Electronic Transactions on Numerical Analysis.
Volume 26, pp. 243-269, 2007.

ISSN 1068-9613.

ETNA
Kent State University
etna@mcs.kent.edu

SOLVING LINEAR SYSTEMS WITH A LEVINSON-LIKE SOLVER*

RAF VANDEBRIL†, NICOLA MASTRONARDI‡, AND MARC VAN BAREL†

Abstract. In this paper we will present a general framework for solving linear systems of equations. The solver is based on the Levinson-idea for solving Toeplitz systems of equations. We will consider a general class of matrices, defined as the class of simple (p_1, p_2)-Levinson conform matrices. This class incorporates, for instance, semiseparable, band, companion, arrowhead and many other matrices. For this class, we will derive a solver of complexity $O(p_1 p_2 n)$. The system solver is written inductively, and uses in every step k, the solution of a so-called kth order Yule-Walker-like equation. The algorithm obtained first has complexity $O(p_1 p_2 n^2)$. Based, however on the specific structure of the simple (p_1, p_2)-Levinson conform matrices, we will be able to further reduce the complexity of the presented method, and get an order $O(p_1 p_2 n)$ algorithm.

Different examples of matrices are given for this algorithm. Examples are presented for: general dense matrices, upper triangular matrices, higher order generator semiseparable matrices, quasiseparable matrices, Givens-vector representable semiseparable matrices, band matrices, companion matrices, confederate matrices, arrowhead matrices, fellow matrices and many more.

Finally, the relation between this method and an upper triangular factorization of the original matrix is given and also details concerning possible look ahead methods are presented.

Key words. Levinson, Yule-Walker, look-ahead, system solving, Levinson conform matrices

AMS subject classifications. 65F05

1. Introduction.

Solving systems of equations is an essential tool in all kinds of applications. Gaussian elimination (see [11, 21, 29]) is a well-known method for solving linear systems, it takes $O(n^3)$ operations. For several applications, however, the coefficient matrices involved are structured, for example, semiseparable, Toeplitz or Hankel matrices. These matrices are essentially determined by $O(n)$ parameters, instead of $O(n^2)$, for an unstructured matrix. Therefore, they often admit faster solvers, of $O(n^2)$, $O(n\log(n))$ or even $O(n)$, than the traditional $O(n^3)$ methods, such as, for example, Gaussian elimination.

Toeplitz systems of equations, for example, can be solved in $O(n^2)$, by using the Durbin and Levinson algorithm. The Levinson algorithm for Toeplitz matrices is widespread and described for example in [21, 24, 25]. Based on a specific block decomposition of the Toeplitz matrix T, one can solve the coupled Yule-Walker equations, which provide enough information for solving linear systems with this Toeplitz matrix. The original method is, however, only applicable for strongly nonsingular Toeplitz matrices, as the inductive procedure, computes solutions of principal leading submatrices of T. Look-ahead procedures exist to overcome numerical instabilities, for matrices which are not or almost not strongly nonsingular; see [7].

*Received August 24, 2005. Accepted for publication December 23, 2006. Recommended by P. Van Dooren. The research was partially supported by the Research Council K.U.Leuven, projects OT/00/16 (SLAP: Structured Linear Algebra Package), OT/05/40 (Large rank structured matrix computations), by the Fund for Scientific Research–Flanders (Belgium), projects G.0078.01 (SMA: Structured Matrices and their Applications), G.0176.02 (ANCILA: Asymptotic aNalysis of the Convergence behavior of Iterative methods in numerical Linear Algebra), G.0184.02 (CORFU: Constructive study of Orthogonal Functions) and G.0455.0 (RHPH: Riemann-Hilbert problems, random matrices and Padé-Hermite approximation), and by the Belgian Programme on Interuniversity Poles of Attraction, initiated by the Belgian State, Prime Minister's Office for Science, Technology and Culture, project IUAPV-22 (Dynamical Systems and Control: Computation, Identification & Modelling). The research of the second author was partially supported by MIUR, grant number 2004015437, by the short term mobility program, Consiglio Nazionale delle Ricerche and by VII Programma Esecutivo di Collaborazione Scientifica Italia–Comunità Francese del Belgio, 2005–2006. The scientific responsibility rests with the authors.

†K.U.Leuven, Dept. Computerwetenschappen, Celestijnenlaan 200A, 3000 Leuven (Heverlee) (raf.vandebri,marc.vanbarell@cs.kuleuven.be)

‡Istituto per le Applicazioni del Calcolo "M.Picone" sez. Bari, National Council of Italy, via G. Amendola 122/D, I-70126 Bari, Italy (n.mastronardi@ba.iac.cnr.it)

ETNA
Kent State University
etna@mcs.kent.edu

A Levinson-like solver for the class of symmetric strongly nonsingular higher order semiseparable plus band, was investigated in [27, 31]. The solution of a system of a (p_1, p_2)-higher order generator representable semiseparable matrix plus an (l_1, l_2)-band matrix was computed in $O((p_1 + l_1)(p_2 + l_2)n)$ operations.

The method presented in this paper, is also based on this Levinson algorithm. A class of matrices called (p_1, p_2)-Levinson conform is defined, which will admit a Levinson-like algorithm. In this paper we focus to a specific subclass, called simple (p_1, p_2)-Levinson conform. This class is called simple, because we will prove that these matrices, admit a solver of complexity $O(p_1 p_2 n)$, which is in fact linear in time. Matrices, such as Toeplitz or Hankel, are not incorporated in the class of simple (p_1, p_2)-Levinson conform matrices. However, as shown in the paper, several classes of matrices, do belong to this class, and hence admit an order $O(p_1 p_2 n)$ solver. For example the matrices considered in [27, 31], fit in this framework, and are given as an example.

The algorithmic idea is exactly the same as for solving Toeplitz systems via the Levinson algorithm. First n systems with a special right-hand-side, called the Yule-Walker like equations need to be solved. Based on these solutions, we can then solve the linear system, with an arbitrary right-hand side.

In [13, 17], the authors investigated a closely related technique for solving systems of equations. The authors restricted themselves to the class of block quasiseparable matrices, which also includes the class of band matrices and semiseparable matrices. The algorithm presented in these manuscripts is based on an efficient computation of the generators of a triangular factorization of the quasiseparable matrix. Using this representation of the factorization, they compute the solution by inverting one upper triangular quasiseparable matrix. The algorithm in these manuscripts can be seen as an analogue of a Schur algorithm for computing the LU-decomposition of the resulting matrix. As several examples provided in this paper naturally fit in the class of quasiseparable matrices, we investigate more closely, in the example section, the relation between the method in [17], and the one presented in this paper. Let us briefly elaborate on the difference. The method in [13, 17], computes an LU-factorization via a Schur-type of algorithm. In this manuscript we use a Levinson-type algorithm. The Levinson-type algorithm can be used to compute an LU^{-1} factorization (this is discussed in more detail in Section 4). But in fact, due to our specific structure we do not need to compute the factors L and U or U^{-1} explicitly. Computing them causes extra, unnecessary work, and by using the Levinson-like approach there is no need in computing these factors. Due to this significant difference, we will see that we can get a speed up of a factor 2 for the quasiseparable case, and moreover a reduction in complexity from $O(p_1 p_2^2 n) + O(p_1^2 p_2 n)$ to a complexity $O(p_1 p_2 n)$ for most of the cases (p_1 and p_2 are matrix dependent values smaller than n).

The paper is organized as follows. In Section 2, the class of simple (p_1, p_2)-Levinson conform matrices is defined. In fact this is just based on a special block decomposition of the matrix involved. In Section 3, the algorithm for solving simple (p_1, p_2)-Levinson matrices is presented. First a direct way to solve systems in this manner is given having complexity $O(p_1 p_2 n^2)$. It will be shown however how one can further reduce this complexity to come to a method which costs $O(p_1 p_2 n)$ operations. It is therefore, important to choose the factors p_1 and p_2 as small as possible. In Section 4, we investigate the upper triangular factorization related to this method. In Section 5, numerous examples are described. The first three examples are related to semiseparable matrices. In a first case the class of Givens-vector representable semiseparable matrices is considered, secondly the class of quasiseparable matrices and finally the class of higher order generator representable semiseparable matrices are investigated. (These results are already extensions of the ones presented in [27, 31].) Next,

ETNA
Kent State University
etna@mcs.kent.edu

the class of band matrices and arrowhead matrices are investigated, they are closely related to semiseparable matrices as well: band matrices can be considered as the inverses of semiseparable matrices and an arrowhead matrix is a semiseparable plus diagonal matrix. Examples for matrices having a nonsymmetric structure are given, such as a unitary Hessenberg matrices. Moreover, the class of simple (p_1, p_2)-Levinson conform matrices is proven to be closed under summation, this means that summations of simple Levinson conform matrices are again simple Levinson conform. Finally, we also prove that upper triangular matrices, dense matrices, companion matrices, comrade matrices as well as fellow matrices are simple Levinson conform and hence admit the proposed solver. We also indicate how to implement the look–ahead version of the algorithm, such that we can omit the strongly nonsingularity condition. The paper closes with conclusions and future research.

2. The matrix block decomposition. In this paper we will develop a Levinson-like solver for structured systems of equations. In order to develop such a solver, our coefficient matrix should admit a specific partitioning of the matrix, making it able to derive the recursive algorithm.

Let us therefore introduce the notion of (simple) p_1-Levinson and (simple) (p_1, p_2)-Levinson conform matrices.

DEFINITION 2.1 (p_1-Levinson conform matrices). *Given a matrix $A = (a_{i,j})$, for $i, j = 1, \ldots, n$, and denote with $A_k = (a_{i,j})$, for $i, j = 1, \ldots, k$ the upper $k \times k$ submatrix of A. The matrix A is said to be p_1-Levinson conform if the matrix can be decomposed in the following way:*

1. *For every $1 \le k \le n-1$, there is a splitting in blocks of the matrix A_{k+1} of the following form:*

$$A_{k+1} = \left[\begin{array}{c|c} A_k & E_k R_k \mathbf{c}_{k+1}^T \\ \hline \mathbf{d}_{k+1} S_k^T F_k^T & a_{k+1,k+1} \end{array}\right]$$

where $\mathbf{c}_{k+1} \in \mathbb{R}^{1\times p_1}$, $\mathbf{d}_{k+1} \in \mathbb{R}^{1\times p_2}$, $R_k \in \mathbb{R}^{k\times p_1}$, $F_k, E_k \in \mathbb{R}^{k\times k}$ $R_k, S_k \in \mathbb{R}^{k\times p_2}$ and $A_k \in \mathbb{R}^{k\times k}$.

2. *The following relation for the matrices R_{k+1} (with $1 \le k \le n-1$) needs to be satisfied:*

$$R_{k+1} = \left[\begin{array}{c} R_k P_k \\ \hline \xi_{k+1} \end{array}\right],$$

where P_k is a matrix of dimension $p_1 \times p_1$ and ξ_{k+1} is a row vector of length p_1.

We call the matrix simple p_1-Levinson conform if the matrix E_k equals the identity matrix of order k and the multiplication of a vector with the matrix P_k can be done in linear time (i.e. only $O(p_1)$ operations are involved).

No conditions were placed on the matrix S_k, if we put similar conditions on S_k as on the matrix R_k we call the matrix (p_1, p_2)-Levinson conform.

DEFINITION 2.2 ((p_1, p_2)-Levinson conform matrices). *A matrix A is called (p_1, p_2)-Levinson conform, if the matrix is p_1-Levinson conform, i.e. that Conditions (1) and (2) from Definition 2.1 are fulfilled and the following condition for the matrices S_k is satisfied:*

3. *The matrices S_{k+1} (with $1 \le k \le n-1$) can be decomposed as:*

$$S_{k+1} = \left[\begin{array}{c} S_k Q_k \\ \hline \eta_{k+1} \end{array}\right],$$

where Q_k is a matrix of dimension $p_2 \times p_2$ and η_{k+1} is a row vector of length p_2.

ETNA
Kent State University
etna@mcs.kent.edu

We call a matrix simple (p_1, p_2)-Levinson conform, if both the matrices E_k and F_k are equal to the identity matrix of order k and the multiplication of a vector with the matrices P_k and Q_k can be performed in respectively $O(p_1)$ and $O(p_2)$ operations.

In fact, every matrix is simple Levinson conform.

LEMMA 2.3. *Suppose an arbitrary matrix $A \in \mathbb{R}^{n\times n}$ is given, the matrix A is simple $(n-1, n-1)$-Levinson conform.*

Proof. The proof is straightforward. Define the $k \times (n-1)$ matrices R_k and S_k as follows:

$$R_k = S_k = [I_k, 0]$$

and assume for every k the matrices E_k, F_k, P_k and Q_k to be equal to the identity. Defining

$$\begin{aligned}\mathbf{d}_k &= [a_{k,1}, a_{k,2}, \ldots, a_{k,k-1}, 0, \ldots, 0],\\ \mathbf{c}_k &= [a_{1,k}, a_{2,k}, \ldots, a_{k-1,k}, 0, \ldots, 0],\end{aligned}$$

with both row vectors of length $n-1$. One can easily check that the conditions of Definition 2.2 are satisfied. □

The Lemma also shows that the choice of $\mathbf{c}_k, \mathbf{d}_k, R_k$ and Q_k is not always unique. But, the notion of simple (p_1, p_2)-Levinson conformity is strongly related to the complexity of the method we will deduce in this paper. The overall solver will have a complexity $O(p_1p_2n)$. Hence it is important to keep the values of p_1 and p_2 as low as possible.

¿From now on we will only focus on Levinson conform matrices for which E_k and F_k are equal to the identity matrix of size k. This excludes in some sense important classes of matrices, such as Toeplitz,Hankel and Vandermonde matrices. In the simple formulation these matrices are $(n-1, n-1)$-Levinson conform, whereas, omitting the assumption of being simple leads to a $(1, 1)$-Levinson conform matrix. In this paper however we will restrict ourselves to the class of simple Levinson conform matrices. This class is already wide enough to admit different types of structures as will be showed later, in the examples section. More information on efficient solvers for Toeplitz, Hankel and Vandermonde matrices, based on their displacement representation, can for example be found in [20].

3. A framework for simple (p_1, p_2)-Levinson conform matrices. In this section we will construct a Levinson-like solver for solving strongly nonsingular linear systems of equations for which the coefficient matrix is simple (p_1, p_2)-Levinson conform. The limitation of being strongly nonsingular can be relaxed; see the section on the look-ahead procedure. In this section we will firstly solve the corresponding Yule-Walker-like systems. The solution of these equations will be used for solving the general system of equations, with an arbitrary right-hand side, based on the Levinson method. A possible algorithm is presented in Section 3.3, followed by complexity reducing remarks in Section 3.4. The final $O(p_1p_2n)$ method is presented, with a detailed complexity count, in Section 3.5.

3.1. The Yule-Walker-like system. Suppose a simple p_1-Levinson conform matrix A is given. The aim of the Yule-Walker step is to solve the following system of equations $AY_n = -R_n$. The system will be solved by induction. Let us assume we know the solution of the kth order Yule-Walker-like problem (with $1 \le k \le n-1$):

$$A_k Y_k = -R_k, \tag{3.1}$$

and we would like to compute the solution of the $(k+1)$th Yule-Walker-like problem. (Note that, in general, Y_k represents a matrix of dimension $k \times p_1$.) The $(k+1)$th system of equations is of the form:

$$A_{k+1}Y_{k+1} = -R_{k+1}.$$

Using the initial conditions put on the matrix A, we can rewrite the equation above as

$$\left[\begin{array}{c|c} A_k & R_k \mathbf{c}_{k+1}^T \\ \hline \mathbf{d}_{k+1} S_k^T & a_{k+1,k+1} \end{array}\right] \left[\begin{array}{c} Z_{k+1} \\ \hline \alpha_{k+1} \end{array}\right] = -\left[\begin{array}{c} R_k P_k \\ \hline \xi_{k+1} \end{array}\right],$$

with $Z_{k+1} \in \mathbb{R}^{k \times p_1}$ and $\alpha_{k+1} \in \mathbb{R}^{1 \times p_1}$. Expanding this equation towards its block-rows, we observe that

$$A_k \; Z_{k+1} + R_k \; \mathbf{c}_{k+1}^T \; \alpha_{k+1} = -R_k P_k, \tag{3.2}$$

and

$$\mathbf{d}_{k+1} \; S_k^T Z_{k+1} + a_{k+1,k+1} \alpha_{k+1} = -\xi_{k+1}. \tag{3.3}$$

Rewriting (3.2) towards Z_{k+1} and using the solution of (3.1) gives

$$\begin{aligned} Z_{k+1} &= -A_k^{-1} R_k \left(P_k + \mathbf{c}_{k+1}^T \; \alpha_{k+1}\right) \\ &= Y_k \left(P_k + \mathbf{c}_{k+1}^T \; \alpha_{k+1}\right). \end{aligned} \tag{3.4}$$

Substituting the latter equation in (3.3) leads to:

$$\mathbf{d}_{k+1} \; S_k^T \; Y_k (P_k + \mathbf{c}_{k+1}^T \alpha_{k+1}) \; + a_{k+1,k+1} \alpha_{k+1} = -\xi_{k+1},$$

from which we can extract α_{k+1} as:

$$\alpha_{k+1} = -\frac{\xi_{k+1} + \mathbf{d}_{k+1} S_k^T Y_k P_k}{\mathbf{d}_{k+1} S_k^T Y_k \mathbf{c}_{k+1}^T + a_{k+1,k+1}}.$$

Using now the vector α_{k+1} in (3.4), we can compute the matrix Z_{k+1}. Based on the formulas for α_{k+1} and Z_{k+1}, we can immediately derive a recursive algorithm, for solving the Yule-Walker-like problems.

To conclude this section, we prove that the denominator in the formula for α_{k+1} is always nonzero, i.e. that the computation of α_{k+1} is well defined. Because our matrix A is assumed to be strongly nonsingular, we know that all the leading principal matrices are nonsingular. This means that for every nonzero vector w: $A_{k+1} w \neq 0$. Taking now $w^T = [\mathbf{c}_{k+1} Y_k^T, 1]$, we have that:

$$\begin{aligned} &\left[\begin{array}{c|c} A_k & R_k \mathbf{c}_{k+1}^T \\ \hline \mathbf{d}_{k+1} S_k^T & a_{k+1,k+1} \end{array}\right] \left[\begin{array}{c} Y_k \mathbf{c}_{k+1}^T \\ 1 \end{array}\right] \\ &= \left[\begin{array}{c} A_k Y_k \mathbf{c}_{k+1}^T + R_k \mathbf{c}_{k+1}^T \\ \mathbf{d}_{k+1} S_k^T Y_k \mathbf{c}_{k+1}^T + a_{k+1,k+1} \end{array}\right] \\ &\neq 0. \end{aligned}$$

Using the fact that $A_k Y_k = -R_k$, we obtain that the first k entries of the vector above are zero. As the total vector needs to be different from zero we have that:

$$\mathbf{d}_{k+1} S_k^T Y_k \mathbf{c}_{k+1}^T + a_{k+1,k+1} \neq 0,$$

which states that the calculation of α_{k+1} is well defined.

Based on the results presented in this section we will derive now the Levinson-like algorithm for solving an arbitrary system of equations.

ETNA
Kent State University
etna@mcs.kent.edu

3.2. The Levinson-like solver. In this section a Levinson-like method is proposed for solving systems of equations for which the coefficient matrix A is simple p_1-Levinson conform. The presented solver uses the solution of all the kth order Yule-Walker-like problems and it is based on an inductive procedure.

Suppose a matrix A which is simple p_1-Levinson conform is given, and use the notation from Definition 2.1 and the one from Section 3.1. We would like to compute a solution $\mathbf{x}$ for the following system of equations $A\mathbf{x} = \mathbf{b}$, where $\mathbf{b}^T = [b_1, \dots, b_n]$ is a general right-hand side. In this section we also assume the matrix A to be strongly nonsingular. As already mentioned, further in the text we will omit this strongly nonsingularity condition.

Assume we know the solution of the kth order Yule-Walker system:

$$A_k Y_k = -R_k, \tag{3.5}$$

and the solution of:

$$A_k \mathbf{x}_k = \mathbf{b}_k, \tag{3.6}$$

where $\mathbf{b}_k^T = [b_1, \dots, b_k]$. We will now solve $A_{k+1}\ \mathbf{x}_{k+1} = \mathbf{b}_{k+1}$, based on the (3.5) and (3.6).

The system we would like to solve can be rewritten in block form as:

$$\left[\begin{array}{c|c} A_k & R_k \mathbf{c}_{k+1}^T \\ \hline \mathbf{d}_{k+1} S_k^T & a_{k+1,k+1} \end{array}\right] \left[\begin{array}{c} \mathbf{w}_{k+1} \\ \hline \mu_{k+1} \end{array}\right] = \left[\begin{array}{c} \mathbf{b}_k \\ \hline b_{k+1} \end{array}\right], \tag{3.7}$$

with $\mathbf{w}_{k+1} \in \mathbb{R}^{k \times 1}$ and μ_{k+1} a scalar.

Expanding (3.7) leads to the following two equations

$$A_k\ \mathbf{w}_{k+1} + \mu_{k+1}\ R_k \mathbf{c}_{k+1}^T\ = \mathbf{b}_k \tag{3.8}$$

and

$$\mathbf{d}_{k+1}\ S_k^T\ \mathbf{w}_{k+1} + \mu_{k+1}\ a_{k+1,k+1} = b_{k+1}. \tag{3.9}$$

Equation (3.8) can be solved for $\mathbf{w}_{k+1}$. Using $A_k^{-1}\mathbf{b}_k = \mathbf{x}_k$ and $A_k^{-1}(-R_k) = Y_k$, we thus get:

$$\begin{aligned} \mathbf{w}_{k+1} &= A_k^{-1}\left(\mathbf{b}_k - \mu_{k+1} R_k \mathbf{c}_{k+1}^T\right) \\ &= \mathbf{x}_k + \mu_{k+1}\ Y_k \mathbf{c}_{k+1}^T. \end{aligned}$$

Substituting the solution for $\mathbf{w}_{k+1}$ into (3.9) and rewriting this leads to the following expression for μ_{k+1}:

$$\mu_{k+1} = \frac{b_{k+1} - \mathbf{d}_{k+1} S_k^T \mathbf{x}_k}{\left(\mathbf{d}_{k+1} S_k^T Y_k \mathbf{c}_{k+1}^T + a_{k+1,k+1}\right)}.$$

The formula for μ_{k+1} is well defined as the denominator is always different from zero; see Section 3.1. Using the relations for computing μ_{k+1} and w_{k+1} one can immediately derive a recursive formula for computing the solution. We remark that for solving the system of equations, with this Levinson-like algorithm, the solution of the nth Yule-Walker-like equation is not needed, hence we do not necessarily need to define the matrix R_n. In the next section this algorithm and the operation count is presented.

ETNA
Kent State University
etna@mcs.kent.edu

3.3. A first algorithm of complexity $O(p_1p_2n^2)$. Based on the previous two sections we can present a first version of a Levinson-like solver for simple p_1-Levinson conform matrices.

The algorithm is given in a simple mathematical formulation. First the problem is initialized and then the main loop is performed. After each computation in the algorithm, the number of flops[1] involved is shown. We remark that the presented algorithm, is not the most efficient implementation. It is written in this way, to clearly see the computationally most expensive steps. For the operation count, we assume that the multiplication of a row vector with any of the matrices P_k has a flop count bounded by $\kappa_1 n + \kappa_2$, with κ_1 and κ_2 two constants.

ALGORITHM 3.1. *Initialize*

$$Y_1 = \alpha_1 = \frac{-R_1}{a_{1,1}}$$
$$\mathbf{x}_1 = \mu_1 = \frac{b_1}{a_{1,1}}$$

For $k = 1, \ldots, n-1$ *do*

1. *Compute and store the following variables:*
   (a) $S_k^T Y_k$ — *Flops:* $p_1p_2(2k-1)$
   (b) $S_k^T \mathbf{x}_k$ — *Flops:* $p_2(2k-1)$
   (c) $\mathbf{d}_{k+1}(S_k^T Y_k)$ — *Flops:* $p_1(2p_2-1)$
   (d) $(\mathbf{d}_{k+1}S_k^T Y_k \mathbf{c}_{k+1}^T + a_{k+1,k+1})$ — *Flops:* $2p_1$
2. $\alpha_{k+1} = -\frac{\xi_{k+1}+\mathbf{d}_{k+1}S_k^T Y_k P}{\mathbf{d}_{k+1}S_k^T Y_k \mathbf{c}_{k+1}^T + a_{k+1,k+1}}$ — *Flops:* $2p_1 + \kappa_1 p_1 + \kappa_2$
3. $Z_{k+1} = Y_k \left(P + \mathbf{c}_{k+1}^T\ \alpha_{k+1}\right)$ — *Flops:* $k(\kappa_1 p_1 + \kappa_2) + k(2p_1 - 1) + 2p_1 k$
4. $\mu_{k+1} = \frac{b_{k+1} - \mathbf{d}_{k+1}S_k^T \mathbf{x}_k}{\mathbf{d}_{k+1}S_k^T Y_k \mathbf{c}_{k+1}^T + a_{k+1,k+1}}$ — *Flops:* $1 + 2p_2$
5. $\mathbf{w}_{k+1} = \mathbf{x}_k + \mu_{k+1}\ Y_k \mathbf{c}_{k+1}^T$ — *Flops:* $2k + k(2p_1 - 1)$
6. $Y_{k+1}^T = [Z_k^T, \alpha_k^T]$ — *Flops:* 0
7. $\mathbf{x}_{k+1}^T = [w_k^T, \mu_k^T]$ — *Flops:* 0

endfor;

Performing an overall complexity count leads us to an algorithm of complexity $O(p_1p_2n^2)$. This means that as long as the factor $p_1p_2 < n$, the method will perform better than Gaussian elimination if the matrices in question are large enough. However, taking a closer look at the involved computations, we can see that the bottlenecks, causing the factor n^2 in the operation count, are the computations of the matrices $S_k^T Y_k$, and $S_k^T \mathbf{x}_k$, and the explicit formation of the matrices Z_{k+1} and vectors $\mathbf{w}_{k+1}$. Assume, now that one could remove the computation of Z_{k+1} and $\mathbf{w}_{k+1}$ out of the most inner loop, and compute the final solution, using only the stored values of α_k and μ_k in $O(n)$ operations (dependent on p_1 and p_2 however). Assume also that one could compute the products $S_k^T \mathbf{x}_k$ and $S_k^T Y_k$ in constant time, independent of k. This would lead to the following algorithm.

ALGORITHM 3.2. *Initialize*

$$Y_1 = \alpha_1 = \frac{-R_1}{a_{1,1}}$$
$$\mathbf{x}_1 = \mu_1 = \frac{b_1}{a_{1,1}}$$

For $k = 1, \ldots, n-1$ *do the same initializations:*

[1] A floating point operation (flop) consists of any of the following operations: $+, -, :, *$. A sign change is not counted as an operation.

ETNA
Kent State University
etna@mcs.kent.edu

1. *Compute and store the following variables:*
   - *(a)* $S_k^T Y_k$ — *Flops: Independent of k*
   - *(b)* $S_k^T \mathbf{x}_k$ — *Flops: Independent of k*
   - *(c)* $(\mathbf{d}_{k+1} S_k^T Y_k \mathbf{c}_{k+1}^T + a_{k+1,k+1})$ — *Flops:* $p_1(2p_2 - 1) + 2p_1$
2. $\alpha_{k+1} = -\frac{\xi_{k+1} + \mathbf{d}_{k+1} S_k^T Y_k P}{\mathbf{d}_{k+1} S_k^T Y_k \mathbf{c}_{k+1}^T + a_{k+1,k+1}}$ — *Flops:* $1 + p_1 + \kappa_1 p_1 + \kappa_2$
3. $\mu_{k+1} = \frac{b_{k+1} - \mathbf{d}_{k+1} S_k^T \mathbf{x}_k}{\mathbf{d}_{k+1} S_k^T Y_k \mathbf{c}_{k+1}^T + a_{k+1,k+1}}$ — *Flops:* $1 + 2p_2$

endfor;

Under these assumptions, the solver has a complexity: $O(p_1 p_2 n)$.

In the next section, we illustrate how to achieve the above complexity, by computing the solution in another loop and by computing the products $S_k^T Y_k$ and $S_k^T \mathbf{x}_k$, in an inductive way, making thereby the computation indepent of k.

3.4. Reduction of the complexity. We know from the previous section that the computations of the matrices $Y_k^T = [Z_k^T, \alpha_k^T]$, the vectors $\mathbf{x}_k = [\mathbf{w}_k^T, \mu_k]$ and the computations of the matrices $S_k^T \mathbf{x}_k$ and $S_k^T Y_k$ in every step of the algorithm are responsible for the n^2 factor in the complexity count. If we could reduce the complexity of these operations, the overall operation count would decrease by a factor n. Let us start with the computation of the solution vector $\mathbf{x}$. Instead of computing for every k the vector $\mathbf{x}_k$, wich incorporates the computation of $\mathbf{w}_k, Z_k$ and Y_k at every step, we will postpone this computation up to the very end, and simple store the factors μ_k and α_k for every k. Extracting the computation of $\mathbf{x}$ out of the loop, the final solution vector can be written in the following form. (Denote with x_i the ith component of the vector $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$.)

$$
\begin{aligned}
\mathbf{x} &= \begin{bmatrix} \vdots \\ \mu_{n-3} + \alpha_{n-3}\left(\mu_{n-2}\mathbf{c}_{n-2}^T + (P_{n-3} + \mathbf{c}_{n-2}^T \alpha_{n-2})\left(\mu_{n-1}\mathbf{c}_{n-1}^T + (P_{n-2} + \mathbf{c}_{n-1}^T \alpha_{n-1})\mu_n \mathbf{c}_n^T\right)\right) \\ \mu_{n-2} + \alpha_{n-2}\left(\mu_{n-1}\mathbf{c}_{n-1}^T + (P_{n-2} + \mathbf{c}_{n-1}^T\alpha_{n-1})\mu_n \mathbf{c}_n^T\right) \\ \mu_{n-1} + \alpha_{n-1}\mu_n \mathbf{c}_n^T \\ \mu_n \end{bmatrix} \\
&= \begin{bmatrix} \vdots \\ \mu_{n-3} + \alpha_{n-3}\left(x_{n-2}\mathbf{c}_{n-2}^T + P_{n-3}\left(x_{n-1}\mathbf{c}_{n-1}^T + P_{n-2}x_n\mathbf{c}_n^T\right)\right) \\ \mu_{n-1} + \alpha_{n-2}\left(x_{n-1}\mathbf{c}_{n-1}^T + P_{n-2}x_n\mathbf{c}_n^T\right) \\ \mu_{n-1} + \alpha_{n-1}\left(x_n \mathbf{c}_{\mathbf{n}}^{\mathbf{T}}\right) \\ \mu_n \end{bmatrix}.
\end{aligned}
$$

The computation of the vector above can easily be rewritten in a recursive manner, as the term following the vector α_{n-1} in the computation of x_{n-1}, can be used in the computation of x_{n-2}. Consecutively, the term following the vector α_{n-2} in the computation of x_{n-2}, can be used in the computation of x_{n-3}, and so on. Implementing this in a recursive way, from bottom to top requires $O(p_1 n)$ operations for computing the solution, instead of the $O(p_1 n^2)$ operations needed if it was incorporated in the main loop. The implementation of this recursion is given the next section.

Secondly, one needs to reduce the complexity of the computation of the matrices $S_k^T Y_k$ and $S_k^T \mathbf{x}_k$ in the loop. This reduction in complexity is related to the structure in the lower triangular part, as the following example shows:

EXAMPLE 3.3. A simple case, considers the class of upper triangular matrices. This means that the matrix $S_k = 0$. Hence all the computations involving the matrix S_k, such as $S_k^T \mathbf{x}_k$ and $S_k^T Y^T$ are removed thereby creating an $O(p_1 n)$ solver for upper triangular, simple p_1-Levinson conform matrices.

We would like to place this however in a more general framework. The class of matrices admitting this reduction in complexity is the class of simple (p_1, p_2)-Levinson conform matrices. Assume a simple (p_1, p_2)-Levinson conform matrix A is given. This means that the lower triangular part is structured in a similar way as the upper triangular part. We have that our matrices S_k satisfy the following relations (for $1 \leq k \leq n-1$):

$$S_{k+1} = \begin{bmatrix} S_k Q_k \\ \eta_{k+1} \end{bmatrix},$$

where Q_k is a matrix of dimension $p_2 \times p_2$ and η_{k+1} is a row vector of length p_2.

Using this relation, the computation of $S_{k+1}^T Y_{k+1}$ can be rewritten in terms of the matrix $S_k^T Y_k$, admitting thereby a recursive computation of these products:

$$\begin{aligned} S_{k+1}^T Y_{k+1} &= [Q_k^T S_k^T, \eta_{k+1}^T] \begin{bmatrix} Z_{k+1} \\ \alpha_{k+1} \end{bmatrix} \\ &= Q_k^T S_k^T Z_{k+1} + \eta_{k+1}^T \alpha_{k+1} \\ &= Q_k^T S_k^T Y_k (P_k + \mathbf{c}_{k+1}^T \alpha_{k+1}) + \eta_{k+1}^T \alpha_{k+1} \\ &= Q_k^T S_k^T Y_k P_k + (Q_k^T S_k^T Y_k \mathbf{c}_{k+1}^T + \eta_{k+1}^T) \alpha_{k+1}. \end{aligned} \tag{3.10}$$

This leads to a recursive calculation for which the computational complexity is independent of k.

In a similar way, a recursive formula for computing the products $S_k^T \mathbf{x}_k$ can be derived:

$$\begin{aligned} S_{k+1}^T \mathbf{x}_{k+1} &= [Q_k^T S_k^T, \eta_{k+1}^T] \begin{bmatrix} \mathbf{w}_{k+1} \\ \mu_{k+1} \end{bmatrix} \\ &= Q_k^T S_k^T \mathbf{x}_k + \mu_{k+1} \left(Q_k^T S_k^T Y_k \mathbf{c}_{k+1}^T + \eta_{k+1}^T \right). \end{aligned} \tag{3.11}$$

This recursive formula for computing $S_k^T \mathbf{x}_k$ in each step of the loop is computationally independent of k.

In the first section we proved that every matrix is simple $(n-1, n-1)$-Levinson conform. Solving a system of equations with a strongly nonsingular simple $(n-1, n-1)$-Levinson conform matrix, therefore costs $O(n^3)$.

In the following section the algorithm for solving strongly nonsingular simple (p_1, p_2)-Levinson conform systems of equations in $O(p_1 p_2 n)$ operations is presented.

3.5. A second algorithm of complexity $O(p_1 p_2 n)$. Before giving examples of matrices solvable via these Levinson-like methods, we present here the general algorithm for simple (p_1, p_2)-Levinson conform matrices. For the computation of $S_k^T Y_k$ and $S_k^T \mathbf{x}_k$ we use the recursive schemes presented in (3.10) and (3.11). We know that for every k the multiplication of a vector with the matrices Q_k and/or P_k can be performed in linear time. Assume that $\kappa_1 p_1 + \kappa_2$ is an upper bound for the number of operations needed to multiply a vector with a matrix P_k, and assume $\gamma_1 p_2 + \gamma_2$ to be an upper bound for the number of operations needed to multiply a vector with a matrix Q_k. The algorithm is presented below. After each computation, the number of involved operations is shown (flop count). The operation count presented here is the worst case scenario, as we do not take into consideration other possible advantages such as sparsity in multiplications and so on. One can see this complexity count as an upper bound.

ALGORITHM 3.4. *Initialize*

$$\alpha_1 = \frac{-R_1}{a_{1,1}} \qquad \textit{Flops: } p_1$$
$$\mu_1 = \frac{b_1}{a_{1,1}} \qquad \textit{Flops: } 1$$
$$S_1^T Y_1 = S_1^T \alpha_1 \qquad \textit{Flops: } p_2 p_1$$
$$S_1^T x_1 = S_1^T \mu_1 \qquad \textit{Flops: } p_1$$

For $k = 1, \ldots, n-1$ *do:*

1. *Compute and store the following variable:*
   - *(a)* $(\mathbf{d}_{k+1} S_k^T Y_k)$ *Flops:* $(2p_2 - 1)p_1$
   - *(b)* $(\mathbf{d}_{k+1} S_k^T Y_k \mathbf{c}_{k+1}^T + a_{k+1,k+1})$ *Flops:* $2p_1$
2. $\alpha_{k+1} = -\frac{\xi_{k+1} + \mathbf{d}_{k+1} S_k^T Y_k P_k}{\mathbf{d}_{k+1} S_k^T Y_k \mathbf{c}_{k+1}^T + a_{k+1,k+1}}$ *Flops:* $2p_1 + \kappa_1 p_1 + \kappa_2$
3. $\mu_{k+1} = \frac{b_{k+1} - \mathbf{d}_{k+1} S_k^T x_k}{\mathbf{d}_{k+1} S_k^T Y_k \mathbf{c}_{k+1}^T + a_{k+1,k+1}}$ *Flops:* $2p_2 + 1$
4. *Compute and store the following variables (if* $k < n-1$*):*
   - *(a)* $Q_k^T S_k^T Y_k c_{k+1}^T + \eta_{k+1}^T$ *Flops:* $2p_2 p_1 + \gamma_1 p_2 + \gamma_2$
   - *(b)* $S_{k+1}^T Y_{k+1} = Q_k^T S_k^T Y_k P_k + \left(Q^T S_k^T Y_k \mathbf{c}_{k+1}^T + \eta_{k+1}^T\right) \alpha_{k+1}$
     Flops: $2p_2 p_1 + p_1(\gamma_1 p_2 + \gamma_2) + p_2(\kappa_1 p_1 + \kappa_2)$
   - *(c)* $S_{k+1}^T \mathbf{x}_{k+1} = Q_k^T S_k^T \mathbf{x}_k + \left(Q_k^T S_k^T Y_k \mathbf{c}_{k+1}^T + \eta_{k+1}^T\right) \mu_{k+1}$
     Flops: $2p_2 + \gamma_1 p_2 + \gamma_2$

endfor;

Computation of the solution vector (h *is a dummy variable)*

$$x_n = \mu_n$$
$$h = \mu_n \mathbf{c}^T \qquad \textit{Flops: } p_1$$

For $k = n-1, -1, 1$ *do*

1. $x_k = \mu_k + \alpha_k h$ *Flops:* $2p_1$
2. $h = x_i \mathbf{c}_i^T + P_{k-1} h$ *Flops:* $2p_1 + \kappa_1 p_1 + \kappa_2$

endfor.

It is clear that an overal complexity count leads to method with complexity $O(p_1 p_2 n)$. A more detailed summation of the involved operations gives us an overall complexity of

$$(n-1)\Big[(6+\gamma_1+\kappa_1)p_1p_2 + (2\kappa_1+\gamma_2+7)p_1 + (2\gamma_1+\kappa_2+4)p_2 + 2\gamma_2 + 2\kappa_2 + 1\Big]$$
$$+(-3-\gamma_1-\kappa_1)p_1p_2 + (3-\gamma_2)p_1 + (-2-2\gamma_1-\kappa_2)p_2 + (1-2\gamma_2).$$

3.6. (p_1, p_2)**-Levinson conform matrices with** $E_k = F_k = I_k$**.** The considered class of (p_1, p_2)-Levinson conform matrices was called simple, because the corresponding solver had a complexity of $O(p_1 p_2 n)$. Suppose now that we omit this simplicity assumption. This would immediately increase the complexity. For example in the Toeplitz case, we cannot extract the computation of the solution vector $\mathbf{x}$ out of the most inner loop and hence we have immediately a complexity of $O(n^2)$. If we assume however that the matrices $E_k = F_k = I_k$, Algorithm 3.4 remains valid, but it will increase in complexity, because we cannot perform multiplications with the matrices P_k or Q_k in linear time anymore. In this section we will assume the matrices E_k and F_k equal to the identity matrix (because then Algorithm 3.4 remains valid), and we will see what the impact of the multiplications with the matrices P_k and Q_k is on the complexity count.

- Assume we cannot multiply vectors with the matrices P_k in linear time, but we can multiply vectors with the matrices Q_k in linear time. This means that in Algorithm 3.4, the following steps increase in complexity:
  - in the main loop, steps
    2. $\alpha_{k+1} = -\frac{\xi_{k+1}+\mathbf{d}_{k+1}S_k^TY_kP_k}{\mathbf{d}_{k+1}S_k^TY_k\mathbf{c}_{k+1}^T+a_{k+1,k+1}}$ Flops: $(2p_1+1)p_1$
    4. (b) $S_{k+1}^TY_{k+1} = Q_k^TS_k^TY_kP_k + \left(Q^TS_k^TY_k\mathbf{c}_{k+1}^T + \eta_{k+1}^T\right)\alpha_{k+1}$
       Flops: $2p_2p_1 + p_1(\gamma_1p_2+\gamma_2) + p_1p_2(2p_1-1)$
  - in the computation of the solution, step
    2. $h = x_i\mathbf{c}_i^T + P_{k-1}h$ Flops: $(2p_1+1)p_1$.

  This means that we will get an Algorithm of complexity $O(p_1^2p_2n)$. More precisely we get the following operation count:

$$(n-1)\Big[(2p_2+4)\,p_1^2 + (\gamma_1+5)\,p_1p_2 + (\gamma_2+5)\,p_1 + (2\gamma_1+4)\,p_2 + 2\gamma_2+1\Big] + O(1).$$

- Assume the multiplication of a vector with the matrices P_k can be done in linear time, but not the multiplication with the matrices Q_k. In Algorithm 3.4, the following steps increase in complexity:
  - in the main loop, steps
    4. (a) $Q_k^TS_k^TY_kc_{k+1}^T + \eta_{k+1}^T$ Flops: $2p_2p_1 + p_2(2p_2-1)$
    4. (b) $S_{k+1}^TY_{k+1} = Q_k^TS_k^TY_kP_k + \left(Q^TS_k^TY_k\mathbf{c}_{k+1}^T + \eta_{k+1}^T\right)\alpha_{k+1}$
       Flops: $2p_2p_1 + p_2(\kappa_1p_1+\kappa_2) + p_1p_2(2p_2-1)$
    4. (c) $S_{k+1}^T\mathbf{x}_{k+1} = Q_k^TS_k^T\mathbf{x}_k + \left(Q_k^TS_k^TY_k\mathbf{c}_{k+1}^T + \eta_{k+1}^T\right)\mu_{k+1}$
       Flops: $p_2(2p_2+1)$
  - in the computation of the solution nothing changes.

  Overall, we get the following operation count:

$$(n-1)\Big[2p_1p_2^2 + 3p_2^2 + (\kappa_1+5)\,p_1p_2 + (2\kappa_1+7)\,p_1 + (\kappa_2+2)\,p_2 + 2\kappa_2+1\Big] + O(1).$$

- Suppose none of the matrices P_k or Q_k admits a multiplication in linear time. Hence the algorithm increases in complexity, and we get:
  - in the main loop, steps
    2. $\alpha_{k+1} = -\frac{\xi_{k+1}+\mathbf{d}_{k+1}S_k^TY_kP_k}{\mathbf{d}_{k+1}S_k^TY_k\mathbf{c}_{k+1}^T+a_{k+1,k+1}}$ Flops: $(2p_1+1)p_1$
    4. (a) $Q_k^TS_k^TY_kc_{k+1}^T + \eta_{k+1}^T$ Flops: $2p_2p_1 + p_2(2p_2-1)$
    4. (b) $S_{k+1}^TY_{k+1} = Q_k^TS_k^TY_kP_k + \left(Q^TS_k^TY_k\mathbf{c}_{k+1}^T + \eta_{k+1}^T\right)\alpha_{k+1}$
       Flops: $2p_2p_1 + p_1p_2(2p_2-1) + p_1p_2(2p_1-1)$
    4. (c) $S_{k+1}^T\mathbf{x}_{k+1} = Q_k^TS_k^T\mathbf{x}_k + \left(Q_k^TS_k^TY_k\mathbf{c}_{k+1}^T + \eta_{k+1}^T\right)\mu_{k+1}$
       Flops: $p_2(2p_2+1)$
  - in the computation of the solution, steps
    2. $h = x_i\mathbf{c}_i^T + P_{k-1}h$ Flops: $(2p_1+1)p_1$.

  If no linear multiplications with vectors are possible, we get the following complexity:

$$(n-1)\Big[2p_1^2p_2 + 2p_1p_2^2 + 4p_1^2 + 4p_1p_2 + 3p_2^2 + 5p_1 + 2p_2 + 1\Big] + O(1).$$

In Section 5 there is a class of matrices included (quasiseparable matrices), which are (p_1,p_2)-Levinson conform, with $E_k = F_k = I_k$, and the matrices P_k and Q_k do not admit a linear multiplication with a vector.

ETNA
Kent State University
etna@mcs.kent.edu

4. An upper triangular factorization. The Levinson-like algorithm as presented in this paper applied to a matrix A, is closely related to an upper triangular factorization of the involved matrix. This is also the case for the Levinson algorithm related to Toeplitz matrices; see, e.g., [21, 24]. Even though a similar construction is possible for general Levinson conform matrices, we will focus here on the case for which $E_k = F_k = I_k$. Suppose we have a Levinson conform matrix A. Writing down the solution of the kth order Yule-Walker-like problem, gives us:

$$A_k Y_k = -R_k.$$

Bringing the matrix R_k to the left-hand side, and multiplying this equation on the right with the vector $\mathbf{c}_{k+1}^T$ gives us:

$$A_k Y_k \mathbf{c}_{k+1}^T + R_k \mathbf{c}_{k+1}^T = 0.$$

This equation can be rewritten in terms of the matrix A_{k+1}:

$$\left[\begin{array}{c|c} A_k & R_k \mathbf{c}_{k+1}^T \\ \hline \mathbf{d}_{k+1} S_k^T & a_{k+1,k+1} \end{array}\right] \left[\begin{array}{c} Y_k \mathbf{c}_{k+1}^T \\ 1 \end{array}\right] = \left[\begin{array}{c} 0 \\ \sigma_{k+1} \end{array}\right],$$

with σ_{k+1} different from zero. If we put the successive equations for the different values of k, in an upper triangular matrix we get (we remark that the products $Y_k \mathbf{c}_{k+1}^T$ are column vectors of dimension k):

$$A \left[\begin{array}{ccccc} 1 & Y_1\mathbf{c}_2^T & Y_2\mathbf{c}_3^T & \dots & Y_{n-1}\mathbf{c}_n^T \\ 0 & 1 & & & \\ 0 & 0 & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ 0 & 0 & \dots & 0 & 1 \end{array}\right] = \left[\begin{array}{ccccc} \sigma_1 & 0 & 0 & \dots & 0 \\ \times & \sigma_2 & 0 & \dots & 0 \\ \vdots & \times & \sigma_3 & & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \times & \times & \dots & \times & \sigma_n \end{array}\right].$$

The $\times$ denote arbitrary elements in the matrix. Rewriting the equations above, defining the upper triangular matrix to the right of A as U and rewriting the matrix on the right-hand side as $L\Lambda$, with $\Lambda = \mathrm{diag}(\sigma_1, \sigma_2, \dots, \sigma_n)$ a diagonal matrix and L lower triangular. We get the following two relations:

$$\begin{aligned} A &= L\Lambda U^{-1}, \\ A^{-1} &= U\Lambda^{-1}L^{-1}, \end{aligned}$$

where U and L are respectively upper and lower triangular matrices, with ones on the diagonal, and Λ a diagonal matrix. Moreover, we do not get the matrices U and L, but in fact we get their generators, in terms of the vectors α_k and μ_k. This factorization can therefore be used to compute the inverse of the matrix A.

We would like to stress once more that the presented Levinson-type method **can** be used to compute the LU^{-1}-decomposition of the involved matrices. One can clearly see that computing this LU^{-1}-decomposition involves the computation of the matrices Y_i, these matrices are however not necessary for solving the systems of equations as shown in the previous sections. These factors L and U^{-1} are not computed in the solver. The fact that in the presented solver the factors L and U^{-1} are not explicitly formed, creates a low cost algorithm for solving several systems of equations.

ETNA
Kent State University
etna@mcs.kent.edu

5. Examples. In this section we will provide several classes of matrices, which are simple (p_1, p_2)-Levinson conform. Hence we can apply our previously designed algorithm and come to an $O(p_1 p_2 n)$ solver for these systems of equations. We do not always derive the complete algorithm, but we define all the necessary matrices to illustrate that the presented matrices are indeed simple (p_1, p_2)-Levinson conform matrices.

5.1. Givens-vector representable semiseparable matrices. Nowadays a lot of attention is being paid to semiseparable, and closely related matrices. Systems of equations with semiseparable plus diagonal coefficient matrices arise for example in differential and integral equations [23, 28], in oscillation theory [19], statistics [22], and so on.

For example, in the papers [8, 12, 14, 15, 18, 26, 30] different methods were proposed for solving semiseparable systems of equations. Some of these papers cover more general structures than the simple semiseparable one, the papers [26, 30] focus explicitely on the class of semiseparable matrices of semiseparability rank 1 plus a diagonal. For this class of matrices, the complexities of some of the different solvers are $54n$ flops for [30], $58n$ flops for [18], $59n$ flops for [8] and $80n$ flops for [14].

As the structure of the simple (p_1, p_2)-Levinson conform matrices, as presented in this paper does not extend towards the diagonal, there is no loss of generality when solving a simple (p_1, p_2)-Levinson conform matrix plus a diagonal. In fact it does not even increase the complexity. This means that the solver we derive below is also applicable for semiseparable plus diagonal matrices.

In [32], an alternative representation for semiseparable matrices of semiseparability rank 1 was presented. The representation is based on a sequence of Givens transformations and a vector. Let us consider here the unsymmetric case, namely an unsymmetric semiseparable matrix of semiseparability rank 1, represented by using two sequences of Givens rotations and a vector. (More information concerning this representation with respect to other types of representations and algorithms related to it can be found in [32].) Let us denote the first sequence of Givens rotations, and the vector for representing the lower triangular part as:

$$G = \begin{bmatrix} c_1 & c_2 & \dots & c_{n-1} \\ s_1 & s_2 & \dots & s_{n-1} \end{bmatrix},$$
$$d = \begin{bmatrix} d_1 & d_2 & \dots & d_n \end{bmatrix}$$

and the second sequence of rotations and the vector, representing the strictly upper triangular part as:

$$H = \begin{bmatrix} r_1 & r_2 & \dots & r_{n-2} \\ t_1 & t_2 & \dots & t_{n-2} \end{bmatrix},$$
$$e = \begin{bmatrix} e_1 & e_2 & \dots & e_{n-1} \end{bmatrix}.$$

The matrices G and H contain in the first row, the cosines of Givens transformations and in the second row the sines of the Givens transformation; every column corresponds to one Givens rotation. The resulting semiseparable matrix S is of the following form

$$S = \begin{bmatrix} c_1 d_1 & r_1 e_1 & r_2 t_1 e_1 & \dots & r_{n-2} t_{n-3} \cdots t_1 e_1 & t_{n-2} t_{n-3} \cdots t_1 e_1 \\ c_2 s_1 d_1 & c_2 d_2 & r_2 e_2 & & r_{n-2} t_{n-3} \cdots t_2 e_1 & t_{n-2} t_{n-3} \cdots t_1 e_1 \\ c_3 s_2 s_1 d_1 & c_3 s_2 d_2 & c_3 d_3 & \ddots & \vdots & \vdots \\ \vdots & & \ddots & \ddots & r_{n-2} e_{n-2} & t_{n-2} e_{n-2} \\ c_{n-1} s_{n-2} \cdots s_1 d_1 & \dots & & & c_{n-1} d_{n-1} & e_{n-1} \\ s_{n-1} s_{n-2} \cdots s_1 d_1 & \dots & & & s_{n-1} d_{n-1} & d_n \end{bmatrix}.$$

ETNA
Kent State University
etna@mcs.kent.edu

We will construct here only the matrices R_k, $\mathbf{c}_k$ and P_k corresponding to the upper triangular part. The matrices corresponding to the lower triangular part, can be constructed in a similar way.

Put $R_1 = e_1$ and define $\mathbf{c}_{k+1} = r_k$ for $k = 1, \ldots, n-2$ and $\mathbf{c}_n = 1$. If $P_k = t_k$ and $\xi_k = e_k$, we get for $k = 2, \ldots, n-1$:

$$R_k = \left[\begin{array}{c} R_{k-1}t_{k-1} \\ \hline e_k \end{array}\right] = \left[\begin{array}{c} e_1t_1t_2\cdots t_{k-1} \\ e_2t_2\cdots t_{k-1} \\ \vdots \\ e_{k-1}t_{k-1} \\ e_k \end{array}\right]$$

and hence (only the upper triangular part is shown)

$$A_{k+1} = \left[\begin{array}{c|c} A_k & R_kr_k \\ \hline & c_{k+1}d_{k+1} \end{array}\right] = \left[\begin{array}{c|c} & r_kt_{k-1}\cdots t_1e_1 \\ & r_kt_{k-1}\cdots t_2e_1 \\ A_k & \vdots \\ & r_ke_k \\ \hline & c_{k+1}d_{k+1} \end{array}\right].$$

Which states that our defined matrices, generate the matrix A. Moreover we remark that the multiplication of a vector with any of the matrices $P_k = t_k$ can clearly be performed in linear time. This defines all the matrices, and therefore, this matrix is simple $(1,1)$-Levinson conform and admits an $O(n)$ Levinson-like method. Using the operation count as presented in Section 3.5, we see that the cost of solving a system of equations is bounded by $24n - 29$ operations (with $\kappa_1 = \gamma_1 = 1$ and $\kappa_2 = \gamma_2 = 0$).

5.2. $(1,1)$**-Quasiseparable matrices.** In this section, the class of $(1,1)$-quasiseparable matrices is considered. General quasiseparable matrices (see [16]) are investigated in Section 5.4 as the general class is slightly different.

Currently a lot of attention is paid to the class of quasiseparable matrices; see, e.g., [12, 13, 17, 18]. Briefly speaking, we can say that a matrix is $(1,1)$-quasiseparable if every subblock taken out of the strictly upper triangular part of the matrix has rank at most 1 (the same holds for the lower triangular part). Hence, the structure does not incorporate the diagonal, in contrast to semiseparable matrices. The class of $(1,1)$ quasiseparable matrices incorporates for example the class of semiseparable matrices of semiseparability rank 1, tridiagonal matrices and unitary Hessenberg matrices. Several algorithms for solving systems of equations with quasiseparable matrices exist. The methods presented in [12, 18] compute a QR-factorization, and solve the system in this way. The methods presented in [13, 17] use a recursive scheme for computing the solution and are in fact closely related to the Levinson method presented here applied to quasiseparable matrices. A comparison between these approaches is given in Section 5.4.

Let us illustrate that $(1,1)$-quasiseparable matrices can be considered as simple $(1,1)$-Levinson conform matrices, and hence admit an $O(n)$ solver of the Levinson type.

The class of $(1,1)$-quasiseparable matrices $R = (r_{i,j})$ consists of matrices of the following form:

$$r_{i,j} = \begin{cases} p_ia_{i,j}^{\times}q_j, & 1 \le i < j \le n, \\ d_i, & 1 \le i = j \le n, \\ g_ib_{i,j}^{\times}h_j, & 1 \le j < i \le n. \end{cases}$$

ETNA
Kent State University
etna@mcs.kent.edu

Where $a^{\times}_{i,j} = a_{i+1} \dots a_{j-1}$ and $b^{\times}_{i,j} = b_{i+1} \dots b_{j-1}$ and all the elements a_i,b_i,p_i,q_i,g_i,h_i and d_i are scalars.

As the stucture of the upper triangular part is exactly the same as the structure from the lower triangular part, we will search for the matrices corresponding to the upper triangular part. The upper triangular part of such a $(1, 1)$-quasiseparable matrix R has the following structure:

$$R = \begin{bmatrix} d_1 & p_1q_2 & p_1a_2q_3 & p_1a_2a_3q_4 & \cdots & p_1a_2 \dots a_{n-1}q_n \\ & d_2 & p_2q_3 & p_2a_3q_4 & \cdots & p_2a_3 \dots a_{n-1}q_n \\ & & d_3 & p_3q_4 & \cdots & p_3a_4 \dots a_{n-1}q_n \\ & & & d_4 & & \vdots \\ & & & & \ddots & p_{n-1}q_n \\ & & & & & d_n \end{bmatrix}.$$

Initializing $R_1 = p_1$ and defining $\mathbf{c}_{k+1} = q_{k+1}$, $P_k = a_{k+1}$ and $\xi_k = p_k$ gives us for $k = 1, \dots, n-1$:

$$R_k = \left[\begin{array}{c} R_{k-1}P_{k-1} \\ \hline \xi_k \end{array}\right] = \left[\begin{array}{c} R_{k-1}a_k \\ \hline p_k \end{array}\right] = \begin{bmatrix} p_1a_2a_3 \cdots a_k \\ p_2a_3 \cdots a_k \\ \vdots \\ p_{k-1}a_k \\ p_k \end{bmatrix}$$

and hence

$$A_{k+1} = \left[\begin{array}{c|c} A_k & R_k\mathbf{c}_{k+1} \\ \hline & a_{k+1,k+1} \end{array}\right] = \left[\begin{array}{c|c} A_k & R_kq_{k+1} \\ \hline & d_{k+1} \end{array}\right] = \left[\begin{array}{c|c} & p_1a_2a_3 \cdots a_kq_{k+1} \\ & p_2a_3 \cdots a_kq_{k+1} \\ A_k & \vdots \\ & p_{k-1}a_kq_{k+1} \\ & p_kq_{k+1} \\ \hline & d_{k+1} \end{array}\right],$$

which gives us the desired matrices. All the conditions are satisfied (including the demands on the multiplication with P_k) to have a simple $(1, 1)$-Levinson conform matrix. Using the operation count as presented in Section 3.5, we see that the cost is bounded by $24n - 29$ operations (with $\kappa_1 = \gamma_1 = 1$ and $\kappa_2 = \gamma_2 = 0$), which is exactly the same number of operations for a semiseparable plus diagonal matrix represented with a sequence of Givens transformations and a vector. In fact both representations are closely related to each other; see [32].

A careful computation of the number of operations involved in Algorithm 5.3 proposed in [17] for solving a $(1, 1)$-quasiseparable matrices gives us, for the number of flops, the following complexity: $49n + O(1)$.

5.3. Higher order generator representable semiseparable matrices. In two recent papers [27, 31], we investigated how to solve symmetric strongly nonsingular systems of higher order semiseparable matrices plus band matrices using a Levinson-like technique. The results provided in this paper are more general. We include here the class of higher order generator representable matrices as an example. Moreover, we prove here that also nonsymmetric matrices fit in this framework. Implementations of these solvers are available; see [27, 31].

ETNA
Kent State University
etna@mcs.kent.edu

Suppose we have the following higher order generator representable semiseparable matrix A:

$$A = \begin{bmatrix} \mathbf{q}_1\mathbf{p}_1^T & \mathbf{q}_1\mathbf{p}_2^T & \cdots & \mathbf{q}_1\mathbf{p}_n^T \\ \mathbf{v}_1\mathbf{u}_1^T & \mathbf{q}_2\mathbf{p}_2^T & & \vdots \\ \vdots & & \ddots & \mathbf{q}_{n-1}\mathbf{p}_n^T \\ \mathbf{v}_1\mathbf{u}_{n-1}^T & \cdots & \mathbf{v}_{n-1}\mathbf{u}_{n-1}^T & \mathbf{q}_n\mathbf{p}_n^T \end{bmatrix},$$

for which all the row vectors $\mathbf{u}_i$ and $\mathbf{v}_i$ are of length p_2 and the row vectors $\mathbf{p}_i$ and $\mathbf{q}_i$ are of length p_1. This matrix is called an (p_1, p_2)-generator representable semiseparable matrix. We will prove now that this matrix is simple (p_1, p_2)-Levinson conform.

Let us define the matrices R_k and S_k as follows:

$$R_k = \begin{bmatrix} \mathbf{q}_1 \\ \vdots \\ \mathbf{q}_k \end{bmatrix} \text{ and } S_k = \begin{bmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_k \end{bmatrix}.$$

Defining $P_k = I_{p_1}$, $Q_k = I_{p_2}$, $\eta_k = \mathbf{v}_k$ and $\xi_k = \mathbf{q}_k$ gives us the following relations (these are Conditions 2 and 3 of Definition 2.2):

$$R_{k+1} = \left[\begin{array}{c} R_k \\ \hline \mathbf{q}_{k+1} \end{array}\right] \text{ and } S_{k+1} = \left[\begin{array}{c} S_k \\ \hline \mathbf{v}_{k+1} \end{array}\right].$$

Moreover the upper left $(k+1) \times (k+1)$ block of A is of the following form (Condition 1 of Definition 2.2):

$$A_{k+1} = \left[\begin{array}{c|c} A_k & R_k\mathbf{p}_{k+1}^T \\ \hline \mathbf{u}_k S_k^T & \mathbf{p}_{k+1}\mathbf{q}_{k+1}^T \end{array}\right].$$

This coefficient matrix satisfies all the properties to come to the desired $O(p_1p_2n)$ Levinson-like solver. Using the operation count as presented in Section 3.5, we see that the number of operations is bounded by

$$(6p_1p_2 + 7p_1 + 4p_2 + 1)(n-1) - 3p_1p_2 + 3p_1 - 2p_2 + 1.$$

5.4. General quasiseparable matrices. References concerning the class of quasiseparable matrices can be found in Section 5.2. In this section we will compare the Levinson-like method, with the algorithm presented in [17], for solving quasiseparable systems of equations $R\mathbf{x} = \mathbf{b}$. Before comparing both methods, we will briefly indicate how quasiseparable matrices fit into this framework. A general (p_1, p_2)-quasiseparable matrix is of the following form

$$r_{i,j} = \begin{cases} p_i a_{i,j}^{\times} q_j^T, & 1 \le i < j \le n, \\ d_i, & 1 \le i = j \le n, \\ g_i b_{i,j}^{\times} h_j^T, & 1 \le j < i \le n. \end{cases}$$

Where $a_{i,j}^{\times} = a_{i+1} \ldots a_{j-1}$,$b_{i,j}^{\times} = b_{i+1} \ldots b_{j-1}$, $p_i, q_i \in \mathbb{R}^{1\times p_1}$, $g_i, h_j \in \mathbb{R}^{1\times p_2}$, $a_i \in \mathbb{R}^{p_1\times p_1}$ and $b_i \in \mathbb{R}^{p_2\times p_2}$ for all i. Combining the techniques presented in Sections 5.3 and

5.2, we see that our quasiseparable matrix is (p_1, p_2)-Levinson conform (we do not include the details). The quasiseparable matrix is however not simple Levinson conform as our matrices $P_k = a_{k+1}$ and $Q_k = b_{k+1}$ and these matrices a_{k+1} and b_{k+1} do not necessarily admit a linear multiplication with a vector. Hence, we are in the case of Section 3.6, leading to a complexity of

$$n\left[2p_1^2p_2 + 2p_1p_2^2 + 4p_1^2 + 4p_1p_2 + 3p_2^2 + 5p_1 + 2p_2 + 1\right] + O(1).$$

for solving a quasiseparable matrix via the Levinson-like solver, as presented in this paper.

The method presented in [17], computes in fact the generators of the inverse of the quasiseparable matrix, and then applies a fast multiplication A^{-1} to the right-hand side $\mathbf{b}$ to compute the solution vector $\mathbf{x}$. Moreover the algorithm produces also the generators for the matrices L and U, in the following factorization of the matrix A:

$$A = L\Lambda U,$$

where L and U are respectively lower and upper triangular matrices, with ones on the diagonal, and Λ is a diagonal matrix. A complexity count of the algorithm proposed in [17], gives us the following number of flops[2]

$$n\left[4p_1^2p_2 + 4p_2^2p_1 + 12p_1p_2 + 8p_1^2 + 8p_2^2 + 5p_1 + 5p_2 + 3\right] + O(1).$$

The method presented in this paper however, does not explicitely compute the generators of the inverse of the quasiseparable matrix. But in some sense it calculates an upper triangular factorization of the following form (see Section 4)

$$AU = L\Lambda,$$

where U and L are respectively upper and lower triangular with ones on the diagonal, and Λ is a diagonal matrix. Moreover comparing both complexity counts, we see that our Levinson-like solver is approximately 2 times faster as the solver presented in [17].

We remark however that the algorithm proposed in [17], is also capable of dealing with block quasiseparable matrices, which our method in the current form is not capable of dealing with. Using the techniques however, which will be presented in the look-ahead section (Section 6), one can derive the block version of the Levinson-like solver as presented in this paper.

5.5. Band matrices. We will now briefly illustrate how we can solve band matrices using the above proposed method. There is a huge variety of methods for solving banded systems of equations: from QR-decompositions, LU-decompositions, Gaussian elimination to parallel methods. Some of these methods can be found in [10], and the references therein. Band matrices can also be considered as quasiseparable matrices, for example a (p_1, p_2) band matrix is also (p_1, p_2)-quasiseparable. But instead of using the quasiseparable approach, the direct approach gives a much faster algorithm: the quasiseparable approach involves the terms $p_1^2p_2$, $p_1p_2^2$ and p_1p_2 in the complexity count, whereas the Levinson-like approach only involves the term p_1p_2 in the complexity count (see Section 5.4, and the complexity count at the end of this section).

Assume we have an (p_1, p_2)-band matrix A of the following form:

[2]In the paper [17] another definition of a flop is used, than the one used throughout this paper. Therefore the operation count as presented in [17], was translated towards the definition of a flop in this paper.

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,p_1+1} & 0 & \dots & 0 \\ a_{2,1} & a_{2,2} & a_{2,3} & \dots & a_{2,p_1+2} & & \vdots \\ \vdots & a_{3,2} & a_{3,3} & & & \ddots & 0 \\ a_{p_2+1,1} & \vdots & & \ddots & & & a_{n-p_1,n} \\ 0 & a_{p_2+2,2} & & & & & \vdots \\ \vdots & & \ddots & & & \ddots & a_{n-1,n} \\ 0 & \dots & 0 & a_{n,n-p_2} & \dots & a_{n,n-1} & a_{n,n} \end{bmatrix},$$

with all the $a_{i,j} \in \mathbb{R}$. This is not the most compact representation of the band matrix A, as many of its elements are zero. Let us therefore introduce the matrix $\bar{A}$. (Similarly we can construct the matrix $\underline{A}$, for the lower triangular part.) which contains the elements in the same row, strictly Let us denote with $\bar{\mathbf{a}}_i$ the ith row out of the matrix $\bar{A}$, this means:

$$\begin{cases} \bar{\mathbf{a}}_i = [0, \dots, 0,\, a_{1,i}\,,\, a_{2,i}\,, \dots,\, a_{i-1,i}] & \text{if} \quad i \leq p_1 \\ \bar{\mathbf{a}}_i = [a_{i-p_1,i}\,,\, a_{i-p_1+1,i}\,, \dots,\, a_{i-1,i}] & \text{if} \quad i > p_1\,. \end{cases}$$

The row vectors $\bar{\mathbf{a}}_i$ are of length p_1 ($\bar{\mathbf{a}}_1 = 0$). It is shown now that the upper triangular part of the matrix satisfies the desired conditions, to be simple (p_1, p_2)-Levinson conform. (The lower triangular part is similar.) Let us define R_k as a $k \times p_1$ matrix of the following form:

$$R_k = \left[\frac{0}{I_{p_1}} \right],$$

where I_{p_1} denotes the identity matrix of size p_1. In the beginning of the algorithm, we have that $k \leq p_1$. In this case we take only the last k lines of I_{p_1}, this means that:

$$R_k = \begin{cases} [0, I_k] & \text{if} \quad k < p_1 \\ I_{p_1} & \text{if} \quad k = p_1 \\ \left[\frac{0}{I_{p_1}} \right] & \text{if} \quad k > p_1\,. \end{cases}$$

For every k the matrix P_k is defined as the shift operator P, which is an $p_1 \times p_1$ matrix of the following form

$$P = \begin{bmatrix} 0 & & & \dots & 0 \\ 1 & 0 & & \dots & 0 \\ 0 & 1 & 0 & & \vdots \\ \vdots & & \ddots & \ddots & \\ 0 & \dots & 0 & 1 & 0 \end{bmatrix}.$$

Multiplying a matrix on the right with this operator, will shift all the columns in this matrix one position to the left, and add a trailing zero column. Using this shift matrix and defining for every k the vector $\xi_k = [0, 0, \dots, 0, 1]$ gives us

$$R_{k+1} = \begin{bmatrix} R_k P \\ [0, \dots, 0, 1] \end{bmatrix}.$$

ETNA
Kent State University
etna@mcs.kent.edu

Using the above definitions and defining $\mathbf{c}_k = \bar{\mathbf{a}}_k$ (define S_k and $\mathbf{d}_k$ similarly as for the upper triangular part.) leads to the following relations:

$$A_{k+1} = \left[\begin{array}{c|c} A_k & R_k \bar{\mathbf{a}}_{k+1}^T \\ \hline \mathbf{d}_{k+1} S_k^T & a_{k+1,k+1} \end{array}\right].$$

This means that our band matrix is simple (p_1, p_2)-Levinson conform, leading to a solver of complexity $O(p_1 p_2 n)$. More precisely we know, as the multiplication on the right with the matrix P_k does not involve operations (just index shuffling), that the number of operations is bounded by

$$(6p_1p_2 + 7p_1 + 4p_2 + 1)(n-1) - 3p_1p_2 + 3p_1 - 2p_2 + 1,$$

as $\kappa_1 = \kappa_2 = \gamma_1 = \gamma_2 = 0$. This is clearly an upper bound, as the special structure of the matrices R_k is not taken into consideration. Exploiting this structure results in further complexity reduction.

5.6. Arrowhead matrices. Arrowhead matrices are often an essential tool for the computation of the eigenvalues, via divide and conquer approaches [5, 9]. They also arise, in block form, in domain decomposition methods, when discretizing partial differential equations where they are used as preconditioners. In this section we will show how arrowhead matrices can be solved efficiently using the presented framework, and we will present the algorithm based on Section 3.5. Let us consider the nonsymmetric arrowhead matrix of the following form:

$$A = \begin{bmatrix} a_1 & \bar{a}_2 & \bar{a}_3 & \dots & \bar{a}_n \\ \underline{a}_2 & a_2 & & & \\ \underline{a}_3 & & a_3 & & \\ \vdots & & & \ddots & \\ \underline{a}_n & & & & a_n \end{bmatrix},$$

where the elements of the form $\bar{a}_i$ denote the elements of the arrow in the upper triangular part, the elements $\underline{a}_i$ denote the elements of the arrow in the lower triangular part, the elements a_i denote the diagonal elements, with $\underline{a}_1 = a_1 = \bar{a}_1$, and the elements not shown are assumed to be zero.

Let us define the matrices R_k and S_k as $S_k^T = R_k^T = [1, 0, \dots, 0]$ which are vectors of length k. Let us define the elements $\mathbf{c}_k = \bar{a}_k$ and $\mathbf{d}_k = \underline{a}_k$, the matrices $P_k = Q_k$ are chosen equal to the identity matrix and the vectors $\xi_k = \eta_k = 0$. One can easily check that this matrix is simple $(1,1)$-Levinson conform.

Based on our solver of Section 3.5, we derive here the solver for the arrowhead matrix.

ALGORITHM 5.1. *Initialize*

$$\alpha_1 = \frac{-1}{a_1} \qquad \textit{Flops: } 1$$

$$\mu_1 = \frac{b_1}{a_1} \qquad \textit{Flops: } 1$$

$$S_1^T Y_1 = S_1^T \alpha_1 = \alpha_1 \quad \textit{Flops: } 0$$

$$S_1^T x_1 = S_1^T \mu_1 = \mu_1 \quad \textit{Flops: } 0$$

For $k = 1, \dots, n-1$ *do:*

ETNA
Kent State University
etna@mcs.kent.edu

1. *Compute and store the following variable:*
   (a) $\underline{a}_{k+1} S_k^T Y_k$ *Flops:* 1
   (b) $(\underline{a}_{k+1} S_k^T Y_k \bar{a}_{k+1} + a_{k+1,k+1})$ *Flops:* 2
2. $\alpha_{k+1} = -\frac{\underline{a}_{k+1} S_k^T Y_k}{\underline{a}_{k+1} S_k^T Y_k \bar{a}_{k+1} + a_{k+1,k+1}}$ *Flops:* 1
3. $\mu_{k+1} = \frac{b_{k+1} - \underline{a}_{k+1} S_k^T x_k}{\underline{a}_{k+1} S_k^T Y_k \bar{a}_{k+1} + a_{k+1,k+1}}$ *Flops:* 3
4. *Compute and store the following variables (if $k < n-1$):*
   (a) $S_k^T Y_k \bar{a}_{k+1}$ *Flops:* 1
   (b) $S_{k+1}^T Y_{k+1} = S_k^T Y_k + \left(S_k^T Y_k \bar{a}_{k+1}\right) \alpha_{k+1}$ *Flops:* 2
   (c) $S_{k+1}^T \mathbf{x}_{k+1} = S_k^T \mathbf{x}_k + \left(S_k^T Y_k \bar{a}_{k+1}\right) \mu_{k+1}$ *Flops:* 2

endfor;

Computation of the solution vector (h is a dummy variable)

$$x_n = \mu_n$$
$$h = \mu_n \bar{a} \qquad \textit{Flops: } 1$$

For $k = n-1, -1, 1$ do
1. $x_k = \mu_k + \alpha_k h$ *Flops:* 2
2. $h = x_i \bar{a}_i + h$ *Flops:* 2

endfor.

This algorithm has as largest term in its complexity $19n$. If one however uses standard Gaussian elimination to solve this system (after having flipped the arrow, so that it points downwards), the complexity is $6n$.

5.7. Unsymmetric structures. The examples presented above were matrices having a symmetric structure. This means that if the upper part was semiseparable also the lower triangular part was semiseparable, if the upper triangular part was from a band matrix, also the lower triangular part was from a band matrix. But in fact, taking a closer look at the conditions, the upper and/or the lower triangular part of the matrix need not to be related in any way.

Dealing with the upper and the lower triangular part of the matrix separately we can create matrices having a nonsymmetric structure:

- An upper triangular semiseparable matrix is simple $(p_1, 0)$-Levinson conform, where p_1 stands for the semiseparability rank of the upper triangular part.
- An upper triangular band matrix is simple $(p_1, 0)$-Levinson conform, where p_1 stands for the bandwidth in the upper triangular part.
- Different matrices from statistical applications, which have different semiseparability ranks in the upper and the lower triangular part plus a band matrix can be found in [22]. These matrices fit naturally in this Levinson framework.
- A matrix for which the upper triangular part is semiseparable, and the lower triangular part is coming from a band matrix. For example a unitary Hessenberg matrix, this matrix has the upper triangular part of semiseparable form and only one subdiagonal different from zero. Hence a unitary Hessenberg matrix is simple $(1, 1)$-Levinson conform.
- A matrix which has a band structure in the upper triangular part and, for which the lower triangular part comes from an arrowhead is $(p_1, 1)$-Levinson conform (p_1 denotes the bandwidth).
- Moreover, one can also combine matrices, for which the upper or the lower triangular part is quasiseparable. In this case the complexity of the Levinson-like solver

ETNA
Kent State University
etna@mcs.kent.edu

changes according to the complexity of the multiplication with the matrices P_k and Q_k; see Section 3.6 for a more detailed analysis of the complexity.

In the following sections, some interesting classes of matrices having an unsymmetric structure are investigated in more detail.

5.8. Upper triangular matrices. Let us apply our Levinson solver to an upper triangular system of equations. We have the matrix $A = (a_{ij})$ where $a_{ij} = 0$ if $i > j$. Let us denote with R_k the matrix of dimension $k \times (n-1)$ where $R_k = [I_k, 0]$ and $\mathbf{c}_k = [a_{1,k}, \dots, a_{k-1,k}, 0, \dots, 0]$ is a row vector of length $1 \times (n-1)$. Moreover we assume the matrix $S_k = 0$ and $P = I$. The matrix is simple $(n-1,0)$-Levinson conform as:

$$A_{k+1} = \left[\begin{array}{c|c} A_k & R_k \mathbf{c}_{k+1}^T \\ \hline 0 & a_{k+1,k+1} \end{array}\right].$$

We know that solving the system of equations in this manner will lead to an $O(n^2)$ method. Moreover, this is a well-known method, as we will show.

Let us take a closer look at the solution generated by this Levinson-like approach. We have all the necessary information to easily caculate the values of α_k and μ_k for every k, as $S_k = 0$. We have that for every k

$$\alpha_k = \frac{e_k^{(n)}}{a_{k,k}}$$
$$\mu_k = \frac{b_k}{a_{k,k}},$$

where $e_k^{(n)}$ is the kth vector of the canonical basis of $\mathbb{R}^n$. Using these variables, we can construct the solution vector $\mathbf{x}$ for the system $A\mathbf{x} = \mathbf{b}$. We will consider the computation of the last three components of $\mathbf{x}$, based on (3.10). The last component x_n has the following form:

$$x_n = \mu_n = \frac{b_n}{a_{n,n}}.$$

The component x_{n-1} is of the following form:

$$\begin{aligned} x_{n-1} &= \frac{b_{n-1}}{a_{n-1,n-1}} + \frac{(-1)}{a_{n-1,n-1}} \frac{b_n}{a_{n,n}} e_{n-1}^{(n)} \mathbf{c}_n^T \\ &= \frac{-1}{a_{n-1,n-1}} \left(b_{n-1} - a_{n-1,n} x_n \right). \end{aligned}$$

Let us conclude with element x_{n-2}, which gives us the following equations:

$$\begin{aligned} x_{n-2} &= \frac{b_{n-2}}{a_{n-2,n-2}} + \frac{e_{n-2}^{(n)}}{a_{n-2,n-2}} \left(\frac{b_{n-1}}{a_{n-1,n-1}} \mathbf{c}_{n-1}^T + (I + \mathbf{c}_{n-1}^T \frac{-1}{a_{n-1,n-1}} e_{n-1}^{(n)}) \frac{b_n}{a_{n,n}} \mathbf{c}_n^T \right) \\ &= \frac{1}{a_{n-2,n-2}} \left(b_{n-2} - a_{n-2,n-1} \frac{b_{n-1}}{a_{n-1,n-1}} - a_{n-2,n} x_n + \frac{a_{n-2,n-1} a_{n-1,n}}{a_{n-1,n-1}} x_n \right) \\ &= \frac{1}{a_{n-2,n-2}} \left(b_{n-2} - a_{n-2,n-1} x_{n-1} - a_{n-2,n} x_n \right). \end{aligned}$$

This means, that rewriting the general formulas for the Levinson-like solver for upper triangular systems of equations, gives us the well-known backward substition algorithm [21]. In a similar way we can derive the solution method for a lower triangular system of equations. This will give us the forward substitution algorithm.

ETNA
Kent State University
etna@mcs.kent.edu

5.9. Dense matrices. Using Lemma 2.3 we know that also strongly nonsingular systems of equations without structure in the coefficient matrix, can be solved in this way. This gives us an algorithm, requiring $O(n^3)$ operations, more precisely $6n^3$, which is of course not efficient enough.

5.10. Summations of Levinson-conform matrices. If we summate different Levinson-conform matrices, we get again a Levinson-conform matrix. This is proved in the next theorem.

THEOREM 5.2. *Suppose we have two matrices $\hat{A}$ and $\tilde{A}$, which are respectively $(\hat{p}_1, \hat{p}_2)$ and $(\tilde{p}_1, \tilde{p}_2)$-Levinson conform. Then the matrix $A = \hat{A} + \tilde{A}$ will be $(\hat{p}_1 + \tilde{p}_1, \hat{p}_2 + \tilde{p}_2)$-Levinson conform.*

Proof. Let us denote all the matrices related to the matrix $\hat{A}$, with a hat and the ones related to the matrix $\tilde{A}$ with a tilde. Let us define the matrices $R_k, c_k, d_k, S_k, \xi_k$ and η_k as follows:

$$\begin{aligned} R_k &= \left[\hat{R}_k, \tilde{R}_k\right], S_k = \left[\hat{S}_k, \tilde{S}_k\right] \\ \mathbf{c}_k &= [\hat{\mathbf{c}}_k, \tilde{\mathbf{c}}_k], \mathbf{d}_k = \left[\hat{\mathbf{d}}_k, \tilde{\mathbf{d}}_k\right] \\ \xi_k &= \left[\hat{\xi}_k, \tilde{\xi}_k\right], \eta_k = [\hat{\eta}_k, \tilde{\eta}_k]\,. \end{aligned}$$

Define the operators P_k and Q_k as

$$P_k = \begin{bmatrix} \hat{P}_k & \\ & \tilde{P}_k \end{bmatrix} \text{ and } Q_k = \begin{bmatrix} \hat{Q}_k & \\ & \tilde{Q}_k \end{bmatrix}.$$

Then it is straightforward to prove that these newly defined matrices and vectors, satisfy the desired conditions, such that the matrix A is $(\hat{p}_1 + \tilde{p}_1, \hat{p}_2 + \tilde{p}_2)$-Levinson conform. □

We remark that if we summate two simple Levinson conform matrix, the resulting matrix will also be simple Levinson conform.

Let us illustrate this with some possible structures which become solvable now:

- One can solve now summations of all previously defined Levinson-conform matrices. For example, the sum of a higher order semiseparable matrix plus a band matrix.
- Moreover it is not necessary that both matrices are strongly nonsingular. As long as the sum of these matrices is strongly nonsingular, the problem can be solved by the standard Levinson-like solver (for the look-ahead method, see Section 6). In this way, we can also add simple Levinson conform matrices which are singular. For example adding a rank 1 matrix to a Levinson conform matrix is feasible.
- For example an arrowhead matrix with a larger band-width is $(p_1 + 1, p_2 + 1)$-Levinson conform, as it can be written as the sum of an arrowhead matrix and an (p_1, p_2)-band matrix.

In the next sections, we will give some more examples of matrices, which can easily be seen as the sum of simple Levinson conform matrices.

5.11. Matrices with errors in structures. Quite often one will deal with matrices, which do not have a perfect structure. For example a matrix, which is semiseparable, except for some elements in one row or column. Or a matrix, which is almost of band form except for some elements which are nonzero. The number of elements desintegrating the structure is often low. If we are able to write now this matrix as the sum of the pure structure (e.g. semiseparable plus band) and the elements destroying the structure, we can decrease the complexity count of the Levinson method related to this matrix. Let us call these matrices destroying the structure, error matrices. If these error matrices are simple (e_1, e_2)-Levinson conform, the complete matrix will be simple $(p_1 + e_1, p_2 + e_2)$-Levinson conform. In case of small e_1 and e_2 this does not lead to a large increase in complexity. Let us illustrate this with some examples of errors in the upper triangular part. (The lower triangular part, can be dealt with in a similar way.)

- The error matrix E has only one column different from zero. Suppose column i: $E_i = [e_1, e_2, \ldots, e_n]^T$ contains the only nonzero elements in the matrix E. If we define $R_1 = e_1$, $R_{k+1}^T = [R_k^T, e_{k+1}]$ and all the $\mathbf{c}_k = 0$, except $\mathbf{c}_{i+1} = 1$, this gives us the structure for the upper triangular part. Defining the lower triangular part similarly gives us a simple $(1,1)$-Levinson conform matrix. Similarly one can consider error matrices with more columns different from zero, or error matrices with one or more rows different from zero. For example a matrix which is of unitary Hessenberg form, except for the elements in the last column. These elements do not belong to the semiseparable structure of the upper triangular part. This matrix can be written as the sum of a unitary Hessenberg matrix plus an error matrix, which has only one column different from zero.
- The error matrix has a super diagonal different from zero. Similarly to the band matrix approach, we can prove that this matrix is simple $(0,1)$-Levinson conform. Multiple super diagonals and/or subdiagonals, can also be considered.
- If the error matrix is unstructured, but contains few elements, one might be able to represent it as a simple (e_1, e_2)-Levinson conform matrix with small e_1 and e_2, but this is of course case dependent.

In the next sections some examples will be given of specific matrices which can be written as the sum of a pure structure plus an error matrix.

5.12. Companion matrices. Companion matrices are often used for computing the zeros of polynomials; see [4]. The companion matrix itself is not suitable for applying this Levinson algorithm, as all the leading principal matrices, except possibly the matrix itself, are singular. We can add this matrix however easily to other Levinson-conform matrices as it is simple $(1,1)$-Levinson conform. Let us consider the companion matrix C corresponding to the polynomial $p(x) = x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$:

$$C = \begin{bmatrix} 0 & 0 & 0 & \cdots & -a_0 \\ 1 & 0 & 0 & \cdots & -a_1 \\ 0 & 1 & 0 & \cdots & -a_2 \\ \vdots & & \ddots & & \vdots \\ & & & 1 & -a_{n-1} \end{bmatrix}.$$

This matrix is clearly simple $(1,1)$-Levinson conform.

5.13. Comrade matrix. Lots of polynomial bases satisfy a three terms recurrence relation. If we would like to compute the roots of a polynomial expressed in such a basis, we can use the comrade matrix [3].

When we have a set of polynomials defined in the following sense:

$$p_i(x) = \sum_{j=0}^{i} p_{ij} x^j, \; i = 0, 1, 2, 3, \ldots$$

ETNA
Kent State University
etna@mcs.kent.edu

which satisfy the following relationships (in fact a three terms recurrence):

$$\begin{aligned}
&p_0(x) = 1, \\
&p_1(x) = \alpha_1 x + \beta_1, \\
&p_i(x) = (\alpha_i x + \beta_i)p_{i-1}(x) - \gamma_i p_{i-2}(x) \text{ for } i \geq 2,
\end{aligned}$$

and suppose we have the following polynomial:

$$a(x) = p_n(x) + a_1 p_{n-1}(x) + \ldots + a_n p_0(x),$$

then the comrade matrix is defined as the matrix C:

$$C = \begin{bmatrix}
\frac{-\beta_1}{\alpha_1} & \frac{1}{\alpha_1} & 0 & \ldots & & & 0 \\
\frac{\gamma_2}{\alpha_2} & \frac{-\beta_2}{\alpha_2} & \frac{1}{\alpha_2} & 0 & \ldots & & 0 \\
0 & \frac{\gamma_3}{\alpha_3} & \frac{-\beta_3}{\alpha_3} & \frac{1}{\alpha_3} & 0 & \ldots & \vdots \\
\vdots & \ddots & \ddots & \ddots & \ddots & & \\
 & & & & \frac{\gamma_{n-1}}{\alpha_{n-1}} & \frac{-\beta_{n-1}}{\alpha_{n-1}} & \frac{1}{\alpha_{n-1}} \\
\frac{-a_n}{\alpha_n} & \frac{-a_{n-1}}{\alpha_n} & \ldots & & \frac{-a_3}{\alpha_n} & \frac{-a_2+\gamma_n}{\alpha_n} & \frac{-a_1-\beta_n}{\alpha_n}
\end{bmatrix}.$$

It is clear that this comrade matrix can be written as the sum of a tridiagonal matrix plus an error matrix, for which one row is different from zero. Hence, the matrix is simple $(1,2)$-Levinson conform.

5.14. Fellow matrices. Fellow matrices are rank 1 perturbations of unitary Hessenberg matrices; see [6]. Finding the roots of a polynomial expressed as a linear combination of Szegö polynomials is related to the eigenvalues of a fellow matrix; see [1, 2]. These matrices are naturally written as the sum of two simple Levinson conform matrices. Suppose F to be a fellow matrix, than we can write $F = H + uv^T$, where H is a unitary Hessenberg matrix, which is simple $(1,1)$-Levinson conform, and the rank one matrix uv^T is also simple $(1,1)$-Levinson conform. A fellow matrix is therefore simple $(2,2)$-Levinson conform.

6. The look-ahead procedure. A limitation of the presented solver is the strongly nonsingularity assumption. In this section we will briefly illustrate how we can overcome this problem by applying a look-ahead technique. Once the main decomposition of the involved matrices is known, it is an easy exercise to derive the complete algorithm. Hence we do not include all the details, we only illustrate it for the Yule-Walker-like equation.

Given a simple p_1-Levinson conform matrix A. Suppose that, after having solved the kth order Yule-Walker-like equation, we encounter some singular or almost singular principal leading matrices A_{k+1} up to A_{k+l-1}. This means that the first nonsingular principal leading matrix is A_{k+l}. We will now solve this $(k+l)$th Yule-Walker-like system, based on the solution of the kth Yule-Walker-like problem:

$$AY_k = -R_k.$$

The coefficient matrix A_{k+l}, can be decomposed in block form as

$$A_{k+l} = \left[\begin{array}{c|c} A_k & R_k C_{k,l}^T \\ \hline D_{k,l} S_k^T & B_{k,l} \end{array}\right],$$

ETNA
Kent State University
etna@mcs.kent.edu

where $B_{k,l} = A(k+1:l, k+1:l)$[3] is an $l \times l$ matrix and $C_{k,l}$ is an $l \times p_1$ matrix of the following form:

$$C_{k,l}^T = \left[\mathbf{c}_{k+1}^T, P_k \mathbf{c}_{k+2}^T, \ldots, P_k P_{k+1} \cdots P_{k+l-2} \mathbf{c}_{k+l}^T\right].$$

Before we can solve the system $A_{k+l} Y_{k+l} = -R_{k+l}$, we also need to block-decompose the matrix R_{k+l}:

$$R_{k+l} = \left[\begin{array}{c} R_k P_k \cdots P_{k+l-1} \\ \hline \Sigma_{k,l} \end{array}\right]$$

with $\Sigma_{k,l} \in \mathbb{R}^{l \times p_1}$ of the form:

$$\Sigma_{k,l} = \left[\begin{array}{r} \xi_{k+1} P_{k+1} P_{k+2} P_{k+3} \cdots P_{k+l-1} \\ \xi_{k+2} P_{k+2} P_{k+3} \cdots P_{k+l-1} \\ \xi_{k+3} P_{k+3} \cdots P_{k+l-1} \\ \vdots \\ \xi_{k+l} \end{array}\right].$$

Using these block-decompositions of the matrices A_{k+l} and R_{k+l}, the $(k+l)$th Yule-Walker-like equation can be written as

$$\left[\begin{array}{c|c} A_k & R_k C_{k,l}^T \\ \hline D_{k,l}^T S_k^T & B_{k,l} \end{array}\right] \left[\begin{array}{c} Z_{k,l} \\ \hline \alpha_{k,l} \end{array}\right] = \left[\begin{array}{c} R_k P_k \cdots P_{k+l-1} \\ \hline \Sigma_{k,l} \end{array}\right],$$

with $Z_{k,l} \in \mathbb{R}^{k \times p_1}$ and $\alpha_{k,l} \in \mathbb{R}^{l \times p_1}$.

Expanding the above equations and solving them towards $Z_{k,l}$ and $\alpha_{k,l}$ leads to the following formulas:

$$\begin{aligned} Z_{k,l} &= Y_k \left(P_k P_{k+1} \cdots P_{k+l-1} + C_{k,l}^T \alpha_{k,l}\right), \\ \alpha_{k,l} &= -\left(D_{k,l} S_k^T Y_k C_{k,l}^T + B_{k,l}\right)^{-1} \left(\Sigma_{k,l} + D_{k,l} S_k^T Y_k P_k \cdots P_{k+l-1}\right). \end{aligned}$$

This means that for computing $\alpha_{k,l}$, we need to solve p_1 systems of size $l \times l$, this can be done by several techniques, at a cost of $O(l^3)$. As long as the value of l, w.r.t. the problem size is small, there is no significant increase in complexity, but this is of course closely related to the problem.

Having computed the solution of the Yule-Walker-like equation, the solving of the corresponding Levinson problem is done similarly. Also the complexity reducing remarks as presented in Section 3.4, can be translated towards this block-version.

An important issue, which we will not address here, is to decide when a principal leading matrix is numerically singular. Or even more, when is it too ill conditioned, such that it might have a large influence on the accuracy of the final result. An easy way to measure the ill-conditioning is to check the value of the denominator in the computation of α_k; as long as this value is not too close to zero (w.r.t. the nominator), the computations are numerically sound. A more detailed analysis should however be done case by case for obtaining a fast and accurate practical implementation.

[3] With the matrix $A(i:j, k:l)$, we denote the submatrix of the matrix A, with rows ranging from i up to j, and with columns ranging from k up to l.

7. Conclusions. In this paper we developed an algorithm for solving simple (p_1, p_2)-Levinson conform matrices. The algorithm was linear in time, w.r.t. the size of the matrix, multiplied with p_1 and p_2. It was shown that different classes of matrices, including semiseparable, quasiseparable, band, ... fit in this framework. We also investigated the relation with an upper triangular factorization, and we pointed out how to design a look-ahead method for this type of algorithm.

REFERENCES

[1] G. S. AMMAR, D. CALVETTI, W. B. GRAGG, AND L. REICHEL, *Polynomial zerofinders based on Szegö polynomials*, J. Comput. Appl. Math., 127 (2001), pp. 1–16.
[2] G. S. AMMAR, D. CALVETTI, AND L. REICHEL, *Continuation methods for the computation of zeros of Szegö polynomials*, Linear Algebra Appl., 249 (1996), pp. 125–155.
[3] S. BARNETT, *Polynomials and Linear Control Systems*, Marcel Dekker Inc, 1983.
[4] D. A. BINI AND V. Y. PAN, *Polynomial and Matrix Computations, vol. 1: Fundamental Algorithms*, Birkhäuser, Boston, 1994.
[5] C. F. BORGES AND W. B. GRAGG, *A parallel divide and conquer algorithm for the generalized real symmetric definite tridiagonal eigenproblem*, in Numerical Linear Algebra and Scientific Computing, L. Reichel, A. Ruttan, and R. S. Varga, eds., de Gruyter, Berlin, 1993, pp. 11–29.
[6] D. CALVETTI, S. KIM, AND L. REICHEL, *The restarted QR-algorithm for eigenvalue computation of structured matrices*, J. Comput. Appl. Math., 149 (2002), pp. 415–422.
[7] T. F. CHAN AND P. C. HANSEN, *A look-ahead Levinson algorithm for indefinite Toeplitz systems*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 490–506.
[8] S. CHANDRASEKARAN AND M. GU, *Fast and stable algorithms for banded plus semiseparable systems of linear equations*, SIAM J. Matrix Anal. Appl., 25 (2003), pp. 373–384.
[9] J. J. M. CUPPEN, *A divide and conquer method for the symmetric tridiagonal eigenproblem*, Numer. Math., 36 (1981), pp. 177–195.
[10] A. M. CUYT AND B. VERDONK, *Different techniques for the construction of multivariate rational interpolants and padé approximants*, Universitaire instelling Antwerpen, 1988.
[11] J. W. DEMMEL, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, 1997.
[12] P. DEWILDE AND A.-J. VAN DER VEEN, *Time-varying systems and computations*, Kluwer Academic Publishers, Boston, June 1998.
[13] Y. EIDELMAN, *Fast recursive algorithm for a class of structured matrices*, Appl. Math. Lett., 13 (2000), pp. 57–62.
[14] Y. EIDELMAN AND I. C. GOHBERG, *Inversion formulas and linear complexity algorithm for diagonal plus semiseparable matrices*, Comput. Math. Appl., 33 (1996), pp. 69–79.
[15] ———, *A look-ahead block Schur algorithm for diagonal plus semiseparable matrices*, Comput. Math. Appl., 35 (1997), pp. 25–34.
[16] ———, *On a new class of structured matrices*, Integral Equations Operator Theory, 34 (1999), pp. 293–324.
[17] ———, *Fast inversion algorithms for a class of block structured matrices*, Contemp. Math., 281 (2001), pp. 17–38.
[18] ———, *A modification of the Dewilde-van der Veen method for inversion of finite structured matrices*, Linear Algebra Appl., 343-344 (2002), pp. 419–450.
[19] F. R. GANTMACHER AND M. G. KREĬN, *Oscillation Matrices and Kernels and Small Vibrations of Mechanical Systems*, AMS Chelsea Publishing, 2002.
[20] I. C. GOHBERG, T. KAILATH, AND I. KOLTRACHT, *Efficient solution of linear systems of equations with recursive structure*, Linear Algebra Appl., 80 (1986), pp. 81–113.
[21] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, Third ed., The Johns Hopkins University Press, Baltimore, 1996.
[22] F. A. GRAYBILL, *Matrices with Applications in Statistics*, Wadsworth International Group, Belmont, California, 1983.
[23] L. GREENGARD AND V. ROKHLIN, *On the numerical solution of two-point boundary value problems*, Comm. Pure Appl. Math., 44 (1991), pp. 419–452.
[24] T. KAILATH AND A. H. SAYED, eds., *Fast Reliable Algorithms for Matrices with Structure*, SIAM, Philadelphia, 1999.
[25] N. LEVINSON, *The Wiener RMS error criterion in filter desing and prediction*, J. Math. Phys., 25 (1947), pp. 261–278.
[26] N. MASTRONARDI, S. CHANDRASEKARAN, AND S. VAN HUFFEL, *Fast and stable two-way algorithm for diagonal plus semi-separable systems of linear equations*, Numer. Linear Algebra Appl., 8 (2001), pp. 7–12.

[27] N. Mastronardi, M. Van Barel, and R. Vandebril, *A Levinson-like algorithm for symmetric positive definite semiseparable plus diagonal matrices*, Tech. Rep. TW423, Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3000 Leuven (Heverlee), Belgium, Mar. 2005.

[28] H. P. J. Starr, *On the numerical solution of one-dimensional integral and differential equations*, PhD thesis, Yale University, 1992. Research Report YALEU/DCS/RR-888.

[29] L. N. Trefethen and D. Bau, *Numerical Linear Algebra*, SIAM, Philadelphia, 1997.

[30] E. Van Camp, N. Mastronardi, and M. Van Barel, *Two fast algorithms for solving diagonal-plus-semiseparable linear systems*, J. Comput. Appl. Math., 164-165 (2004), pp. 731–747.

[31] R. Vandebril, N. Mastronardi, and M. Van Barel, *A Levinson-like algorithm for symmetric strongly nonsingular higher order semiseparable plus band matrices*, J. Comput. Appl. Math., 198 (2007), pp. 75–97.

[32] R. Vandebril, M. Van Barel, and N. Mastronardi, *A note on the representation and definition of semiseparable matrices*, Numer. Linear Algebra Appl., 12 (2005), pp. 839–858.