A general Levinson-like solver Framework

Raf Vandebril, Marc Van Barel & Nicola Mastronardi

Department of Computer Science
K.U.Leuven

May 2006

Outline

Introduction

(p_1, p_2)-Levinson conform matrices

The algorithm

Examples

Conclusions, Future work

Introduction
System solving
Levinson for Toeplitz matrices
Examples

(p_1, p_2)-Levinson conform matrices

The algorithm

Examples

Conclusions, Future work

System solving

- We would like to solve (strongly nonsingular) systems:

$$A\mathbf{x} = \mathbf{b},$$

  by induction.
- If A_k is the upper left $k \times k$ block of A, and we know

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

  with $\mathbf{b}_k = [b_1, b_2, \ldots, b_k]^T$.
- Then by induction we use this to solve:

$$A_{k+1}\mathbf{x}_{k+1} = \mathbf{b}_{k+1}.$$

- Which leads to the solution of $A_n\ \mathbf{x} = \mathbf{b}_n$.

Toeplitz matrices

Suppose we have a symmetric Toeplitz matrix T:

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

we would like to solve:

$$T\mathbf{x} = \mathbf{b}.$$

Block partitioning

Suppose we know $T_3\mathbf{x}_3 = \mathbf{b}_3$ (or $\mathbf{x}_3 = T_3^{-1}\mathbf{b}_3$) and we want to solve now the 4th-Levinson system:

$$T_4\mathbf{x}_4 = \mathbf{b}_4$$

$$\left[\begin{array}{ccc|c} t_0 & t_1 & t_2 & t_3 \\ t_1 & t_0 & t_1 & t_2 \\ t_2 & t_1 & t_0 & t_1 \\ \hline t_3 & t_2 & t_1 & t_0 \end{array}\right] \left[\begin{array}{c} w_1 \\ w_2 \\ w_3 \\ \hline \mu_4 \end{array}\right] = \left[\begin{array}{c} b_1 \\ b_2 \\ b_3 \\ \hline b_4 \end{array}\right]$$

$$\left[\begin{array}{c|c} T_3 & E_3\mathbf{t}_3 \\ \hline \mathbf{t}_3^T E_3 & t_0 \end{array}\right] \left[\begin{array}{c} \mathbf{w}_4 \\ \mu_4 \end{array}\right] = \left[\begin{array}{c} \mathbf{b}_3 \\ b_4 \end{array}\right].$$

Block partitioning

And we have

$$E_3\mathbf{t}_3 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} = \begin{bmatrix} t_3 \\ t_2 \\ t_1 \end{bmatrix}.$$

The matrix E_3 is the counteridentity matrix and flips the vector upside down.

Two equations

This system

$$\left[\begin{array}{c|c} T_3 & E_3\mathbf{t}_3 \\ \hline \mathbf{t}_3^T E_3 & t_0 \end{array}\right] \left[\begin{array}{c} \mathbf{w}_4 \\ \mu_4 \end{array}\right] = \left[\begin{array}{c} \mathbf{b}_3 \\ b_4 \end{array}\right].$$

leads to two equations:

$$T_3\mathbf{w}_4 + E_3\mathbf{t}_3\mu_4 = \mathbf{b}_3,$$
$$\mathbf{t}_3^T E_3\mathbf{w}_4 + t_0\mu_4 = b_4.$$

the first equation ($T_3^{-1}*$) gives us (using $T_3^{-1}E_3 = E_3T_3^{-1}$):

$$\mathbf{w}_4 + E_3T_3^{-1}\mathbf{t}_3\mu_4 = T_3^{-1}\mathbf{b}_3 = \mathbf{x}_3.$$

We know that $T_3^{-1}\mathbf{b}_3 = \mathbf{x}_3$, but we do not know $-T_3^{-1}\mathbf{t}_3 = \mathbf{y}_3$, otherwise we could easily compute μ_4.

Two equations

Suppose we had $\mathbf{y}_3$, then we could deduce the following formulas:

$$\begin{aligned} \mu_4 &= \frac{(b_4 - \mathbf{t}_3 E_3 \mathbf{x}_3)}{\left(1 + \mathbf{t}_3^T \mathbf{y}_3\right)}, \\ \mathbf{w}_4 &= \mathbf{x}_4 + \mu_4 E_3 \mathbf{t}_3^T. \end{aligned}$$

But: we don't yet know

$$-T_3^{-1}\mathbf{t}_3 = \mathbf{y}_3.$$

Yule-Walker equations

Let us try the same approach.
Suppose we know

$$T_k \mathbf{y}_k = -[t_1, \ldots, t_k]^T = -\mathbf{t}_k,$$

can we then easily compute $\mathbf{y}_{k+1}$:

$$\begin{aligned} T_{k+1}\mathbf{y}_{k+1} &= -\mathbf{t}_{k+1}, \\ \left[\begin{array}{c|c} T_k & E_k\mathbf{t}_k \\ \hline \mathbf{t}_k^T E_k & t_0 \end{array}\right] \left[\begin{array}{c} \mathbf{z}_{k+1} \\ \alpha_{k+1} \end{array}\right] &= -\left[\begin{array}{c} \mathbf{t}_k \\ t_{k+1} \end{array}\right]. \end{aligned}$$

Yule-Walker equations

So the equation

$$\left[\begin{array}{c|c} T_k & E_k\mathbf{t}_k \\ \hline \mathbf{t}_k^T E_k & t_0 \end{array}\right] \left[\begin{array}{c} \mathbf{z}_{k+1} \\ \hline \alpha_{k+1} \end{array}\right] = -\left[\begin{array}{c} \mathbf{t}_k \\ \hline t_{k+1} \end{array}\right]$$

leads in a similar way to:

$$\begin{aligned} \mathbf{z}_{k+1} + E_k T_k^{-1}\mathbf{t}_k\alpha_{k+1} &= -T_k^{-1}\mathbf{t}_k, \\ \mathbf{t}_k^T E_k\mathbf{z}_{k+1} + t_0\alpha_{k+1} &= -t_{k+1}. \end{aligned}$$

Now we can compute $\mathbf{y}_{k+1}$ and hence also $\mathbf{x}_{k+1}$ leading to a solution for $\mathbf{x}$.

Levinson algorithm

The algorithm consists of two steps (for every k):

- Compute the solution of

$$T_k \mathbf{y}_k = -\mathbf{t}_k.$$

- Use this solution for computing:

$$T_k \mathbf{x}_k = \mathbf{b}_k$$

Why does it work?

Because the columns are highly related to each other!

$$T_{k+1}\mathbf{y}_{k+1} = -\mathbf{t}_{k+1},$$

$$\left[\begin{array}{c|c} T_k & E_k\mathbf{t}_k \\ \hline \mathbf{t}_k^T E_k & t_0 \end{array}\right] \begin{bmatrix} \mathbf{z}_{k+1} \\ \alpha_{k+1} \end{bmatrix} = -\begin{bmatrix} \mathbf{t}_k \\ t_{k+1} \end{bmatrix}.$$

And because

$$E_k T_k^{-1} = T_k^{-1} E_k.$$

Why does it work?

Because the columns are highly related to each other!

$$\left[\begin{array}{cccc|c} t_0 & t_1 & t_2 & t_3 \\ t_1 & t_0 & t_1 & t_2 \\ t_2 & t_1 & t_0 & t_1 \\ \hline t_3 & t_2 & t_1 & t_0 \end{array}\right]$$

Namely:

$$\begin{bmatrix} t_3 \\ t_2 \\ t_1 \end{bmatrix} \rightarrow \begin{bmatrix} t_4 \\ t_3 \\ t_2 \\ t_1 \end{bmatrix} \rightarrow \begin{bmatrix} t_5 \\ t_4 \\ t_3 \\ t_2 \\ t_1 \end{bmatrix}$$

Why does it work?

But due the equation:

$$\left[\begin{array}{c|c} T_k & E_k\mathbf{t}_k \\ \hline \mathbf{t}_k^T E_k & t_0 \end{array}\right] \begin{bmatrix} \mathbf{z}_{k+1} \\ \alpha_{k+1} \end{bmatrix} = -\begin{bmatrix} \mathbf{t}_k \\ t_{k+1} \end{bmatrix}$$

we can only add at the bottom:

$$\begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} \rightarrow \begin{bmatrix} t_1 \\ t_2 \\ t_3 \\ t_4 \end{bmatrix} \rightarrow \begin{bmatrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \end{bmatrix}$$

That is why we use the matrices E_k to flip these vectors upside down.

Why does it work?

Attention because:

- The multiplication with E_k was performed on the left.
- This caused some problems due to $T_k^{-1} E_k \mathbf{t}_k$.
- These problems were solved by the equation:

$$T_k^{-1} E_k = E_k T_k^{-1}.$$

- An operation on the right of the column would not have caused this problem.

Examples

Used to solve systems of equations with

- Toeplitz matrices;
- Hankel matrices.

References:

- G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, third edition, 1996.
- T. Kailath and A. H. Sayed, editors. *Fast reliable algorithms for matrices with structure*. SIAM, Philadelphia, PA, USA, May 1999. ISBN 0-89871-431-1.

Introduction

(p_1, p_2)-Levinson conform matrices

- What matrices do we search for
- p_1-Levinson conform
- (p_1, p_2)-Levinson conform
- Goal
- Example

The algorithm

Examples

Conclusions, Future work

What matrices do we search for

We search for matrices A, such that the successive columns satisfy:

- Column K_4 (up to the diagonal, not including the diagonal) satisfies:

$$K_4 = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} \mathbf{c}_4 = R_3 \mathbf{c}_4.$$

- Column K_5 satisfies:

$$K_5 = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ \hline \xi_4 \end{bmatrix} \mathbf{c}_5 = \begin{bmatrix} R_3 \\ \hline \xi_4 \end{bmatrix} \mathbf{c}_5 = R_4 \mathbf{c}_5.$$

Example

Directly applying this constraint onto a matrix gives us.

$$K_2 = \begin{bmatrix} v_1 \end{bmatrix} u_2; \quad K_3 = \begin{bmatrix} v_1 \\ \hline v_2 \end{bmatrix} u_3; \quad K_4 = \begin{bmatrix} v_1 \\ v_2 \\ \hline v_3 \end{bmatrix} u_4.$$

This leads to a matrix:

$$\begin{bmatrix} \times & v_1u_2 & v_1u_3 & v_1u_4 \\ & \times & v_2u_3 & v_2u_4 \\ & & \times & v_3u_4 \\ & & & \times \end{bmatrix}$$

This is a matrix having the strictly upper triangular part of semiseparable form (i.e. all blocks are of rank 1.).

Adding a multiplier

We search for matrices A, such that the successive columns satisfy (with an extra multiplier):

- Column K_4 (up to the diagonal) satisfies:

$$K_4 = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} \mathbf{c}_4 = R_3 \mathbf{c}_4$$

- Column K_5 satisfies (with P_3 a matrix):

$$K_5 = \begin{bmatrix} R_3 P_3 \\ \hline \xi_4 \end{bmatrix} \mathbf{c}_5 = R_4 \mathbf{c}_5$$

- Multiplication with P_k should be easy.

Example

For example we get (with $P_k = 0$):

$$K_2 = \begin{bmatrix} 1 \end{bmatrix} c_1 = [c_1]$$

$$K_3 = \begin{bmatrix} \frac{1 * 0}{1} \end{bmatrix} c_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} c_2 = \begin{bmatrix} 0 \\ c_2 \end{bmatrix}$$

$$K_4 = \begin{bmatrix} \frac{\begin{bmatrix} 0 \\ 1 \end{bmatrix} * 0}{1} \end{bmatrix} c_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} c_3 = \begin{bmatrix} 0 \\ 0 \\ c_3 \end{bmatrix}.$$

Example

This leads to a matrix:

$$\begin{bmatrix} \times & c_1 & 0 & 0 \\ & \times & c_2 & 0 \\ & & \times & c_3 \\ & & & \times \end{bmatrix}$$

This is a tridiagonal matrix.

Generalizing to matrices

Generalizing a little more:

- Column K_{k+1} satisfies:

$$K_{k+1} = R_k \mathbf{c}_{k+1}^T.$$

- Column K_{k+2} satisfies:

$$K_{k+2} = R_{k+1} \mathbf{c}_{k+2}^T,$$

with

$$R_{k+1} = \left[\frac{R_k P_k}{\xi_{k+1}} \right],$$

where $\mathbf{c}_{k+1} \in \mathbb{R}^{1 \times p_1}$, $R_k \in \mathbb{R}^{k \times p_1}$, $\xi_{k+1} \in \mathbb{R}^{1 \times p_1}$, and $P_k \in \mathbb{R}^{p_1 \times p_1}$

Example

For example we get $p_1 = 2$ and P_k is the 2×2 shift left operator:

$$K_2 = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ c_1 \end{bmatrix} = [c_1]$$

$$K_3 = \left[\frac{[0 \;\; 1] \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}}{[0 \;\; 1]} \right] \begin{bmatrix} d_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} d_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} d_1 \\ c_2 \end{bmatrix}$$

$$K_4 = \left[\frac{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}}{[0 \;\; 1]} \right] \begin{bmatrix} d_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} d_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 0 \\ d_2 \\ c_3 \end{bmatrix}$$

Example

This leads to a matrix:

$$\begin{bmatrix} \times & c_1 & d_1 & & & \\ & \times & c_2 & d_2 & & \\ & & \times & c_3 & d_3 & \\ & & & \times & c_4 & d_4 \\ & & & & \times & c_5 \\ & & & & & \times \end{bmatrix}$$

This is a band matrix.

p_1-Levinson conform (slide 1)

Definition (p_1-Levinson conform matrices)

Given a matrix A, with A_k the upper left $k \times k$ block of A.
A is said to be p_1-Levinson conform if

- For every $0 \leq k \leq n$, there is a splitting in blocks of the matrix A_{k+1} of the following form:

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

p_1-Levinson conform (slide 2)

Thus:

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

whith

$$\begin{aligned} c_{k+1} &\in \mathbb{R}^{1\times p_1}, \\ d_{k+1} &\in \mathbb{R}^{1\times p_2}, \\ R_k &\in \mathbb{R}^{k\times p_1}, \\ F_k, E_k &\in \mathbb{R}^{k\times k} \\ S_k &\in \mathbb{R}^{k\times p_2}, \\ A_k &\in \mathbb{R}^{k\times k}. \end{aligned}$$

We will work without E_k and F_k.

p_1-Levinson conform (slide 3)

So, A is said to be p_1-Levinson conform if

- The following relation for the R_k is satisfied

$$R_{k+1} = \left[\frac{R_k P_k}{\xi_{k+1}} \right],$$

where

$$\begin{aligned} P_k &\in \mathbb{R}^{p_1 \times p_1}, \\ \xi_k &\in \mathbb{R}^{1 \times p_1}. \end{aligned}$$

The multiplication of P_k with a vector has linear complexity.

(p_1, p_2)-Levinson conform (slide 1)

Definition ((p_1, p_2)-Levinson conform matrices)

Given a matrix A, with
A is said to be (p_1, p_2)-Levinson conform if

- A is p_1-Levinson conform.
- The following relation for the S_k needs to be satisfied:

$$S_{k+1} = \left[\frac{S_k Q_k}{\eta_{k+1}} \right],$$

whith $Q_k \in \mathbb{R}^{p_2 \times p_2}$ and $\eta_k \in \mathbb{R}^{1 \times p_2}$. And the multiplication of a vector and Q_k has linear complexity.

Goal

To develop for strongly nonsingular (p_1, p_2)-Levinson conform matrices.
A solver of complexity:

$$O(p_1 p_2 n)$$

Example

Theorem

Every matrix is at least (n, n)-Levinson conform.

Proof.

The proof is straightforward. Define the $k \times n$ matrices R_k and S_k as follows:

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

One can easily calculate $\mathbf{d}_k$ and $\mathbf{c}_k$ such that the conditions of the Definition are satisfied. □

Introduction

(p_1, p_2)-Levinson conform matrices

The algorithm

- The Yule-Walker-like system
- The Levinson-like system
- Algorithm 1
- Reduction of the complexity
- Algorithm 2

Examples

Conclusions, Future work

The Yule-Walker-like system (slide 1)

Matrix A is p_1-Levinson conform.
We will solve $AY_n = -R_n$, by induction.
Assume we know Y_k (with $Y_k \in \mathbb{R}^{k\times p}$) such that:

$$A_k Y_k = -R_k.$$

We will solve $A_{k+1}Y_{k+1} = -R_{k+1}$:

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

The Yule-Walker-like system (slide 2)

Expanding the equation

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

we observe that

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

and

$$\mathbf{d}_{k+1} \, S_k^T Z_{k+1} + a_{k+1,k+1} \alpha_{k+1} = -\xi_{k+1}. \tag{2}$$

The Yule-Walker-like system (slide 3)

Rewriting Equation (1) to Z_{k+1} and using $A_k Y_k = -R_k$:

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

Substituting the latter equation in Equation (2) leads to:

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

from which we can extract α_{k+1} as:

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

This gives us an inductive algorithm. (The denominator is always well-defined due to the strongly nonsingularity of A.)

The Yule-Walker-like system (slide 4)

Solving this system we get the following recursive formulas:

$$Z_{k+1} = Y_k \left(P_k + \mathbf{c}_{k+1}^T \alpha_{k+1} \right),$$

with

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

The Levinson-like system (slide 1)

Use the Yule-Walker-like solution to solve the general case. Assume we know the solutions:

$$\begin{aligned} A_k Y_k &= R_k \\ A_k \mathbf{x}_k &= \mathbf{r}_k \end{aligned}$$

whith $\mathbf{r}_k^T = [b_1, \ldots, b_k]$.

We will now solve:

$$A_{k+1}\ \mathbf{x}_{k+1} = \mathbf{r}_{k+1}$$

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

The Levinson-like system (slide 2)

Expanding

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

gives us

$$A_k \; w_{k+1} + \mu_{k+1} \; R_k \mathbf{c}_{k+1}^T = \mathbf{r}_k$$

and

$$\mathbf{d}_{k+1} \; S_k^T \; w_{k+1} + \mu_{k+1} \; a_{k+1,k+1} = b_{k+1}.$$

The Levinson-like system (slide 3)

Leading to (in a similar way as in the previous slides)

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

and

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

This gives us an inductive algorithm for computing the solution.

Algorithm 1 (slide 1)

Initialization (solution of the upper left 1 by 1 system).

$$Y_1 = \alpha_1 = \frac{1}{a_{1,1}}(-R_1)$$

$$x_1 = \mu_1 = \frac{1}{a_{1,1}} b_1$$

Algorithm 1 (slide 2)

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

1. Compute and store the following variables:

1.1 $S_k^T Y_k,\ S_k^T x_k$	$p_1 p_2 (2k-1), p_2(2k-1)$
1.2 $\mathbf{d}_{k+1}(S_k^T Y_k)$	$p_1(2p_2-1)$
1.3 $(\mathbf{d}_{k+1} S_k^T Y_k \mathbf{c}_{k+1}^T + a_{k+1,k+1})$	$2p_1$

2. $\alpha_{k+1} = \dfrac{(-1)(\xi_{k+1} + \mathbf{d}_{k+1} S_k^T Y_k P)}{(\mathbf{d}_{k+1} S_k^T Y_k \mathbf{c}_{k+1}^T + a_{k+1,k+1})}$ $\quad 1 + p_1 + p_1(2p_1 - 1)$
3. $Z_{k+1} = Y_k \left(P + \mathbf{c}_{k+1}^T\ \alpha_{k+1} \right)$ $\quad kp_1(2p_1-1) + k(2p_1-1) + 2p_1 k$
4. $\mu_{k+1} = \dfrac{b_{k+1} - \mathbf{d}_{k+1} S_k^T x_k}{(\mathbf{d}_{k+1} S_k^T Y_k \mathbf{c}_{k+1}^T + a_{k+1,k+1})}$ $\quad 1 + 2p_2$
5. $w_{k+1} = x_k + \mu_{k+1}\ Y_k \mathbf{c}_{k+1}^T$ $\quad 2k + k(2p_1 - 1)$
6. $Y_k^T = [Z_k^T, \alpha_k^T]$ $\quad 0$
7. $x_k^T = [w_k^T, \mu_k^T]$ $\quad 0$

endfor;

Algorithm 1 (slide 3)

Performing an overall complexity count leads to:

$$O(p_1^2 n^2) + O(p_1 p_2 n^2)$$

The bottlenecks are the computations involving the

$$S_k^T Y_k, S_k^T x_k$$

and the explicit formation of the variables

$$Z_{k+1} \text{ and } w_{k+1}.$$

Reduction of the complexity (slide 1)

Postpone the computation of x to the end (hence no need to compute Z_k or w_k anymore).
Store in the loop: the factors μ_k and α_k for every k. We get the following recursive formula for x (using its components):

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

The computation of the solution is of complexity: $O(p_1 n)$.

Reduction of the complexity (slide 2)

For (p_1, p_2)-Levinson conform matrices, we can easily rewrite the computations of

$$S_k^T Y_k, S_k^T x_k$$

recursively.
We have:

$$S_{k+1} = \begin{bmatrix} S_k Q_k \\ \eta_{k+1} \end{bmatrix} \quad R_{k+1} = \left[\begin{array}{c} R_k P_k \\ \hline \xi_{k+1} \end{array}\right]$$

Using these relations we can recursively compute the products $S_k^T x_k$ and $S_k^T Y_k$.

Reduction of the complexity (slide 3)

Let us illustrate $S_k^T Y_k$.

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

This leads to a recursive calculation of the inner-product and is independent of k.
Similar one can derive now a recursive formula for computing the products $S_k^T x_k$.

Algorithm 2 (slide 1)

The general algorithm for (p_1, p_2)-Levinson conform matrices:
For $k = 1, \ldots, n-1$ (with the same initializations):

1. Compute and store the following variables:
   - 1.1 $Q^T S_{k-1}^T Y_{k-1} c_k^T + \eta_k^T$ $\quad 2p_2p_1 + O(p_1)$
   - 1.2 $S_k^T Y_k$ $\quad p_2 O(p_1) + O(p_2) + 2p_1p_2$
   - 1.3 $S_k^T x_k$ $\quad O(p_2) + 2p_2$
   - 1.4 $(\mathbf{d}_{k+1} S_k^T Y_k \mathbf{c}_{k+1}^T + a_{k+1,k+1})$ $\quad (2p_2 - 1)p_1 + (2p_1 - 1) + 1$
2. $\alpha_{k+1} = \frac{(-1)(\xi_{k+1} + \mathbf{d}_{k+1} S_k^T Y_k P)}{(\mathbf{d}_{k+1} S_k^T Y_k \mathbf{c}_{k+1}^T + a_{k+1,k+1})}$ $\quad 1 + p_1 + O(p_1) + p_1(2p_2 - 1)$
3. $\mu_{k+1} = \frac{b_{k+1} - \mathbf{d}_{k+1} S_k^T x_k}{(\mathbf{d}_{k+1} S_k^T Y_k \mathbf{c}_{k+1}^T + a_{k+1,k+1})}$ $\quad 2p_2 + 1$

endfor;

Algorithm 2 (slide 1)

Computation of the solution. h is a dummy variable.

$$\begin{aligned} x_n &= \mu_n \\ h &= \mu_n \mathbf{c}^T \end{aligned}$$

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

1. $x_k = \mu_k + \alpha_k h$ $\qquad 2p_1$
2. $h = x_i \mathbf{c}_i^T + Ph$ $\qquad O(p_1)$

endfor.

This gives us the desired $O(p_1 p_2 n)$ complexity.

Introduction

(p_1, p_2)-Levinson conform matrices

The algorithm

Examples

- Semiseparable matrices
- Band matrices
- Arrowhead matrices
- Uppertriangular matrices
- Unsymmetric structures
- Summations

Semiseparable matrices (slide 1)

Suppose we have a higher order semiseparable matrix A,

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

for which $\mathbf{u}_i$ and $\mathbf{v}_i$ are row vectors of length l_1
$\mathbf{p}_i$ and $\mathbf{q}_i$ are of length l_2.
This matrix is (l_2, l_1)-Levinson conform and hence has therefore a $O(l_2 l_1 n)$ Levinson-like solver.

Semiseparable matrices (slide 2)

Let us define R_k and S_k as follows:

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

Then we have the following relations:

$$R_{k+1} = \begin{bmatrix} R_k \\ \hline \mathbf{q}_{k+1} \end{bmatrix} \qquad S_{k+1} = \begin{bmatrix} S_k \\ \hline \mathbf{v}_{k+1} \end{bmatrix}.$$

Semiseparable matrices (slide 3)

The upper left $(k+1) \times (k+1)$ block of A equals:

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

Therefore we can develop the desired $O(l_1 l_2 n)$ algorithm.
More precisely: $6 l_1 l_2 n$ plus lower order terms.

Band matrices (slide 1)

Suppose we have a band matrix A:

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

Band matrices (slide 2)

Let us denote with $\bar{\mathbf{a}}_i$ the following vectors:

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

The row vectors $\bar{\mathbf{a}}_i$ are of length l ($\bar{\mathbf{a}}_0 = 0$). (Similarly we define the vectors $\underline{\mathbf{a}}_i$.)
Let us define R_k as a $k \times l$ matrix of the following form:

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

Band matrices (slide 3)

Let us define the shift operator P as the $l \times l$ matrix:

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

Then we have:

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

Band matrices (slide 3)

Defining $\mathbf{c}_k = \bar{\mathbf{a}}_k$ and define S_k and $\mathbf{d}_k$ similarly as for the upper triangular part. This leads to the following relations:

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

This means that our band matrix is (l,k)-Levinson conform. This leads to and $O(lkn)$ method. More precisely $6lkn$ plus lower order terms.

Arrowhead matrices (slide 1)

Let us consider the nonsymmetric arrowhead matrix:

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

Arrowhead matrices (slide 1)

Let us define the matrices R_k and S_k as $S_k = R_k = [1, 0, \ldots, 0]$ which is a vector of length k.
Let us define the elements $\mathbf{c}_k = \bar{a}_k$ and $\mathbf{d}_k = \underline{a}_k$.
Then one can easily see that the conditions for solving this system with a Levinson-like algorithm are satisfied.
This means that our matrix is $(1, 1)$-Levinson conform, and hence we have a $O(n)$ solver.

Uppertriangular matrices

Let us solve now an upper triangular system of equations: $A = (a_{ij})$ whith $a_{ij} = 0$ if $i > j$. Let us denote with R_k the matrix where $R_k = [I_k, 0]$ and $\mathbf{c}_k = [a_{1,k}, \ldots, a_{k-1,k}, 0, \ldots, 0]$. The matrix $S_k = 0$.
Then the matrix is $(n, 1)$-Levinson conform. This leads to an $O(n^2)$ method.
Solving the system in this way leads to the well-known backward substitution!

Unsymmetric structures

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

- An upper triangular semiseparable matrix.
- An upper triangular band matrix.
- A matrix for which the upper triangular part is semiseparable, and the lower triangular part is coming from a band matrix.
- An upper triangular band matrix, for which the lower triangular part comes from an arrowhead.

Summations of Levinson conform matrices (slide 1)

Theorem

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

For example:

- One can solve now the sum of higher order nonsymmetric semiseparable matrices plus nonsymmetrix band matrices.
- We can solve now easily arrowhead matrices with a larger band-width by writing them as the sum of an arrowhead and a band matrix.
- Important the terms in the summation can be singular, as long as the summation is strongly nongsinular it is ok.

Summations of Levinson conform matrices (slide 2)

For example

$$A = \begin{bmatrix} \times & & & p_4q_1 & p_5q_1 \\ \times & \times & & p_4q_2 & p_5q_2 \\ & \times & \times & & \\ & & \times & \times & \\ & & & \times & \times \end{bmatrix}$$

The matrix A can be written as a lower bidiagonal matrix plus a matrix of rank 1, having only the upper right block different from zero.

Errors in structures

- A matrix which has one row or column not satisfying the structure.
- A matrix having one (sub) superdiagonal not satisfying the structure.
- For large matrices, few random unstructured distortions.

Introduction

(p_1, p_2)-Levinson conform matrices

The algorithm

Examples

Conclusions, Future work

Conclusions and current work

- For (p_1, p_2)-Levinson conform matrices we developped an $O(p_1 p_2 n)$ solver.
- Incorporation of the Levinson methods like the Toeplitz, Vandermonde Hankel.
- Combinations with $E_k \neq 0$ and $F_k \neq 0$.
- Block version and look ahead.

References

- 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.
- N. Mastronardi, M. Van Barel, and R. Vandebril. A levinson-like algorithm for symmetric strongly nonsingular higher order semiseparable plus band matrices. Technical report, Dept. Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3001 Heverlee (Leuven), Belgium, April 2005.