

**A Levinson-like algorithm for
symmetric strongly nonsingular higher
order semiseparable plus band matrices**

Raf Vandebril

Nicola Mastronardi

Marc Van Barel

Report TW 427, April 2005



Katholieke Universiteit Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

A Levinson-like algorithm for symmetric strongly nonsingular higher order semiseparable plus band matrices

Raf Vandebril

Nicola Mastronardi

Marc Van Barel

Report TW 427, April 2005

Department of Computer Science, K.U.Leuven

Abstract

In this paper we will derive a solver for a symmetric strongly nonsingular higher order generator representable semiseparable plus band matrix. The solver we will derive is based on the Levinson algorithm, which is used for solving strongly nonsingular Toeplitz systems.

In a first part an $O(p^2n)$ solver for a semiseparable matrix of semiseparability rank p is derived, and in a second part we derive an $O(l^2n)$ solver for a band matrix with bandwidth $2l + 1$. Both solvers are constructed in a similar way: firstly a Yule-Walker-like equation needs to be solved, and secondly this solution is used for solving a linear equation with an arbitrary right-hand side.

Finally a combination of the above methods is presented to solve linear systems with semiseparable plus band coefficient matrices. The overall complexity of this solver is $6(l + p)^2n$ plus lower order terms.

In a final section numerical experiments are performed. Attention is paid to the timing and the accuracy of the described methods.

Keywords : Yule-Walker, Levinson, higher order generator representable semiseparable matrices, band matrices, symmetric strongly nonsingular, upper triangular factorisation of the inverse

AMS(MOS) Classification : Primary : 65F05

A Levinson-like algorithm for symmetric strongly nonsingular higher order semiseparable plus band matrices *

Raf Vandebril, Nicola Mastronardi, Marc Van Barel

26th April 2005

Abstract

In this paper we will derive a solver for a symmetric strongly nonsingular higher order generator representable semiseparable plus band matrix. The solver we will derive is based on the Levinson algorithm, which is used for solving strongly nonsingular Toeplitz systems.

In a first part an $O(p^2n)$ solver for a semiseparable matrix of semiseparability rank p is derived, and in a second part we derive an $O(l^2n)$ solver for a band matrix with bandwidth $2l + 1$. Both solvers are constructed in a similar way: firstly a Yule-Walker-like equation needs to be solved, and secondly this solution is used for solving a linear equation with an arbitrary right-hand side.

Finally a combination of the above methods is presented to solve linear systems with semiseparable plus band coefficient matrices. The overall complexity of this solver is $6(l + p)^2n$ plus lower order terms.

In a final section numerical experiments are performed. Attention is paid to the timing and the accuracy of the described methods.

Keywords: Yule-Walker, Levinson, higher order generator representable semiseparable matrices, band matrices, symmetric strongly nonsingular, upper triangular factorisation of the inverse

1 Introduction

Currently there is a growing interest in semiseparable matrices. Attention is paid to different theoretical or algorithmic descriptions related to these matrices, for example eigenvalue and singular value algorithms [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] relations with orthogonal rational functions [11], reduction algorithms related to this matrices [12, 13, 14], and many more. A more elaborate overview of publications related to semiseparable matrices can be found in [1, 15].

Also a lot of attention is paid to system solving, where the coefficient matrix is of semiseparable (plus a diagonal) form. Linear systems of equations of semiseparable (plus diagonal) form arise for example in the solution of differential and integral equations [16, 17], in oscillation theory [18], statistics [19], and so on. In the following publications, algorithms are proposed for solving semiseparable systems of equations [20, 21, 22, 23, 24, 25, 26]. The publications [21, 24] are based on the QR -decomposition for more general classes of matrices. In the papers [20, 26, 27] several methods based on the QR (or URV)-decomposition of these matrices are proposed.

In a recent paper [28], we proposed a method based on the Levinson idea for solving strongly nonsingular semiseparable plus diagonal matrices. The presented solver used only $19n - 17$ operations. The method was however only suitable for semiseparable matrices of semiseparability rank 1, i.e. their lower

*The work of the first and third author was partially supported by the Fund for Scientific Research–Flanders (Belgium), 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 IUAP V-22 (Dynamical Systems and Control: Computation, Identification & Modelling). This work was partially supported by MIUR, grant number 2004015437 (second author). The scientific responsibility rests with the authors

triangular part and their upper triangular part are coming from a rank one matrix. Therefore we present a more general solver in this paper, based on the same idea as in [28]. In fact we will present three solvers in this paper. A first solver will solve higher order strongly nonsingular semiseparable systems of equations. If the semiseparability rank of the considered matrix equals p , the solver will have an overall complexity of $6p^2n$ operations plus lower order terms. Secondly we present a solver based on a similar idea for solving strongly nonsingular band systems of equations. If the bandwidth is $2l + 1$, the solver has a complexity of $6l^2n$ plus lower order terms. Finally we will combine both of these algorithms and derive a solver for higher order semiseparable plus band matrices, having as complexity $6(l + p)^2n$ plus lower order terms.

The proposed method is based on the Durbin and Levinson algorithm for solving Toeplitz systems of equations. The Levinson algorithm for Toeplitz matrices is widespread and described for example in [29, 30, 31]. We will not reconsider the Levinson algorithm for Toeplitz matrices in this paper.

The paper is organised as follows. In the following section we will briefly introduce some notation related to higher order semiseparable matrices and band matrices. This notation will be used throughout the remainder of the paper. In Section 3 we will derive the solver for the higher order semiseparable matrices. This section is split up into 4 parts. A first part constructs a solution method for the Yule-Walker part of these equations. An implementation of this solution method is given in the second part, combined with the operation count for this algorithm. In a third part we derive the solver for the overall problem with an arbitrary right-hand side. An implementation together with the operation count is given in the fourth part. In Section 4, the solver for the band matrices is derived. This section is split up in a similar way as Section 3. In Section 5, both methods are combined to tackle the general problem. The relation between the presented solvers and the upper triangular factorisation of the inverse are shown in Section 6. In a last section, numerical experiments are given. Two types of experiments are performed. In the first type of experiments we perform timings for the different solvers for various sizes, and varying thereby also the semiseparability rank and the bandwidth. A second type of experiments checks the accuracy of the proposed methods. All the proposed methods in this paper are freely available and can be downloaded from <http://www.cs.kuleuven.ac.be/~marc/>.

2 Some notation

As already mentioned in the abstract, the general solver for the semiseparable plus band system is based on the solvers for band and for semiseparable matrices separately. Before starting the construction of the Levinson solvers for these systems, we will introduce briefly some notation which will be used throughout the remainder of the paper.

2.1 Higher order semiseparable matrices

Let us define first what is meant with a higher order generator representable semiseparable matrix. Higher order generator representable matrices are characterised by the fact that the part below (and including) the diagonal is coming from a certain rank k matrix, the same statement holds for the upper triangular part including the diagonal. This means in fact that the higher order semiseparable matrix can be written as the sum of generator representable semiseparable matrices of rank 1. In the remainder of the manuscript, for simplicity, we will talk about semiseparable matrices, but in fact generator representable semiseparable matrices are discussed. A more elaborate study on the difference between generator representable semiseparable and semiseparable matrices can be found in [32]. Let us assume the rank of the higher order semiseparable matrix to be equal to p , this is briefly called the semiseparability rank. For the semiseparable matrix S of semiseparability rank p and of dimension $n \times n$, we get

$$S = S_1 + S_2 + S_3 + \dots + S_p, \quad (1)$$

where all the matrices S_j are semiseparable matrices of rank 1, i.e. they are of the following form: (only the lower triangular part of the matrices is shown, as we consider symmetric matrices).

$$S_j = \begin{bmatrix} u_{1j}v_{1j} & & & & \\ u_{2j}v_{1j} & u_{2j}v_{2j} & & & \\ \vdots & & \ddots & & \\ u_{nj}v_{1j} & u_{nj}v_{2j} & \dots & u_{nj}v_{nj} & \end{bmatrix}.$$

Let us define the following two matrices $U = (u_{ij})$ and $V = (v_{ij})$, with $i = 1, \dots, n$ and $j = 1, \dots, p$ and let us use the following notation

$$U = \begin{bmatrix} \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_n \end{bmatrix} \text{ and } V = \begin{bmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_n \end{bmatrix}, \quad (2)$$

where for $i = 1, \dots, n$ $\mathbf{v}_i = [v_{i1}, \dots, v_{ip}]$ and $\mathbf{u}_i = [u_{i1}, \dots, u_{ip}]$ denote the i th row out of U and V respectively. Using this new notation, one can easily rewrite the matrix S of Equation (1) as

$$S = \begin{bmatrix} \mathbf{u}_1 \mathbf{v}_1^T & & & & \\ \mathbf{u}_2 \mathbf{v}_1^T & \mathbf{u}_2 \mathbf{v}_2^T & & & \\ \vdots & & \ddots & & \\ \mathbf{u}_n \mathbf{v}_1^T & \mathbf{u}_n \mathbf{v}_2^T & \dots & \mathbf{u}_n \mathbf{v}_n^T & \end{bmatrix}.$$

This gives us a similar structure as for the semiseparable matrices of semiseparability rank 1. Similarly as in the semiseparability rank 1 case we will call the matrices U and V the generator matrices of the higher order generator representable semiseparable matrix S .

2.2 Band matrices

Even though most of the readers are already familiar with band matrices, we would briefly like to introduce some notations related to band matrices, which we will use in this paper.

Let us consider a band matrix B with bandwidth $2l + 1$, where $l \geq 1$. This means that our matrix B has the following form (only consider the lower triangular part, because the considered matrices are symmetric):

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

with all the $a_{i,j} \in \mathbb{R}$.

This is of course not the most compact representation of the band matrix B , as many of its elements are zero. Let us introduce the matrix A , which contains as columns the elements on the same super or sub-diagonal of the matrix B :

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

Let us denote with \mathbf{a}_i the i th row out of the matrix A , this means:

$$\begin{cases} \mathbf{a}_i = [0, \dots, 0, a_{i,1}, \dots, a_{i,i}] & \text{if } i \leq l \\ \mathbf{a}_i = [a_{i,i-l}, \dots, a_{i,i}] & \text{if } i > l \end{cases}$$

The row vectors \mathbf{a}_i are of length $l + 1$.

This is a more compact notation, which we will use throughout the paper, and which is also used in the effective implementation.

3 A Levinson-like solver for higher order semiseparable matrices

In this section we will solve a higher order semiseparable system of equations based on the Levinson idea. Firstly we will solve in fact k systems of Yule-Walker-like equations. The solution of this system will afterwards be used for solving a system with an arbitrary right-hand side.

3.1 A Yule-Walker-like solver for higher order semiseparable matrices

Let us consider a higher order semiseparable matrix S of rank p with generators U and V , as in Equation (2). The Yule-Walker-like problems we would like to solve are the following ones:

$$S_k Y_k = -R_k, \quad (3)$$

where S_k denotes the upper left $k \times k$ sub-block of the semiseparable matrix S . The right-hand side R_k is a $k \times p$ matrix of the following form:

$$R_k = \begin{bmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_k \end{bmatrix}.$$

This Yule-Walker-like equation is similar to the one considered in [28]. An important difference however is the fact that the right-hand side is not a vector anymore but a matrix of dimension $k \times p$. The solution Y_k of the k th order Yule-Walker system is therefore also a $k \times p$ matrix and plays an important role in solving the general problem with an arbitrary right-hand side. Even though we have to solve now in fact p problems, we will limit the overall complexity to $O(p^2 n)$.

First we will derive the recursion formulas, which will give us a $O(p^2 n^2)$ method. Afterwards we will remove the most expensive steps in the recursion, which will reduce the overall complexity by a factor n . The system is solved, as already mentioned, in a recursive manner. Assume we have the solution of Equation (3), and we would like to find the solution now of the $(k + 1)$ th system of equations:

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

Rewriting the above formula in block form leads to:

$$\left[\begin{array}{c|c} S_k & R_k \mathbf{u}_{k+1}^T \\ \hline \mathbf{u}_{k+1} R_k^T & \mathbf{u}_{k+1} \mathbf{v}_{k+1}^T \end{array} \right] \begin{bmatrix} Z_{k+1} \\ \alpha_{k+1} \end{bmatrix} = - \begin{bmatrix} R_k \\ \mathbf{v}_{k+1} \end{bmatrix}, \quad (4)$$

with $Z_{k+1} \in \mathbb{R}^{k \times p}$ and $\alpha_{k+1} \in \mathbb{R}^{1 \times p}$. Expanding Equation (4), we observe that

$$S_k Z_{k+1} + R_k \mathbf{u}_{k+1}^T \alpha_{k+1} = -R_k \quad (5)$$

and

$$\mathbf{u}_{k+1} R_k^T Z_{k+1} + \mathbf{u}_{k+1} \mathbf{v}_{k+1}^T \alpha_{k+1} = -\mathbf{v}_{k+1}. \quad (6)$$

Rewriting Equation (5) towards the matrix Z_{k+1} gives us

$$\begin{aligned} Z_{k+1} &= -S_k^{-1} R_k (I_p + \mathbf{u}_{k+1}^T \alpha_{k+1}) \\ &= Y_k (I_p + \mathbf{u}_{k+1}^T \alpha_{k+1}), \end{aligned} \quad (7)$$

where I_p denotes the identity matrix of size p . Substituting the latter equation in Equation (6) leads to:

$$\mathbf{u}_{k+1} R_k^T Y_k (I_p + \mathbf{u}_{k+1}^T \alpha_{k+1}) + \mathbf{u}_{k+1} \mathbf{v}_{k+1}^T \alpha_{k+1} = -\mathbf{v}_{k+1},$$

from which we can extract α_{k+1} as:

$$\alpha_{k+1} = \frac{(-1)(\mathbf{v}_{k+1} + \mathbf{u}_{k+1} R_k^T Y_k)}{(\mathbf{u}_{k+1} R_k^T Y_k \mathbf{u}_{k+1}^T + \mathbf{u}_{k+1} \mathbf{v}_{k+1}^T)}.$$

Using now the vector α_{k+1} in equation (7), 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, having complexity $O(p^2 n^2)$. The goal is however to come to an $O(p^2 n)$ algorithm, so we have to eliminate two expensive, in terms of operations, steps. Before showing a possible reduction in complexity, we will briefly prove that the denominator in the formula for α_{k+1} is always nonzero. Because our matrix S is assumed to be strongly nonsingular, we know that all the principal $k \times k$ matrices are nonsingular. This means that for every nonzero vector w : $w^T S_k w \neq 0$. Taking now $w^T = [\mathbf{u}_{k+1} Y_k^T, 1]$, we can derive the following equality:

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

Using the fact that $S_k Y_k = -R_k$ we come to

$$\mathbf{u}_{k+1} R_k^T Y_k \mathbf{u}_{k+1}^T + \mathbf{u}_{k+1} \mathbf{v}_{k+1}^T \neq 0$$

which states that the calculation of α_{k+1} is well defined. Let us focus now again on reducing the complexity.

The calculation in each step of the algorithm of the matrix $R_k^T Y_k$, is too expensive. For each k this is a multiplication of a $p \times k$ matrix with a $k \times p$ matrix, which costs $p^2(2k-1)$ operations. Calculating the inner-product in this way, in every step of the algorithm, leads to an overall complexity of $O(p^2 n^2)$. If we rewrite the formula, we can reduce this complexity. Using the following relations we come to a recursion for calculating this inner-product:

$$\begin{aligned} R_{k+1}^T Y_{k+1} &= [R_k^T, \mathbf{v}_{k+1}^T] \left[\begin{array}{c} Z_{k+1} \\ \alpha_{k+1} \end{array} \right] \\ &= R_k^T Z_{k+1} + \mathbf{v}_{k+1}^T \alpha_{k+1} \\ &= R_k^T Y_k (I_p + \mathbf{u}_{k+1}^T \alpha_{k+1}) + \mathbf{v}_{k+1}^T \alpha_{k+1} \\ &= R_k^T Y_k + (R_k^T Y_k \mathbf{u}_{k+1}^T + \mathbf{v}_{k+1}^T) \alpha_{k+1}. \end{aligned}$$

This leads to a recursive calculation of the inner-product. It costs each step in the algorithm $p(2p-1) + p + p^2 + p^2 = 4p^2$, and hence is independent of k .

Secondly, the computation of Z_k and hence Y_k is too expensive to compute in each step. It costs at step k : $kp(2p-1) + O(p^2)$ operations, which leads again to an $O(p^2n^2)$ algorithm. If we postpone this computation up to the very end, we can reduce its complexity. The only thing needed to compute the final solution is the storage of the vectors α_k in each step of the algorithm. Using the equations $Y_{k+1} = [Z_{k+1}^T, \alpha_{k+1}^T]^T$ and $Z_{k+1} = Y_k (I_p + \mathbf{u}_k^T \alpha_k)$, we can write the final solution of the Yule-Walker system as:

$$Y = \begin{bmatrix} \alpha_1 (I_p + \mathbf{u}_2^T \alpha_2) \dots (I_p + \mathbf{u}_{n-1}^T \alpha_{n-1}) (I_p + \mathbf{u}_n^T \alpha_n) \\ \vdots \\ \alpha_{n-2} (I_p + \mathbf{u}_{n-1}^T \alpha_{n-1}) (I_p + \mathbf{u}_n^T \alpha_n) \\ \alpha_{n-1} (I_p + \mathbf{u}_n^T \alpha_n) \\ \alpha_n \end{bmatrix}.$$

This can be recursively rewritten, starting from the last element. The most expensive step in this calculation is the computation of the matrix products. Let us define the $p \times p$ matrix F_i as the matrix $F_i = (I_p + \mathbf{u}_i^T \alpha_i) \dots (I_p + \mathbf{u}_{n-1}^T \alpha_{n-1}) (I_p + \mathbf{u}_n^T \alpha_n)$. This means that the matrix $F_{i-1} = (I_p + \mathbf{u}_{i-1}^T \alpha_{i-1}) F_i$. Rewriting this product as $F_{i-1} = F_i + \mathbf{u}_{i-1}^T (\alpha_{i-1} F_i)$, we can obtain a constant complexity of $p^2 + p(2p-1) + p^2 = 4p^2 - p$ for computing one row in the solution vector Y (more details can be found in the following subsection).

3.2 The algorithm

Included is the Matlab¹ implementation for solving the Yule-Walker system in a way explained in the previous section. The software is publicly available and can be found on <http://www.cs.kuleuven.ac.be/~marc>. After each line of operations, the complexity for that computation in the algorithm is given. Note, that in the computation of the number of flops, we did not count a sign change as an operation nor did we count computations involving indices.

The input of the function consists of two matrices U and V , which generate the higher order semiseparable matrix S . Both U and V are of dimension $n \times p$, where p stands for the semiseparability rank of S .

```

1  ----- Yule-Walker solver for higher order semiseparable -----
2  FUNCTION Y=YWSOLVERHSS(U,V)
3
4  %-----PART 1: SOLVE THE SYSTEM-----
5  % Initialisation:n=size,p=semiseparability rank
6  [n,p]=size(U);
7
8  % Initialising alpha and the p x p matrix R'*Y
9  alpha(1,:)=-V(1,:)/(U(1,:)*V(1,:)); % FLOPS: 3p-1
10 ry=V(1,:)*alpha(1,:); % FLOPS: p^2
11
12 % The main loop in which we use always the previous solution
13 for k=2:n
14 % The product U(k,:)*ry appears twice
15  ury=U(k,:)*ry; % FLOPS: p(2p-1)
16
17 % Calculate the denominator and alpha
18  denominator=ury*U(k,:)+U(k,:)*V(k,:); % FLOPS: 4p-2
19  alpha(k,:)=(-1)*(V(k,:)+ury)/denominator; % FLOPS: 2p
20
21 % Calculate the new p x p matrix R'*Y for the next loop
22  ry=ry+(ry*U(k,:)+V(k,:))*alpha(k,:); % FLOPS: 4p^2
23 end
24
25 %-----PART 2: CALCULATE THE SOLUTION -----
26 % Initialisation of y and F
27 Y(n,:)=alpha(n,:);
28 F=eye(p);

```

¹Matlab is a registered trademark of the Mathworks inc.

```

29
30 % Constructing the n x p matrix Y
31 for i=n-1:-1:1
32     F= F+ U(i+1,:)'* Y(i+1,:);           % FLOPS: 2p^2
33     Y(i,:)=alpha(i,:)*F;                 % FLOPS: 2p^2-p
34 end
35 end
36

```

This leads to an operation count of $6np^2 + 5np - 2n - 5p^2 - 2p + 1$ flops for part 1 in the algorithm, and $4np^2 - np - 4p^2 + p$ flops for computing the solution. The overall number of operations is hence $10np^2 + 4np - 2n - 9p^2 - p + 1$ which gives us the desired $O(np^2)$ complexity.

3.3 The Levinson solver for higher order semiseparable matrices

In this section we will solve a system of equations for which the coefficient matrix is a higher order strongly nonsingular symmetric semiseparable matrix. The solver we will propose here is based on the Levinson idea for solving Toeplitz systems. The method is recursive and in every step k of the algorithm we need the solution of the k th order Yule-Walker-like equation from Section 3.1

We would like to find a solution x for the following system of equations $Sx = b$.

Assume we know the solution of the $(k+1)$ th order Yule-Walker system, and assume that we also know the solution of :

$$S_k x_k = \mathbf{d}_k,$$

where $\mathbf{d}_k^T = [b_1, \dots, b_k]$ and S_k is again the upper left $k \times k$ sub-block of the matrix S . Based on the solutions of the previous two equations we will now solve $S_{k+1} x_{k+1} = \mathbf{d}_{k+1}$.

We start deducing the formulas in a similar way as in Section 3.1. The system we would like to solve can be rewritten in block form as:

$$\left[\begin{array}{c|c} S_k & R_k \mathbf{u}_{k+1}^T \\ \hline \mathbf{u}_{k+1} R_k^T & \mathbf{u}_{k+1} \mathbf{v}_{k+1}^T \end{array} \right] \begin{bmatrix} w_{k+1} \\ \mu_{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{d}_k \\ b_{k+1} \end{bmatrix}. \quad (8)$$

We remark that in contrast to the previous section, the right-hand side is now a column vector instead of a matrix, therefore also x_k and x_{k+1} are column vectors.

Expanding the above formula (8) leads to the following two equations

$$S_k w_{k+1} + \mu_{k+1} R_k \mathbf{u}_{k+1}^T = \mathbf{d}_k \quad (9)$$

and

$$\mathbf{u}_{k+1} R_k^T w_{k+1} + \mu_{k+1} \mathbf{u}_{k+1} \mathbf{v}_{k+1}^T = b_{k+1}. \quad (10)$$

Equation (9) can be rewritten towards w_{k+1} , using thereby $S_k^{-1} \mathbf{d}_k = x_k$ and $S_k^{-1} R_k = Y_k$:

$$w_{k+1} = x_k + \mu_{k+1} Y_k \mathbf{u}_{k+1}^T.$$

Substituting the equation for w_{k+1} into Equation (10) and rewriting this leads to the following expression for μ_{k+1} :

$$\mu_{k+1} = \frac{b_{k+1} - \mathbf{u}_{k+1} R_k^T x_k}{(\mathbf{u}_{k+1} R_k^T Y_k \mathbf{u}_{k+1}^T + \mathbf{u}_{k+1} \mathbf{v}_{k+1}^T)}.$$

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} we can derive an order $O(p^2 n^2)$ method. We will reduce this complexity to $O(p^2 n)$. One can compare these results for higher order semiseparable matrices, with the available results for semiseparable plus diagonal matrices [28], and see that the formulas of both methods are quite similar.

If we want to come to an order $O(p^2 n)$ solver we need to adapt two things in the algorithm, namely the computation of the product $R_k^T x_k$, and we have to postpone the computation of the solution vector x up to the very end.

The computation of the vector $R_k^T x_k$ can be rewritten in a recursive manner:

$$\begin{aligned} R_{k+1}^T x_{k+1} &= [R_k^T, \mathbf{v}_{k+1}^T] \begin{bmatrix} w_{k+1} \\ \mu_{k+1} \end{bmatrix} \\ &= R_k^T x_k + \mu_{k+1} (R_k^T Y_k \mathbf{u}_{k+1}^T + \mathbf{v}_{k+1}^T). \end{aligned}$$

This recursive formula for computing $R_k^T x_k$ in each step of the algorithm requires a fixed number of operations independent of k .

Suppose we postpone the computation of the solution vector x up to the very end, and we store all the factors α_k and μ_k . Then we come to the following formula for computing the solution:

$$x = \begin{bmatrix} \vdots \\ \mu_{n-3} + \alpha_{n-3} (\mu_{n-2} \mathbf{u}_{n-2}^T + (I_p + \mathbf{u}_{n-2}^T \alpha_{n-2}) (\mu_{n-1} \mathbf{u}_{n-1}^T + (I_p + \mathbf{u}_{n-1}^T \alpha_{n-1}) \mu_n \mathbf{u}_n^T)) \\ \mu_{n-2} + \alpha_{n-2} (\mu_{n-1} \mathbf{u}_{n-1}^T + (I_p + \mathbf{u}_{n-1}^T \alpha_{n-1}) \mu_n \mathbf{u}_n^T) \\ \mu_{n-1} + \alpha_{n-1} \mu_n \mathbf{u}_n^T \\ \mu_n \end{bmatrix}.$$

The computation of the vector above can easily be rewritten in a recursive manner from bottom to top requiring $O(p^2 n)$ operations for computing the solution, instead of the $O(p^2 n^2)$ operations needed if it was incorporated in the first recursion.

Using the recursive calculation of the product $R_k^T x_k$ and postponing the computation of the solution vector x up to the very end, leads to an order $O(p^2 n)$ method for solving the system of equations with a higher order semiseparable coefficient matrix. In the next section an effective implementation of the method is described, together with the operation count for this method.

3.4 The algorithm

The software is publicly available and can be found at the same site as mentioned in Section 3.2. The input of the function consists of two $n \times p$ matrices U and V , which are the generators of the semiseparable matrix S of semiseparability rank p . The input vector b of length n is the right-hand side. The function returns as output the solution vector x of the system $Sx = b$.

Similar as in the previous algorithm the complexity is given after each operation and operations involving indices and sign changes are not counted.

```

1  ----- Levinson solver for higher order semiseparable -----
2  FUNCTION x=LEVSOLVERHSS(U,V,b)
3
4  %-----PART 1: SOLVE THE SYSTEM-----
5  % Initialisation:n=size,p=semiseparability rank
6  [n,p]=size(U);
7
8  % Initialising alpha and mu
9  denominator=U(1,:)*V(1,:)' ; % FLOPS: 2p-1
10 alpha(1,:)=-V(1,:)./denominator; % FLOPS: p
11 mu(1)=b(1)./denominator; % FLOPS: 1
12
13 % Initialising the p x p matrix R'*Y and the 1 x p vector R'*X
14 ry=V(1,:)'*alpha; % FLOPS: p^2
15 rx=V(1,:)'*mu(1); % FLOPS: p
16
17 % The main loop in which we use always the previous solution
18 for k=2:n
19 % Initialising help variables which appear twice in the loop
20 ury=U(k,:)*ry; % FLOPS: p(2p-1)
21 ryu=ry*U(k,:)' ; % FLOPS: p(2p-1)
22
23 % Calculate alpha and mu
24 denominator=ury*U(k,:)' + U(k,:)*V(k,:)' ; % FLOPS: 4p-2

```

```

25     mu(k)=(b(k)-U(k,:)*rx)/denominator;           % FLOPS: 2p+1
26     alpha(k, :)=(-1)*(V(k,:)+ury)/denominator;     % FLOPS: 2p
27
28     % Update the products R'*Y and R'*X
29     rx=rx+mu(k)*(ryu+V(k,:))';                   % FLOPS: 3p
30     ry=ry+(ryu+V(k,:))*alpha(k,:);               % FLOPS: 2p2+p
31 end
32
33 %-----PART 2: CALCULATE THE SOLUTION -----
34 % Initialisation of x and auxiliary variable G
35 x(n)=mu(n);
36 G=mu(n)*U(n,:);                                 % FLOPS: p
37
38 % Constructing the solution vector x
39 for i=n-1:-1:1
40     x(i)=mu(i)+alpha(i,:)*G;                     % FLOPS: 2p
41     G=x(i)*U(i,:)+G;                             % FLOPS: 2p
42 end
43 end
44

```

The overall operation count leads to $6np^2 + 10np - n - 5p^2 - 6p + 1$ operations for part 1 of the algorithm and $4np - 3p$ operations for constructing the solution. The overall complexity is therefore $O(np^2)$, more precisely the method performs $6np^2 + 14np - n - 5p^2 - 9p + 1$ flops.

4 A Levinson-like solver for band matrices

In this section we will develop a Levinson-like system solver for band matrices. We will derive this solver in a similar way as for the higher order semiseparable matrices. This means that we will first solve a Yule-Walker-like system and secondly we will use this solution for solving a band system with an arbitrary right-hand side.

4.1 A Yule-Walker-like solver for band matrices

Let us denote the leading $k \times k$ matrix of B with B_k . The Yule-Walker-like equations we would like to solve for this band matrix are the following ones:

$$B_k Y_k = -T_k$$

where T_k is a $k \times l$ matrix of the following form:

$$T_k = \begin{bmatrix} 0 \\ I_l \end{bmatrix},$$

where I_l denotes the identity matrix of size l . If we are in the beginning of the algorithm, we have that $k \leq l$. In this case we take only the last k lines of I_l , this means that:

$$T_k = \begin{cases} [0, I_k] & \text{if } k < l \\ I_l & \text{if } k = l \\ \begin{bmatrix} 0 \\ I_l \end{bmatrix} & \text{if } k > l \end{cases}$$

This means that also here we have to solve in fact l equations to obtain the solution of the Yule-Walker-like equation. Firstly we will derive an $O(l^2 n^2)$ method, but afterwards we will reduce this complexity to come to an $O(l^2 n)$ algorithm.

Note that for this particular Yule-Walker system k first rows of the right-hand side of the equations changes in every step. This was not the case in the system corresponding to the higher order semiseparable matrix, in that case the right-hand side was increased in dimension every time by adding one more row.

Let us solve now the $(k + 1)$ th order Yule-Walker-like equation, assuming we have the solution of the k th order system. The $(k + 1)$ th system $B_{k+1}Y_{k+1} = -T_{k+1}$ can be rewritten in block form, using the representation matrix A , which represents the band matrix B (see Section 2.2):

$$\left[\begin{array}{c|c} B_k & T_k \bar{\mathbf{a}}_{k+1}^T \\ \hline \bar{\mathbf{a}}_{k+1} T_k^T & a_{k+1,l+1} \end{array} \right] \begin{bmatrix} Z_{k+1} \\ \boldsymbol{\alpha}_{k+1} \end{bmatrix} = -T_{k+1},$$

where $\bar{\mathbf{a}}_k$ equals the first l elements of the vector \mathbf{a}_k , so in fact $\mathbf{a}_k = [\bar{\mathbf{a}}_k, a_{k,l+1}]$. Writing the right-hand side in block form does not give us something of the following form: $[T_k^T, \times]^T$. To be able to use the right-hand side in the recursion formulas we need to use an operator. Let us define the shift operator P as the $l \times l$ matrix of the following form

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

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 the right-hand side T_{k+1} can be written as

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

Therefore we can extract the following two formulas in terms of the previous right-hand side T_k :

$$B_k Z_{k+1} + T_k \bar{\mathbf{a}}_{k+1}^T \boldsymbol{\alpha}_{k+1} = -T_k P$$

and

$$\bar{\mathbf{a}}_{k+1} T_k^T Z_{k+1} + a_{k+1,l+1} \boldsymbol{\alpha}_{k+1} = -[0, \dots, 0, 1]. \quad (11)$$

Rewriting the first equation towards Z_{k+1} leads to

$$Z_{k+1} = -B_k^{-1} T_k (P + \bar{\mathbf{a}}_{k+1}^T \boldsymbol{\alpha}_{k+1}) = Y_k (P + \bar{\mathbf{a}}_{k+1}^T \boldsymbol{\alpha}_{k+1}).$$

Combined with Equation (11) we can determine $\boldsymbol{\alpha}_{k+1}$,

$$\boldsymbol{\alpha}_{k+1} = (-1) \frac{[0, \dots, 0, 1] + \bar{\mathbf{a}}_{k+1} T_k^T Y_k P}{\bar{\mathbf{a}}_{k+1} T_k^T Y_k \bar{\mathbf{a}}_{k+1}^T + a_{k+1,l+1}}.$$

Similarly like in Section 3 one can prove non zeroness of the denominator in the equation for $\boldsymbol{\alpha}_{k+1}$. The relations for $\boldsymbol{\alpha}_{k+1}$ and Z_{k+1} give us a recursive algorithm of complexity $O(l^2 n^2)$. We will further reduce this complexity by rewriting the calculation of $T_k^T Y_k$ in a recursive manner, and by postponing the calculation of the solution matrix Y .

Let us derive a recursive formula for the product $T_k^T Y_k$:

$$\begin{aligned} T_{k+1}^T Y_{k+1} &= \begin{bmatrix} P^T T_k^T, \\ \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \end{bmatrix} \begin{bmatrix} Z_{k+1} \\ \boldsymbol{\alpha}_{k+1} \end{bmatrix} \\ &= P^T T_k^T Y_k (P + \bar{\mathbf{a}}_{k+1}^T \boldsymbol{\alpha}_{k+1}) + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \boldsymbol{\alpha}_{k+1}. \end{aligned}$$

Using this relation we can write the product $T_{k+1}^T Y_{k+1}$ in terms of the product of the previous step $T_k^T Y_k$. The updating of the product in every step takes now a number of operations independent of k .

The last important resources taking computation is the calculation of the vector Z_k and hence Y_k in every step. If we store however the vectors α_k for every k we can compute this vector in an easy manner, similarly as for the higher order semiseparable case:

$$Y = \begin{bmatrix} \alpha_1 (P + \bar{\mathbf{a}}_2^T \alpha_2) \dots (P + \bar{\mathbf{a}}_{n-1}^T \alpha_{n-1}) (P + \bar{\mathbf{a}}_n^T \alpha_n) \\ \vdots \\ \alpha_{n-2} (P + \bar{\mathbf{a}}_{n-1}^T \alpha_{n-1}) (P + \bar{\mathbf{a}}_n^T \alpha_n) \\ \alpha_{n-1} (P + \bar{\mathbf{a}}_n^T \alpha_n) \\ \alpha_n \end{bmatrix}.$$

Constructing the solution matrix Y from bottom to top costs now $O(l^2 n)$ instead of $O(l^2 n^2)$. The overall algorithm has therefore a complexity of $O(l^2 n)$.

4.2 The algorithm

In this section a Matlab implementation of the algorithm for solving the Yule-Walker-like problem for band matrices is given. The input of the algorithm is the matrix A , which contains in fact the representation of the band matrix B . The output Y is the solution matrix of the Yule-Walker-like equation. The number of flops corresponding to a particular equation are given at the end of every line.

```

----- Yule-Walker solver for band matrices -----
1
2 FUNCTION Y=YWSOLVERBAND(A)
3
4 %-----PART 1: SOLVE THE SYSTEM-----
5 % Initialisation:n=size,2l+1=bandwidth
6 [n,l]=size(A);l=l-1;
7
8 % Initialising alpha, l x l matrix T'*Y
9 alpha(1,l)=-1/A(1,l+1);ty(1,l)=alpha(1,l); % FLOPS: 1
10
11 % The main loop in which we use always the previous solution
12 for k=2:n
13 % Because the inner-product A(k,1:l)* ty appears twice
14 bty=A(k,1:l)*ty; % FLOPS: 2l^2 - l
15
16 % Calculate alpha
17 denominator=(bty*B(k,1:l)'+A(k,l+1)); % FLOPS: 3l - 1
18 alpha(k,:)=[-bty(:,2:l),-1]/denominator; % FLOPS: l
19
20 % Calculation of the T'*Y for the next loop
21 tyb=ty(2:l,:)*A(k,1:l)'; % FLOPS: (l-1)(2l-1)
22 ty(1:l-1,1:l)=[ty(2:l,2:l)+tyb*alpha(k,1:l-1),tyb*alpha(k,l)]; % FLOPS: 2(l-1)^2+l-1
23
24 ty(1,:)=alpha(k,:);
25 end
26
27 %-----PART 2: CALCULATE THE SOLUTION -----
28 % Initialisation of Y and F
29 Y(n,:)=alpha(n,:);
30 F=eye(l);
31
32 % Construction of the n x l matrix Y
33 for i=n-1:-1:1
34 F=[A(i+1,l)*Y(i+1,:)+F(1:l-1,:)+A(i+1,2:l)'+Y(i+1,:)]; % FLOPS: 2l(l-1)+l
35 Y(i,:)=alpha(i,:)*F; % FLOPS: l(2l-1)
36 end
37
38 end
39

```

The number of operations for the first part in the algorithm is $6nl^2 - 2nl - 6l^2 + 2l + 1$, the number of operations needed to calculate the solution is $4nl^2 - 3nl - 4l^2 + 3l$. The overall complexity is $O(l^2n)$, more precisely $10nl^2 - 5nl - 10l^2 + 5l + 1$ flops are needed to compute the solution of the Yule-Walker-like equation for band matrices.

4.3 The Levinson solver for band matrices

In this section we will construct a Levinson solver for band matrices. Again this method is based on the Yule-Walker solution as presented in the previous section. Let us consider the right-hand side b , and use the same notation as for the Yule-Walker-like solver for band matrices, the system we would like to solve is $Bx = b$.

Assume the solution in step k :

$$B_k x_k = \mathbf{d}_k,$$

where $d_k^T = [b_1, \dots, b_k]$, is known. Moreover also the solution of the $(k+1)$ th order Yule-Walker-like equation is known.

We start deducing the formulas in a similar way as in the previous section. The $(k+1)$ th system can be written as:

$$B_{k+1} x_{k+1} = \mathbf{d}_{k+1}$$

$$\left[\begin{array}{c|c} B_k & T_k \bar{\mathbf{a}}_{k+1}^T \\ \hline \bar{\mathbf{a}}_{k+1} T_k^T & a_{k+1,l+1} \end{array} \right] \begin{bmatrix} w_{k+1} \\ \mu_{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{d}_k \\ b_{k+1} \end{bmatrix}.$$

Expanding towards the rows, leads to the following two equations:

$$B_k w_{k+1} + \mu_{k+1} T_k \bar{\mathbf{a}}_{k+1}^T = \mathbf{d}_k,$$

$$\bar{\mathbf{a}}_{k+1} T_k^T w_{k+1} + \mu_{k+1} a_{k+1,l+1} = b_{k+1}.$$

In a straightforward way, we can solve the system towards w_{k+1} and substitute this solution in the equation above. This gives us for μ_{k+1} and w_{k+1} the following relations:

$$w_{k+1} = x_k + \mu_{k+1} Y_k \bar{\mathbf{a}}_{k+1}^T$$

$$\mu_{k+1} = \frac{b_{k+1} - \bar{\mathbf{a}}_{k+1} T_k^T x_k}{(\bar{\mathbf{a}}_{k+1} T_k^T Y_k \bar{\mathbf{a}}_{k+1}^T + a_{k+1,l+1})}.$$

This formula is well defined because the matrix T is strongly nonsingular. This leads to an $O(l^2n^2)$ method. But we can again decrease the complexity: the computation of the vector $T_k^T x_k$ can be rewritten in a recursive manner:

$$T_{k+1}^T x_{k+1} = \begin{bmatrix} P^T T_k^T & \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \end{bmatrix} \begin{bmatrix} w_{k+1} \\ \mu_{k+1} \end{bmatrix}$$

$$= P^T T_k^T (x_k + \mu_{k+1} Y_k \bar{\mathbf{a}}_{k+1}^T) + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \mu_{k+1}.$$

Suppose we postpone the computation of the solution vector x up to the very end, and we store all the factors α_k and μ_k . This gives us the following formula for computing the solution:

$$x = \begin{bmatrix} \vdots \\ \mu_{n-3} + \alpha_{n-3} (\mu_{n-2} \bar{\mathbf{a}}_{n-2}^T + (P + \bar{\mathbf{a}}_{n-2}^T \alpha_{n-2}) (\mu_{n-1} \bar{\mathbf{a}}_{n-1}^T + (P + \bar{\mathbf{a}}_{n-1}^T \alpha_{n-1}) \mu_n \bar{\mathbf{a}}_n^T)) \\ \mu_{n-2} + \alpha_{n-2} (\mu_{n-1} \bar{\mathbf{a}}_{n-1}^T + (P + \bar{\mathbf{a}}_{n-1}^T \alpha_{n-1}) \mu_n \bar{\mathbf{a}}_n^T) \\ \mu_{n-1} + \alpha_{n-1} \mu_n \bar{\mathbf{a}}_n^T \\ \mu_n \end{bmatrix}.$$

The computation of the vector above can easily be rewritten to come to a complexity of $O(l^2n)$.

In the following section an implementation of the solver for band matrices based on the Levinson idea is given.

4.4 The algorithm

The function below solves a symmetric band system of equations. The input consists of the right-hand side b and the matrix A representing the band matrix B . The output of the algorithm is the vector $x = B^{-1}b$. The number of operations in each step is given at the end of the line.

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43

```

Levinson solver for band matrices

```

FUNCTION x=LEVSOLVERBAND(A,b)
%-----PART 1: SOLVE THE SYSTEM-----
% Initialisation:n=size,2l+1=bandwidth
[n,l]=size(A);l=l-1;
% Initialisation of alpha and mu
mu(1)=b(1)/B(1,l+1);alpha(1,l)=-1/B(1,l+1);           % FLOPS: 2
% Initialisation of the matrix T'*Y and the vector T'*x
ty(1,l)=alpha(1,l);tx(1,l)=mu(1);
% The main loop in which we use always the previous solution
for k=2:n
% Because the products A(k,1:l)*ty and ty(2:l,:)*A(k,1:l)' appear twice
  bty=A(k,1:l)*ty;                                     % FLOPS: 2l-1
  tyb=ty(2:l,:)*A(k,1:l)';                             % FLOPS: (l-1)(2l-1)
% Calculate alpha and mu
  denominator=(bty*A(k,1:l)'+A(k,l+1));                % FLOPS: 3l-1
  alpha(k,:)=[-bty(:,2:l),-1]/denominator;             % FLOPS: l
  mu(k)=(b(k)-A(k,1:l)*tx)/denominator;                % FLOPS: 2l+1
% Updating ty and tx
  tx=[tx(2:l)+mu(k)*tyb;mu(k)];                         % FLOPS: 2(l-1)
  ty(1:l-1,1:l)=[ty(2:l,2:l)+tyb*alpha(k,1:l-1),tyb*alpha(k,1)]; % FLOPS: 2(l-1)^2+l-1
  ty(1,:)=alpha(k,:);
end
%-----PART 2: CALCULATE THE SOLUTION -----
% Initialisation of x and M
M=zeros(l,l);
x(n)=mu(n);
% Constructing of x
for i=n-1:-1:1
  M=[A(i+1,1)'+x(i+1);A(i+1,2:l)'+x(i+1)+M(1:l-1)]; % FLOPS: 2(l-1)+1
  x(i)=mu(i)+alpha(i,:)*M;                             % FLOPS: 2l
end
end

```

The total number of operations for the first part of the algorithm is $6nl^2 + nl - 6l^2 - l + 2$, the number of operations for calculating the solution vector x is $4nl - n - 4l + 1$. This means that the overall complexity of the algorithm is $O(l^2n)$, more precisely $6nl^2 + 5nl - n - 6l^2 - 5l + 3$ flops are needed.

5 A Levinson solver for higher order semiseparable plus band matrices.

In this section the results of the previous two sections will be combined, to come to a system solver where the coefficient matrix is a higher order semiseparable plus band matrix. Again we first have to derive a solution for the Yule-Walker-like problem and afterwards we will solve the general problem with a Levinson-like algorithm.

5.1 The Yule-Walker-like problem

As the construction of the solution is always rather similar, we do not go into the details of this construction. We only give the essential formulas for solving the problem.

Suppose we have a band matrix B of bandwidth $2l + 1$ and a semiseparable matrix S of semiseparability rank p . The Yule-Walker-like problems we would like to solve are the following ones:

$$(S_k + B_k)Y_k = -[R_k, T_k],$$

where the matrices R_k and T_k are respectively of dimension $k \times p$ and $k \times l$. The matrices are the same as the ones used for solving the semiseparable problem and the band problem separately. This means that:

$$R_k = \begin{bmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_k \end{bmatrix} \quad \text{and} \quad T_k = \begin{cases} [0, I_k] & \text{if } k < l \\ I_l & \text{if } k = l \\ \begin{bmatrix} 0 \\ I_l \end{bmatrix} & \text{if } k > l \end{cases}.$$

Suppose we know the solution of the k th problem and we would like to solve the $(k+1)$ th Yule-Walker-like equation. Let us introduce some extra notation for simplifying the coming formulas. Let $Q_k = [R_k, T_k] \in \mathbb{R}^{k \times (l+p)}$ and let $\mathbf{c}_k = [\mathbf{u}_k, \bar{\mathbf{a}}_k] \in \mathbb{R}^{1 \times (p+l)}$. Rewriting the $(k+1)$ th equation in block form leads to:

$$\begin{aligned} B_{k+1}Y_{k+1} &= -Q_{k+1}, \\ \left[\begin{array}{c|c} B_k & Q_k \mathbf{c}_{k+1}^T \\ \mathbf{c}_{k+1} Q_k^T & \mathbf{u}_{k+1} \mathbf{v}_{k+1}^T + a_{k+1, l+1} \end{array} \right] \begin{bmatrix} Z_{k+1} \\ \alpha_{k+1} \end{bmatrix} &= -Q_{k+1}. \end{aligned} \quad (12)$$

The right-hand side Q_{k+1} still needs to be rewritten in block form. Let us define the operator \tilde{P} as the block diagonal matrix having as diagonal blocks I_p and P , with I_p the identity matrix of size p and P the left shift matrix from the previous section. This leads to the following block splitting for the matrix Q_{k+1} :

$$Q_{k+1} = \begin{bmatrix} Q_k \tilde{P} \\ [\mathbf{v}_k, 0, \dots, 0, 1] \end{bmatrix}.$$

In a completely similar way as in the previous sections we can derive now the equations determining α_{k+1} and Z_{k+1} :

$$\begin{aligned} Z_{k+1} &= Y_k (\tilde{P} + \mathbf{c}_{k+1}^T \alpha_{k+1}) \\ \alpha_{k+1} &= \frac{-[\mathbf{v}_k, 0, \dots, 1] - \mathbf{c}_{k+1} Q_k^T Y_k \tilde{P}}{\mathbf{c}_{k+1} Q_k^T Y_k \mathbf{c}_{k+1}^T + \mathbf{u}_{k+1} \mathbf{v}_{k+1}^T + a_{k+1, l+1}}. \end{aligned}$$

Also the formulas for computing the product $Q_k^T Y_k$ in each step of the algorithm, and the computation of the solution Y_k at the very end, are done in a similar way as before and hence not included. The algorithm is however included.

5.2 The algorithm

The function below, takes as input the generators U and V of the semiseparable matrix and the generator matrix A of the band matrix B . The output is the solution matrix Y of the Yule-Walker-like equation described in the previous section.

Yule-Walker solver

```

1
2 FUNCTION Y=YWSOLVERBANDHSS(U,V,A)
3
4 %-----PART 1: SOLVE THE SYSTEM-----
5 % Initialisation:n=size,2l+1=bandwidth,p=semiseparability rank
6 [n,l]=size(A);l=l-1;
7 [n,p]=size(U);
8
9 % Initialisation of alpha and Q'Y
10 alpha(1,[1:p,p+1])=[-V(1,:),-1]./(U(1,:)*V(1,:)+A(1,l+1));% FLOPS: 3p+1
11 qy=[V(1,:),zeros(1,l-1),1]'*alpha(1,:); % FLOPS: p(p+1)
12
13 % The main loop in which we use always the previous solution
14 for k=2:n
15 % Initialisation of help variable c, product c*qy
16 c=[U(k,:),A(k,1:l)];cqy=c*qy; % FLOPS: (p+l)(2(p+l)-1)
17
18 % Calculate alpha
19 denominator=cqy*c'+U(k,:)*V(k,:)+A(k,l+1); % FLOPS: (2(p+l)-1)+2p
20 alpha(k,:)=[-V(k,:)-cqy(1:p),-cqy(p+2:l+p),-1]/denominator; % FLOPS: l+2p
21
22 % Calculation of the new Product Q'Y
23 qyc=qy([1:p,p+2:l+p],:)*c'; % FLOPS: (p+l-1)(2(p+l)-1)
24 qyc(1:p)=qyc(1:p)+V(k,:); % FLOPS: p
25 qy=[qy([1:p,p+2:l+p],[1:p,p+2:l+p])+qyc*alpha(k,1:l+p-1),qyc*alpha(k,l+p)]; % FLOPS: 2(p+l-1)^2+p+l-1
26
27 qy=[qy;alpha(k,:)];
28 end
29
30 %-----PART 2: CALCULATE THE SOLUTION -----
31 % Initialisation of Y and F
32 Y(n,:)=alpha(n,:);
33 F=eye(l+p);
34
35 % Construction of Y
36 for i=n-1:-1:1
37 F(1:p,:)=F(1:p,:)+U(i+1,:)*Y(i+1,:); % FLOPS: 2p(l+p)
38 F(p+2:l+p,:)=F(p+1:l+p-1,:)+A(i+1,2:l)*Y(i+1,:); % FLOPS: 2(l-1)(l+p)
39 F(p+1,:)=A(i+1,1)*Y(i+1,:); % FLOPS: l+p
40 Y(i,:)=alpha(i,:)*F; % FLOPS: (2(l+p)-1)(l+p)
41 end
42

```

Combining the operations, leads to a total of $6n(p+l)^2 - 4nl + n - 6(p+l)^2 + 4l + p^2 + 4p$ operations for executing part 1 in the algorithm, and $4n(p+l)^2 - 2n(p+l)$ operations for part 2. The complete method is of order $O((l+p)^2n)$.

5.3 The Levinson solver for higher order semiseparable plus band matrices

As the computation of the formulas is again a straightforward generalization of the previous sections, we do not include the complete derivation. Only the relations determining w_{k+1} and μ_{k+1} are given. We want to solve the following system: $(B+S)x = b$. Suppose we know the solution of the k th order system $(S_k + B_k)x_k = \mathbf{d}_k$ and of the k th order Yule-Walker-like equation $(S_k + B_k)Y_k = -Q_k$. Then we can write for the following system:

$$(S_{k+1} + B_{k+1}) \begin{bmatrix} w_{k+1} \\ \mu_{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{d}_k \\ b_{k+1} \end{bmatrix}$$

the solutions of w_{k+1} and μ_{k+1} as:

$$\begin{aligned} w_{k+1} &= x_k + \mu_k Y_k \mathbf{c}_{k+1} \\ \mu_{k+1} &= \frac{b_{k+1} - \mathbf{c}_{k+1} Q_k^T x_k}{\mathbf{c}_{k+1} Q_k^T Y_k \mathbf{c}_{k+1}^T + \mathbf{u}_{k+1} \mathbf{v}_{k+1}^T + a_{k+1,l+1}}. \end{aligned}$$

Similarly like in the previous sections we can write formulas for calculating $Q_k^T x_k$ such that this computation is recursively. Moreover also the computation of the solution vector x can be postponed up to the very end, to reduce the complexity of the algorithm.

5.4 The algorithm

The function below takes as input the generators U and V of the semiseparable matrix S , the matrix A which represents the band matrix B and the right-hand side b . The output of the method is the solution x of the linear system $(S+B)x = b$.

```

----- Levinson solver -----
1
2 FUNCTION x=LEVSOLVERBANDHSS(U,V,A,b)
3
4 %-----PART 1: SOLVE THE SYSTEM-----
5 % Initialisation:n=size,2l+1=bandwidth,p=semiseparability rank
6 [n,l]=size(A);l=l-1;
7 [n,p]=size(U);
8
9 % Initialisation of alpha and mu
10 alpha(1,[1:p,p+1])=[-V(1,:),-1]./(U(1,:)*V(1,.)'+A(1,l+1)); % FLOPS: 3p+1
11 mu(1)=b(1)/(A(1,l+1)+U(1,:)*V(1,.)');
12
13 % initialisation of the product Q'*Y and Q'*x
14 qy=[V(1,:),zeros(1,l-1),1]'*alpha(1,:); % FLOPS: p(p+1)
15 qx=[V(1,:),zeros(1,l-1),1]'*mu(1); % FLOPS: p+1
16
17 % the main loop in which we use always the previous solution
18 for k=2:n
19 % Initialisation of help variable c and two products
20 c=[U(k,:),A(k,1:l)];cqy=c*qy; % FLOPS: (p+l)(2(p+l)-1)
21 qyc=qy([1:p,p+2:l+p],:)*c'; % FLOPS: (2(p+l)-1)(p+l-1)
22 qyc(1:p)=qyc(1:p)+V(k,:)' ; % FLOPS: p
23
24 % Calculate alpha and mu
25 denominator=cqy*c'+U(k,:)*V(k,.)'+A(k,l+1); % FLOPS: (2(p+l)-1)+2p
26 alpha(k,:)=[-V(k,)-cqy(1:p),-cqy(p+2:l+p),-1]/denominator; % FLOPS: l+2p
27 mu(k)=(b(k)-c*qx)/denominator; % FLOPS: 2(p+l)+1
28
29 % Calculation of the new Product Q'*x and Q'*y
30 qx=[qx([1:p,p+2:l+p])+qyc*mu(k);mu(k)]; % FLOPS: 2(l+p-1)
31 qy=[qy([1:p,p+2:l+p],[1:p,p+2:l+p])+qyc*alpha(k,1:l+p-1),qyc*alpha(k,l+p)];
32 % FLOPS: 2(p+l-1)^2+p+l-1
33 qy=[qy;alpha(k,:)];
34 end
35
36 %-----PART 2: CALCULATE THE SOLUTION -----
37 % Initialisation of x
38 x(n)=mu(n);
39
40 % Constructing x
41 for i=n-1:-1:1
42 M(1:p,1)=M(1:p)+U(i+1,:)'*x(i+1); % FLOPS: 2p
43 M(p+2:l+p,1)=M(p+1:l+p-1)+A(i+1,2:l)'*x(i+1); % FLOPS: 2(l-1)
44 M(p+1,1)=A(i+1,1)*x(i+1); % FLOPS: 1
45 x(i)=mu(i)+alpha(i,:)*M; % FLOPS: 2(l+p)
46 end
47

```

The first part of the method performs $(6(p+l)^2 + 4p - 1)(n-1) + p^2 + 5p + 2$ operations, while the second one performs $(n-1)(4(p+l)-1)$ operations. The overall complexity is therefore $6n(p+l)^2$.

6 An upper triangular factorization of the inverse

The method presented in this paper is closely related to an upper triangular factorization of the inverse of the matrix on which we applied our solver, this was expectable as also the Levinson algorithm for Toeplitz matrices has this relation [30]. In this section we will show this relation. As the construction for all three shown methods is completely similar we will only deduce the formulas for the inverse of the higher order semiseparable plus band matrix. The other two cases are dealt with in a similar way. We use the same notation as in the previous section. Let us denote the solution of the k th order Yule-Walker type equation as

$$(S_k + B_k)Y_k = -Q_k.$$

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

$$(S_k + B_k)Y_k \mathbf{c}_{k+1}^T + Q_k \mathbf{c}_{k+1}^T = 0.$$

This equation can be rewritten towards the matrix $S_{k+1} + B_{k+1}$, the above system is similar to:

$$\left(\begin{array}{c|c} S_k + B_k & Q_k \mathbf{c}_{k+1}^T \\ \hline \mathbf{c}_{k+1}^T Q_k^T & \mathbf{u}_{k+1} \mathbf{v}_{k+1}^T + a_{k+1,l+1} \end{array} \right) \begin{pmatrix} Y_k \mathbf{c}_{k+1}^T \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ \sigma_{k+1} \end{pmatrix}.$$

If we put the successive equations above in an upper triangular matrix we get (we note that the products $Y_k \mathbf{c}_{k+1}^T$ are column vectors of dimension k):

$$(S+B) \begin{pmatrix} 1 & Y_1 \mathbf{c}_2^T & Y_2 \mathbf{c}_3^T & \dots & Y_{n-1} \mathbf{c}_n^T \\ 0 & 1 & & & \\ 0 & 0 & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ 0 & 0 & \dots & 0 & 1 \end{pmatrix} = \begin{pmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ \times & \sigma_2 & 0 & \dots & 0 \\ \vdots & \times & \sigma_3 & & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \times & \times & \dots & \times & \sigma_n \end{pmatrix}.$$

The \times denote arbitrary elements in the matrix. Rewriting the equations above, defining the upper triangular matrix to the right of $(S+B)$ as U and defining the matrix Λ as the diagonal matrix with on its diagonal the elements σ_i , we get the following upper triangular factorization of the inverse of $(S+B)$

$$(S+B)^{-1} = U \Lambda^{-1} U^*.$$

Moreover, in fact we get the representation of the upper triangular factor U , which gives us an easy way to construct the generators of the inverse of $(S+B)$. More information on this factorization can be found in [28, 29].

7 Numerical Experiments

In this section two types of numerical experiments are performed. For all the different algorithms proposed in this paper we performed timing experiments and accuracy experiments.

7.1 Timings for the solvers

In a first experiment we considered timings in Matlab. We performed experiments for different sizes of the matrices: $[5 \cdot 10^2, 5 \cdot 10^3, 5 \cdot 10^4, 5 \cdot 10^5]$. These experiments were repeated for several values of the semiseparability rank: for every matrix size, experiments were constructed for ranks 1, 2, 4, 8, 16. In Figure 1, the different sizes of the matrices are plotted in a log scale on the x-axis, while the y-axis contains the cputime in a log scale needed for calculating the solution. The different lines correspond to the different values of the semiseparability rank.

The second experiment is similar to the previous one. Here we timed the solver for band matrices. The experiments performed are the same as above, but varying the bandwidths instead of the semiseparability ranks.

We did not include timing tests for the combined solver, as the behaviour is completely similar.

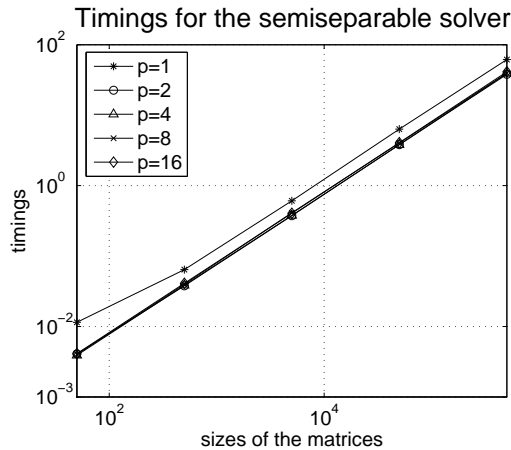


Figure 1: Timings for semiseparable matrices with different values of p .

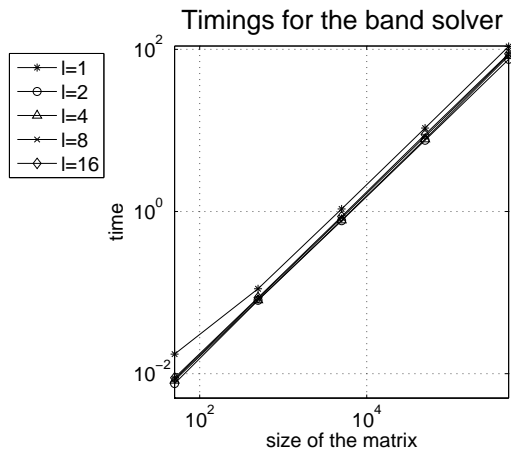


Figure 2: Timings for band matrices with different bandwidths $2l + 1$.

7.2 Accuracy tests for the solvers

To test the accuracy of the semiseparable solver, experiments were performed for different sizes of matrices, and also for different semiseparability ranks. The sizes are $[5 \cdot 10^2, 5 \cdot 10^3, 5 \cdot 10^4, 5 \cdot 10^5]$ and the different ranks were equal to $[1, 2, 4, 8, 16]$. We generated a solution vector x and calculated then the right-hand side $b = Bx$. For every problem we calculated two different things. When we denote with \tilde{x} the computed solution, we computed the relative residual $\|B\tilde{x} - b\|/\|b\|$ and we calculated also the relative error $\|\tilde{x} - x\|/\|x\|$. The observations are shown in Figure 3.

To test the accuracy of the band solver we performed similar experiments as for the semiseparable solver. Instead of changing the semiseparability rank, we changed now the bandwidth. The results are shown in Figure 4.

Also tests were performed to check the accuracy of the solver for higher order semiseparable plus band matrices. We performed experiments for $p = [1, 2, 4, 8]$ thereby varying for every p the bandwidth of the examples $l = [1, 2, 4, 8]$. For every every choice of p and l experiments were performed of different sizes $[10, 10^2, \dots, 10^5]$. The results are depicted in Figure 5.

In the last experiment, we tested some matrices with a principal leading submatrix which is almost singular. We generated a test matrix of dimension 20. The matrix A is the sum of a higher order genera-

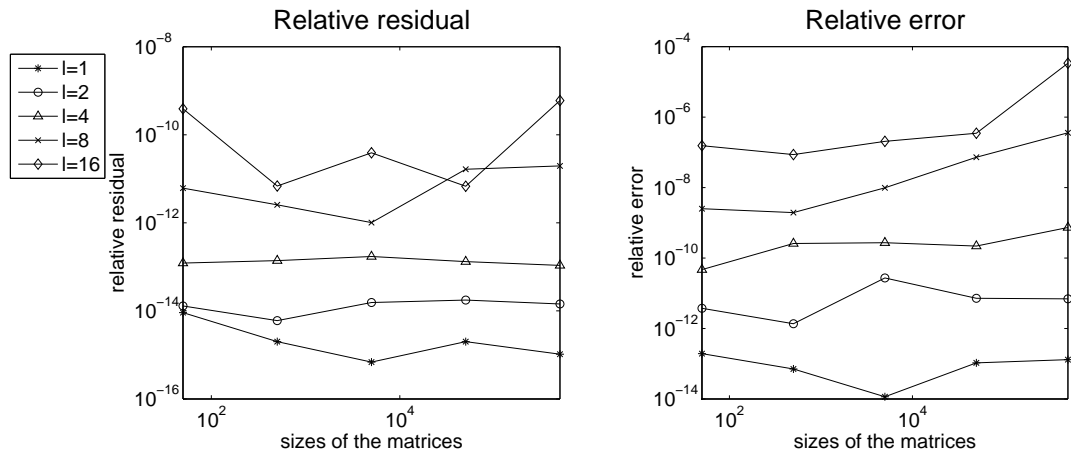


Figure 3: Accuracy graphs for the semiseparable solver

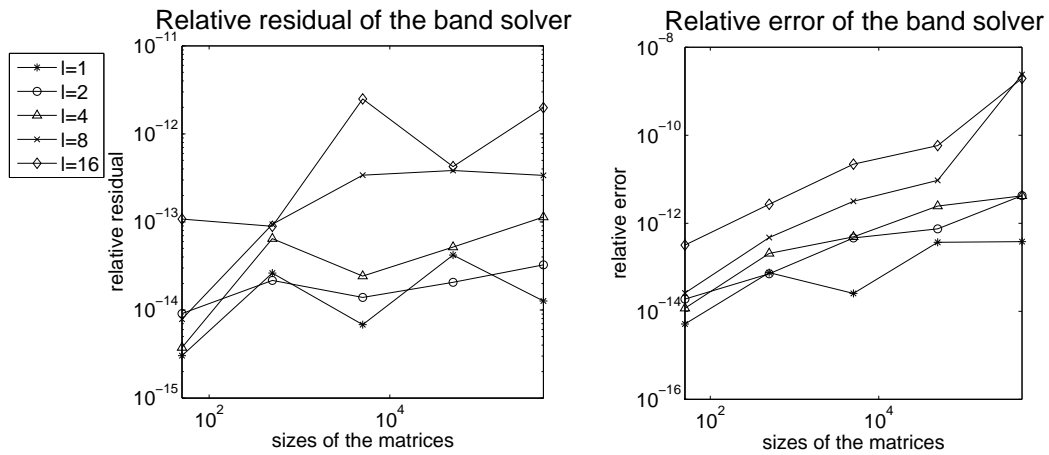


Figure 4: Accuracy graphs for the band solver

tor representable semiseparable matrix of semiseparability rank 2 and a bandmatrix of bandwidth 2. We changed the upper left 10×10 block of the matrix, by subtracting a diagonal from it. This diagonal was chosen to be very close to the smallest eigenvalue of this leading 10×10 matrix. We varied this diagonal to obtain different condition numbers for the upper left block. In the following table the results of this experiment are shown. From left to right we see the condition number of the complete matrix, the condition number of the leading 10×10 matrix, the relative error and finally the relative residual. The relative error and the relative residual are computed similarly as in the other experiments.

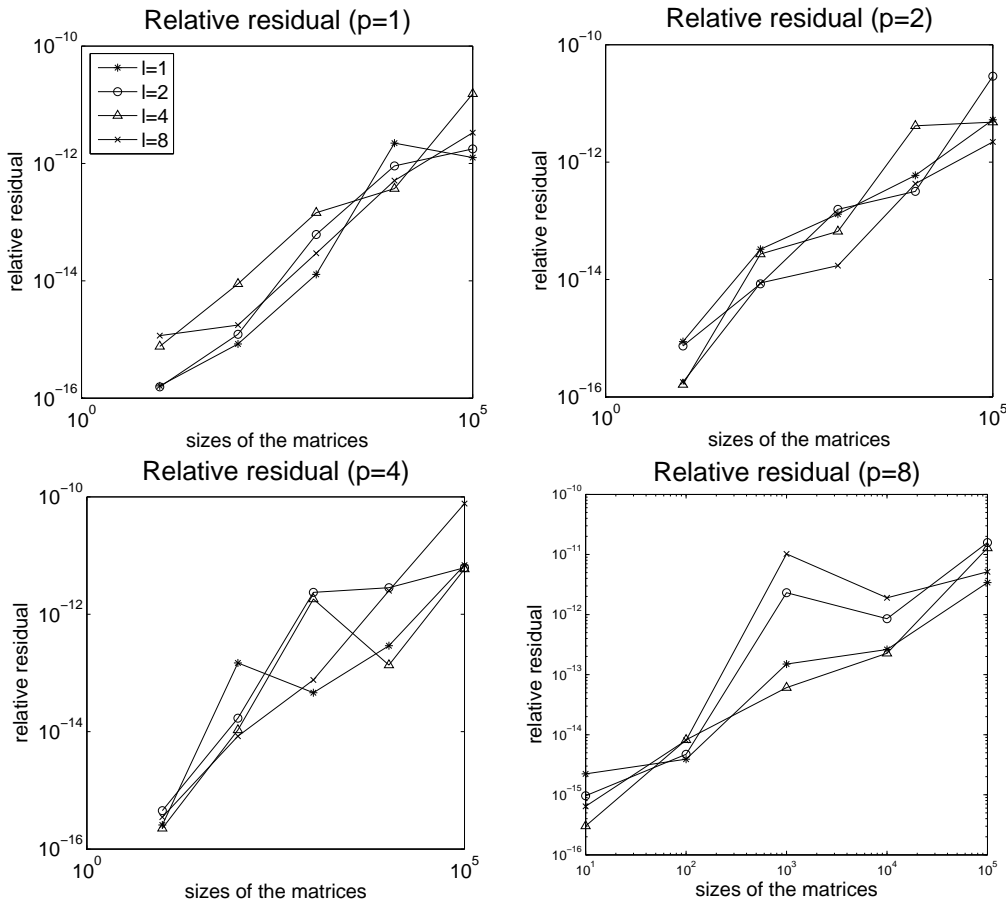


Figure 5: Accuracy graphs for the band solver

cond(A)	cond(A(1:10,1:10))	Relative Error	Relative Residual
9.8826702011e+02	7.2137658055e+06	7.9371451652e-12	1.5876136765e-13
9.8834568681e+02	7.2137666893e+07	1.2899057031e-11	8.1869391232e-13
9.8835355416e+02	7.2137668126e+08	1.3731988886e-09	1.6178975264e-11
9.8835434090e+02	7.2137678326e+09	1.7523244367e-08	1.9521650543e-10
9.8835441958e+02	7.2137522541e+10	4.9245637551e-08	7.5282322736e-10
9.8835442745e+02	7.2134203263e+11	1.0598060169e-06	1.5041387982e-08
9.8835442823e+02	7.2125761709e+12	8.4718778442e-06	1.4160532102e-07
9.8835442831e+02	7.1769836736e+13	1.0353474474e-04	1.3557724944e-06
9.8835442832e+02	7.0936256361e+14	7.6248083160e-04	1.4000530881e-05
9.8835442832e+02	5.4957030404e+15	9.4706661128e-03	1.1029369701e-04

The accuracy is therefore highly dependent on the condition number of the leading submatrices of A .

8 Conclusions

In this paper a new method based on the Levinson idea was presented for solving strongly nonsingular systems of linear equations, where the coefficient matrix is a higher order semiseparable matrix plus a band matrix. Denoting the semiseparability rank with p and the bandwidth with $2l + 1$, the overall complexity of the presented solver is $O((p + l)^2 n)$. Numerical experiments have shown that the presented approach is stable.

Future work will focus on a look-ahead method. In this way, we can also solve problems for which the matrix has one or more ill-conditioned principal leading submatrices.

References

- [1] R. Vandebril. *Semiseparable matrices and the symmetric eigenvalue problem*. PhD thesis, Dept. of Computer Science, K.U.Leuven, Celestijnenlaan 200A, 3000 Leuven, May 2004.
- [2] N. Mastronardi, M. Van Barel, and E. Van Camp. Divide and conquer algorithms for computing the eigendecomposition of symmetric diagonal-plus-semiseparable matrices. *Numerical Algorithms*, 39(4):379–398, 2005.
- [3] S. Chandrasekaran and M. Gu. A divide and conquer algorithm for the eigendecomposition of symmetric block-diagonal plus semi-separable matrices. *Numerische Mathematik*, 96(4):723–731, February 2004.
- [4] D. Fasino and L. Gemignani. Direct and inverse eigenvalue problems, for diagonal-plus-semiseparable matrices. *Numerical Algorithms*, 34:313–324, 2003.
- [5] D. A. Bini, L. Gemignani, and V. Y. Pan. *QR-like algorithms for generalized semiseparable matrices*. Technical Report 1470, Department of Mathematics, University of Pisa, 2004.
- [6] R. Vandebril, M. Van Barel, and N. Mastronardi. An implicit QR algorithm for semiseparable matrices to compute the eigendecomposition of symmetric matrices. Report TW 367, Department of Computer Science, K.U.Leuven, Leuven, Belgium, August 2003. To appear in *Numerical Linear Algebra with Applications*.
- [7] R. Vandebril, M. Van Barel, and N. Mastronardi. A QR-method for computing the singular values via semiseparable matrices. *Numerische Mathematik*, 99:163–195, October 2003.
- [8] D. Fasino. Rational Krylov matrices and QR-steps on Hermitian diagonal-plus-semiseparable matrices. To appear in *Numerical Linear Algebra and its Applications*, 2004.
- [9] L. Gemignani. A unitary Hessenberg QR-based algorithm via semiseparable matrices. url: fi.bonacci.dm.unipi.it/~gemignan/papers/unihess.pdf, 2004.
- [10] Y. Eidelman, I. C. Gohberg, and V. Olshevsky. The QR iteration method for Hermitian quasiseparable matrices of an arbitrary order. *Linear Algebra and Its Applications*, 2005. To appear.
- [11] D. Fasino, L. Gemignani, N. Mastronardi, and M. Van Barel. Orthogonal rational functions and diagonal plus semiseparable matrices. In F. T. Luk, editor, *Advanced Signal Processing Algorithms, Architectures, and Implementations XII*, volume 4791 of *Proceedings of SPIE*, pages 167–170, 2002.
- [12] D. Fasino, N. Mastronardi, and M. Van Barel. Fast and stable algorithms for reducing diagonal plus semiseparable matrices to tridiagonal and bidiagonal form. *Contemporary Mathematics*, 323:105–118, 2003.
- [13] R. Vandebril, M. Van Barel, and N. Mastronardi. An orthogonal similarity reduction of a matrix to semiseparable form. Report TW 355, Department of Computer, Science, K.U.Leuven, Leuven, Belgium, 2003. To appear in *SIAM Journal on Matrix Analysis and its Applications*.
- [14] N. Mastronardi, S. Chandrasekaran, and S. Van Huffel. Fast and stable algorithms for reducing diagonal plus semiseparable matrices to tridiagonal and bidiagonal form. *BIT*, 41(1):149–157, 2003.
- [15] R. Vandebril, M. Van Barel, G. H. Golub, and N. Mastronardi. A bibliography on semiseparable matrices. Technical Report TW412, Katholieke Universiteit Leuven, Dept. Computerwetenschappen, Celestijnenlaan 200A, 3001 Heverlee (Leuven), Belgium, December 2005.

- [16] L. Greengard and V. Rokhlin. On the numerical solution of two-point boundary value problems. *Communications on Pure and Applied Mathematics*, 44:419–452, 1991.
- [17] H. P. Jr. Starr. *On the numerical solution of one-dimensional integral and differential equations*. PhD thesis, Yale University, 1992. Research Report YALEU/DCS/RR-888.
- [18] F. R. Gantmacher and M. G. Krein. *Oscillation matrices and kernels and small vibrations of mechanical systems*. AMS Chelsea publishing, 2002.
- [19] F. A. Graybill. *Matrices with applications in statistics*. Wadsworth international group, Belmont, California, 1983.
- [20] S. Chandrasekaran and M. Gu. Fast and stable algorithms for banded plus semiseparable systems of linear equations. *SIAM Journal on Matrix Analysis and its Applications*, 25(2):373–384, 2003.
- [21] P. Dewilde and A.-J. van der Veen. *Time-varying systems and computations*. Kluwer academic publishers, Boston, June 1998.
- [22] Y. Eidelman and I. C. Gohberg. Inversion formulas and linear complexity algorithm for diagonal plus semiseparable matrices. *Computers & Mathematics with Applications*, 33(4):69–79, August 1996.
- [23] Y. Eidelman and I. C. Gohberg. A look-ahead block Schur algorithm for diagonal plus semiseparable matrices. *Computers & Mathematics with Applications*, 35(10):25–34, 1997.
- [24] Y. Eidelman and I. C. Gohberg. A modification of the Dewilde-van der Veen method for inversion of finite structured matrices. *Linear Algebra and Its Applications*, 343-344:419–450, April 2002.
- [25] D. Fasino, N. Mastronardi, and M. Van Barel. Fast and stable algorithms for reducing diagonal plus semi-separable matrices to tridiagonal and bidiagonal form. *Contemporary Mathematics*, 323:105–119, 2003.
- [26] N. Mastronardi, S. Chandrasekaran, and S. Van Huffel. Fast and stable two-way algorithm for diagonal plus semi-separable systems of linear equations. *Numerical Linear Algebra with Applications*, 8(1):7–12, 2001.
- [27] E. Van Camp, N. Mastronardi, and M. Van Barel. Two fast algorithms for solving diagonal-plus-semiseparable linear systems. *Journal of Computational and Applied Mathematics*, 164-165:731–747, 2004.
- [28] N. Mastronardi, M. Van Barel, and R. Vandebril. A Levinson-like algorithm for symmetric positive definite semiseparable plus diagonal matrices. Technical Report TW 423, Dept. Computerwetenschappen, K.U.Leuven, March 2005.
- [29] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, third edition, 1996.
- [30] T. Kailath and A. H. Sayed, editors. *Fast reliable algorithms for matrices with structure*. SIAM, Philadelphia, PA, USA, May 1999. ISBN0-89871-431-1.
- [31] N. Levinson. The Wiener RMS error criterion in filter design and prediction. *Journal of Mathematical Physics*, 25:261–278, 1947.
- [32] R. Vandebril, M. Van Barel, and N. Mastronardi. A note on the representation and definition of semiseparable matrices. *Numerical Linear Algebra with Applications*, March 2005.