

**A Levinson-like algorithm for  
symmetric positive definite  
semiseparable plus diagonal matrices**

*Nicola Mastronardi*

*Raf Vandebril*

*Marc Van Barel*

*Report TW 423, March 2005*



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

# A Levinson-like algorithm for symmetric positive definite semiseparable plus diagonal matrices

*Nicola Mastronardi*

*Raf Vandebril*

*Marc Van Barel*

*Report TW 423, March 2005*

Department of Computer Science, K.U.Leuven

## **Abstract**

In this paper a Levinson-like algorithm is derived for solving symmetric positive definite semiseparable plus diagonal systems of equations. In a first part we solve a Yule-Walker-like system of equations. Based on this  $O(n)$  solver an algorithm for a general right-hand side is derived. The new method has a linear complexity and takes  $19n - 13$  operations.

The relation between the algorithm and an upper triangular decomposition of the inverse of the semiseparable plus diagonal matrices is investigated. Numerical experiments are included.

**Keywords :** Yule-Walker, Levinson, Durbin, semiseparable plus diagonal, symmetric positive definite, upper triangular factorization of the inverse

**AMS(MOS) Classification :** Primary : 65F05,

# A Levinson-like algorithm for symmetric positive definite semiseparable plus diagonal matrices <sup>\*</sup>

Nicola Mastronardi <sup>†</sup>, Raf Vandebril <sup>‡</sup>, Marc Van Barel <sup>§</sup>

31st March 2005

## Abstract

In this paper a Levinson-like algorithm is derived for solving symmetric positive definite semiseparable plus diagonal systems of equations. In a first part we solve a Yule-Walker-like system of equations. Based on this  $O(n)$  solver an algorithm for a general right-hand side is derived. The new method has a linear complexity and takes  $19n - 13$  operations.

The relation between the algorithm and an upper triangular decomposition of the inverse of the semiseparable plus diagonal matrices is investigated. Numerical experiments are included.

**Keywords:** Yule-Walker, Levinson, Durbin, semiseparable plus diagonal, symmetric positive definite, upper triangular factorization of the inverse

## 1 Introduction

Recently different algorithms for solving systems of equations, with semiseparable plus diagonal coefficient matrices, were proposed. In some of the following publications, more general rank structured matrices were used, but the proposed algorithms are also suitable for semiseparable plus diagonal systems of equations [1, 2, 4, 5, 6, 7, 14]. The publications [2, 6] are based on the  $QR$ -decomposition for more general classes of matrices. In the papers [1, 14, 16] several methods based on the  $QR$ (or  $URV$ )-decomposition of these matrices are proposed. The complexities of the different solvers are  $54n$  ops for [16],  $58n$  ops for [6],  $59n$  ops for [1] and  $80n$  ops for [4]. To our knowledge, these methods are the fastest currently available for solving general semiseparable plus diagonal systems of equations. In this paper a new method is derived for solving positive definite symmetric semiseparable plus diagonal systems of equations. The proposed method has a complexity of  $19n - 13$ .

Linear systems of equations of semiseparable plus diagonal form arise for example in the solution of differential and integral equations [11, 15], in oscillation theory [8], statistics [10], and so on. More references can be found in [17].

The method proposed in this paper, is based on the underlying idea of the Durbin and Levinson algorithm for solving Toeplitz systems of equations. Assume we would like to solve the system  $Ax = b$  with  $A$  of dimension  $n$ . The Levinson idea is to solve  $n$  systems of equations of increasing size. At step  $k$  of

---

<sup>\*</sup>The work of the second and third author was partially supported by K.U.Leuven, project OT/00/16 (SLAP: Structured Linear Algebra Package), 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 (first author). The scientific responsibility rests with the authors

<sup>†</sup>n.mastronardi@ba.iac.cnr.it

<sup>‡</sup>raf.vandebril@cs.kuleuven.ac.be

<sup>§</sup>marc.vanbarel@cs.kuleuven.ac.be

the Levinson algorithm a system of dimension  $k \times k$  is solved with as coefficient matrix the upper left  $k \times k$  block of the matrix  $A$ , and right-hand side the first  $k$  elements of the vector  $b$ .

The Levinson algorithm for Toeplitz matrices is widespread and described for example in [9, 12, 13], it takes  $O(k)$  operations in each step of the loop. The overall complexity to solve the Toeplitz equations using Levinson is therefore  $O(n^2)$ . The Levinson-like algorithm described here for symmetric positive definite semiseparable plus diagonal matrices uses 15 operations in each step of the loop. The loop does not generate intermediate solutions like in the Toeplitz case, but it generates two vectors from which we can construct the solution at the end. The construction of the solution takes  $4n - 4$  operations.

The paper is organized as follows. In the following introductory Section 1.1, we will briefly refresh the notions Yule-Walker, Levinson and Durbin algorithm, in relation with Toeplitz matrices. In Section 2 and 3, we derive the analogue Durbin and Levinson algorithms respectively, for semiseparable plus diagonal coefficient matrices. The relation with an upper triangular factorization of the inverse of the semiseparable plus diagonal matrix is investigated in Section 4. The last section of this paper presents numerical experiments proving the  $O(n)$  complexity, and investigating the accuracy of the algorithm.

## 1.1 The Yule-Walker problem and the Durbin and Levinson algorithms

Traditionally this problem and the algorithms are used in relation with Toeplitz matrices. In our paper we will place them in a similar context related to semiseparable plus diagonal matrices. Let us briefly recapitulate the meaning of these words. This introduction is based on [9].

Suppose we have a symmetric positive definite Toeplitz matrix of order  $n$ :

$$T = \begin{pmatrix} t_0 & t_1 & \cdots & t_{n-1} \\ t_1 & t_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & t_1 \\ t_{n-1} & \cdots & t_1 & t_0 \end{pmatrix}.$$

Let us denote with  $T_k$  the principal leading  $k \times k$  submatrix of  $T$ . The Yule-Walker problem of dimension  $k$  is the following one:

$$T_k y_k = -[t_1, \dots, t_k]^T.$$

To solve these systems for  $k = 1, \dots, n$ , one can use the Durbin algorithm (see [3]).

The Levinson algorithm solves in fact a general system of equations  $Tx = b$ , with an arbitrary right-hand side. The Levinson algorithm also works inductively, increasing the dimension of the coefficient matrix in each step, using thereby the solutions of the  $k$ th order Yule-Walker system of equations (see [13]).

## 1.2 Semiseparable plus diagonal matrices

Before starting the algorithmic description of the Levinson and Durbin-like algorithms for semiseparable matrices, we will briefly define the class of matrices we will be working with. Throughout the complete paper we consider symmetric and positive definite semiseparable plus diagonal matrices.

Suppose we have three vectors  $u, v$  and  $d$  of length  $n$ . The semiseparable plus diagonal matrices<sup>1</sup> we consider here are of the following form:

$$S + D = \begin{pmatrix} u_1 v_1 & u_2 v_1 & \cdots & u_n v_1 \\ u_2 v_1 & u_2 v_2 & & u_n v_2 \\ \vdots & & \ddots & \vdots \\ u_n v_1 & \cdots & & u_n v_n \end{pmatrix} + \begin{pmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_n \end{pmatrix}.$$

<sup>1</sup>The matrices considered in [1, 4, 5, 14] have the main diagonal of the semiseparable matrix set to zero. The Levinson algorithm, developed in this paper, costs  $n$  ops less if applied to this class of matrices, because the computation of the main diagonal of the semiseparable part is avoided.

We remark that the class of semiseparable matrices considered here, is less general than the class of semiseparable matrices as defined in [18]. In fact this is the class of generator representable semiseparable matrices. But because every symmetric semiseparable matrix as defined in [18], can be seen as a block diagonal matrix with the block diagonal matrices of the generator representable form (see [18]), there is no loss in generality here. In this paper we solve systems, and solving a system with a block diagonal matrix reduces in a natural way to the solution of the separate blocks. Therefore, we will continue working with the class of generator representable semiseparable matrices as defined above. Let us call the vectors  $u$  and  $v$  the generators of the matrix  $S$  and the vector  $d$  contains the diagonal elements of the matrix  $D$ .

## 2 A Yule-Walker-like problem for semiseparable plus diagonal matrices

In this first section of the paper, we will solve a Yule-Walker-like problem for semiseparable plus diagonal matrices.

The Yule-Walker-like equation we would like to solve is the following one:

$$(S + D)y = -v,$$

where  $S$  is a semiseparable matrix generated by  $u$  and  $v$ , and the diagonal elements of  $D$  are stored in the vector  $d$ . We will solve this problem in an analogous way as for the Toeplitz case [9]. In the first subsection we will develop, completely similar, an  $O(n^2)$  method. A slight adaptation in the second subsection will lead to an  $O(n)$  solver.

### 2.1 The method

We solve the system inductively. Let us denote with  $S_k$  the upper left  $k \times k$  submatrix of  $S$ , a similar notation is used for  $D$ .

Let us assume by induction that we have solved the  $k$ th order Yule-Walker problem:

$$(S_k + D_k)y_k = -r_k,$$

where  $r_k = [v_1, v_2, \dots, v_k]$ . The  $(k + 1)$ th order Yule-Walker-like system of equations:

$$\begin{aligned} (S_{k+1} + D_{k+1})y_{k+1} &= -r_{k+1}, \\ \left( \begin{array}{c|c} S_k + D_k & u_{k+1}r_k \\ \hline u_{k+1}r_k^T & u_{k+1}v_{k+1} + d_{k+1} \end{array} \right) \begin{pmatrix} z_{k+1} \\ \alpha_{k+1} \end{pmatrix} &= - \begin{pmatrix} r_k \\ v_{k+1} \end{pmatrix}, \end{aligned} \quad (1)$$

will be solved in  $O(k)$  flops. Using equation (1), we observe that

$$(S_k + D_k) z_{k+1} + u_{k+1} r_k \alpha_{k+1} = -r_k \quad (2)$$

and secondly

$$u_{k+1} r_k^T z_{k+1} + \alpha_{k+1} (u_{k+1} v_{k+1} + d_{k+1}) = -v_{k+1}. \quad (3)$$

This gives us for equation (2):

$$\begin{aligned} z_{k+1} &= (1 + u_{k+1} \alpha_{k+1})(S_k + D_k)^{-1}(-r_k) \\ &= (1 + u_{k+1} \alpha_{k+1})y_k. \end{aligned} \quad (4)$$

Substituting the latter equation in (3) leads to:

$$u_{k+1} r_k^T (1 + u_{k+1} \alpha_{k+1}) y_k + \alpha_{k+1} (u_{k+1} v_{k+1} + d_{k+1}) = -v_{k+1},$$

which can be rewritten towards  $\alpha_{k+1}$  as:

$$\alpha_{k+1} = \frac{(-1)(v_{k+1} + u_{k+1} r_k^T y_k)}{(u_{k+1}^2 r_k^T y_k + u_{k+1} v_{k+1} + d_{k+1})}. \quad (5)$$

Using  $\alpha_{k+1}$  in equation (4), we can compute the vector  $z$ .

The denominator in (5) is always different from zero, because our initial matrix  $S + D$  is positive definite. As  $S + D$  is positive definite, also all the leading principal submatrices are positive definite, hence also  $S_{k+1} + D_{k+1}$ . This means that  $w^T(S_{k+1} + D_{k+1})w > 0$  for every  $w$  different from zero. Taking  $w^T = [u_{k+1}y_k^T, 1]$  gives us the following relations:

$$\begin{aligned} w^T (S_{k+1} + D_{k+1}) w &= [u_{k+1} y_k^T, 1] (S_{k+1} + D_{k+1}) \begin{pmatrix} u_{k+1} y_k^T \\ 1 \end{pmatrix} \\ &= [u_{k+1} y_k^T, 1] \left( \begin{array}{c|c} S_k + D_k & u_{k+1} r_k \\ \hline u_{k+1} r_k^T & u_{k+1} v_{k+1} + d_{k+1} \end{array} \right) \begin{pmatrix} u_{k+1} y_k^T \\ 1 \end{pmatrix} \\ &= u_{k+1}^2 y_k^T (S_k + D_k) y_k + u_{k+1}^2 y_k^T r_k + u_{k+1}^2 r_k^T y_k + u_{k+1} v_{k+1} + d_{k+1} > 0. \end{aligned}$$

Using now the relation  $y_k^T (S_k + D_k) y_k = -y_k^T r_k$  in the equation above, we get that

$$u_{k+1}^2 r_k^T y_k + u_{k+1} v_{k+1} + d_{k+1} > 0,$$

and hence the formula for calculating  $\alpha_{k+1}$  is well defined.

Clearly the computation of the inner product  $r_k^T y_k$  at each step and the computation of  $z_k$  via the multiplication of a vector with a scalar, leads to an order  $k$  complexity for solving the  $(k + 1)$ th order Yule-Walker-like type of equation. Overall this leads to an order  $O(n^2)$  solver.

## 2.2 How to achieve an $O(n)$ complexity

Two bottlenecks arise when solving the Yule-Walker equation of order  $(k + 1)$ . These are: the computation of  $r_k^T y_k$  and the computation of  $z_k$  in each step. We will show now how to change the algorithm from the previous section in order to come to a step with a fixed number of flops.

Suppose we just solved the  $(k + 1)$ th Yule-Walker system of equations and we start now solving the  $(k + 2)$ nd system. Let us denote the solution and the right-hand side of the  $(k + 1)$ th system as follows:  $(S_{k+1} + D_{k+1})y_{k+1} = -r_{k+1}$ . For the new  $(k + 2)$ nd system we need the inner product  $r_{k+1}^T y_{k+1}$ . This corresponds to the inner product (using hereby the notation from Section 2.1):

$$r_{k+1}^T y_{k+1} = [r_k^T, v_{k+1}] \begin{pmatrix} z_{k+1} \\ \alpha_{k+1} \end{pmatrix}.$$

Rewriting the equation above, using the expression for  $z_{k+1} = (1 + u_{k+1}\alpha_{k+1})y_k$ , leads to the following recursive formula for the inner product:

$$r_{k+1}^T y_{k+1} = (1 + u_{k+1}\alpha_{k+1})r_k^T y_k + v_{k+1}\alpha_{k+1}.$$

It is clear that the inner product  $r_{k+1}^T y_{k+1}$  for solving the Yule-Walker equation of order  $(k + 2)$ , can already be computed at the end of step  $(k + 1)$  in a constant number of operations if  $r_k^T y_k$  is known. Incorporating this inductive construction of the inner product in the algorithm, we remove the  $O(k)$  flops in each step connected to the calculation of the inner product.

The only remaining slow step is the computation of the vector  $z_k$ , and hence  $y_k$ , in each step of the algorithm. Let us denote the solution of the  $k$ th Yule-Walker-like type of equations with  $y_k^T = [z_k^T \ \alpha_k]$ . In fact, we calculate consecutively the following solutions  $y_i$ , (where the  $f_i$ 's denote different factors  $f_i = (1 + u_i\alpha_i)$ ):

$$\begin{aligned} y_1 &= \alpha_1 \\ y_2^T &= [z_2, \alpha_2] = [f_2 y_1, \alpha_2] = [f_2 \alpha_1, \alpha_2] \\ y_3^T &= [z_3, \alpha_3] = [f_3 y_2^T, \alpha_3] = [f_3 f_2 \alpha_1, f_3 \alpha_2, \alpha_3] \\ &\vdots \\ y_n^T &= [z_n, \alpha_n] = [f_n y_{n-1}^T, \alpha_n] = [f_n f_{n-1} \dots f_2 \alpha_1, f_n \dots f_3 \alpha_2, \dots, \alpha_n] \end{aligned}$$

Instead of calculating now in each step the vector  $y_i$  and  $z_i$ , we will store the factors  $f_i$  and the numbers  $\alpha_i$ . Storing only these two numbers at each step of the algorithm, we achieve a constant complexity for solving a  $k$ th order Yule-Walker-like type of equations. The only remaining problem is the computation in order  $O(n)$  of the solution vector  $y_n$ . This is done rather easily by starting the construction of the vector  $y_n$  from bottom to top, and updating in each step a certain factor. This is a rather straightforward computation, and therefore not included, the complete algorithm is given in the next subsection.

### 2.3 The algorithm

Two separate loops are needed for solving the Yule-Walker problem. One loop “solves” the  $k$ th order Yule-Walker type of equations. We say “solves” because in fact the solution is not calculated, but only the factors determining the solution are calculated. A second loop will construct the solution, given the two sequences of numbers constructed by the first loop.

Given the generators  $u, v$  and the diagonal  $d$  of the symmetric positive definite semiseparable matrix  $S + D$ , the following algorithm computes the solution  $y$  such that  $(S + D)y = -v$ . Two extra vectors are needed for the algorithm. In the vector  $a$  we will store the separate values of the parameter  $\alpha$ , and in the vector  $f$ , the consecutive factors  $f_i$ .

```
%---- First Part: Generate the two vectors ----
% Initialization, the system for i=1 is solved
a(1)=-v(1)/(d(1)+u(1)*v(1));
f(1)=1;
% Calculate the inner product for the next loop
ip=a(1)*v(1);
% The main loop
for i=2:n
    t=u(i)*ip+v(i);
    % Store the new calculated alpha
    a(i)=-t/(u(i)*t+d(i));
    % Store the new factor f
    f(i)=1+u(i)*a(i);
    % Calculation of the inner product for the next loop
    ip=f(i)*ip+v(i)*a(i);
end

%---- Second Part: Construct the solution-----
% The vectors f and a are given,
% construct now the solution with these vectors.
% Initialization before the loop
y(n)=a(n);
factor=f(n);
% Construct the solution vector from bottom to top
for i=n-1:-1:1
    y(i)=factor*a(i);
    factor=factor*f(i);
end
```

So the overall cost is  $12n - 8$  operations<sup>2</sup> for solving the Yule-Walker-like system of equations for positive definite symmetric semiseparable plus diagonal matrices.

<sup>2</sup>We did not count the changing of a sign as an operation.

### 3 A Levinson-type algorithm for semiseparable plus diagonal matrices.

Based on the results of the previous section, a Levinson-like solver for solving semiseparable plus diagonal systems of equations, will be constructed here. The system we would like to solve in a similar way as above is the following one:

$$(S + D)x = b,$$

where  $b$  is an arbitrary right-hand side.

#### 3.1 The method

Let us assume that we know the solutions of the  $(k + 1)$ th order Yule-Walker-like type of equations. Use the same notations as in Section 2.1. Moreover assume that we also know the solution of the system:

$$(S_k + D_k)x_k = d_k,$$

where  $d_k^T = [b_1, \dots, b_k]$ .

The system we would like to solve now in  $O(k)$  operations (in the next subsection we reduce this cost to a constant amount of flops), is the following one:

$$\begin{aligned} (S_{k+1} + D_{k+1})x_{k+1} &= d_{k+1} \\ \left( \begin{array}{c|c} S_k + D_k & u_{k+1}r_k \\ \hline u_{k+1}r_k^T & u_{k+1}v_{k+1} + d_{k+1} \end{array} \right) \begin{pmatrix} w_{k+1} \\ \mu_{k+1} \end{pmatrix} &= \begin{pmatrix} d_k \\ b_{k+1} \end{pmatrix}. \end{aligned} \quad (6)$$

The idea is identical to the one in the previous section. Expanding formula (6) leads to the following two equations

$$(S_k + D_k)w_{k+1} + \mu_{k+1}u_{k+1}r_k = d_k \quad (7)$$

and

$$u_{k+1}r_k^T w_{k+1} + \mu_{k+1}(u_{k+1}v_{k+1} + d_{k+1}) = b_{k+1}. \quad (8)$$

Equation (7) can be rewritten towards  $w_{k+1}$ :

$$w_{k+1} = x_k + \mu_{k+1}u_{k+1}y_k.$$

Substituting the equation for  $w_{k+1}$  into equation (8) and rewriting this equation leads to the following expression for  $\mu_{k+1}$ :

$$\mu_{k+1} = \frac{b_{k+1} - u_{k+1}r_k^T x_k}{(u_{k+1}^2 r_k^T y_k + u_{k+1}v_{k+1} + d_{k+1})}.$$

The denominator is always positive, see Section 2. Again one can clearly see that there are two computations in every step which will lead to a complexity of  $O(k)$ , namely the computation of the vector  $w_{k+1}$  and the computation of the inner product  $r_k^T x_k$ . But again we can simplify the calculation of the inner product, and we can postpone the calculation of the solution vector up to the very end.

#### 3.2 Reduction of the complexity

We will clarify how we can reduce the complexity in the previous step, to come to a step which takes a constant amount of arithmetic work.

Firstly we will derive a recursion for calculating the inner product  $r_k^T x_k$ . Suppose we just solved the  $(k + 1)$ th Levinson system of equations, and we start now solving the  $(k + 2)$ nd system. Let us denote the solution and the right-hand side of the  $(k + 1)$ th system as follows:  $(S_{k+1} + D_{k+1})x_{k+1} = d_{k+1}$ , with  $r_{k+1}^T = [r_k^T, v_{k+1}]$ . For the new  $(k + 2)$ nd system we need the inner product  $r_{k+1}^T x_{k+1}$ . This corresponds to the inner product:

$$r_{k+1}^T x_{k+1} = [r_k^T, v_{k+1}] \begin{pmatrix} w_{k+1} \\ \mu_{k+1} \end{pmatrix}.$$

Similarly as in the previous section we can rewrite this in terms of the previous inner product  $r_k^T x_k$  plus the inner product  $r_k^T y_k$ .

$$r_{k+1}^T x_{k+1}^T = r_k^T x_k + \mu_{k+1} (u_{k+1} r_k^T y_k + v_{k+1})$$

Using this recursive formula, already decreases the amount of work in one step.

Instead of constructing the intermediate solutions of all the  $k$ th order Levinson equations, one can construct the overall solution in  $O(n)$  operations at the end of the algorithm. This construction is similar as the one described for solving the Yule-Walker type equations. We will briefly illustrate it. Let us denote with  $x_k$  the solution of the  $k$ th order Levinson equation, similarly we denote with  $\mu_k$  the last component of  $x_k$ . The  $g_i$ 's denote factors. We illustrate the computation for a matrix of dimension  $4 \times 4$ . We get the following equations with  $g_i = \mu_i u_i$ :

$$\begin{aligned} x_1 &= \mu_1 \\ x_2^T &= [x_1 + g_2 y_1, \mu_2] \\ x_3^T &= [x_2^T + g_3 y_2^T, \mu_3] \\ x_4^T &= [x_3^T + g_4 y_3^T, \mu_4]. \end{aligned}$$

Expanding now everything towards vector  $x_4$ , gives us the following vector:

$$x_4 = \begin{pmatrix} \mu_1 + (g_2 + f_2(g_3 + f_3 g_4))\alpha_1 \\ \mu_2 + (g_3 + f_3 g_4)\alpha_2 \\ \mu_3 + g_4 \alpha_3 \\ \mu_4 \end{pmatrix}.$$

The latter equation gives a clear clue on how to compute the solution in linear time. This means that for a fast construction of the solution we need to store 4 vectors, namely 2 vectors which store the values of  $\alpha$  and  $\mu$ , and two more vectors for storing the values of the factors  $f_i$  and  $g_i$ .

### 3.3 The algorithm

Let us denote with  $u$  and  $v$  the generators of the semiseparable matrix. The vector  $d$  contains the diagonal elements and  $b$  is an arbitrary right-hand side. The following algorithm solves the following system of equations  $(S + D)x = b$ , using thereby the Yule-Walker-like results as developed in the previous section. We have to store now 4 extra vectors for recalculating the solution, namely  $a$  and  $\mu$  for storing the values of  $\alpha$  and  $\mu$  in each step and  $f$  and  $g$  for storing the values of  $f_i$  and  $g_i$  in each step.

```
%---- First part: construction of the vectors -----
% initialization for the different vectors
denominator=d(1)+u(1)*v(1);
a(1)=-v(1)/denominator;
mu(1)=b(1)/denominator;
f(1)=1;
g(1)=1;
% initialization for the different inner products
ip=a(1)*v(1); % the inner product r^T y
ipl=mu(1)*v(1); % the inner product r^T x
% The main loop
for i=2:n
    % initialize
    t=u(i)*ip+v(i);
    denominator=(u(i)*t+d(i));
    % Calculation of alpha
    a(i)=-t/denominator;
    % Calculation of the factor f
    f(i)=(1+u(i)*a(i));
```

```

% The calculation of mu
mu(i)=(b(i)-u(i)*ipl)/denominator;
% The factor g
g(i)=mu(i)*u(i);
% the inner product r^T x
ipl=ipl+mu(i)*t;
% the inner product r^T y
ip=a(i)*t+ip;
end

%---- Second part: construction of the solution ----
% we construct now the solution with the 4 vectors.
x(n)=mu(n);
factor=g(n);
for i=n-1:-1:1
    x(i)=mu(i)+factor*a(i);
    factor=g(i)+f(i)*factor;
end

```

An easy calculation reveals that the complexity of this solver is  $19n - 13$  operations.

## 4 An upper triangular factorization of the inverse

Similarly to the Toeplitz case, the Durbin algorithm presented here is closely related to an upper triangular factorization of the inverse of the semiseparable plus diagonal matrix.

Let us denote the solution of the  $k$ th order Yule-Walker type equation as

$$(S_k + D_k)y_k = -r_k. \quad (9)$$

This equation can be rewritten towards the matrix  $S_{k+1} + D_{k+1}$ , by bringing the factor  $r_k$  to the other side. The system (9) is similar to

$$(S_{k+1} + D_{k+1}) \begin{pmatrix} y_k \\ 1/u_{k+1} \end{pmatrix} = \left( \begin{array}{c|c} S_k + D_k & u_{k+1}r_k \\ \hline u_{k+1}r_k^T & u_{k+1}v_{k+1} + d_{k+1} \end{array} \right) \begin{pmatrix} y_k \\ 1/u_{k+1} \end{pmatrix} = \begin{pmatrix} 0 \\ \sigma_{k+1} \end{pmatrix}.$$

If we put the successive equations above in an upper triangular matrix we get (let us denote with  $y_{k,j}$  the successive components in the vector  $y_k$ ):

$$(S_n + D_n) \begin{pmatrix} 1/u_1 & y_{1,1} & y_{2,1} & \cdots & y_{n-1,1} \\ 0 & 1/u_2 & y_{2,2} & \cdots & y_{n-1,2} \\ 0 & 0 & 1/u_3 & & y_{n-1,3} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 1/u_n \end{pmatrix} = \begin{pmatrix} \sigma_1 & 0 & 0 & \cdots & 0 \\ \times & \sigma_2 & 0 & \cdots & 0 \\ \vdots & \times & \sigma_3 & & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \times & \times & \cdots & \times & \sigma_n \end{pmatrix}.$$

The  $\times$  denote arbitrary elements in the matrix. Rewriting the equations above and defining the upper triangular matrix to the right of  $(S_n + D_n)$  as  $U$  and the right-hand side as  $L$ , we get the following equations ( $\hat{L}$  is another lower triangular matrix):

$$\begin{aligned} (S+D)U &= L \\ (S+D)U &= U^{-*}\hat{L} \\ U^*(S+D)U &= \hat{L}. \end{aligned}$$

Because  $\hat{L}$  is lower triangular, and because of symmetry, the matrix  $\hat{L}$  is a diagonal matrix  $\Lambda$ . Inverting the equation above gives:

$$\begin{aligned} U^{-1}(S+D)^{-1}U^{-*} &= \Lambda^{-1} \\ (S+D)^{-1} &= U\Lambda^{-1}U^* \end{aligned}$$

which gives us a triangular factorization of the inverse of the semiseparable plus diagonal matrix.

## 5 Some general remarks

The results as presented in this paper are formulated for positive definite symmetric semiseparable plus diagonal matrices. The results can be extended towards the nonsymmetric case, similarly as in [9]. The implementations as given here, are based on the generator representation. This representation is suitable here for solving systems of equations. If one however wants to use this upper triangular decomposition as presented here for an iterative algorithm (e.g. for computing the spectrum), one should be aware of numerical instabilities that can occur. More information on possible problems with this representation can be found in [18].

## 6 Numerical experiments

The software can be downloaded from <http://www.cs.kuleuven.ac.be/~marc/software>. The experiments are performed using Matlab<sup>3</sup> 7, on a linux platform.

In a first experiment, we used the matlab commands tic and toc, to check the time needed to compute a solution of the system using the Levinson solver. We performed 15 experiments, every experiment repeated 40 times, for sizes of matrices  $10^4 : 10^5 : 10^6$ . The line drawn, represents the average of all the timings. One can clearly see that the timings are linear.

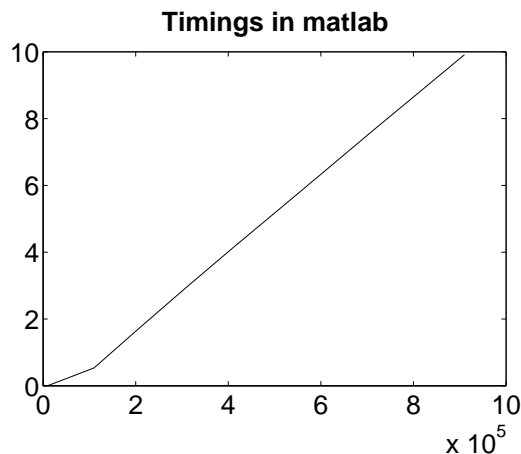


Figure 1: Timings of the Levinson solver

In a second experiment, we generated positive definite semiseparable plus diagonal matrices of sizes  $10^4 : 10^5 : 10^6$ , for every dimension we performed 5 experiments. We started from a known solution  $x$ , calculated the right-hand side  $b = Ax$ , and then computed the solution  $\tilde{x}$  of the system  $A\tilde{x} = b$ . In the left figure the norm of  $\|\tilde{x} - x\|_2$  is plotted for every experiment. In the right figure the residual  $\|A\tilde{x} - b\|_2 / \|b\|_2$  is plotted. The line represents the average error over the 5 different experiments.

## Conclusions

In this paper we developed a solver for symmetric positive definite semiseparable systems of equations based on the Levinson idea for solving Toeplitz matrices. The connection with the upper triangular factorization of the inverse of the semiseparable plus diagonal matrix was given, and numerical experiments illustrating the accuracy and speed of the algorithm were included.

<sup>3</sup>Matlab is a registered trademark of the MathWorks Inc.

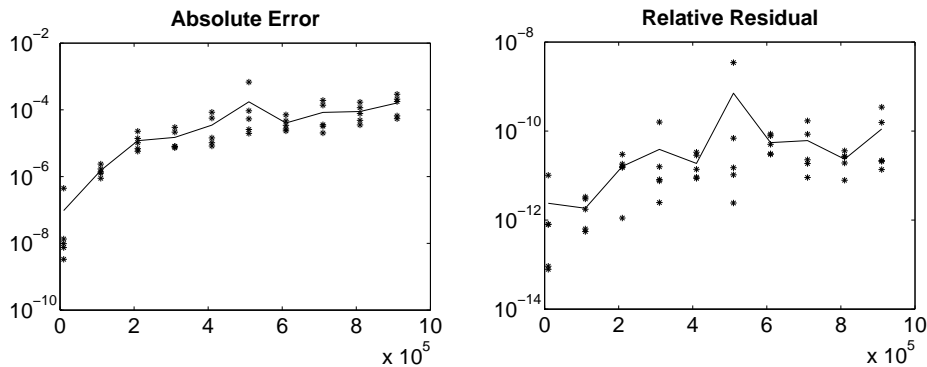


Figure 2: Accuracy of the Levinson solver

## References

- [1] 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.
- [2] P. Dewilde and A.-J. van der Veen. *Time-varying systems and computations*. Kluwer academic publishers, Boston, June 1998.
- [3] J. Durbin. The fitting of time series in models. *Review of the International Statistical Institute*, 28:233–243, 1960.
- [4] 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.
- [5] 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.
- [6] 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.
- [7] 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.
- [8] F. R. Gantmacher and M. G. Krein. *Oscillation matrices and kernels and small vibrations of mechanical systems*. AMS Chelsea publishing, 2002.
- [9] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, third edition, 1996.
- [10] F. A. Graybill. *Matrices with applications in statistics*. Wadsworth international group, Belmont, California, 1983.
- [11] 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.
- [12] T. Kailath and A. H. Sayed, editors. *Fast reliable algorithms for matrices with structure*. SIAM, Philadelphia, PA, USA, May 1999. ISBN0-89871-431-1.
- [13] N. Levinson. The Wiener RMS error criterion in filter desing and prediction. *Journal of Mathematical Physics*, 25:261–278, 1947.

- [14] 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.
- [15] 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.
- [16] 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.
- [17] R. Vandebril. *Semiseparable matrices and the symmetric eigenvalue problem*. PhD thesis, Dept. of Computer Science, K.U.Leuven, Celestijnenlaan 200A, 3000 Leuven, May 2004.
- [18] R. Vandebril, M. Van Barel, and N. Mastronardi. A note on the representation and definition of semiseparable matrices. To appear in *Numerical Linear Algebra with Applications* (Report TW393), May 2004.