{n+1-i}),$$

so that $c_i^- = c_i^+$. Consequently,

$$(t-1)\mathbf{z}_-(t) = (t+1)\mathbf{z}_+(t) - 2t\mathbf{z}'(t) + c\,\mathbf{w}_{n+1}(t)$$

for some $c \in \mathbb{F}$. This implies (10.8). □

10.9. Split Schur algorithm

The Schur counterpart of the split Levinson is designed in the same way as the classical Schur from the classical Levinson algorithm. Let T_k^+ be defined by (4.1) and

$$T_k^+\mathbf{w}_k = \mathbf{t}_k.$$

If $\mathbf{t}_k = (t_{ik})_{i=1}^{n-k+1}$, then, in particular, $t_{1k} = 1$ and $t_{2k} = \gamma_k$. From Theorem 10.1 we immediately obtain the following.

Theorem 10.7 *For $k = 2, \dots, n-1$, the residual vectors $\mathbf{t}_k$ satisfy the recursion*

$$\mathbf{t}_{k+1} = \frac{1}{\tau_k}\,((I_+ + I_-)\mathbf{t}_k - I_{+-}\mathbf{t}_{k-1}),$$

where $\tau_k = 1 + t_{2,k} - t_{2,k-1}$, and $I_\pm$, I_{+-} are defined in (4.3), (4.4), respectively.

By Theorem 10.2 we also have a double-step recursion as follows.

Theorem 10.8 *For $k = 2, \dots, m-1$, the residual vectors $\mathbf{t}_{2k}$ satisfy the recursion*

$$\mathbf{t}_{2k+2} = \frac{1}{\sigma_k}((I_{++} + I_{--} + \alpha_k I_{+-})\mathbf{t}_{2k} - I_{++--}\mathbf{t}_{2k-2}),$$

where

$$\alpha_k = t_{2,2k-2} - t_{2,2k}, \quad \sigma_k = 1 + t_{3,2k} - t_{3,2k-2} + t_{2,2k}(t_{2,2k-2} - t_{2,2k}).$$

Similar recursions hold for the residuals $\mathbf{t}_k^- = T_k^+ \mathbf{w}_k^-$ of the solutions $\mathbf{w}_k^-$ of $T_k \mathbf{w}_k^- = 2\,P_- \mathbf{e}_k$.

11. Split algorithms for skewsymmetric Toeplitz matrices

This section is dedicated to nonsingular skewsymmetric Toeplitz matrices $T_n = [a_{i-j}]_{i,j=1}^n$, $a_{-j} = -a_j$. Like in the previous two sections we assume that the entries of the matrix belong to a field with a characteristic not equal to 2. Since skewsymmetric matrices of odd order are always singular, n must be even. Suppose that $n = 2m$. For the same reason, a skewsymmetric matrix cannot be strongly nonsingular, which means that the classical Schur and Levinson algorithms cannot be applied. We show that, however, the double-step split algorithms for symmetric Toeplitz matrices of the previous section have skewsymmetric counterparts.

Instead of strong nonsingularity we assume, throughout this section, that all leading principal submatrices of even order T_{2k} are nonsingular. This is equivalent to the centro-nonsingularity of T_n. In this case the nullspaces of T_{2k-1} $(k = 1, \dots, m)$ are one-dimensional. Let $\mathbf{x}_k$ be a vector spanning this subspace,

$$\ker\, T_{2k-1} = \operatorname{span}\{\mathbf{x}_k\}.$$

We can normalize $\mathbf{x}_k$ in different ways, for example we could assume that $\mathbf{x}_k$ is monic, since the last component must be different from zero. However, for convenience we use another normalization by assuming that

$$\begin{bmatrix} a_{2k-1} & \dots & a_1 \end{bmatrix} \mathbf{x}_k = 1.$$

This can be done, since the inner product on the left-hand side is nonzero. Otherwise T_{2k} would be singular. Clearly, with this normalization the vector $\mathbf{x}_k$ is unique.

11.1. Splitting and symmetry property of the nullspaces

Recall from Section 2.1 that a skewsymmetric Toeplitz matrix is also centro-skewsymmetric, which means that $T_n^J = -T_n$. A centro-skewsymmetric matrix maps $\mathbb{F}_+^n$ to $\mathbb{F}_-^n$ and $\mathbb{F}_-^n$ to $\mathbb{F}_+^n$. So a general system $T_n\mathbf{z} = \mathbf{b}$ splits into the two systems $T_n\mathbf{z}_\mp = P_\pm\mathbf{b}$, where $\mathbf{z}_\pm = P_\pm\mathbf{z}$, i.e. $\mathbf{z} = \mathbf{z}_+ + \mathbf{z}_-$.

Furthermore, we conclude from this property that with the vector $\mathbf{x}_k$ also the vector $\mathbf{x}_k^J$ belongs to the nullspace of T_{2k-1}. Thus $\mathbf{x}_k$ is either symmetric or skewsymmetric. We show that the latter is not possible.

Lemma 11.1 *The vector $\mathbf{x}_k$ is symmetric.*

Proof. Let $\mathbf{w} \in \mathbb{F}^{2k}$ be the vector defined by $\mathbf{w}(t) = (t+1)\mathbf{x}_k(t)$. Then

$$T_{2k}\mathbf{w} = \begin{bmatrix} -1 \\ \mathbf{0} \\ 1 \end{bmatrix}.$$

Since the right-hand side is skewsymmetric, $\mathbf{w}$ must be symmetric. This implies that $\mathbf{x}_k$ is symmetric. □

This lemma has the following consequence.

Corollary 11.2 *For any skewsymmetric* $\mathbf{b}_- \in \mathbb{F}^{2k-1}$ *the system* $T_{2k-1}\mathbf{z}_+ = \mathbf{b}_-$ *is solvable.*

11.2. First and last columns of inverses

The following is a peculiar property of skewsymmetric Toeplitz matrices. Define

$$\mathbf{x}_k^+ = \begin{bmatrix} \mathbf{x}_k \\ 0 \end{bmatrix} \quad \text{and} \quad \mathbf{x}_k^- = \begin{bmatrix} 0 \\ \mathbf{x}_k \end{bmatrix}. \tag{11.1}$$

Then we have

$$T_{2k}\mathbf{x}_k^+ = \mathbf{e}_{2k} \quad \text{and} \quad T_{2k}\mathbf{x}_k^- = -\mathbf{e}_1.$$

11.3. Levinson-type algorithm

We have

$$T_{2k+1}\begin{bmatrix} \mathbf{x}_k & 0 & 0 \\ 0 & \mathbf{x}_k & 0 \\ 0 & 0 & \mathbf{x}_k \end{bmatrix} = \begin{bmatrix} 0 & -1 & -r_k \\ 0 & 0 & -1 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ 1 & 0 & 0 \\ r_k & 1 & 0 \end{bmatrix} \quad \text{and} \quad T_{2k+1}\begin{bmatrix} 0 \\ 0 \\ \mathbf{x}_{k-1} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -r_{k-1} \\ -1 \\ \mathbf{0} \\ 1 \\ r_{k-1} \end{bmatrix},$$

where

$$r_k = \begin{bmatrix} a_{2k} & \dots & a_2 \end{bmatrix}\mathbf{x}_k.$$

From this we conclude the following.

Theorem 11.3 *For* $k = 2, \dots, n-1$, *the vectors* $\mathbf{x}_k$ *satisfy the recursion*

$$\mathbf{x}_{k+1} = \frac{1}{\alpha_k}\left(\begin{bmatrix} \mathbf{0}_2 \\ \mathbf{x}_k \end{bmatrix} + \begin{bmatrix} \mathbf{x}_k \\ \mathbf{0}_2 \end{bmatrix} - (r_k - r_{k-1})\begin{bmatrix} 0 \\ \mathbf{x}_k \\ 0 \end{bmatrix} - \begin{bmatrix} \mathbf{0}_2 \\ \mathbf{x}_{k-1} \\ \mathbf{0}_2 \end{bmatrix}\right),$$

where

$$\alpha_k = r_k' - r_{k-1}' - r_k(r_k - r_{k-1})$$

with

$$r_k' = \begin{bmatrix} a_{2k+1} & \dots & a_3 \end{bmatrix}\mathbf{x}_k.$$

In polynomial language this recursion can be written as

$$\mathbf{x}_{k+1}(t) = \frac{1}{\alpha_k}(t^2 - (r_k - r_{k-1})t + 1)\mathbf{x}_k(t) - t^2\mathbf{x}_{k-1}(t).$$

The recursion can be started with an empty $\mathbf{x}_0$, $r_0 = 0$ and $\mathbf{x}_1 = \frac{1}{a_1}$.

11.4. Schur-type algorithm

We define residual vectors $\mathbf{r}_k \in \mathbb{F}^{n-2k+1}$ of $\mathbf{x}_k$ as

$$\mathbf{r}_k = (r_{j,k})_{j=1}^{n-2k+1}, \quad r_{j,k} = \begin{bmatrix} a_{2k+j-2} & \dots & a_j \end{bmatrix}\mathbf{x}_k.$$

In particular we have $r_{1,k} = 1$ and $r_{2,k} = r_k$. From Theorem 11.3 we conclude the following.

Theorem 11.4 *For $k = 2, \dots, n-1$, the vectors $\mathbf{r}_k$ satisfy the recursion*

$$\mathbf{r}_{k+1} = \frac{1}{\alpha_k}\left((I_{++} + I_{--} - (r_{2,k} - r_{2,k-1})I_{+-})\mathbf{r}_k - I_{++--}\mathbf{r}_{k-1}\right).$$

With the projections P_k introduced in (4.5) this can be written in polynomial language,

$$\mathbf{r}_{k+1}(t) = \frac{1}{\alpha_k} P_{n-2k-1}\left((1 + t^{-2} - (r_{2,k} - r_{2,k-1})t^{-1})\mathbf{r}_k(t) - t^{-2}\mathbf{r}_{k-1}(t)\right).$$

Complexity Since the Levinson-type and Schur-type algorithms for skewsymmetric Toeplitz matrices have the same structure as the double-step algorithms for symmetric Toeplitz matrices, they have the same complexity which is $\frac{1}{2}n^2$ (M) plus $\frac{7}{8}n^2$ (A).

11.5. Solution of systems

For solving a system with a skewsymmetric Toeplitz coefficient matrix, it is recommendable, like for a symmetric Toeplitz system, to split the right-hand side into its symmetric and skewsymmetric parts and then to apply centrosymmetric bordering, as it was explained in Section 10.2. For correction we need the solution of equations

$$T_{2k}\mathbf{w}_{2k}^{\mp} = \rho_k(\mathbf{e}_{2k} \pm \mathbf{e}_1).$$

These solutions are given by the vectors $\mathbf{x}_k^{\pm}$ of (11.1),

$$\mathbf{w}_{2k}^{\mp} = \mathbf{x}_k^{+} \pm \mathbf{x}_k^{-}.$$

The structure of this method is in principle the same as for the symmetric case, so we expect the same complexity. However, we have in addition the amount for computing the vectors $\mathbf{w}_{2k}^{\mp}$ which consists in two vector additions. This adds the term $\frac{1}{4}n^2$ (A) to the overall complexity.

This additional term can be avoided if we find a bordering that uses the vectors $\mathbf{x}_k$ directly. For this we need solutions of systems of odd order. Let us introduce skewsymmetric vectors $\mathbf{c}_- \in \mathbb{F}_-^{n+1}$ and $\mathbf{d}_- \in \mathbb{F}_-^{n+1}$ by $\mathbf{c}_-(t) = (t-1)\mathbf{b}_+(t)$, and $\mathbf{d}_-(t) = (t+1)\mathbf{b}_-(t)$, where $\mathbf{b}^{\pm} = P_{\pm}\mathbf{b}$. We consider the equations

$$T_{n+1}\mathbf{p} = \mathbf{c}_- \quad \text{and} \quad T_{n+1}\mathbf{q} = \mathbf{d}_-,$$

where T_{n+1} is any $(n+1)\times(n+1)$ skewsymmetric Toeplitz extension of T_n. These two systems can be solved using centro-skewsymmetric bordering in which the vectors $\mathbf{x}_k$ are used for correction. We have to show how the (symmetric) solutions $\mathbf{p}$ and $\mathbf{q}$ are related to the solutions $\mathbf{z}_{\mp}$ of $T_n\mathbf{z}_{\mp} = \mathbf{b}_{\pm}$.

We have

$$T_{n+1}\left(\begin{bmatrix}\mathbf{z}_+\\ 0\end{bmatrix} + \begin{bmatrix}0\\ \mathbf{z}_+\end{bmatrix}\right) = \begin{bmatrix}\mathbf{b}_-\\ *\end{bmatrix} + \begin{bmatrix}*\\ \mathbf{b}_-\end{bmatrix}.$$

Since the nullspace of T_{n+1} is spanned by $\mathbf{x}_{m+1}$, we conclude from this that

$$(t+1)\mathbf{z}_+(t) = (t+1)\mathbf{q}(t) + \alpha_1\,\mathbf{x}_{m+1}(t) + \beta_1\,t\,\mathbf{x}_m(t).$$

Analogously,

$$(t-1)\mathbf{z}_-(t) = (t-1)\mathbf{p}(t) + \alpha_2\,\mathbf{x}_{m+1}(t) + \beta_2\,t\,\mathbf{x}_m(t).$$

It remains to find the coefficients α_i and β_i, $(i = 1, 2)$. For this we can put $t = 1$ and $t = -1$ or apply test functionals.

12. Split algorithms for Hermitian Toeplitz matrices

In this section, we consider Hermitian Toeplitz matrices $T_n = [a_{i-j}]_{i,j=1}^n$, $a_{-j} = \overline{a}_j$. Such matrices can be represented as $T_n = T_n^r + \mathrm{i}\,T_n^i$, where T_n^r is a real symmetric and T_n^i a real skewsymmetric

Toeplitz matrix. Thus real symmetric and real skewsymmetric Toeplitz matrices can be considered as special cases. Our aim is to design algorithms for strongly nonsingular Hermitian Toeplitz matrices that exploit the additional symmetry property of the matrix, like it was done in the previous sections for symmetric and skewsymmetric Toeplitz matrices. However, it turns out that a direct generalization is not possible, due to the different nature of splitting, which will be discussed next. Nevertheless algorithms with about the same gain in complexity as in the symmetric and skewsymmetric cases do exist. Applying them to the real symmetric or skewsymmetric cases we will obtain algorithms that are different from those presented in the previous sections.

Like in the symmetric case, we assume for the sake of simple notation that n is even, $n = 2m$. The case of odd n can be treated analogously.

12.1. Splitting

To begin with, let us explain the nature of splitting for a general centro-Hermitian matrix A. Remember that an $n \times n$ matrix A is said to be centro-Hermitian if $A^{\#} = J_n \overline{A} J_n = A$. Let $\mathbb{C}^n_*$ denote the set of conjugate-symmetric vectors in $\mathbb{C}^n$. The set $\mathbb{C}^n_*$ is not a subspace of $\mathbb{C}^n$ if this space is considered as a complex vector space, but it is a subspace if $\mathbb{C}^n$ is considered as a vector space over the reals. Furthermore, $\mathbb{C}^n = \mathbb{C}^n_* \oplus \mathrm{i}\mathbb{C}^n_*$. The subspaces $\mathbb{C}^n_*$ and $\mathrm{i}\mathbb{C}^n_*$ are invariant under a centro-Hermitian matrix A. Thus the system $A\mathbf{z} = \mathbf{b}$ is equivalent to the two systems $A\mathbf{z}_\pm = \mathbf{b}_\pm$ with conjugate-symmetric right-hand sides $\mathbf{b}_+ = \frac{1}{2}(\mathbf{b} + \mathbf{b}^{\#})$ and $\mathbf{b}_- = \frac{1}{2\mathrm{i}}(\mathbf{b} - \mathbf{b}^{\#})$ and $\mathbf{z} = \mathbf{z}_+ + \mathrm{i}\mathbf{z}_-$.

All what was just said applies to an Hermitian Toeplitz matrix T_n, because any Hermitian Toeplitz matrix is also centro-Hermitian (see Section 2.1).

12.2. Centro-Hermitian bordering

We know from Section 10.2 that for the solution of a system with a centrosymmetric coefficient matrix A symmetric bordering is more efficient than usual bordering. Thus it can be expected that for a centro-Hermitan coefficient matrix centro-Hermitian bordering leads to a reduction of complexity. This turns out to be true. However, the situation is somehow different here. The reason for this is that multiplication of a conjugate-symmetric vector by a number is conjugate-symmetric again only if the number is real.

Suppose that A is a centro-nonsingular, centro-Hermitian matrix, and a system $A\mathbf{z} = \mathbf{b}$ with a conjugate-symmetric right-hand side $\mathbf{b} \in \mathbb{C}^n_*$ has to be solved. Let A^c_k, $\mathbf{z}^c_k$, and $\mathbf{b}^c_k$ be defined as in Section 10.2. Then $A^c_{k+1} \begin{bmatrix} 0 \\ \mathbf{z}^c_k \\ 0 \end{bmatrix}$ is equal to $\mathbf{b}^c_{k+1}$, except for the first and last components. For correction we now need solutions $\mathbf{w}^{(l)}_{2k} \in \mathbb{C}^{2k}_*$ $(l = 1, 2)$ of equations

$$A^c_k \mathbf{w}^{(l)}_{2k} = \begin{bmatrix} \overline{\alpha}_{2k} \\ \mathbf{0} \\ \alpha_{2k} \end{bmatrix}$$

for *two* values $\alpha_{2k} = \alpha^{(1)}_k$ and $\alpha_{2k} = \alpha^{(2)}_k$ which are linearly independent over the reals. With these solutions one can find $\mathbf{z}^c_{k+1}$ via

$$\mathbf{z}^c_{k+1} = \begin{bmatrix} 0 \\ \mathbf{z}^c_k \\ 0 \end{bmatrix} + \xi^{(1)}_k \mathbf{w}^{(1)}_{2k+2} + \xi^{(2)}_k \mathbf{w}^{(2)}_{2k+2}$$

with real $\xi_k^{(1)}$ and $\xi_k^{(2)}$. Applying A_{k+1}^c to both sides we obtain that the latter equality holds if and only if

$$\xi_k^{(1)}\alpha_{k+1}^{(1)} + \xi_k^{(2)}\alpha_{k+1}^{(2)} = b_{m-k} - \beta_k,$$

where $\beta_k = \mathbf{g}_k^T \begin{bmatrix} 0 \\ \mathbf{z}_k^c \\ 0 \end{bmatrix}$ with $\mathbf{g}_k^T$ being the last row A_{k+1}^c.

This is equivalent to the following 2×2 system with a nonsingular coefficient matrix

$$\begin{bmatrix} \operatorname{Re}\alpha_{k+1}^{(1)} & \operatorname{Re}\alpha_{k+1}^{(2)} \\ \operatorname{Im}\alpha_{k+1}^{(1)} & \operatorname{Im}\alpha_{k+1}^{(2)} \end{bmatrix} \begin{bmatrix} \xi_k^{(1)} \\ \xi_k^{(2)} \end{bmatrix} = \begin{bmatrix} \operatorname{Re}(b_{m-k} - \beta_k) \\ \operatorname{Im}(b_{m-k} - \beta_k) \end{bmatrix}.$$

Complexity Let us compare the amount of centro-Hermitian bordering with ordinary bordering. First we have to compute 1 inner product of a general vector and a conjugate-symmetric vector of length $2k$. This costs $4k$ (RM) plus $6k$ (RA). To form the correction vector we have to evaluate 1 real linear combination of two conjugate-symmetric vectors, which costs $4k$ (RM) plus $2k$ (RA). Finally we have 1 addition of conjugate-symmetric vectors for the amount of $2k$ (RA). Since k runs from 1 to $m = n/2$ the total amount is n^2 (RM) plus $\frac{5}{4}n^2$ (RA). Recall that a general system is reduced to two systems with conjugate-symmetric right-hand sides. Hence solving a system with a nonsingular centro-Hermitian coefficient matrix costs $2\,n^2$ (RM) plus $\frac{5}{2}n^2$ (RA), compared with $4\,n^2$ (RM) plus $4\,n^2$ (RA) for ordinary bordering.

The problem with this approach is that for its application one has to compute two families of solution $\mathbf{w}_{2k}$, which seems to be too costly. We now show that in the case of a strongly nonsingular Hermitian Toeplitz matrix T_n one family of solutions is sufficient to carry out centro-Hermitian bordering.

Suppose we have one family of solutions of

$$T_k\mathbf{w}_k = \begin{bmatrix} \overline{\alpha}_k \\ \mathbf{0} \\ \alpha_k \end{bmatrix} \tag{12.1}$$

for the leading principal submatrices T_k ($k = 1, \ldots, n$). The idea is that we first solve recursively systems of the form

$$T_{2k}\widetilde{\mathbf{z}}_k^c = \mathbf{b}_k^c + \begin{bmatrix} \overline{\gamma}_k \\ \mathbf{0} \\ \gamma_k \end{bmatrix},$$

with arbitrary γ_k. For this we are looking for a recursion of the form

$$\widetilde{\mathbf{z}}_{k+1}^c = \begin{bmatrix} 0 \\ \widetilde{\mathbf{z}}_k^c \\ 0 \end{bmatrix} + c_k \begin{bmatrix} \mathbf{w}_{2k+1} \\ 0 \end{bmatrix} + \overline{c}_k \begin{bmatrix} 0 \\ \mathbf{w}_{2k+1} \end{bmatrix}. \tag{12.2}$$

Applying T_{2k+2} to both sides we see that the recursion is satisfied if we choose c_k as

$$c_k = \frac{b_{m-k+1} - \beta_k'}{\alpha_{2k+1}},$$

where $\beta_k' = \mathbf{g}_k'^T \begin{bmatrix} 0 \\ \widetilde{\mathbf{z}}_k^c \\ 0 \end{bmatrix}$ and $\mathbf{g}_k'^T$ is the last but one row of T_{2k+2}. Finally we correct the first and last components with the help of two linearly independent solutions of (12.1) once, for $k = n$. That means we need a second solution of (12.1) only at the very last step.

Complexity The amount for centro-Hermitian Toeplitz bordering, which was just explained, is approximately the same as for general centro-Hermitian bordering, since multiplication of a conjugate-symmetric vector by a complex number and its conjugate complex costs as much as multiplication of two conjugate-symmetric vectors by real numbers.

12.3. Recursion for solutions $\mathbf{w}_k$

We now show how solutions of equations (12.1) can be recursively found .

Theorem 12.1 *For* $k = 2, \dots, n-1$ *the recursion*

$$\mathbf{w}_{k+1}(t) = (\overline{\mu}_k + \mu_k t)\mathbf{w}_k(t) - t\mathbf{w}_{k-1}(t),$$

with $\mu_k = \frac{\alpha_{k-1}}{\alpha_k}$ *produces solutions of equation (12.1). Furthermore,*

$$\alpha_{k+1} = \overline{\mu}_k\beta_k + \alpha_{k-1} - \beta_{k-1},$$

where

$$\beta_k = \begin{bmatrix} a_k & \dots & a_1 \end{bmatrix}\mathbf{w}_k.$$

Proof. The assertion follows from the relations

$$T_{k+1}\begin{bmatrix} \mathbf{w}_k & 0 \\ 0 & \mathbf{w}_k \end{bmatrix} = \begin{bmatrix} \overline{\alpha}_k & \overline{\beta}_k \\ 0 & \overline{\alpha}_k \\ \mathbf{0} & \mathbf{0} \\ \alpha_k & 0 \\ \beta_k & \alpha_k \end{bmatrix} \quad \text{and} \quad T_{k+1}\begin{bmatrix} 0 \\ \mathbf{w}_{k-1} \\ 0 \end{bmatrix} = \begin{bmatrix} \overline{\beta}_{k-1} \\ \overline{\alpha}_{k-1} \\ \mathbf{0} \\ \alpha_{k-1} \\ \beta_{k-1} \end{bmatrix}. \quad \square$$

For a reason that will be clear at once, it is reasonable to compute also a vector $\mathbf{w}_{n+1}$ and to start the recursion with an empty $\mathbf{w}_1$ and $\mathbf{w}_2 = \begin{bmatrix} -\mathrm{i} \\ \mathrm{i} \end{bmatrix}$. If we choose $\mu_2 = (a_0 - a_1)^{-1}$, then $\mathbf{w}_3$ defined by $\mathbf{w}_3(t) = (\mu_2 t + \overline{\mu}_2)\mathbf{w}_2(t)$ is a solution of an equation (12.1) for $k = 3$. With this initialization all $\mathbf{w}_k(t)$ will have the property $\mathbf{w}_k(1) = 0$.

12.4. Computing a second family of solutions $\mathbf{w}_k$

We show how a second family of solutions of (12.1) can be constructed that is linearly independent of the first one. Recall that for bordering according to (12.2) we need a second solution only for $k = n$. However, the whole family will be needed to build factorizations, as discussed in the next section.

Since $\mathbf{w}_k(1) = 0$ we can define vectors $\mathbf{q}_{k-1}$ for $k = 2, \dots, n+1$ by

$$\mathbf{q}_{k-1}(t) = \frac{\mathrm{i}}{t-1}\mathbf{w}_k(t). \tag{12.3}$$

Obviously, these vectors are conjugate-symmetric and

$$\mathbf{w}_k = \mathrm{i}\left(\begin{bmatrix} \mathbf{q}_{k-1} \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ \mathbf{q}_{k-1} \end{bmatrix}\right).$$

The following can easily be checked.

Lemma 12.2 *For* $k = 1, \dots, n$, *the vectors* $\mathbf{q}_k$ *satisfy the equations*

$$T_k\mathbf{q}_k = \theta_k\mathbf{e}, \tag{12.4}$$

where θ_k *are nonzero real numbers and* $\mathbf{e} = (1, 1, \dots, 1)$.

Proposition 12.3 *For $k = 2, \dots, n+1$, let $\mathbf{w}_k$ be a solution of (12.1), $\mathbf{q}_{k-1}$ be given by (12.3), and*

$$\theta_k = \begin{bmatrix} a_{k-1} \dots a_0 \end{bmatrix} \mathbf{q}_k.$$

Then the vector $\widetilde{\mathbf{w}}_k$ defined by

$$\widetilde{\mathbf{w}}_k(t) = (1+t)\mathbf{q}_{k-1}(t) - \frac{2\theta_{k-1}}{\theta_k}\,\mathbf{q}_k(t)$$

is a solution of

$$T_k \widetilde{\mathbf{w}}_k = \begin{bmatrix} \overline{\alpha}_k \mathrm{i} \\ \mathbf{0} \\ -\alpha_k \mathrm{i} \end{bmatrix}.$$

Proof. We have

$$T_k \begin{bmatrix} \mathbf{q}_{k-1} & 0 \\ 0 & \mathbf{q}_{k-1} \end{bmatrix} = \begin{bmatrix} \theta_{k-1} & \bar{s}_{k-1} \\ \theta_{k-1}\mathbf{e} & \theta_{k-1}\mathbf{e} \\ s_{k-1} & \theta_{k-1} \end{bmatrix},$$

where

$$s_k = \begin{bmatrix} a_k \dots a_1 \end{bmatrix} \mathbf{q}_k. \tag{12.5}$$

Combining these relations and taking (12.3) and(12.4) into account we obtain

$$T_k \mathbf{w}_k = \begin{bmatrix} -\mathrm{i}(\bar{s}_{k-1} - \theta_{k-1}) \\ \mathbf{0} \\ \mathrm{i}(s_{k-1} - \theta_{k-1}) \end{bmatrix} \quad \text{and} \quad T_k \widetilde{\mathbf{w}}_k = \begin{bmatrix} \bar{s}_{k-1} - \theta_{k-1} \\ \mathbf{0} \\ s_{k-1} - \theta_{k-1} \end{bmatrix},$$

which is just the assertion. □

Clearly, the vectors $\mathbf{w}_k$ and $\widetilde{\mathbf{w}}_k$ are linearly independent, since the right-hand sides of the corresponding equations have this property.

Now we have all ingredients for a Levinson-type algorithm that computes the solution of an Hermitian Toeplitz system with conjugate-symmetric right-hand side. First we compute the family of solutions $\mathbf{w}_k$ by Theorem 12.1, then we apply the Toeplitz centro-Hermitian bordering, and finally we compute $\widetilde{\mathbf{w}}_n$ (use (12.3) for $k = n, n+1$) to correct the first and last components.

Complexity Let us compare the complexity of this with the classical Levinson algorithm. In each step of the recursion according to Theorem 12.1 we have first 1 inner product of a general vector and a conjugate-symmetric vector, which requires $2k$ (RM) plus $3k$ (RA) (see Section 2.2). Then we have to multiply a conjugate-symmetric vector by a complex number and by its conjugate complex. This is equivalent to 4 real number times vector multiplications and 4 real vector additions, where the vectors are symmetric or skewsymmetric, which requires $2k$ (RM) plus $2k$ (RA). In addition we have 2 complex vector additions with conjugate-symmetric sums which costs $2k$ (RA). This results in a total amount of $2\,n^2$ (RM) plus $\frac{7}{2}\,n^2$ (RA), compared with $4\,n^2$ (RM) plus $4\,n^2$ (RA) for the classical Levinson recursion.

If an Hermitian Toeplitz system is solved with the help of Theorem 12.1 and Toeplitz centro-Hermitian bordering, then the amount will be $4\,n^2$ (RM) and $6\,n^2$ (RA). The amount for computing the vector $\widetilde{\mathbf{w}}_n$ is $O(n)$ and can be neglected. Solving a system by the classical Levinson algorithm and bordering costs $8\,n^2$ (RM) and $8\,n^2$ (RA). Thus we have 50% savings in multiplications and 25% savings in additions.

12.5. Solutions for the right-hand side **e**

If we want to find two linearly independent families of solutions of (12.1), then it is reasonable to compute the solutions $\mathbf{q}_k$ of (12.4) instead of the $\mathbf{w}_k$ recursively. The following theorem is an immediate consequence of (12.3) and Theorem 12.1.

Theorem 12.4 *For $k = 1, \dots, n-1$, the polynomials $\mathbf{q}_k(t)$ satisfy the recursion*

$$\mathbf{q}_{k+1}(t) = (\nu_k + \overline{\nu}_k t)\mathbf{q}_k(t) - t\mathbf{q}_{k-1}(t),$$

where

$$\nu_k = \frac{s_{k-1} - \theta_{k-1}}{s_k - \theta_k}$$

and s_k is given by (12.5). Furthermore,

$$\theta_{k+1} = (\nu_k + \overline{\nu}_k)\theta_k - \theta_{k-1}.$$

The recursion can be started with an empty $\mathbf{q}_0$, $\mathbf{q}_1 = 1$ and $\nu_1 = a_0 - \overline{a}_1$.

The vectors $\mathbf{q}_k$ provide two families of solutions of (12.1) as the following proposition shows. It follows from the discussion above, but can also be verified directly.

Proposition 12.5 *Let the vectors $\mathbf{w}_k^{(l)}$, $l = 1, 2$, be defined by the solutions $\mathbf{q}_k$ via*

$$\mathbf{w}_k^{(1)}(t) = \mathrm{i}(1-t)\mathbf{q}_{k-1}(t),$$
$$\mathbf{w}_k^{(2)}(t) = (1+t)\mathbf{q}_{k-1}(t) - \frac{2\theta_{k-1}}{\theta_k}\,\mathbf{q}_k(t).$$

Then $\mathbf{w}_k^{(l)}$ are (linearly independent over the reals) solutions of (12.1) with $\alpha_k = \alpha_k^l$ and

$$\alpha_k^1 = \mathrm{i}(s_{k-1} - \theta_{k-1}), \quad \alpha_k^2 = s_{k-1} - \theta_{k-1},$$

where s_k is defined by (12.5).

Instead of the vector $\mathbf{w}_k^{(2)}$ the vector $\mathbf{w}_k^{(3)}$ defined by

$$\mathbf{w}_k^{(3)}(t) = t\mathbf{q}_{k-2}(t) - \frac{\theta_{k-2}}{\theta_k}\,\mathbf{q}_k(t)$$

can be used. In fact, this vector solves also the equation (12.1) with $\alpha_k = \alpha_k^3$ and

$$\alpha_k^3 = s_{k-2} - \theta_{k-1}.$$

The advantage to take $\mathbf{w}_k^{(3)}$ instead of $\mathbf{w}_k^{(2)}$ is that its computation requires less additions. However, the vectors $\mathbf{w}_k^{(1)}$ and $\mathbf{w}_k^{(3)}$ are only linearly independent if the number $\frac{s_{k-2}-\theta_{k-2}}{s_{k-1}-\theta_{k-1}}$ is not purely imaginary.

12.6. Recursion for the residuals

All Levinson-type recursions described in this section have Schur counterparts. We here describe only the Schur counterpart of the recursion of the solutions $\mathbf{q}_k$.

Let T_k^+ be defined by (4.1) and

$$T_k^+ \mathbf{q}_k = \mathbf{s}_k. \tag{12.6}$$

If $\mathbf{s}_k = (s_{ik})_{i=1}^{n-k+1}$, then, in particular, $s_{1,k} = \theta_k$ and $s_{2,k} = s_k$. From Theorem 12.4 we immediately obtain the following.

Theorem 12.6 *For $k = 2, \dots, n-1$, the residual vectors $\mathbf{s}_k$ satisfy the recursion*

$$\mathbf{s}_{k+1} = (\nu_k I_+ + \overline{\nu}_k I_-)\mathbf{s}_k - I_{+-}\mathbf{s}_{k-1},$$

where

$$\nu_k = \frac{s_{2,k-1} - s_{1,k-1}}{s_{2,k} - s_{1,k}},$$

with $I_\pm$ being defined in (4.3).

Complexity In each step of the algorithm we have to multiply a complex vector by a complex number and by its conjugate complex number. This is equivalent to 4 real number times vector multiplications and 4 real vector additions and requires $4k$ (RM) plus $4k$ (RA). In addition we have 2 complex vector additions requiring $4k$ (RA). This results in a total amount of $2\,n^2$ (RM) plus $4\,n^2$ (RA).

Corollary 12.7 *With the help of the residuals $\mathbf{s}_k$ we can find the residuals $\mathbf{t}_k^{(l)}$ of the solutions $\mathbf{w}_k^{(l)}$ ($l = 1, 2, 3$) given by*

$$T_k^+ \mathbf{w}_k^{(l)} = \mathbf{t}_k^{(l)}$$

via the relations

$$\mathbf{t}_k^{(1)} = \mathrm{i}(I_+\mathbf{s}_{k-1} - I_-\mathbf{s}_{k-1}),$$

$$\mathbf{t}_k^{(2)} = I_+\mathbf{s}_{k-1} + I_-\mathbf{s}_{k-1} - \frac{2s_{1,k-1}}{s_{1,k}}\,\mathbf{s}_k,$$

$$\mathbf{t}_k^{(3)} = I_{+-}\mathbf{s}_{k-2} - \frac{s_{1,k-2}}{s_{1,k}}\,\mathbf{s}_k.$$

Note that these vectors are needed only for even k.

13. Butterfly factorization

The triangular factorization of a matrix has the disadvantage that properties like centrosymmetry are not inherited in the factors. But there is another type of factorization which has this property. We call it *butterfly factorization* in view of the shape of the factors. This kind of factorization is the subject of this section. It turns out that butterfly factorization is the background for the split Levinson-type and the split Schur-type algorithms, like triangular factorization is for the classical Levinson and Schur algorithms.

13.1. Z-, W-, and X-matrices

A matrix $A = [a_{ij}]_{i,j=1}^n$ is called a *W-matrix* (or a bow tie matrix) if $a_{ij} = 0$ for all (i, j) for which $i > j$ and $i + j > n + 1$ or $i < j$ and $i + j \le n$. The matrix A will be called a *unit W-matrix* if in addition $a_{ii} = 1$ and $a_{i,n+1-i} = 0$ for $i = 1, \dots, n$. The transpose of a W-matrix is called a *Z-matrix* (or hourglass matrix). A matrix which is both a Z- and a W-matrix will be called an *X-matrix*. A matrix which is either a Z-matrix or a W-matrix will be called *butterfly matrix*.

These names are suggested by the shapes of the set of all possible positions for nonzero entries, which are as follows:

$$W = \begin{bmatrix} \bullet & & & & \bullet \\ \bullet & \circ & & \circ & \bullet \\ \bullet & \circ & \circ & \circ & \bullet \\ \bullet & \circ & \bullet & \circ & \bullet \\ \bullet & \bullet & & \bullet & \bullet \\ \bullet & & & & \bullet \end{bmatrix}, \quad Z = \begin{bmatrix} \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ & \circ & \circ & \circ & \bullet & \\ & & \circ & \bullet & & \\ & & \bullet & \circ & & \\ & \bullet & \circ & \circ & \circ & \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \end{bmatrix},$$

$$X = \begin{bmatrix} \bullet & & & & \bullet \\ & \bullet & & \bullet & \\ & & \bullet\ \bullet & & \\ & & \bullet\ \bullet & & \\ & \bullet & & \bullet & \\ \bullet & & & & \bullet \end{bmatrix}.$$

For an $n \times n$ matrix A, the following facts are easily verified.

- If A is a Z-, W- or X-matrix, then A^J is a Z-, W- or X-matrix again, respectively.
- The inverse of a nonsingular Z-, W- or X-matrix is again a Z-, W- or X-matrix, respectively.
- If Z is a Z- and X is an X-matrix of the same order, then ZX and XZ are Z-matrices.
- If Z is a nonsingular Z-matrix, then there exist unique nonsingular X-matrices X_1 and X_2 such that ZX_1 and X_2Z are unit Z-matrices.

From now on we assume again, for simplicity of notation, that the order n of the matrices is even, $n = 2m$.

In order to describe X-matrices we introduce a notation that is motivated by the "diag" notation for diagonal matrices. If $M_k = \begin{bmatrix} \alpha_k & \beta_k \\ \gamma_k & \delta_k \end{bmatrix}$ $(k = 1, \dots, m)$, then we set

$$\operatorname{xma}(M_k)_{k=1}^m = \begin{bmatrix} \alpha_m & & & & \beta_m \\ & \ddots & & \iddots & \\ & & \alpha_1 & \beta_1 & \\ & & \gamma_1 & \delta_1 & \\ & \iddots & & \ddots & \\ \gamma_m & & & & \delta_m \end{bmatrix}.$$

Clearly, $\operatorname{xma}(M_k)_{k=1}^m$ is nonsingular if and only if all M_k are nonsingular and

$$\left(\operatorname{xma}(M_k)_{k=1}^m\right)^{-1} = \operatorname{xma}(M_k^{-1})_{k=1}^m.$$

13.2. ZW-factorization and centro-nonsingularity

A representation of the nonsingular matrix A in the form $A = ZXW$ in which Z is a Z-, X is an X- and W is a W-matrix is called *ZW-factorization*. If Z and W are unit, then the factorization is referred to as *unit*. Analogously, a WZ-factorization is defined. A factorization which is either a ZW- or WZ-factorization is called *butterfly factorization*. The following is the analogue of Proposition 5.1.

Proposition 13.1 *A necessary and sufficient condition for a matrix A to admit a ZW-factorization is that A is centro-nonsingular. Among all ZW-factorization there is a unique unit one.*

If A is nonsingular and $A = ZXW$ a ZW-factorization, then $A^{-1} = W^{-1}X^{-1}Z^{-1}$ is WZ-factorization of A^{-1}. Conversely, any WZ-factorization of A^{-1} produces a ZW-factorization of A.

13.3. Symmetry properties

Let $A = ZXW$ be the unit ZW-factorization of A. Then we immediately obtain a unit ZW-factorization of A^J as $A^J = Z^JX^JW^J$. Taking the uniqueness of the unit ZW-factorization into account we conclude the following.

1. If A is centrosymmetric, then Z, X, and W are also centrosymmetric.
2. If A is centro-skewsymmetric, then Z and W are centrosymmetric and X is centro-skewsymmetric. The later means that X is a skewsymmetric antidiagonal matrix.
3. If A is centro-Hermitian, then Z, X, and W are also centro-Hermitian.

In addition, the unit ZW-factorization has similar properties as the unit LU-factorization.

1. If A is symmetric, then $W = Z^T$ and X is symmetric.
2. If A is skewsymmetric, then $W = Z^T$ and X is a skewsymmetric antidiagonal matrix.
3. If A is Hermitian, then $W = Z^*$ and X is Hermitian.

13.4. Solution of centrosymmetric Z-systems

The advantage of the ZW-factorization over the LU-factorization is that in the former symmetry properties are inherited in the factors. Now we show how this symmetry properties can be exploited. We describe this for a centrosymmetric Z-system. For a W-system the situation is analogous.

Let Z be a centrosymmetric Z-matrix. Then Z is of the form

$$Z = \begin{bmatrix} L_0^J & J_m L_1 \\ L_1 J_m & L_0 \end{bmatrix}, \tag{13.1}$$

where L_0 and L_1 are $m \times m$ lower triangular matrices. The following is now easily checked.

Proposition 13.2 *For Z of the form (13.1), the solution of a system $Z\mathbf{u} = \mathbf{b}$ with a symmetric or skewsymmetric right-hand side $\mathbf{b} = \begin{bmatrix} \pm\mathbf{c}^J \\ \mathbf{c} \end{bmatrix}$ is given by $\mathbf{u} = \begin{bmatrix} \pm\mathbf{v}^J \\ \mathbf{v} \end{bmatrix}$, where $\mathbf{v}$ is the solution of the triangular system*

$$(L_0 \pm L_1)\mathbf{v} = \mathbf{c}. \tag{13.2}$$

Thus solving a centrosymmetric Z-system requires to solve 2 triangular systems of half size. This reduces the number of operations from $\frac{1}{2}n^2$ (M) plus $\frac{1}{2}n^2$ (A) (needed for an unstructured Z-system) to $\frac{1}{4}n^2$ (M) plus $\frac{1}{4}n^2$ (A). However, we have the additional amount of forming the matrices $L_0 \pm L_1$, which is $\frac{1}{4}n^2(A)$. We will see that in all cases we are interested in this additional amount can be avoided.

13.5. Unit ZW-factorization of skewsymmetric Toeplitz matrices

We start with the case of a skewsymmetric Toeplitz matrix, since only for this case we will compute the unit ZW-factorization. In the cases of symmetric and Hermitian Toeplitz matrices we will consider some modifications.

We show how the Schur-type algorithm described in Theorem 11.4 provides the unit ZW-factorization of a skewsymmetric Toeplitz matrix T_n. For simplification of notation, let us agree upon identifying any symmetric vector $\mathbf{u} \in \mathbb{F}^{n-2k}$ with the vector $\begin{bmatrix} \mathbf{0}_k \\ \mathbf{u} \\ \mathbf{0}_k \end{bmatrix} \in \mathbb{F}^n$. From the vectors (11.1) we form the matrix

$$V = \begin{bmatrix} \mathbf{x}_m^+ & \dots & \mathbf{x}_1^+ & \mathbf{x}_1^- & \dots & \mathbf{x}_m^- \end{bmatrix}.$$

Obviously, V is a centrosymmetric W-matrix. We investigate now the matrix T_nV. For $n = 6$ we have

$$T_nV = \begin{bmatrix} 0 & -1 & -r_{21} & -r_{31} & -r_{22} & -1 \\ 0 & 0 & -1 & -r_{21} & -1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & r_{21} & 1 & 0 & 0 \\ 1 & r_{22} & r_{31} & r_{21} & 1 & 0 \end{bmatrix}.$$

We see that $Z = T_6VJ_6 \operatorname{diag}(-I_3, I_3)$ is a unit centrosymmetric Z-matrix and conclude that this matrix is just the Z-factor of the unit ZW-factorization of T_6. This generalizes to arbitrary $n = 2m$. In particular, the Z-factor of the unit ZW-factorization of T_n is given by

$$Z = T_nVJ_n \operatorname{diag}(-I_m,\ I_m).$$

The middle factor X can be extracted from V. We obtain

$$X = \operatorname{xma}\left(\begin{bmatrix} 0 & -\xi_k^{-1} \\ \xi_k^{-1} & 0 \end{bmatrix}\right)_{k=1}^{m}, \tag{13.3}$$

where ξ_k is the last component of $\mathbf{x}_k$. Recall from Theorem 11.3 that the numbers ξ_k satisfy the recursion $\xi_{k+1} = \alpha_k^{-1}\xi_k$.

Theorem 13.3 *The Z-factor of the unit ZW-factorization $T_n = ZXZ^T$ is of the form (13.1), where L_0 and L_1 are given by*

$$L_0 = \begin{bmatrix} r_{1,1} & & & \\ r_{2,1} & r_{1,2} & & \\ \vdots & \vdots & \ddots & \\ r_{m,1} & r_{m,2} & \dots & r_{1,m} \end{bmatrix}, \quad L_1 = S_mL_0.$$

Here the $r_{i,k}$ are the residuals defined in Section 11.4, and S_m is the forward shift in $\mathbb{F}^m$ introduced in (6.2). The X-factor is given by (13.3).

Note that the factors of the unit WZ-factorization of T_n^{-1}, $T_n^{-1} = W\widetilde{X}W^T$ are given by $W = VJ_n\operatorname{diag}(-I_m, I_m)X$ and $\widetilde{X} = X^{-1}$.

Complexity The ZW-factorization can be used to solve a system $T_n\mathbf{z} = \mathbf{b}$. Of course, we first split the right-hand side into its symmetric and skewsymmetric parts. For each part we have to solve a centrosymmetric Z-system. This reduces to two systems (13.2) of size m, which in our case turn into

$$(I_m \pm S_m)L_0\mathbf{v} = \mathbf{c}.$$

We first solve $(I_m \pm S_m)\mathbf{d} = \mathbf{c}$ and then $L_0\mathbf{v} = \mathbf{d}$. Since the system with the coefficient matrix $I_m \pm S_m$ can be solved with m additions there is no additional amount for forming $L_0 \pm L_1$. So the complexity for solving both the Z- and the corresponding W-system is $\frac{1}{2}n^2$ (M) plus $\frac{1}{2}n^2$ (A). If we add this to the amount for the Schur-type algorithm we obtain a complexity of n^2 (M) plus $\frac{11}{8}n^2$ (A).

13.6. Split ZW-factorization of centrosymmetric matrices

The split Schur algorithm for symmetric Toeplitz matrices does not produce the unit ZW-factorization but a version of it which we call *split factorization*.

We say that a matrix A of order $n = 2m$ is *split* if the first m columns of A are skewsymmetric and the last m columns are symmetric. Split and centrosymmetric matrices are closely related. In fact, let Λ denote the matrix

$$\Lambda = \operatorname{xma}\left(\begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix}, \dots, \begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix}\right).$$

Then it is easy to check that A is centrosymmetric if and only $A\Lambda$ is split. A split Z-matrix is called *unit split Z-matrix* if the last nonzero element in each row is equal to 1, i.e.

$$a_{i,n+1-i} = a_{m+i,m+i} = 1 \quad \text{for } i = 1, 2, \dots, m.$$

A ZW-factorization of A of the form $A = Z_s X_s Z_s^T$ in which Z_s is unit split will be called *unit split ZW-factorization*. The unit ZW-factorization is easily transformed into the unit split ZW-factorization, and vice versa. In fact, let $A = ZXZ^T$ be the unit ZW-factorization, then the factors of the unit split factorization are given by

$$Z_s = Z\Lambda \qquad X_s = \frac{1}{4}\Lambda X \Lambda.$$

The X-factor X_s of the split factorization is diagonal. This follows from the fact that the X-factor of the unit ZW-factorization is built from blocks of the form $\begin{bmatrix} a & b \\ b & a \end{bmatrix}$ and that

$$\begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} a & b \\ b & a \end{bmatrix}\begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix}$$

is diagonal.

We show how a system with a split Z-coefficient matrix can be solved. A split Z-matrix Z_s is of the form

$$Z_s = \begin{bmatrix} -L_-^J & J_m L_+ \\ L_- J_m & L_+ \end{bmatrix}, \tag{13.4}$$

where $L_\pm$ are $m \times m$ lower triangular matrices.

Proposition 13.4 *Let Z_s be the split Z-matrix (13.4). The system $Z_s\mathbf{u} = \mathbf{b}$ with symmetric or skewsymmetric right-hand side $\mathbf{b} = \begin{bmatrix} \pm\mathbf{c}^J \\ \mathbf{c} \end{bmatrix}$ is equivalent to $L_+\mathbf{v}_+ = \mathbf{c}$ or $L_-\mathbf{v}_- = \mathbf{c}$, where $\mathbf{u} = \begin{bmatrix} \mathbf{0} \\ \mathbf{v}_+ \end{bmatrix}$ or $\mathbf{u} = \begin{bmatrix} \mathbf{v}_-^J \\ \mathbf{0} \end{bmatrix}$, respectively.*

Thus the amount for solving a split Z-system reduces by 50% compared with an unstructured Z-system.

13.7. Solution of symmetric Toeplitz systems

It is obvious that the Schur-type algorithms for a symmetric Toeplitz matrix T_n described in Theorems 10.7 and 10.8 and their counterparts for the residuals of the vectors $\mathbf{w}_k^-$ produce the Z-factor of the unit split factorization $T_n = Z_s X_s Z_s^T$. This Z-factor is given by (13.4), where

$$L_+ = \begin{bmatrix} t_{1,2} & & & \\ t_{2,2} & t_{1,4} & & \\ \vdots & \vdots & \ddots & \\ t_{m,2} & t_{m-1,4} & \dots & t_{1,2m} \end{bmatrix}$$

and L_- is analogously defined. The X-factor is the diagonal matrix

$$X_s = \operatorname{diag}\left(\omega_{2m}^-, \dots, \omega_2^-, \omega_2, \dots, \omega_{2m}\right),$$

where ω_{2k} is the last component of $\mathbf{w}_{2k}$ and ω_{2k}^- the last component of $\mathbf{w}_{2k}^-$. The ω_{2k} can be found recursively by $\omega_{2k+2} = \frac{1}{\sigma_k}\,\omega_{2k}$, where σ_k is given in Theorem 10.8.

In Sections 10.7 and 10.8 it was shown that the solution of a symmetric Toeplitz system with a general right-hand side can be reduced to two systems with symmetric right-hand sides. Since for a symmetric right-hand side we obtain a symmetric solution $\mathbf{z}_+$, the first m components of $Z_s^T\mathbf{z}_+$ are zero. Hence for the product $Z_sX_sZ_s^T\mathbf{z}_+$ only the $m\times m$ matrices L_+ and diag $(\omega_2, \dots, \omega_{2m})$ are relevant. That means we have to compute only the matrix L_+ and the numbers ω_{2k}.

Complexity The solution of $T_n\mathbf{z} = \mathbf{b}$ reduces to 2 systems with coefficient matrix L_+ and 2 systems with coefficient matrix L_+^T. Thus the amount is $\frac{1}{2}n^2$ (M) plus $\frac{1}{2}n^2$ (A). Together with the double-step Schur-type algorithm this results in n^2 (M) plus $\frac{11}{8}\,n^2$ (A), compared with 2 n^2 (M) plus 2 n^2 (A) for the classical Schur algorithm and LU-factorization (cf. Sections 4.3 and 5.4).

13.8. Solution of Z-systems with conjugate symmetries

For centro-Hermitian Z-systems the situation is similar to that for centrosymmetric Z-systems. These systems can be reduced to triangular systems. Also in this case it is convenient to consider the systems in their split form.

A square matrix will be called *column conjugate-symmetric* if all columns are conjugate-symmetric.

We introduce the matrix

$$\Sigma = \operatorname{xma}\left(\begin{bmatrix} -\mathrm{i} & 1 \\ \mathrm{i} & 1 \end{bmatrix}, \dots, \begin{bmatrix} -\mathrm{i} & 1 \\ \mathrm{i} & 1 \end{bmatrix}\right).$$

Then it is easy to check that A is centro-Hermitian if and only if $A\Sigma$ is column conjugate-symmetric.

A column conjugate-symmetric Z-matrix is called *unit column conjugate-symmetric Z-matrix* if the X-matrix built from its diagonal and antidiagonal is equal to Σ. A centro-Hermitian ZW-factorization $A = ZXZ^*$ can be transformed into a ZW-factorization $A = Z_hX_hZ_h^*$ in which Z_h is unit column conjugate-symmetric. We will call this factorization *unit column conjugate-symmetric ZW-factorization*. Concerning the factor X_h we obtain $X_h = \frac{1}{4}\,\Sigma^*X\Sigma$. For X is Hermitian, X_h is Hermitian. Moreover, X_h is real. In fact, we have

$$\overline{X}_h = \frac{1}{4}\,\overline{\Sigma^*X\Sigma} = \frac{1}{4}\,\overline{\Sigma}^*J_nXJ_n\overline{\Sigma} = \frac{1}{4}\,\Sigma^*X\Sigma = X_h.$$

Obviously, a column conjugate-symmetric Z-matrix Z_h has the form

$$Z_h = \begin{bmatrix} J_m\overline{L}_1J_m & J_m\overline{L}_0 \\ L_1J_m & L_0 \end{bmatrix}, \tag{13.5}$$

where $L_0 = L_{0,r} + \mathrm{i}L_{0,i}$, $L_1 = L_{1,r} + \mathrm{i}L_{1,i}$ are lower triangular matrices with $L_{j,r}, L_{j,i}$ $(j = 1, 2)$ being real matrices. In the case where Z_h is unit column conjugate-symmetric $L_{0,r}$ and $L_{1,i}$ are unit, and $L_{0,i}$ and $L_{1,r}$ have zeros on their main diagonal.

From the representation (13.5) it can be seen that Z_h transforms real vectors to conjugate-symmetric vectors, so that the solution of $Z_h\mathbf{u} = \mathbf{b}_+$ with conjugate-symmetric $\mathbf{b}_+$ is real.

Suppose that $\mathbf{b}_+ = \begin{bmatrix} \mathbf{c}^{\#} \\ \mathbf{c} \end{bmatrix}$, $\mathbf{c} = \mathbf{c}_r + \mathrm{i}\mathbf{c}_i$ with $\mathbf{c}_r,\ \mathbf{c}_i \in \mathbb{R}^m$, and let $\mathbf{u} = \begin{bmatrix} \mathbf{v} \\ \mathbf{w} \end{bmatrix}$ with $\mathbf{v},\ \mathbf{w} \in \mathbb{R}^m$ be the solution of $Z_h\mathbf{u} = \mathbf{b}_+$. Then we have

$$Z_h \begin{bmatrix} \mathbf{v} \\ \mathbf{w} \end{bmatrix} = \begin{bmatrix} \mathbf{c}_r^J \\ \mathbf{c}_r \end{bmatrix} + \mathrm{i} \begin{bmatrix} -\mathbf{c}_i^J \\ \mathbf{c}_i \end{bmatrix},$$

which is equivalent to

$$L_{1,i}J_m\mathbf{v} + L_{0,i}\mathbf{w} = \mathbf{c}_i,$$
$$L_{1,r}J_m\mathbf{v} + L_{0,r}\mathbf{w} = \mathbf{c}_r.$$

This system can be written as a real Z-system

$$Z_h' \begin{bmatrix} \mathbf{v} \\ \mathbf{w} \end{bmatrix} = \begin{bmatrix} \mathbf{c}_i^J \\ \mathbf{c}_r \end{bmatrix},$$

where

$$Z_h' = \begin{bmatrix} J_mL_{1,i}J_m & J_mL_{0,i} \\ L_{1,r}J_m & L_{0,r} \end{bmatrix}.$$

If Z_h is unit column conjugate-symmetric then Z_h' is a real unit Z-matrix.

Complexity We just have shown that the solution of system with a unit column conjugate-symmetric Z-coefficient matrix reduces, after splitting the right-hand side as explained in Section 12.1, to 2 real unit Z-systems. That means that the solution requires n^2 (RM) plus n^2 (RA). The same is true for a system with the adjoint coefficient matrix.

13.9. Solution of Hermitian Toeplitz systems

The Schur-type algorithm for an Hermitian Toeplitz matrix T_n described by Theorem 12.6 produces the Z-factor of a column conjugate-symmetric factorization $T_n = Z_hX_hZ_h^*$. Indeed, we find the residual vectors $\mathbf{t}_k^{(1)}$ and $\mathbf{t}_k^{(2)}$ of the solutions $\mathbf{w}_k^{(1)}$ and $\mathbf{w}_k^{(2)}$ defined in Proposition 12.5 via the relations of Corollary 12.7 from the residual vectors of the solutions $\mathbf{q}_k$.

Now the factor Z_h is given by (13.5), where L_0, L_1 are the lower triangular matrices the kth columns ($k = 1, \dots, m$) of which are

$$\begin{bmatrix} \mathbf{0}_{k-1} \\ \mathbf{t}_k^{(1)} \end{bmatrix}, \quad \begin{bmatrix} \mathbf{0}_{k-1} \\ \mathbf{t}_k^{(2)} \end{bmatrix},$$

respectively.

It remains to describe the middle factor X_h. Let ν_k^1, ν_k^2, denote the last components of $\mathbf{w}_k^{(1)}$, $\mathbf{w}_k^{(2)}$,

$$\nu_k^j = \mathbf{e}_k^T\mathbf{w}_k^{(j)} \quad \text{for } j = 1, 2.$$

We take advantage of the relation

$$\begin{bmatrix} \overline{z}_1 & \overline{z}_2 \\ z_1 & z_2 \end{bmatrix} = \begin{bmatrix} -\mathrm{i} & 1 \\ \mathrm{i} & 1 \end{bmatrix} \begin{bmatrix} \operatorname{Im} z_1 & \operatorname{Im} z_2 \\ \operatorname{Re} z_1 & \operatorname{Re} z_2 \end{bmatrix} \tag{13.6}$$

for complex numbers z_1, z_2, and observe that Z_hX_α with

$$X_\alpha = \operatorname{xma}\left(\begin{bmatrix} \operatorname{Im}\alpha_k^1 & \operatorname{Im}\alpha_k^2 \\ \operatorname{Re}\alpha_k^1 & \operatorname{Re}\alpha_k^2 \end{bmatrix}^{-1} \right)_{k=1}^{m}$$

is a unit column conjugate-symmetric Z-matrix. From the uniqueness of the unit column conjugate-symmetric ZW-factorization we now conclude

$$X_h = \frac{1}{2}\,\mathrm{xma}\left(\begin{bmatrix} \mathrm{Im}\,\nu_k^1 & \mathrm{Im}\,\nu_k^2 \\ \mathrm{Re}\,\nu_k^1 & \mathrm{Re}\,\nu_k^2 \end{bmatrix}^{-1} \begin{bmatrix} \mathrm{Im}\,\alpha_k^1 & \mathrm{Im}\,\alpha_k^2 \\ \mathrm{Re}\,\alpha_k^1 & \mathrm{Re}\,\alpha_k^2 \end{bmatrix}^{-T}\right)_{k=1}^{m}.$$

(The factor $\frac{1}{2}$ appears in view of this factor in $\Sigma^{-1} = \frac{1}{2}\,\Sigma^*$.)

From Proposition 12.5 we know

$$\alpha_k^1 = \mathrm{i}(s_{2,k-1} - s_{1,k-1}), \quad \alpha_k^2 = s_{2,k-1} - s_{1,k-1},$$

where $\mathbf{s}_k$ is defined by (12.6). For the last components of $\mathbf{w}_k^{(j)}$ we obtain

$$\nu_k^1 = -\mathrm{i}\,\eta_{k-1}, \quad \nu_k^2 = \eta_{k-1} - 2\eta_k \frac{s_{1,k-1}}{s_{1,k}},$$

where η_k are the last components of the vectors $\mathbf{q}_k$, a recursion of which is given in Theorems 12.4 and 12.6.

14. Split algorithms for centrosymmetric and centro-skewsymmetric Toeplitz-plus-Hankel matrices

In this section, we consider $n \times n$ matrices R_n which are the sum of a Toeplitz matrix $T_n = T_n(\mathbf{a})$, $\mathbf{a} = (a_i)_{i=1-n}^{n-1}$ and a Hankel matrix $H_n = T_n(\mathbf{b})J_n$, $\mathbf{b} = (b_i)_{i=1-n}^{n-1}$,

$$R_n = T_n(\mathbf{a}) + T_n(\mathbf{b})J_n = \begin{bmatrix} a_0 & \dots & a_{1-n} \\ \vdots & \ddots & \vdots \\ a_{n-1} & \dots & a_0 \end{bmatrix} + \begin{bmatrix} b_{1-n} & \dots & b_0 \\ \vdots & \iddots & \vdots \\ b_0 & \dots & b_{n-1} \end{bmatrix}. \tag{14.1}$$

Note that the *chess-board matrices*,

$$B = \begin{bmatrix} c & d & c & \cdots \\ d & c & d & \cdots \\ c & d & c & \cdots \\ \vdots & \vdots & \vdots & \end{bmatrix} \qquad (c, d \in \mathbb{F})$$

are both Toeplitz and Hankel matrices. Thus the representation (14.1) is not unique.

Hereafter, we also use a representation of R_n which involves the projections $P_\pm = \frac{1}{2}\,(I_n \pm J_n)$ onto $\mathbb{F}_\pm^n$ and the vectors

$$\mathbf{c} = (c_j)_{j=1-n}^{n-1} = \mathbf{a} + \mathbf{b}, \quad \mathbf{d} = (d_j)_{j=1-n}^{n-1} = \mathbf{a} - \mathbf{b},$$

namely

$$R_n = T_n(\mathbf{c})P_+ + T_n(\mathbf{d})P_-. \tag{14.2}$$

From now on we restrict ourselves to the case where R_n is a centrosymmetric or centro-skewsymmetric matrix. (The general case is beyond the scope of the present paper.)

Proposition 14.1 *An $n \times n$ $T+H$ matrix R_n is centrosymmetric or centro-skewsymmetric if and only if it admits a representation (14.2) (or, equivalently, (14.1)) in which* $\mathbf{c}$ *and* $\mathbf{d}$ ($\mathbf{a}$ *and* $\mathbf{b}$) *are symmetric or skewsymmetric, respectively.*

Proof. It is easily checked that $J_nT_n(\mathbf{a})J_n = T_n(\mathbf{a}^J)$, with $\mathbf{a}^J = J_{2n-1}\mathbf{a}$. Hence

$$\begin{aligned} J_nR_nJ_n &= (J_nT_n(\mathbf{c})J_n)(J_nP_+J_n) + (J_nT_n(\mathbf{d})J_n)(J_nP_-J_n) \\ &= T_n(\mathbf{c}^J)P_+ + T_n(\mathbf{d}^J)P_-. \end{aligned}$$

If now R_n is centrosymmetric or centro-skewsymmetric, then

$$R_n = \pm J_nR_nJ_n = T_n\left(\frac{\mathbf{c}\pm\mathbf{c}^J}{2}\right)P_+ + T_n\left(\frac{\mathbf{d}\pm\mathbf{d}^J}{2}\right)P_-,$$

respectively. It remains to mention that $\frac{\mathbf{c}\pm\mathbf{c}^J}{2}$ and $\frac{\mathbf{d}\pm\mathbf{d}^J}{2}$ are symmetric or skewsymmetric, respectively. Representation (14.1) can be considered in an analogous way. The other direction of the assertion is obvious. □

Corollary 14.2 *An $n \times n$ T+H matrix R_n is centrosymmetric or centro-skewsymmetric if and only if there is a representation (14.2) such that the Toeplitz matrices $T_n(\mathbf{c})$, $T_n(\mathbf{d})$ (or $T_n(\mathbf{a})$, $T_n(\mathbf{b})$ in (14.1)) are both symmetric or both skewsymmetric, respectively.*

Remark It follows from Corollary 14.2 and can also be shown directly that a centrosymmetric $n \times n$ T+H matrix is also symmetric. On the contrary, a centro-skewsymmetric $n \times n$ T+H matrix need not to be neither symmetric nor skewsymmetric. In particular, a skewsymmetric T+H matrix is always a pure Toeplitz matrix.

Obviously, a centrosymmetric T+H matrix can be written in the form

$$R_n = P_+T_n(\mathbf{c})P_+ + P_-T_n(\mathbf{d})P_-$$

and a centro-skewsymmetric T+H matrix in the form

$$R_n = P_-T_n(\mathbf{c})P_+ + P_+T_n(\mathbf{d})P_-.$$

Besides the matrix $R_n = [r_{ij}]_{i,j=1}^n$ we also consider the central submatrices $R_{n-2l}^c = [r_{ij}]_{i,j=l+1}^{n-l}$ for $l = 0, 1, \ldots, l < n/2$. Recall that these matrices are nested, viz.

$$R_{n-2l+2}^c = \begin{bmatrix} * & * & * \\ * & R_{n-2l}^c & * \\ * & * & * \end{bmatrix}$$

and inherit the centrosymmetry properties of R_n. Furthermore, the following is obvious.

Proposition 14.3 *If R_n is centrosymmetric or centro-skewsymmetric and given by (14.2), then*

$$R_{n-2l}^c = T_{n-2l}^+P_{n-2l}^+ + T_{n-2l}^-P_{n-2l}^-,$$

where $T_{n-2l}^+ = [c_{i-j}]_{i,j=l}^{n-2l}$ and $T_{n-2l}^- = [d_{i-j}]_{i,j=l}^{n-2l}$.

We assume that R_n is centro-nonsingular, n is even, $n = 2m$. Let us restrict ourselves to the centrosymmetric case. (The centro-skewsymmetric case is analogous.) We want to solve the equation

$$R_n\mathbf{f} = \mathbf{b}$$

by centrosymmetric bordering (see Section 10.2).

We observe first that $R_n\mathbf{f} = \mathbf{b}$ is equivalent to the two Toeplitz systems

$$T_n^+\mathbf{f}^+ = \mathbf{b}^+, \quad T_n^+\mathbf{f}^- = \mathbf{b}^-, \tag{14.3}$$

where $\mathbf{b}^\pm = P_\pm\mathbf{b}$ and $\mathbf{f}^\pm = P_\pm\mathbf{f}$.

Let $\mathbf{b}^{\pm} = (b_j^{\pm})_{j=1}^{n}$, $\mathbf{b}_k^{\pm} = (b_j^{\pm})_{j=m-k+1}^{m+k}$ for $k = 1, 2, \ldots, m$, and let $\mathbf{f}_k^{\pm}$ be the solutions of

$$T_{2k}^{\pm}\mathbf{f}_k^{\pm} = \mathbf{b}_k^{\pm}. \tag{14.4}$$

These solutions exist and are unique. Indeed, since R_n is centro-nonsingular, $R_k^c\mathbf{f}_k^{\pm} = \mathbf{b}_k^{\pm}$ is uniquely solvable, and we have according to (14.2) $R_k^c\mathbf{f}_k^{\pm} = T_k^{\pm}\mathbf{f}_k^{\pm}$. For centrosymmetric bodering we need the (unique) solutions $\mathbf{x}_k^{\pm} \in \mathbb{F}_{\pm}^k$ of

$$T_{2k}^{\pm}\mathbf{x}_k^{\pm} = P_{\pm}\mathbf{e}_k, \tag{14.5}$$

Theorem 14.4 *The solutions of the equations (14.4) satisfies the recursions*

$$\mathbf{f}_{k+2}^{\pm} = \begin{bmatrix} 0 \\ \mathbf{f}_k^{\pm} \\ 0 \end{bmatrix} + \left(b_{n-l+1}^{\pm} - \beta_k^{\pm}\right)\mathbf{x}_{k+2}^{\pm},$$

where

$$\beta_k^{+} = [c_k \ldots c_1]\,\mathbf{f}_k^{+}, \quad \beta_k^{-} = [d_k \ldots d_1]\,\mathbf{f}_k^{-}.$$

The recursion starts with an empty $\mathbf{f}_0^{\pm}$.

It remains to mention that the solutions $\mathbf{x}_n^{\pm}$ can be computed using the recursions for $\mathbf{w}_n$ and $\mathbf{w}_n^{-}$ introduced in Section 10. In particular, the double-step split Levinson algorithm looks (in the notation here) as follows.

Theorem 14.5 *Let R_n be a centro-nonsingular, centrosymmetric T+H matrix. Then the solutions $\mathbf{x}_k^{\pm}$ of the equations (14.5) ($k = 1, 2, \ldots, m$) satisfy the recursion*

$$\mathbf{x}_{k+2}^{\pm} = \frac{1}{2\alpha_k^{\pm}}\left(\begin{bmatrix} 0 \\ 0 \\ \mathbf{x}_k^{\pm} \end{bmatrix} + \begin{bmatrix} \mathbf{x}_k^{\pm} \\ 0 \\ 0 \end{bmatrix} - 2\left(r_{k+1,k}^{\pm} - r_{k-1,k-2}^{\pm}\right)\begin{bmatrix} 0 \\ \mathbf{x}_k^{\pm} \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ \mathbf{x}_{k-2}^{\pm} \\ 0 \\ 0 \end{bmatrix}\right),$$

where

$$r_{jk}^{+} = [c_{j-1} \ldots c_{j-k}]\,\mathbf{x}_k^{+}, \quad r_{jk}^{-} = [d_{j-1} \ldots d_{j-k}]\,\mathbf{x}_k^{-},$$

and

$$\alpha_k^{\pm} = r_{k+2,k}^{\pm} - r_{k,k-2}^{\pm} - 2r_{k+1,k}^{\pm}\left(r_{k+1,k}^{\pm} - r_{k-1,k-2}^{\pm}\right) + \frac{1}{2}.$$

Note that $\alpha_k^{\pm} \neq 0$, since otherwise R_{k+2}^c would be singular.

Finally, for the sake of completeness we should mention the following facts. For a centro-skewsymmetric T+H matrix $R_n = T_n(\mathbf{c})P_{+} + T_n(\mathbf{d})P_{-}$, we have that R_n is nonsingular if and only if $T_n(\mathbf{c})$ and $T_n(\mathbf{d})$ are nonsingular.

This is different to the centrosymmetric case. In this case R_n is nonsingular if $T_n(\mathbf{c})$ and $T_n(\mathbf{d})$ are nonsingular. But the converse is not true. Take, for example, $\mathbf{c} = (1,\ 1,\ 1)$ and $\mathbf{d} = (-1,\ 1,\ -1)$. Then $T_n(\mathbf{c})$ and $T_n(\mathbf{d})$ are singular, whereas $R_2 = 2\,I_2$ is nonsingular.

One might conjecture that for a nonsingular R_n there is always such a representation with nonsingular $T_n(\mathbf{c})$ and $T_n(\mathbf{d})$. For $n = 2$ this is true. But this fails to be true for $n = 3$. Take, for example,

$\mathbf{c} = (1,\ 0,\ 1,\ 0,\ 1)$ and $\mathbf{d} = (0,\ 0,\ 1,\ 0,\ 0)$. Then $T_n(\mathbf{c})$ is singular for all representations, but

$$R_3 = T_3(\mathbf{c})P_+ + T_3(\mathbf{d})P_- = \frac{1}{2}\begin{bmatrix} 3 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 0 & 3 \end{bmatrix}$$

is nonsingular. Note that it can be shown that surprisingly for $n = 4$ there is always a representation with nonsingular $T(\mathbf{c})$ and $T(\mathbf{d})$.

A full understanding of such statements requires a deeper insight into the fascinating structure of Toeplitz-plus-Hankel matrices. We hope that we have made the reader inquisitive.

Exercises

1. Show that the set of all nonsingular, upper triangular Toeplitz matrices of order n forms a (multiplicative) group.
2. Consider a block Toeplitz matrix $T_n = [a_{i-j}]_{i,j=1}^n$, where a_i $(i = -n+1, \dots, n-1)$ are matrices of order $k \ll n$. Generalize the Levinson algorithm of Theorem 3.1 and the Schur algorithm of Theorem 4.1 to this case.
3. Consider the tridiagonal symmetric Toeplitz matrix
$$T_n = \begin{bmatrix} 2 & -1 & & 0 \\ -1 & 2 & \ddots & \\ & \ddots & 2 & -1 \\ 0 & & -1 & 2 \end{bmatrix} \Bigg\} n.$$
Then it is easily verified that, with the notations of Section 3.2,
$$\mathbf{u}_k^+ = \frac{1}{k}\,(i)_{i=1}^k, \quad \rho_k = \frac{k+1}{k}, \quad \text{and} \quad \alpha_k^+ = \alpha_k^- = -\frac{1}{k}.$$
Show that the Levinson algorithm of Section 3.3 also gives these results.
4. Find the unit LU-factorization of the matrix $A_n = T_n - \mathbf{e}_1\mathbf{e}_1^T$, where T_n is defined in Exercise 3, as well as the unit UL-factorization of A_n^{-1}. Compute the vector $\mathbf{v}$ so that $A_n^{-1} - T_n^{-1} = \mathbf{v}\mathbf{v}^T$.
5. Consider the $n \times n$ Toeplitz matrix $T_n(a, b)$ the first column of which is $(a^{j-1})_{j=1}^n$ and the first row of which is $(b^{j-1})_{j=1}^n$, where $ab \neq 1$. Show that $T_n(a, b)$ is strongly nonsingular, compute the first and last rows of its inverse and the co-unit UL-factorization of $T_n(a, b)^{-1}$. Conclude that $T_n(a, b)^{-1}$ is, in the nontriangular case $ab \neq 0$, the sum of a tridiagonal Toeplitz matrix and a matrix of rank 2.
6. Show that the symmetric Toeplitz matrix $T_n = [t_{|i-j|}]_{i,j=1}^n$ with $t_0 = 0, \quad t_j = \frac{1}{2^{j-1}}$ is singular if and only if $n \equiv 1 \mod 3$.
7. In Section 3.1 it was shown that the nonsingularity of T_k and T_{k+1} implies the nonsingularity of Γ_k. Show that, conversely, the nonsingularity of T_k and Γ_k implies the nonsingularity of T_{k+1}.
8. For a Toeplitz matrix, show the following relations between the polynomials $\mathbf{u}_k^+(t)$ and $\mathbf{u}_k^-(t)$ introduced in Section 3.2:
$$\mathbf{u}_k^-(t) = 1 - t\sum_{i=1}^{k-1} \gamma_i^- \mathbf{u}_i^+(t), \quad \mathbf{u}_k^+(t) = t^{k-1} - \sum_{i=1}^{k-1} \gamma_i^+ t^{k-1-i}\mathbf{u}_i^-(t).$$
9. Find the matrix of the operator of multiplication by t acting from $\mathbb{F}^{n-1}(t)$ to $\mathbb{F}^n(t)$ in the bases $\{\mathbf{u}_k^\pm(t)\}$ introduced in Section 3.2.

10.
   (a) Let A be a strongly nonsingular $n \times n$ matrix and $A^{-1} = UDL$, $U = U(\mathbf{u}_k^+)$, $L^T = U(\mathbf{u}_k^-)$, the unit UL-factorization of A^{-1}. Show that if the $\mathbf{u}_k^\pm$ satisfy a recursion as in Theorem 3.2, then A is Toeplitz.
   (b) Let A be a symmetric matrix and $A^{-1} = UDU^T$, $U = U(\mathbf{u}_k)$ the unit UL-factorization of A^{-1}. Show that if the vectors $\mathbf{u}_k$ satisfy a recursion as in Theorem 7.2, then A is Hankel.
11. Describe the co-unit LU-factorization of a Hankel matrix H_n and the unit UL-factorization of its inverse H_n^{-1} in terms of the vectors $\mathbf{r}_k$ and $\mathbf{u}_k$ that are computed by the algorithms described in Theorems 7.5 and 7.2, respectively.
12. Let $\mathbf{u}_k$, $\widetilde{\mathbf{u}}_k$, $\mathbf{s}_k$, $\widetilde{\mathbf{s}}_k$ be as in Section 5.1.
   (a) Find the matrices of the basis change between the bases $\{\mathbf{u}_k\}$ and $\{\widetilde{\mathbf{u}}_k\}$.
   (b) Find a relationship similar to that of Proposition 5.6 between the vectors $\mathbf{s}_k$ and $\widetilde{\mathbf{s}}_k$.
13. Show that inverses of Toeplitz matrices are, in general, not Toeplitz but quasi-Toeplitz matrices.
14. Consider instead of the transformation ∇_+ defined in (6.1) the transformation

$$\nabla_-(A) = A - S_n^T A S_n \tag{14.6}$$

   defined for $n \times n$ matrices A.
   (a) Show that rank $\nabla_-(A) \leq 2$ if and only if $J_n A J_n$ is quasi-Toeplitz.
   (b) Give a general description of matrices A satisfying rank $\nabla_-(A) \leq 2$.
   (c) Find examples for rank $\nabla_+(A) \neq$ rank $\nabla_-(A)$.
   (d) Find sufficient conditions for rank $\nabla_+(A) =$ rank $\nabla_-(A) \leq 2$.
15. Prove Proposition 6.2 and its generalization to Toeplitz-like matrices.
16. Show that an $n \times n$ matrix A is quasi-Toeplitz if and only if A admits a representation $A = LTU$ in which L is lower triangular Toeplitz, U is upper triangular Toeplitz and T is Toeplitz.
17. Show that a nonsingular $n \times n$ matrix A is quasi-Toeplitz if and only if the matrix $J_n A^{-1} J_n$ is quasi-Toeplitz.

   Hint. Write two Schur complement formulas for the matrix $\begin{bmatrix} A & S_n \\ S_n^T & A^{-1} \end{bmatrix}$.
18. Let $H_n = [h_{i+j-1}]_{i,j=1}^n$ be a strongly nonsingular Hankel matrix and L_{2n-1} the lower triangular Toeplitz matrix $L_{2n-1} = [h_{i-j+1}]_{i,j=1}^{2n-1}$ ($h_i = 0$ if $i < 1$). Let $(s_i)_{i=1}^{2n-1}$ be the first column of L_{2n-1}^{-1}. Show that

$$H_n = L_n \begin{bmatrix} s_1 & 0 \\ 0 & -H' \end{bmatrix} L_n^T,$$

   where $H' = [s_{i+j+1}]_{i,j=1}^{n-1}$ and L_n is the $n \times n$ leading principal submatrix of L_{2n-1}.
19. Let T_n be a real symmetric Toeplitz matrix, and let $T_\pm$ denote the restriction of it, as a linear operator, to the invariant subspace $\mathbb{R}_\pm^n$, respectively. Furthermore, let W be the operator defined by

$$(W\mathbf{x}_-)(t) = \frac{t+1}{t-1}\mathbf{x}_-(t) \quad (\mathbf{x}_- \in \mathbb{R}_-^n),$$

   mapping $\mathbb{R}_-^n$ into $\mathbb{R}_+^n$. Show that T_+ and T_- are related via

$$T_+ W - W T_- = 2\mathbf{e}\,\mathbf{a}^T (S_n - I_n)^{-1},$$

   where $\mathbf{a} = [a_n \ldots a_1]$, $\mathbf{e} = [1 \ldots 1]$ and S_n is the forward shift introduced in (6.2).
20. Find a recursion for the solution of a skewsymmetric Toeplitz system by centrosymmetric bordering as follows. Correct in each step only the last component of the right-hand side by using $\mathbf{x}_k^+$ of (11.1). After the last step correct the first component by using $\mathbf{x}_m^-$. Compare the complexity of the resulting algorithm with the complexity of the algorithms discussed in Section 11.5.
21. Let A be a $n \times n$ centrosymmetric matrix, $n = 2m$, and $Q_n = \begin{bmatrix} -J_m & I_m \\ J_m & I_m \end{bmatrix}$. Show that $Q_n^T A Q_n$ is of the form

$$Q_n^T A Q_n = \begin{bmatrix} B_- & O \\ O & B_+ \end{bmatrix}$$

for some $m \times m$ matrices $B_\pm$. Investigate how ZW-factorization of A is related to triangular factorization of B_+ and B_-.

22. Let Z be a Z-matrix of even order $2m$. Introduce the X-matrix X built from the diagonal and antidiagonal entries of Z,

$$X = \operatorname{xma}\left(\begin{bmatrix} \alpha_k & \beta_k \\ \gamma_k & \delta_k \end{bmatrix}\right)_{k=1}^{m},$$

and show that $\det Z = \det X = \prod_{k=1}^{m} (\alpha_k \delta_k - \beta_k \gamma_k)$.

23. Let T_n be the tridiagonal symmetric Toeplitz matrix T_n of Exercise 3. Compute the factors of its unit LU-factorization, its unit split ZW-factorization, and the unit split WZ-factorization of its inverse.

24. Find a unit split WZ-factorization for the inverse of a symmetric, centro-nonsingular Toeplitz matrix T_n using the solutions $\mathbf{w}_k$ of (10.1) and $\mathbf{w}_k^-$ and the double-step Levinson algorithm.

Comments and references

2., 3., and 4. The Levinson algorithm appeared in the famous paper [34] about filter design as an algorithm for solving a general linear system with a positive definite Toeplitz coefficient matrix. The algorithm that solves only the Yule–Walker equation is often attributed to Durbin in connection with his paper [16] on linear prediction problems.

However, it should be pointed out that the recursions of the Levinson algorithm are almost identical to the recursions for orthogonal polynomials on the unit circle discovered by Szegő [41].

The Schur algorithm appeared in the famous paper of Schur [40] as an algorithm in complex function theory to check whether an analytic function on the unit disk maps the unit disk into itself (see also [29] and the references therein). As a factorization algorithm for Toeplitz matrices it was designed in Bareiss' paper [3] (not mentioning Schur).

Practical experience and theoretical results indicate that, in general, Schur-type algorithms have better stability behavior than Levinson-type algorithms (see e.g. the contribution of Brent in [32]).

As already mentioned in the Introduction, the Levinson-type algorithms produce the parameters needed in Bezoutian formulas for the inverses of Toeplitz or Hankel matrices (see e.g. [19]). These Bezoutian formulas are the basis for constructing superfast algorithms (see [21–23,39]).

Readers who want to learn more about a special case of Toeplitz matrices, the circulants, should study the nice book of Davis [12].

Anyone who wants to venture into the vast literature on Toeplitz (Hankel and other structured) matrices could first read the book [19], where e.g. recursions in the not strongly regular case, inversion formulas, and results on other structured matrices are presented.

We refer the reader who is interested in numerical aspects to the books [17,5,39,32,4] and the references therein.

5. The concept of displacement structure was first introduced in [30] (see also [31]) using a displacement operator of the form (14.6). In [18] a displacement operator of the form (6.1) was considered (called there UV-reduction operator), and in [19] it was shown that replacing the shifts by other matrices (e.g. diagonal matrices) allows us to consider also other types of structured matrices (e.g. matrices which are the sum of a Toeplitz and a Hankel matrix or which are generalizations of Vandermonde or Cauchy matrices) in a similar way.

In the subsequent years a huge number of papers followed which documents the success in the field of displacement structured matrices.

6. The Levinson-type and Schur-type algorithms for Hankel matrices show that there are essential differences between Toeplitz and Hankel matrices. A first algorithm for Hankel matrices was given in a

paper of Trench [43]]. The recursions given there are tree-term recursions and at the end of the paper their connection with orthogonal polynomials on the real axis is discussed.

7. Padé approximation has a very long history. Due to the connection with continued fractions one can assume that it starts with Euclid's algorithm for computing the greatest common divisor of two integers. This was in about the year 300 BC.

In 1892 Padé defended his thesis at the Sorbonne in Paris. It was the first systematic investigation of what today is called Padé approximant and reveals, in particular, connections with continued fractions ([36,37], see also [38]). Padé's thesis was very much influenced by his teacher Hermite, who developed a general theory of interpolation by rational functions. We refer the reader who is interested in more details of this very exciting history to [8].

In the last decades there is a lot of activities on Padé approximation not only in pure mathematics and numerical analysis but also in applications to physics. Two important monographs are [11,2]. In [9] Brezinski and Van Iseghem give an instructive survey concerning relevant aspects of Padé approximation which can also be used as a preliminary tutorial guide.

8. In 1950 Lanczos [33] proposed a method for computing the eigenvalues of a matrix by reducing this matrix to a tridiagonal form, from which the eigenvalues can be determined. The interested reader should study Chapter 9 of [17], which is dedicated to Lanczos methods.

To learn more about the connection between Hankel algorithms and Lanczos methods we refer to [7] and the references therein. The authors there write: "The resulting recursion formulae to factorize a strongly nonsingular Hankel matrix have appeared in several papers under different guises, going all the way back to Tchebycheff [42]".

9., 10., and 11. The idea to "split" the classical Levinson and Schur algorithms for real symmetric Toeplitz matrices goes back to Delsarte and Genin [13,14], but the splitting property was utilized before in other fields, for example in the reduction of the trigonometric moment problem with real data to a moment problem on the interval $[-1, 1]$ (see [1]), in efficient root location tests (see [6]), and also in signal processing and seismology (see [10]).

12. The ZW-factorization is closely related to the "quadrant interlocking" or WZ-factorization, which was originally introduced and studied by Evans and his coworkers for the parallel solution of tridiagonal systems.

The ZW-factorization for real symmetric Toeplitz matrices was first mentioned by Demeure in [15]. In our papers [24,26,27] ZW-factorizations for skewsymmetric Toeplitz, centrosymmetric and centro-skewsymmetric Toeplitz–plus–Hankel, and general Toeplitz-plus-Hankel matrices were described, respectively. Note that for skewsymmetric Toeplitz matrices (and thus also for purely imaginary hermitian Toeplitz matrices) the factors of the ZW-factorization have some surprising additional symmetry properties which are not shared by the symmetric case.

In the paper [28] it is shown that for hermitian Toeplitz matrices the ZW-factorization leads to more efficient algorithms for the solution of linear systems than LU-factorization and that the ZW-factorization reflects, in contrast to the LU-factorization, both symmetry properties. This leads to a computational gain in solving linear systems.

13. First important results on Toeplitz-plus-Hankel matrices were the construction of their inverses (see [35,19]) and the discovery of the Bezoutian structure of their inverses [20]. Here we deal with the cases of centrosymmetric or centro-skewsymmetric matrices and offer split algorithms which require the application of the afore-mentioned algorithms for pure Toeplitz matrices.

If the reader's interest is piqued in the structure of Toeplitz-plus-Hankel matrices we recommend to study also [25,26].

References

[1] N.I. Akhiezer, The Classical Moment Problem and Some Related Questions in Analysis (N. Kemmer, Trans.), Hafner Publishing Co., New York, 1965.
[2] G.A. Baker Jr., P. Graves-Morris, Padé approximants, in: Encyclopedia of Mathematics and its Applications, vol. 59, Cambridge University Press, Cambridge, 1996.
[3] E.H. Bareiss, Numerical solution of linear equations with Toeplitz and vector Toeplitz matrices, Numer. Math. 13 (1969) 404–424.

[4] D. Bini, V. Mehrmann, V. Olshevsky, E. Tyrtyshnikov, M. van Barel (Eds.), Numerical methods for structured matrices and applications, in: Operator Theory: Advances and Applications, vol. 199, Birkhäuser Verlag, Basel, 2010. The Georg Heinig memorial volume.
[5] D. Bini, V.Y. Pan, Polynomial and matrix computations, in: Progress in Theoretical Computer Science, vol. 1, Birkhäuser Boston Inc., Boston, MA, 1994. Fundamental algorithms.
[6] Y. Bistritz, Zero location of polynomials wit respect to the unit circle of discrete-time linear system polynomials, Proc. IEEE 72 (1984) 1131–1142.
[7] D.L. Boley, T.J. Lee, F.T. Luk, The Lanczos algorithm and Hankel matrix factorization, Linear Algebra Appl. 172 (1992) 109–133. Second NIU Conference on Linear Algebra, Numerical Linear Algebra and Applications, DeKalb, IL, 1991.
[8] C. Brezinski, History of continued fractions and Padé approximants, in: Springer Series in Computational Mathematics, vol. 12, Springer-Verlag, Berlin, 1991.
[9] C. Brezinski, J. Van Iseghem, A taste of Padé approximation, in: Acta Numerica 1995, Acta Numer., Cambridge University Press, Cambridge, 1995, pp. 53–103.
[10] K.P. Bube, R. Burridge, The one-dimensional inverse problem of reflection seismology, SIAM Rev. 25 (4) (1983) 497–559.
[11] A. Bultheel, Laurent series and their Padé approximations, in: Operator Theory: Advances and Applications, vol. 27, Birkhäuser Verlag, Basel, 1987.
[12] P.J. Davis, Circulant Matrices, John Wiley & Sons, New York–Chichester–Brisbane, 1979. A Wiley-Interscience Publication, Pure and Applied Mathematics.
[13] P. Delsarte, Y.V. Genin, The split Levinson algorithm, IEEE Trans. Acoust. Speech Signal Process. 34 (3) (1986) 470–478.
[14] P. Delsarte, Y.V. Genin, On the splitting of classical algorithm, IEEE Trans. Acoust. Speech Signal Process. 35 (1987) 645–653.
[15] C.J. Demeure, Bowtie factors of Toeplitz matrices by means of splitz algorithms, IEEE Trans. Acoust. Speech Signal Process. 37 (10) (1989) 1601–1603.
[16] J. Durbin, The fitting of time-series models, Revue Inst. Int. De Stat. 28 (1960) 233–244.
[17] G.H. Golub, C.F. Van Loan, Matrix computations, in: Johns Hopkins Studies in the Mathematical Sciences, third ed., Johns Hopkins University Press, Baltimore, MD, 1996.
[18] G. Heinig, K. Rost, Invertierung einiger Klassen von Matrizen und Operatoren. I. Endliche Toeplitzmatrizen und ihre Verallgemeinerungen, Wissenschaftliche Informationen [Scientific Information], vol. 12, Technische Hochschule Karl-Marx-Stadt Sektion Mathematik, Karl-Marx-Stadt, 1979.
[19] G. Heinig, K. Rost, Algebraic methods for Toeplitz-like matrices and operators, in: Operator Theory: Advances and Applications, vol. 13, Birkhäuser Verlag, Basel, 1984.
[20] G. Heinig, K. Rost, On the inverses of Toeplitz-plus-Hankel matrices, Linear Algebra Appl. 106 (1988) 39–52.
[21] G. Heinig, K. Rost, DFT representations of Toeplitz-plus-Hankel Bezoutians with application to fast matrix-vector multiplication, Linear Algebra Appl. 284 (1–3) (1998) 157–175. ILAS Symposium on Fast Algorithms for Control, Signals and Image Processing, Winnipeg, MB, 1997.
[22] G. Heinig, K. Rost, Hartley transform representations of symmetric Toeplitz matrix inverses with application to fast matrix-vector multiplication, SIAM J. Matrix Anal. Appl. 22 (1) (2000) 86–105 (Electronic).
[23] G. Heinig, K. Rost, Efficient inversion formulas for Toeplitz-plus-Hankel matrices using trigonometric transformations, in: Structured Matrices in Mathematics, Computer Science, and Engineering, II (Boulder, CO, 1999), Contemp. Math., vol. 281, Amer. Math. Soc., Providence, RI, 2001, pp. 247–264.
[24] G. Heinig, K. Rost, Fast algorithms for skewsymmetric Toeplitz matrices, in: Toeplitz matrices and singular integral equations (Pobershau, 2001), Oper. Theory Adv. Appl., vol. 135, Birkhäuser, Basel, 2002, pp. 193–208.
[25] G. Heinig, K. Rost, Centrosymmetric and centro-skewsymmetric Toeplitz-plus-Hankel matrices and Bezoutians, Linear Algebra Appl. 366 (2003) 257–281. Special issue on structured matrices: analysis, algorithms and applications, Cortona, 2000.
[26] G. Heinig, K. Rost, Fast algorithms for centro-symmetric and centro-skewsymmetric Toeplitz-plus-Hankel matrices, Numer. Algorithms 33 (2003) 305–317. International Conference on Numerical Algorithms, vol. I, Marrakesh, 2001.
[27] G. Heinig, K. Rost, New fast algorithms for Toeplitz-plus-Hankel matrices, SIAM J. Matrix Anal. Appl. 25 (3) (2003) 842–857 (Electronic).
[28] G. Heinig, K. Rost, Schur-type algorithms for the solution of Hermitian Toeplitz systems via factorization, in: Recent Advances in Operator Theory and its Applications, Oper. Theory Adv. Appl., vol. 160, Birkhäuser, Basel, 2005, pp.233–252.
[29] T. Kailath, A theorem of I. Schur and its impact on modern signal processing, in: I. Schur Methods in Operator Theory and Signal Processing, Oper. Theory Adv. Appl., vol. 18, Birkhäuser, Basel, 1986, pp. 9–30.
[30] T. Kailath, S.Y. Kung, M. Morf, Displacement ranks of matrices and linear equations, J. Math. Anal. Appl. 68 (2) (1979) 395–407.
[31] T. Kailath, A.H. Sayed, Displacement structure: theory and applications, SIAM Rev. 37 (3) (1995) 297–386.
[32] T. Kailath, A.H. Sayed (Eds.), Fast Reliable Algorithms for Matrices with Structure, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1999
[33] C. Lanczos, An iteration method for the solution of the eigenvalue problem of linear differential and integral operators, J. Res. Nat. Bur. Standards 45 (1950) 255–282.
[34] N. Levinson, The Wiener RMS (root mean square) error criterion in filter design and prediction, J. Math. Phys. Mass. Inst. Tech. 25 (1947) 261–278.
[35] A.B. Nersesyan, A.A. Papoyan, Construction of a matrix inverse to the sum of Toeplitz and Hankel matrices, Izv. Akad. Nauk Armyan. SSR Ser. Mat. 18 (2) (1983) 150–160.
[36] H. Padé, Sur la représentation approchée d'une fonction par des fractions rationnelles, Ann. Sci. École Norm. Sup. (3) (1892) 3–93.
[37] H. Padé, Sur les séries entières convergentes ou divergentes et les fractions continues rationnelles, Acta Math. 18 (1) (1894) 97–111.
[38] H. Padé, Œuvres, Bibliothèque Scientifique Albert Blanchard (Albert Blanchard Scientific Library), in: Claude Brezinski (Ed.), Librairie Scientifique et Technique Albert Blanchard, Paris, 1984.
[39] V.Y. Pan, Structured Matrices and Polynomials, Birkhäuser Boston Inc., Boston, MA, 2001. Unified Superfast Algorithms.
[40] I. Schur, Über Potenzreihen, die im Innern des Einheitskreises beschränkt sind, J. Reine Angew. Math. 147 (1917) 205–232.

[41] G. Szegő, Orthogonal Polynomials, American Mathematical Society, New York, 1939. American Mathematical Society Colloquium Publications, vol. 23.
[42] P. Tchebycheff, Sur l'interpolation par la méthode des moindres carrés, Mém. Acad. Impér. Sci. St. Pétersbourg 1 (15) (1859) 1–24.
[43] W.F. Trench, An algorithm for the inversion of finite Hankel matrices, J. Soc. Indust. Appl. Math. 13 (1965) 1102–1107.