UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

AD-A238 977 (2)

REPORT DOCUMENTA

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY NA	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for Public Release; Distribution Unlimited
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE NA	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) STANFORD UNIVERSITY	5. MONITORING ORGANIZATION REPORT NUMBER(S) AFOSR-TR- 90 0987
6a. NAME OF PERFORMING ORGANIZATION STANFORD UNIVERSITY 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION AFOSR/NM
6c. ADDRESS (City, State, and ZIP Code) Department of Electrical Engr. Durand 117 Stanford, CA 94305	7b. ADDRESS (City, State, and ZIP Code) Bldg. 410 Bolling AFB, DC 20332-6448
8a. NAME OF FUNDING / SPONSORING ORGANIZATION AFOSR 8b. OFFICE SYMBOL (If applicable) NM	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER AFOSR 88-0327A (Contract or Grant Number)
8c. ADDRESS (City, State, and ZIP Code) Bldg. 410 Bolling AFB, DC 20332-6448	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO.: 61102F PROJECT NO.: 2304 TASK NO.: A6 WORK UNIT ACCESSION NO.:

11. TITLE (Include Security Classification)
Fast Array Algorithms for Structured Matrices

12. PERSONAL AUTHOR(S)
Joohwan Chun

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Thesis	FROM 1989 TO 1990	1989, June	149

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Schur complements, displacement structure, Toeplitz, Hankel, and Vandermonde matrices

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

See Back

91-05959

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL [illegible]	22b. TELEPHONE (Include Area Code) 202/ 767-4939 22c. OFFICE SYMBOL AFOSR/NM

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

Abstract

Many engineering or mathematical problems require to factorize structured matrices (Toeplitz, Hankel, Vandermonde, products of such matrices and their inverses, Schur complements, etc) either in explicit or in disguised form. Consequently there exist various analytic tools regarding structured matrices as well as several fast factorization algorithms. In this thesis, we show that many of these results and several significant generalizations can be obtained in a very constructive way. The generic form is to use elementary circular and hyperbolic transformations to triangularize a certain array of numbers derived from the displacement representation of the given structured matrix; the desired results can then be read off from the resulting array. These "fast array algorithms" require $O(mn)$ operations for LU and QR factorizations of $m \times n$ structured matrices, and $O(mn)$ or even $O(n \log^2 n)$ operations for solving matrix equations. Also the array form suggests various alternative algorithms, depending upon the order in which the transformations are applied; these variations can have different numerical properties and lead to different implementations.

Our algorithm is based on a generalized definition of displacement for block-Toeplitz (Hankel) and Toeplitz (Hankel)-block matrices slightly extending the previous definitions of Kailath, Kung and Morf (1979) and Lev-Ari and Kailath (1984). An important property of displacement structure is that it is preserved under Schur complementations. It will turn out that Toeplitz-(Hankel)-derived (near-Toeplitz, Toeplitz-like, etc) matrices are perhaps best regarded as particular Schur complements obtained from suitably defined block matrices. The displacement structure is used to obtain a generalized Schur algorithm for the fast triangular and orthogonal factorizations of all such matrices, and well structured fast solutions of the corresponding exact and overdetermined systems of linear equations.

Accession For
Distribution/
Availability Codes
Avail and/or
Dist Special
A-1

FAST ARRAY ALGORITHMS FOR STRUCTURED MATRICES

JOOHWAN CHUN

STANFORD UNIVERSITY

FAST ARRAY ALGORITHMS FOR STRUCTURED MATRICES

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

By
Joohwan Chun
June 1989

© Copyright by Joohwan Chun 1989

All Rights Reserved

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Thomas Kailath

Prof. Thomas Kailath (Principal Advisor)

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Gene F. Franklin

Prof. Gene Franklin

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Allen M. Peterson

Prof. Allen Peterson

Approved for the University Committee
on Graduate Studies:

Dean of Graduate Studies

Abstract

Many engineering or mathematical problems require to factorize structured matrices (Toeplitz, Hankel, Vandermonde, products of such matrices and their inverses, Schur complements, etc) either in explicit or in disguised form. Consequently there exist various analytic tools regarding structured matrices as well as several fast factorization algorithms. In this thesis, we show that many of these results and several significant generalizations can be obtained in a very constructive way. The generic form is to use elementary circular and hyperbolic transformations to triangularize a certain array of numbers derived from the displacement representation of the given structured matrix; the desired results can then be read off from the resulting array. These "fast array algorithms" require $O(mn)$ operations for LU and QR factorizations of $m \times n$ structured matrices, and $O(mn)$ or even $O(n \log^2 n)$ operations for solving matrix equations. Also the array form suggests various alternative algorithms, depending upon the order in which the transformations are applied; these variations can have different numerical properties and lead to different implementations.

Our algorithm is based on a generalized definition of displacement for block-Toeplitz (Hankel) and Toeplitz (Hankel)-block matrices slightly extending the previous definitions of Kailath, Kung and Morf (1979) and Lev-Ari and Kailath (1984). An important property of displacement structure is that it is preserved under Schur complementations. It will turn out that Toeplitz-(Hankel)-derived (near-Toeplitz, Toeplitz-like, etc) matrices are perhaps best

regarded as particular Schur complements obtained from suitably defined block matrices. The displacement structure is used to obtain a generalized Schur algorithm for the fast triangular and orthogonal factorizations of all such matrices, and well structured fast solutions of the corresponding exact and overdetermined systems of linear equations.

Acknowledgments

I would like to express my deepest gratitude to Professor Thomas Kailath for providing me with the opportunity to pursue my studies at Stanford University. His open-mindedness, vast knowledge, interest and above all his patience and consistent support during several years at Stanford have been vital to the completion of this dissertation. This dissertation has evolved from the process of answering and re-answering his short but incisive questions.

Professor Gene Franklin was kind enough to agree to serve as the associate advisor, and helped me with applications of fast algorithms. I am also grateful to Professor Allen Peterson for serving as the chairman of my Ph.D. orals committee and for his useful comments on the dissertation. Thanks are also due to Professor John Cioffi who served on my committee.

This dissertation owes much in kind to the encouragement and support that I have received from Dr. Hanoch Lev-Ari. He was always available and ready to help with any questions. I would also like to take this opportunity to thank Barbara Mckee for her help and friendship.

I would like to thank to Vwani Roychowdhury for invaluable friendship and discussions on wide range of topics. Danny Abramovitch has been one of my best friends since we took the quals together. Reuven Ackner was always willing to give me good advice. Conversation with Juan Jover was always pleasant. Frequent visits of Dirk Slock to my office at night helped me keep awake. Richard Roy shared with me his vast experience with extended

Kalman filters. I also would remember the great companionship with Tie-Jun Shan, King Pang, Alon Orlitsky, Glen Dudevoir, Neil Jablon, Mike Workman, Kevin Fisher, Seyi Farotimi, Sanjay Kasturia, Jim Aslanis, Miki Genossar, Xiaoming Li, Marc Goldberg and Ragu Balakrishnan.

Although their efforts do not directly impact this dissertation, I bear a debt to my past teachers, Wangsan Chu, Prof. Youngduk Kim, Prof. Walter Ku, Prof. Frank Luk and Prof. Myunghwan Kim. Their efforts have provided me with a solid intellectual framework and a valuable set of skills.

Finally, and most importantly, to my parents who have sacrificed a lot for my education and without whose constant encouragement I would not be where I stand today, I dedicate this dissertation.

This work was supported in part by the National Science Foundation under Grant ECS-850246, by the Air Force Office of Scientific Research, Air Force Systems Command under Contract AF83-0228, the U.S. Army Research Office, under Contract DAAL03-86-K-0045, and by the SDIO/IST, managed by the Army Research Office under Contract DAAL03-87-K-0033.

Table of Contents

1. Introduction 1

   1.1 Prototype Examples 3
   1.2 Some Examples of Structured Matrices 9
   1.3 Outline of the Thesis 18
   References 19

2. Generalized Displacement Structure for Block-Toeplitz, Toeplitz-block and Toeplitz-derived Matrices 25

   2.1 Introduction 26
   2.2 Displacement of Matrices 28
   2.3 Generalized Schur Algorithm and Partial Triangularization 31
   2.4 Applications to non-Toeplitz Matrices 36
   2.5 Construction of Proper Generators 39
   2.6 Concluding Remarks 43
   A.1 Appendix 1 43
   A.2 Appendix 2 45
   A.3 Appendix 3 46
   A.4 Appendix 4 48
   References 51

3. Generalization of Gohberg-Semencul formulas 54

   3.1 Introduction 55
   3.2 A Constructive Proof of two Gohberg-Semencul formulas 58
   3.3 Generalized Displacement Representations and Schur Complements 64
   3.4 Some Generalized Gohberg-Semencul Formulas 70
   3.5 Concluding Remarks 73
   A.1 Appendix 75
   References 75

4. Displacement Structure for Hankel, Vandermonde and Related (Derived) Matrices 78

4.1 Introduction 79
4.2 Some General Properties of Displacement Operators 81
4.3 Fast Partial Triangular Factorization using Hankel Generators 86
4.4 Fast Triangularization of Vandermonde Matrices 92
4.5 Fast Triangularization of close-to-Hankel Matrices 97
4.6 Fast QR Factorization of Vandermonde Matrices 103
4.7 Concluding Remarks 104
References 105

5. Divide-and-Conquer Solutions for Least Squares Problems 108

5.1 Introduction 109
5.2 Generalized Schur Algorithm 113
5.3 Divide-and-Conquer Implementation 119
5.4 Polynomial Products with Fast Convolutions 127
5.5 Concluding Remarks 130
A.1 Appendix 131
References 133

6. Concluding Remarks 136

6.1 Summary of Results 136
6.2 Some Known and Unknown Numerical Properties of the fast Algorithms 139
6.3 Other Open Research Problems 145
A . Appendix 146
References 148

List of Illustrations

5-1 Sequence of Computations for Example 4 121
5-2 Sequence of Computations for Example 5 122
6-1 Generating Toeplitz Matrix given Reflection Coefficients 147

Chapter 1.

Introduction.

Fast algorithms for factorizing *structured matrices* that include Toeplitz, Hankel and Vandermonde matrices have a long history. The earliest known fast algorithm is probably the Euclidean algorithm, which has recently been recognized as providing a fast factorization of Hankel matrices [14]. Factorizations of Hankel matrices also underlie the criteria and the fast algorithms (due to Hermite (1856), Hurwitz (1895) and Routh (1875)) for checking the root distributions of a polynomial with respect to imaginary axis (See [45] for recent generalizations). More recently, in the context of decoding BCH codes Berlekamp and Massey [5], [48] gave a fast algorithm that factorizes the inverse of a Hankel matrix (See also [9], [14], [41], [54]).

Fast algorithms for Toeplitz matrices have an even richer history [33-34]. Caratheodory (1911) and Toeplitz (1911) showed that the *positive-realness* of certain functions is equivalent to the positive-definiteness of certain Toeplitz matrices [1]. Later, Schur (1917) gave a fast algorithm that checks the positive-realness, and in fact, also factorizes *close-to-Toeplitz* matrices [33-34], [44], [59]. The Schur algorithm has also appeared in different contexts notably in seismic deconvolution problems as the so-called "layer-peeling" method [11], [35], [57], in orthogonal filter synthesis [17], [56] and in checking the root location of a polynomial with respect to the unit circle [33-34]. Bareiss [4] also rediscovered the Schur algorithm as a

fast method of solving Toeplitz systems of equations.

There is another class of fast algorithms that factorize the *inverse* of Toeplitz matrices. They include the recursions of the Szego orthogonal polynomials [62] and the Levinson algorithm [46]. As closely related results, there are the Gohberg-Semencul formulas [21-23] and the Trench recursion [63].

In 1972, Kailath [29] [31] developed fast algorithms for Kalman filters associated with continuous-time constant parameter state-space models. These algorithms replaced the nonlinear Riccati differential equations of the Kalman filter with another set of nonlinear equations that he dubbed the Chandrasekhar equations because equations of somewhat the same form had been developed by Chandrasekhar and Ambarzumian for solving certain Wiener-Hopf integral equations encountered in radiative transfer theory [12], [60]. The discrete-time versions of these results were developed by Kailath, Morf and Sidhu (see [37], [38]). Various extensions were made jointly by them along with Ljung and Friedlander, and nice interpretations were found in terms of scattering theory. In the course of this work, it became clear that there were close relations between these state-space results and the Levinson and Schur algorithms for solving Toeplitz equations and factoring Toeplitz matrices (see the review paper [30], [32], which contains many references). As noted therein, Kailath *et. al.* found that the key concept enabling the different fast algorithms was what they called DISPLACEMENT STRUCTURE. This is in many ways a natural generalization of Toeplitz structure; for example, the inverse of a Toeplitz matrix is not in general Toeplitz, but all matrices and their inverses have the same displacement rank. Structured matrices (Toeplitz, Hankel, Vandermonde, products of such matrices and their inverses, Schur complements with respect to various entries, etc) all have low displacement rank. The complexity of numerical computations with structured matrices depends upon their displacement rank. The concept of displacement rank has been developed, extended and applied in many ways by Kailath and his students and colleagues (Morf, Sidhu,

Dickinson, Ljung, Kung, Friedlander, Verghese, Vieira, Levy, Lee, Lev-Ari, Delosme, Porat, Cioffi, Bruckstein, Citron, Bistritz, Rao, Dewilde, Dym, DePrettere, Pal) - see the review paper [34].

As the reader may have anticipated from the above discussion, the various results mentioned therein have been developed and presented using a variety of algebraic and analytic tools. The main contribution of this thesis is to show that the above results, and several significant generalizations, can be obtained in a very constructive (or algorithmic) way. (At least, we have so far shown this for many of the earlier results; with more work, one might anticipate being able to replace "many" by "all". - see the remarks in the last chapter).

The generic form is to use elementary circular and hyperbolic transformations to triangularize a certain array of numbers derived from the displacement representation of the given structured matrix; the desired results can then be read off from the resulting array. This new array form suggests various alternative algorithms, depending upon the order in which the transformations are applied; these variations can have different numerical properties and lead to different implementations.

The basic ideas can be seen from the simple examples in the next section. However, it may be noted here that such array form algorithms were introduced into least-squares problems independently by Golub [24] and by Dyer and McReynolds [18], and further developed by Bierman [6], among others. Generalizations for Riccati and Chandrasekhar recursions were introduced under the name *square-root algorithms* by Morf and Kailath [50].

1. Prototype Examples.

In this Chapter, we shall explain the basic idea of the fast algorithms for structured matrices. To provide some motivation, we shall also briefly introduce several problems that involve structured matrices.

Simultaneous Factorization of T and T^{-1}.

Let T be a Toeplitz matrix

$$T = \begin{bmatrix} c_0 & c_1 & \cdot & c_{n-1} \\ c_1 & c_0 & \cdot & c_{n-2} \\ \cdot & \cdot & \cdot & \cdot \\ c_{n-1} & c_{n-2} & \cdot & c_0 \end{bmatrix}, \qquad c_0 = 1.$$

Then it is easy to check that T can be expressed in the so-called *displacement form* [33-36],

$$T = L(\mathbf{x}_1)L^T(\mathbf{x}_1) - L(\mathbf{x}_2)L^T(\mathbf{x}_2), \tag{1}$$

where $L(\mathbf{x})$ denotes the lower-triangular Toeplitz matrix with the first column $\mathbf{x}$, and

$$\mathbf{x}_1 = [1, c_1, c_2, \cdots, c_{n-1}]^T, \quad \mathbf{x}_2 = [0, c_1, c_2, \cdots, c_{n-1}]^T.$$

Now we form a *pre-array*

$$\Delta = \begin{bmatrix} L(\mathbf{x}_1) & L(\mathbf{x}_2) \\ I & I \end{bmatrix}, \tag{2}$$

and post-multiply Δ with any *J-orthogonal matrix* Θ, viz., one that satisfies

$$\Theta J \Theta^T = J, \quad J \equiv \begin{bmatrix} I_n & O \\ O & -I_n \end{bmatrix},$$

that will yield a triangular *post-array*

$$\Delta\Theta = \begin{bmatrix} L_1 & O \\ U & L_2 \end{bmatrix} \equiv \tilde{\Delta}, \quad \text{say.} \tag{3}$$

Then it turns out that

$$T = L_1 L_1^T, \qquad T^{-1} = UU^T = L_2 L_2^T. \tag{4}$$

The proof is very simple. We just compare entries in the identity

$$\Delta J \Delta^T = \Delta\Theta J \Theta^T \Delta^T = \tilde{\Delta} J \tilde{\Delta}^T.$$

From (2) and (3), we have

$$\Delta J \Delta^T = \begin{bmatrix} L(\mathbf{x}_1)L^T(\mathbf{x}_1) - L(\mathbf{x}_2)L^T(\mathbf{x}_2) & L(\mathbf{x}_1) - L(\mathbf{x}_2) \\ L^T(\mathbf{x}_1) - L^T(\mathbf{x}_2) & O \end{bmatrix}, \tag{5a}$$

$$\tilde{\Delta} J \tilde{\Delta}^T = \begin{bmatrix} L_1 L_1^T & L_1 U^T \\ U L_1^T & UU^T - L_2 L_2^T \end{bmatrix}. \tag{5b}$$

Now equating corresponding entries gives

$$
\begin{aligned}
L_1 L_1^T &= L(\mathbf{x}_1)L^T(\mathbf{x}_1) - L(\mathbf{x}_2)L^T(\mathbf{x}_2) \\
UL_1^T &= L^T(\mathbf{x}_1) - L^T(\mathbf{x}_2) = I \\
UU^T &- L_2 L_2^T = 0.
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
T &= L_1 L_1^T, \\
T^{-1} &= L_1^{-T} L_1^{-1} = UU^T = L_2 L_2^T.
\end{aligned}
$$

Remark. Determining the AR parameters of a random process requires solving a special Toeplitz systems of equations called the *Yule-Walker equation* (or Normal equation). The Yule-Walker equation has a special right-side vector, *viz.*, the last column of the Toeplitz matrix shifted by one position. One can easily prove that the solution of Yule-Walker equation is the normalized last column of the upper triangular matrix U, where $T^{-1} = UU^T$. Similarly, decoding BCH codes requires solving the Hankel system of equations whose right-side vector is the last column of the Hankel matrix shifted by one position. The factorization of the inverse of the Hankel matrix also gives the solution for such equations.

We still need to show how to find such a matrix Θ. This can be done in many ways. One is as a sequence of *hyperbolic rotations*,

$$
H_{ij}(k) = \frac{1}{(1-k^2)^{1/2}} \begin{bmatrix} 1 & & & \\ & 1 & -k & \\ & -k & 1 & \\ & & & 1 \end{bmatrix}, \quad |k| < 1,
$$

where k is called the *reflection coefficient* or *Schur parameter*. Let us consider a row vector $\mathbf{x}^T \in R^{1\times n}$, and $H_{ij}(k)$, where $k = x_j/x_i$, the ratio of the jth and ith elements. With this choice, we will have

$$
[\cdots x_i \cdots x_j \cdots]H_{ij}(k) = [\cdots x_i' \cdots 0 \cdots] \equiv \mathbf{x}'^T, \quad x_i' \equiv \sqrt{x_i^2 - x_j^2},
$$

where only the ith and jth elements of $\mathbf{x}$ are altered. Clearly, we can introduce a zero at any position in the pre-array by post-multiplying the pre-array with an appropriate hyperbolic rotation.

We shall illustrate the annihilation procedure with a 3×3 Toeplitz matrix,

$$\mathbf{T} = \begin{bmatrix} 1 & c_1 & c_2 \\ c_1 & 1 & c_1 \\ c_2 & c_1 & 1 \end{bmatrix}.$$

We first form a pre-array Δ, and post-multiply Δ by $H_{2,4}(c_1)$ to annihilate an element c_1 in the (1, 2) block, resulting in Δ_1:

$$\Delta \equiv \left[\begin{array}{ccc|ccc} 1 & & & & & \\ c_1 & 1 & & c_1 & & \\ c_2 & c_1 & 1 & c_2 & c_1 & \\ \hline 1 & & & 1 & & \\ & 1 & & & 1 & \\ & & 1 & & & 1 \end{array}\right], \quad \Delta H_{2,4}(c_1) = \left[\begin{array}{ccc|ccc} 1 & & & & & \\ c_1 & d_1 & & & & \\ c_2 & d_2 & 1 & d_3 & c_1 & \\ \hline 1 & d_4 & & d_5 & & \\ & d_5 & & d_4 & 1 & \\ & & 1 & & & 1 \end{array}\right] \equiv \Delta_1.$$

Now, we annihilate the remaining c_1 in the (1, 2) block with $H_{3,5}(c_1)$ and the last element d_3 in the (1, 2) block with $H_{3,4}(d_3/d_1)$ resulting in the post-array $\bar{\Delta}$:

$$\Delta_1 H_{3,5}(c_1) = \left[\begin{array}{ccc|ccc} 1 & & & & & \\ c_1 & d_1 & & & & \\ c_2 & d_2 & d_1 & d_3 & & \\ \hline 1 & d_4 & & d_5 & & \\ & d_5 & d_4 & d_4 & d_5 & \\ & & d_5 & & d_4 & 1 \end{array}\right] \equiv \Delta_2 \quad \Delta_2 H_{3,4}\left(\frac{d_3}{d_1}\right) = \left[\begin{array}{ccc|ccc} 1 & & & & & \\ c_1 & d_1 & & & & \\ c_2 & d_2 & e_1 & & & \\ \hline 1 & d_4 & e_2 & e_4 & & \\ & d_5 & e_3 & e_3 & d_5 & \\ & & e_4 & e_2 & d_4 & 1 \end{array}\right] \equiv \bar{\Delta}.$$

Note that the Toeplitz structure allows the whole subdiagonal in the (1, 2) block to be nulled-out with hyperbolic rotations having the same reflection coefficient k_i. This suggests that we can keep only two columns and operate on them as shown below:

$$X \equiv \left[\begin{array}{cc} 1 & 0 \\ c_1 & c_1 \\ c_2 & c_2 \\ \hline 1 & 1 \\ 0 & 0 \\ 0 & 0 \end{array}\right], \quad X^{(s)} \equiv \left[\begin{array}{cc} 0 & 0 \\ 1 & c_1 \\ c_1 & c_2 \\ \hline 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{array}\right], \quad X^{(s)}H_{1,2}(c_1) = \left[\begin{array}{cc} 0 & 0 \\ d_1 & 0 \\ d_2 & d_3 \\ \hline d_4 & d_5 \\ d_5 & d_4 \\ 0 & 0 \end{array}\right] \equiv X_1, \tag{6a}$$

$$X_1^{(s)} \equiv \left[\begin{array}{cc} 0 & 0 \\ 0 & 0 \\ d_1 & d_3 \\ \hline 0 & d_5 \\ d_4 & d_4 \\ d_5 & 0 \end{array}\right], \quad X_1^{(s)}H_{1,2}(\frac{d_3}{d_1}) = \left[\begin{array}{cc} 0 & 0 \\ 0 & 0 \\ e_1 & 0 \\ \hline e_2 & e_4 \\ e_3 & e_3 \\ e_4 & e_2 \end{array}\right] \equiv X_2. \tag{6b}$$

The entries in X, X_1 and X_2 completely determine the matrices of L_1, L_2 and U. This construct can be regarded as a combined form of Schur algorithm [33-34], [59] (see also [4], [42-44], [55]) and Levinson algorithm [46]. We remark also on the simplicity of the algorithm description and justification.

By inspecting (6), the overall operation count for finding L_1, L_2 and U for an $n \times n$ Toeplitz matrix T can be seen to be

$$2\sum_{i=1}^{n} 4i = 4n^2 + O(n).$$

This count can be reduced by half if we use fast rotations (see [13] for details).

QR factorization of T [13].

Similar ideas can be used for finding the fast QR factorization of a Toeplitz matrix T. First, it is not hard to check that $T^T T$ and T have the displacement forms,

$$T^T T = L(\mathbf{w}_1)L^T(\mathbf{w}_1) + L(\mathbf{w}_2)L^T(\mathbf{w}_2) - L(\mathbf{w}_3)L^T(\mathbf{w}_3) - L(\mathbf{w}_4)L^T(\mathbf{w}_4),$$
$$T = L(\mathbf{v}_1)L^T(\mathbf{w}_1) + L(\mathbf{v}_2)L^T(\mathbf{w}_2) - L(\mathbf{v}_3)L^T(\mathbf{w}_3) - L(\mathbf{v}_4)L^T(\mathbf{w}_4),$$

where the vectors $\mathbf{w}_i$ and $\mathbf{v}_i$ are defined as

$$\mathbf{w}_1 = T^T\mathbf{t}_1/\|\mathbf{t}_1\|, \quad \mathbf{w}_2 = \mathbf{t}_2, \quad \mathbf{w}_3 = Z_n Z_n^T \mathbf{w}_1, \quad \mathbf{w}_4 = Z_n \mathbf{l}_1,$$
$$\mathbf{v}_1 = -\mathbf{v}_3 = \mathbf{t}_1/\|\mathbf{t}_1\|, \quad \mathbf{v}_2 = \mathbf{e}_1, \quad \mathbf{v}_4 = \mathbf{0},$$

where Z_n denotes the $n \times n$ *shift matrix* (i.e., the matrix with 1's on the first sub-diagonal and 0's elsewhere), and

$$\mathbf{t}_1 = [t_0, t_1, \cdots, t_{m-1}]^T, \quad \mathbf{t}_2 = [0, t_{-1}, \cdots, t_{1-n}]^T, \quad \mathbf{l}_1 = [t_{m-1}, \cdots, t_{m-n}]^T.$$

Now we form a *pre-array*

$$\Gamma = \begin{bmatrix} L(\mathbf{w}_1) & L(\mathbf{w}_2) & L(\mathbf{w}_3) & L(\mathbf{w}_4) \\ L(\mathbf{v}_1) & L(\mathbf{v}_2) & L(\mathbf{v}_3) & L(\mathbf{v}_4) \end{bmatrix},$$

and post-multiply Γ with any *J-orthogonal matrix*, where

$$J \equiv \begin{bmatrix} I_n & & & \\ & I_n & & \\ & & -I_n & \\ & & & -I_n \end{bmatrix},$$

that will annihilate the (1, 2), (1, 3) and (1, 4) blocks of Γ, and triangularize the (1, 1) block of Γ:

$$\Gamma\Theta = \begin{bmatrix} R^T & O & O & O \\ Q & * & * & * \end{bmatrix} \equiv \bar{\Gamma}.$$

Again by equating each entry of $\Gamma J \Gamma^T$ and $\bar{\Gamma} J \bar{\Gamma}^T$, we shall have

$$T^T T = R^T R, \qquad T = QR.$$

Note that the matrix Q is orthogonal because

$$Q^T Q = R^{-T} T^T T R^{-1} = R^{-T} R^T R R^{-1} = I.$$

In this application, the J-orthogonal matrix Θ can be constructed as a sequence of hyperbolic rotations *and* circular (or Givens rotations),

$$G_{i,j}(k) = \frac{1}{(1+k^2)^{1/2}} \begin{bmatrix} 1 & & & \\ & 1 & -k & \\ & k & 1 & \\ & & & 1 \end{bmatrix}.$$

Simplicity of the above algorithm and its justification may be compared with the previously

known fast QR factorization algorithms.

2. Some Examples of Structured Matrices.

How about the factorization of non-Toeplitz matrices and their inverses? In many signal processing problems, one needs to solve *structured matrix* equations or to factorize structured matrices either implicitly or explicitly [33-34], [42-44]. Some examples [33-34] involving Toeplitz or *Toeplitz-like* matrices are the Schur-Cohn test (for checking if a polynomial has a root outside the unit disk), orthogonal filter synthesis, finding AR filter [39] and certain inverse scattering problems [11], [35]. On the other hand, certain decoding algorithms for BCH codes [14], [41] require the factorization of Hankel matrices, and finding interpolating polynomials needed to solve Vandermonde matrix equations. In this section we shall present some well-known examples bringing in structured matrices.

Example 1 Two Dimensional Filtering with finite Samples.

Let $\{y_{i,j}\}$ and $\{\eta_{i,j}\}$ be uncorrelated Gaussian random images, and suppose we observe $\{y_{i,j}\}$,

$$y_{i,j} = x_{i,j} + \eta_{i,j}.$$

Let the image planes be *stationary*, i.e.,

$$E[y_{i,j}y_{k,l}] = d_{k-i,l-j}, \quad E[\eta_{i,j}\eta_{k,l}] = f_{k-i,l-j}.$$

It is desired to find the estimator based on the measurements in the square region centered at (i, j)

$$\begin{aligned} \hat{x}_{i,j} &= E[x_{i,j} | y_{k,l} : i-m \le k \le i+m,\ j-m \le l \le j+m] \\ &= \sum_{k=i-m}^{i+m} \sum_{l=j-m}^{j+m} a_{k,l} y_{k,l}. \end{aligned} \tag{7}$$

The coefficients in (7) can be found from the *orthogonality principle* [32],

$$E[(\hat{x}_{i,j} - x_{i,j})y_{i,j}] = 0$$

leading to the following block-Toeplitz, Toeplitz-block (or doubly Toeplitz) system of

equations,

$$\begin{bmatrix} T_0 & T_1 & \cdot & T_{2m} \\ T_1 & T_0 & \cdot & T_{2m-1} \\ \cdot & \cdot & \cdot & \cdot \\ T_{2m} & T_{2m-1} & \cdot & T_0 \end{bmatrix} \begin{bmatrix} \mathbf{a}_{i-m} \\ \mathbf{a}_{i-m+1} \\ \cdot \\ \mathbf{a}_{i+m} \end{bmatrix} = \begin{bmatrix} \mathbf{g}_{-m} \\ \mathbf{g}_{-m+1} \\ \cdot \\ \mathbf{g}_m \end{bmatrix}, \tag{8}$$

where

$$[T_k]_{ij} = d_{k,i-j} \in R^{(2m+1)\times(2m+1)},$$
$$\mathbf{a}_k = [a_{k,i-m}, \cdots, a_{k,i+m}]^T \in R^{(2m+1)\times 1},$$
$$\mathbf{g}_k = [g_{k,-m} \cdots, g_{k,m}]^T \in R^{(2m+1)\times 1}, \qquad g_{ij} \equiv d_{ij} - f_{ij}.$$

The algorithms to be described in Chapter 2 can be used to solve (8) with $O(m^5)$ operations.

Example 2 Numerical Solution of Integral Equations [19], [47], [52], [58].

In some signal processing applications, we need to solve certain integral equations, such as the Wiener-Hopf equation,

$$r_{xy}(t+\lambda) = \int_{-\infty}^{\infty} r_{yy}(t-\tau)h(\tau)d\tau, \quad t > 0, \tag{9a}$$

or the equation that arises in *inverse filtering* [27] for image restoration,

$$g(x, y) = \int_a^b \int_c^d h(x-\alpha, y-\beta) f(\alpha, \beta) d\alpha d\beta. \tag{9b}$$

The equation (9b) is usually solved numerically after discretization using some quadrature formula, which will yield a matrix equation. It is known that solving an integral equation of the form (9) is inherently an ill-posed (ill-conditioned) problem. For simplicity let us consider the single variable case,

$$g(x) = \int_a^b K(x, y) f(y) dy. \tag{10}$$

Phillips [52] gives a good discussion of the difficulty involved. Following Phillips, let $f_m(y) \equiv \sin(my)$. Then for any integrable kernel $K(x, y)$, it is known that

$$g_m \equiv \int_a^b K(x, y) f_m(y) dy \;\rightarrow\; 0 \quad \text{as} \quad m \rightarrow \infty.$$

Therefore, an infinitesimal perturbation g_m in g can cause a finite perturbation f_m in f. Also, one would expect that $g_m \rightarrow 0$ (as $m \rightarrow \infty$) faster for flat smooth kernels than for sharply

peaked kernels. Let

$$K\mathbf{f} = \mathbf{g} \tag{11}$$

be the matrix equation obtained from (10) by some discretization procedure. If we refine the discretization, then the ill-conditioned kernel would appear as an ill-conditioned matrix. On the other hand, if we use a large mesh width, the transformation from the integral equation to the matrix equation can be ill-conditioned, and therefore we may be solving a (possibly well-conditioned) different problem.

One way to try to overcome this difficulty is via *regularization*. The ill-conditioning manifests itself as an oscillatory solution $f(x)$. Therefore, it is reasonable to constrain the solution to have some smoothness, *e.g.*, to require that

$$\int_a^b f^{(n)}(x)dx \le \Gamma,$$

or after discretization

$$\|L^{(n)}\mathbf{f}\|_2 \le \gamma. \tag{12}$$

For example when $n = 1$ and 2,

$$f^{(1)}(x)\Delta x \approx f(x + \Delta x) - f(x), \quad f^{(2)}\Delta x \approx f(x + \Delta x) - 2f(x) + f(x - \Delta x).$$

Therefore,

$$L^{(1)} = \begin{bmatrix} 1 & -1 & & & \\ & 1 & -1 & & \\ & & \ddots & \ddots & \\ & & & 1 & -1 \end{bmatrix}, \quad L^{(2)} = \begin{bmatrix} 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \end{bmatrix}.$$

Now, we solve the following constrained minimization problem rather than (11),

$$\min_{\mathbf{f}} \|K\mathbf{f} - \mathbf{g}\|_2 \quad \text{subject to } \dagger \quad \|L^{(n)}\mathbf{f}\|_2 = \gamma. \tag{13}$$

To solve (13), we form the Lagrange function

† The constraint in (12) is usually active.

$$F(\mathbf{f}, \eta) \equiv ||K\mathbf{f} - \mathbf{g}||^2 + \eta(||L\mathbf{f}||^2 - \gamma),$$

and from $\dfrac{\partial F(\mathbf{f}, \eta)}{\partial \mathbf{f}} = 0$, we get

$$(K^T K + \eta L^T L)\mathbf{f} = K^T \mathbf{g}. \tag{14a}$$

This can be recognized as seeking the least squares solution to the linear system

$$\begin{bmatrix} K \\ \eta L \end{bmatrix} \mathbf{f} = \begin{bmatrix} \mathbf{g} \\ 0 \end{bmatrix}. \tag{14b}$$

The Lagrange multiplier η is usually chosen by trial and error. If L is the identity matrix, then this method reduces to the so-called "damped least squares method" (also see [25, pp. 145, P6.1-9] for computing approximate pseudo inverses with this technique).

For the convolutional kernel, the matrix K is Toeplitz, and the matrix $\begin{bmatrix} K \\ \eta L \end{bmatrix}$ is close-to-Toeplitz. The algorithms to be discussed in Chapter 2 can be used to solve (14) in $O(n^2)$ operations.

Example 3 Maximum Likelihood Estimation of ARMA Parameters [3], [40, pp. 125-127].

Let $\{y(t)\}$ and $\{e(t)\}$ satisfy the difference equation

$$y(t) + a_1 y(t-1) + \cdots + a_p y(t-p) = e(t) + c_1 e(t-1) + \cdots + c_q e(t-q) \tag{15}$$

where $e(t)$ is a zero-mean white Gaussian process with variance σ^2,

$$A(z) \equiv z^p + a_1 z^{p-1} + \cdots + a_p \neq 0 \quad \text{for} \quad |z| < 1,$$
$$C(z) \equiv z^q + c_1 z^{q-1} + \cdots + c_q \neq 0 \quad \text{for} \quad |z| < 1,$$

and $\{a_i\}$, $\{c_i\}$ are unknown *deterministic* constants. Given measurements

$$\mathbf{y}_N \equiv [y(N), \cdots, y(0), y(-1), \cdots, y(-p)]^T$$

it is desired to find an estimator for σ^2 and

$$\boldsymbol{\theta} = [\theta_1, \theta_2, \cdots, \theta_n]^T = [a_1, \cdots, a_p, c_1, \cdots, c_q]^T.$$

Note that

$$P(\mathbf{y}_N) = P(y(N)|\mathbf{y}_{N-1})P(\mathbf{y}_{N-1}).$$

Therefore,

$$P(\mathbf{y}_N) = [\prod_{t=1}^{N} P(y(t)|\mathbf{y}_{t-1})]P(y(0), \cdots, y(-p)) \tag{16}$$

Because $\{e(t)\}$ is Gaussian, $P(y(t)|\mathbf{y}_{t-1})$ is also Gaussian, and

$$P(y(t)|\mathbf{y}_{t-1}) = \frac{1}{\sqrt{2\pi\rho^2}}\exp\left[-\frac{[y(t)-\hat{y}(t)]^2}{2\rho^2}\right], \tag{17}$$

where

$$\hat{y}(t) = E[y(t)|\mathbf{y}_{t-1}] \tag{18}$$
$$\rho^2 = E[(y(t)-\hat{y}(t))(y(t)-\hat{y}(t))] = E[e(t)e(t)] = \sigma^2$$

First, note that the probability density $P(y(0), \cdots, y(p))$ is a complicated function of $\{y(0), \cdots, y(-p)\}$, θ and σ, and therefore, it is difficult to find the maximum likelihood estimate of θ using $P(\mathbf{y}_N)$ in (16). Instead, we maximize the conditional probability density function

$$P[\mathbf{y}_N|(y(0), \cdots, y(-p))] = \prod_{t=1}^{N} P(y(t)|\mathbf{y}_{t-1}). \tag{19}$$

To use (19) for maximum likelihood estimation, we need to express $\hat{y}(t)$ in (17) in terms of θ. If we assume that we know $\{e(t)\}$, then

$$\begin{aligned}\hat{y}(t) &= E[y(t)|\mathbf{y}_{t-1}] \\ &= a_1 y(t-1) + \cdots + a_p y(t-p) + c_1 e(t-1) + \cdots + c_q e(t-q).\end{aligned}$$

However, we do not know $\{e(t)\}$, so we approximate $e(t)$ with $\varepsilon(t)$ that is computed recursively by

$$\varepsilon(t) = y(t) + a_1 y(t-1) + \cdots + a_p y(t-p) - c_1\varepsilon(t-1) - \cdots - c_q\varepsilon(t-q). \tag{20}$$

With this approximation note that

$$y(t) - \hat{y}(t) = \varepsilon(t).$$

From (19), the (conditional) likelihood function L is given by

$$-\log L = \frac{1}{2\sigma^2}\sum_{k=1}^{N}\varepsilon^2(t) + N\log\sigma + \frac{N}{2}\log 2\pi.$$

Note that maximization of L with respect to the parameters σ and θ can be done separately.

Maximization of L with respect to θ is equivalent to minimization of

$$V(\theta) \equiv \frac{1}{2}\sum_{k=1}^{N}\varepsilon^2(t).$$

With the following notations,

$$Df(\theta) \equiv [\ \frac{\partial f}{\partial \theta_1}, \cdots, \frac{\partial f}{\partial \theta_n}\ ]^T \in R^{n\times 1}, \qquad D^2 f(\theta) \equiv [\ \frac{\partial^2 f}{\partial \theta_i \partial \theta_j}\ ] \in R^{n\times n},$$

notice that

$$DV(\theta) = \sum_{t=1}^{N}\varepsilon(t)\cdot D\varepsilon(t) = B^T\mathbf{b},$$

$$D^2V(\theta) = \sum_{t=1}^{N}D\varepsilon(t)D^T\varepsilon(t) + \sum_{t=1}^{N}\varepsilon(t)D^2\varepsilon(t) \approx B^TB,$$

where

$$B = \begin{bmatrix} [D\varepsilon(1)]^T \\ [D\varepsilon(2)]^T \\ \cdot \\ \cdot \\ [D\varepsilon(N)]^T \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \varepsilon(1) \\ \varepsilon(2) \\ \cdot \\ \cdot \\ \varepsilon(N) \end{bmatrix} \tag{21}$$

Using (20), one can easily verify that

$$\frac{\partial\varepsilon(t)}{\partial a_i} = y(t-i) - c_1\frac{\partial\varepsilon(t-1)}{\partial a_i} - c_2\frac{\partial\varepsilon(t-2)}{\partial a_i} - \cdots - c_q\frac{\partial\varepsilon(t-q)}{\partial a_i},$$

$$\frac{\partial\varepsilon(t)}{\partial c_i} = \varepsilon(t-i) - c_1\frac{\partial\varepsilon(t-1)}{\partial c_i} - c_2\frac{\partial\varepsilon(t-2)}{\partial c_i} - \cdots - c_q\frac{\partial\varepsilon(t-q)}{\partial c_i}.$$

If we define the matrices Y and E,

$$Y \equiv \begin{bmatrix} y(0) & y(-1) & \cdot & y(1-p) \\ y(1) & y(0) & \cdot & y(2-p) \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ y(m-1) & y(m-2) & \cdot & y(m-p) \end{bmatrix} \quad E \equiv -\begin{bmatrix} \varepsilon(0) & \varepsilon(-1) & \cdot & \varepsilon(1-q) \\ \varepsilon(1) & \varepsilon(0) & \cdot & \varepsilon(2-q) \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \varepsilon(m-1) & \varepsilon(m-2) & \cdot & \varepsilon(m-q) \end{bmatrix}$$

then the matrix B in (21) has the form

$$B = C^{-1}[Y\ E], \qquad C \equiv \begin{bmatrix} 1 & & & & & & \\ c_1 & 1 & & & & & \\ \cdot & c_1 & \ddots & & & & \\ c_q & \cdot & & \ddots & & & \\ & c_q & \ddots & & \ddots & & \\ & & & c_q & \cdot & c_1 & 1 \end{bmatrix}.$$

Now the parameter $\boldsymbol{\theta}$ that minimizes $V(\boldsymbol{\theta})$ can be obtained iteratively by using Newton's method.

(Conditional) Maximum Likelihood Estimate

Set initial estimate $\boldsymbol{\theta} = 0$;
repeat
Compute $\varepsilon(t)$ with (20);
Solve $B\mathbf{s} = \mathbf{b}$;
$\boldsymbol{\theta} := \boldsymbol{\theta} + \mathbf{s}$
until convergence.

The matrices $C^{-1}Y$ or $C^{-1}E$ are not Toeplitz, but they are close-to-Toeplitz.

Example 4 Instrumental Variable Method [20].

Consider an ARMA model as in (15), and let $\boldsymbol{\theta}_a = [a_1, \cdots, a_p]^T$. The parameter $\boldsymbol{\theta}_a$ can be obtained by solving

$$T_2^T T_1 \boldsymbol{\theta} = T_2^T \mathbf{y}, \tag{22}$$

where

$$T_2^T = \begin{bmatrix} y(q) & y(q+1) & \cdots & y(q+N) \\ y(q-1) & y(q) & \cdots & y(q+N+1) \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ y(q-p+1) & \cdot & \cdots & \cdot \end{bmatrix}, \quad T_1 = \begin{bmatrix} y(0) & y(-1) & \cdots & y(1-p) \\ y(1) & y(0) & \cdots & y(2-p) \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ y(N-1) & \cdot & \cdots & y(N-p) \end{bmatrix}$$

$$\mathbf{y} = [\, y(1), y(2), \cdots, y(N)\,]^T$$

Again the matrix $T_2^T T_1$ is close-to-Toeplitz, and the fast algorithms in this thesis can be used to solve (22).

Example 5 The Euclidean algorithm and Hankel Matrix Factorization. †

The Euclidean algorithm that tests if two polynomials are relatively prime or not in fact factorizes Hankel matrices. To see this, let us consider the 3×3 Hankel matrix

$$H = \begin{bmatrix} 5 & 3 & 2 \\ 3 & 2 & 1 \\ 2 & 1 & 4 \end{bmatrix}$$

We define the polynomial

$$p(x) = 5x^4 + 3x^3 + 2x^2 + x + 4,$$

whose coefficients are the 1st column and the 1st row of H. Also let

$$q(x) = x^5.$$

We repeatedly divide as follows.

$$\begin{array}{r|l} & \frac{1}{5}x \\ \hline 5x^4+3x^3+2x^2+x+4 & 1x^5 \\ & x^5+\frac{3}{5}x^4+\frac{2}{5}x^3+\frac{1}{5}x^2+\frac{4}{5}x \\ \hline & -\frac{3}{5}x^4-\frac{2}{5}x^3-\frac{1}{5}x^2-\frac{4}{5}x \end{array} \tag{22a}$$

$$\begin{array}{r|l} & -\frac{25}{3} \\ \hline -\frac{3}{5}x^4-\frac{2}{5}x^3-\frac{1}{5}x^2-\frac{4}{5}x & 5x^4+3x^3+2x^2+x+4 \\ & 5x^4+\frac{10}{3}x^3+\frac{5}{3}x^2+\frac{20}{3}x \\ \hline & -\frac{1}{3}x^3+\frac{1}{3}x^2-\frac{17}{3}x+4 \end{array}$$

† The division algorithm shown here is slightly different from the classical Euclidean algorithm. However, classical Euclidean algorithm can be also used for this purpose.

$$
\begin{array}{r|l}
 & \frac{9}{5}x \\
\hline
-\frac{1}{3}x^3+\frac{1}{3}x^2-\frac{17}{3}x+4 & -\frac{3}{5}x^4-\frac{2}{5}x^3-\frac{1}{5}x^2-\frac{4}{5}x \\
 & -\frac{3}{5}x^4+\frac{3}{5}x^3-\frac{51}{5}x^2+\frac{36}{5}x \\
\hline
 & -x^3+10x^2-8x
\end{array}
\tag{22b}
$$

$$
\begin{array}{r|l}
 & \frac{1}{3} \\
\hline
-x^3+10x^2-8x & -\frac{1}{3}x^3+\frac{1}{3}x^2-\frac{17}{3}x+4 \\
 & -\frac{1}{3}x^3+\frac{10}{3}x^2-\frac{8}{3}x \\
\hline
 & -3x^2-3x+4
\end{array}
$$

$$
\begin{array}{r|l}
 & \frac{1}{3}x \\
\hline
-3x^2-3x+4 & -x^3+10x^2-8x \\
 & -x^3-x^2+\frac{3}{4}x \\
\hline
 & 11x^2-\frac{35}{4}x
\end{array}
\tag{22c}
$$

Now consider the truncated divisor polynomials in (22),

$$p_0(x) \equiv 5x^4+3x^3+2x^2, \quad p_1(x) \equiv -\frac{1}{3}x^3+\frac{1}{3}x^2, \quad p_2(x) \equiv -3x^2,$$

and the highest degree terms of the dividend polynomials in (22),

$$q_0(x) \equiv x^5, \quad q_1(x) \equiv -\frac{3}{5}x^4, \quad q_2(x) \equiv -x^3.$$

If we multiply $p_i(x)$ by the coefficients of $q_i(x)$, then it turns out the resulting polynomials form the columns of Cholesky factors of H. Namely,

$$H = \begin{bmatrix} 5 & & \\ 3 & 1/5 & \\ 2 & -1/5 & 3 \end{bmatrix} \begin{bmatrix} 1/5 & & \\ & 5 & \\ & & 1/3 \end{bmatrix} \begin{bmatrix} 5 & 3 & 2 \\ & 1/5 & -1/5 \\ & & 3 \end{bmatrix}.$$

The computational effort is $O(n^2)$. Further discussion of the problem will be given in Chapter 4.

Example 6 Polynomial Interpolation [15, pp. 38-46].

Given n distinct points $x_1, x_2, \cdot, \cdot, x_n$, we can find a polynomial

$$p(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_1 x + a_0 \tag{23}$$

of degree at most $n-1$ by solving

$$\begin{bmatrix} 1 & x_1 & \cdot & x_1^{n-1} \\ 1 & x_2 & \cdot & x_2^{n-1} \\ \cdot & \cdot & \cdot & \cdot \\ 1 & x_n & \cdot & x_n^{n-1} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \cdot \\ a_{n-1} \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ y_n \end{bmatrix}, \qquad y_i \equiv p(x_i).$$

Instead of finding the coefficients of the polynomial in (23), we could as well find the coefficients of the so-called *Newton form*,

$$p(x) = c_0 + c_1(x-x_0) + c_2(x-x_0)(x-x_1) + \cdots + c_n(x-x_0)\cdots(x-x_{n-1}).$$

One can check that the coefficient c_k depends only on the values of $f(x)$ at the points $x_0, x_1, \cdots, x_k$; it is called the *kth divided difference* of $f(x)$ at the points $x_0, x_1, \cdots, x_k$ and is denoted by $f[x_0, \cdots, x_k]$. Also one can check that

$$f[x_0, \cdots, x_k] = \frac{f[x_1, \cdots, x_k] - f[x_0, \cdots, x_{k-1}]}{x_k - x_0} \tag{24}$$

After finding the Newton form, we can recursively compute the coefficients a_i in (23), if we wish, by using the identity

$$\begin{aligned} c_0 + c_1(x-x_0) + c_2(x-x_0)(x-x_1) + c_3(x-x_0)(x-x_1)(x-x_2) \\ = c_0 + (c_1 + (c_2 + c_3(x-x_2))(x-x_1))(x-x_0). \end{aligned} \tag{25}$$

The computations in (24) and (25) need $O(n^2)$ operations, and in fact factorize the Vandermonde system of equations [8]. The factorization algorithms (for Vandermonde matrices) in Chapter 4 are closely related to this method.

3. Outline of the Thesis.

The idea in Sec 1 is extended in Chapter 2 to find triangular factorizations and QR factorizations of block-Toeplitz and Toeplitz-block matrices. In Chapter 3, we slightly change the

pre-array in Sec 1 to constructively obtain certain well-known Toeplitz inverse expressions called the Gohberg-Semencul formulas. We also generalize the Gohberg-Semencul formulas [22-23] to a large class of matrices. Some related results are in [21], [26], [39], [42-43], [63]. In Chapter 4, we show how to factorize close-to-Hankel matrices such as Hankel, block-Hankel, Hankel-block and Vandermonde matrices. Some previous results are [5], [9], [41], [48], [53-54], [61]. In Chapter 5, we present a divide-and-conquer approaches for finding solutions of block-Toeplitz and Toeplitz-block matrices; for related results, see [2], [10], [16], [49], [51]. Some concluding remarks are offered in Chapter 6. Each chapter is self-contained, and therefore, readers can essentially read them in any order.

REFERENCES

[1]. N. Akhiezer, *The classical moment problem*, Hafner Pub. Company, New York, 1965.

[2]. G. Ammar and W. Gragg, *Superfast solution of real positive definite Toeplitz systems*, SIAM J. Matrix Anal., Appl., Vol. 9, No. 1, Jan, (1988), pp. 61-76.

[3]. K. Astrom, *Maximum likelihood and prediction error methods*, pp. 145-168, in Time Series and Linear Systems, eds S. Bittanti, in Lecture Notes in Control and Information Sciences 86, eds M. Thoma and A. Wyner, Springer-Verlag, New York.

[4]. E. Bareiss, *Numerical solution of linear equations with Toeplitz and vector Toeplitz matrices*, Numer. Math., 13: (1969), 404-424.

[5]. E. Berlekamp, *Algebraic coding theory*, McGraw-Hill, New York, 1968.

[6]. G. Bierman, *Factorization methods for discrete sequential estimation*, New York, Academic, 1977.

[7]. G. Bitmead and B. D. O. Anderson, *Asymptotically fast solution of Toeplitz and related*

systems of linear equations, Linear Algebra and its Appli., 34 (1980), pp. 103-116.

[8]. A. Bjorck and V. Pereyra, *Solution of Vandermonde systems of equations*, Math Comp., 24 (1980).

[9]. R. Blahut, *Theory and practice of error control codes*, Addison-Wesley, Reading, MA, 1983.

[10]. R. Brent, F. Gustavson and D. Yun *Fast Solution of Toeplitz Systems of Equations and Computation of Pade Approximants* Journal of Algorithms, 1, (1980), pp. 259-295

[11]. A. Bruckstein and T. Kailath, *Inverse scattering for discrete transmission-line models*, SIAM Review, (1986).

[12]. S. Chandrasekhar, *Radiative Transfer*, New York, Dover Publications, 1960.

[13]. J. Chun, T. Kailath and H. Lev-Ari, *Fast parallel algorithms for QR and triangular factorization*, SIAM J. Sci. Stat. Comput., vol. 8, No. 6, Nov., (1987), pp. 899-913.

[14]. T. Citron, *Algorithms and architectures for error correcting codes*, Ph.D. Thesis, Stanford Univ., August 1986.

[15]. S. Conte and C. de Boor, *Elementary numerical analysis*, An algorithmic approach, 3rd ed., McGraw-Hill, New York, 1980.

[16]. F. De Hoog, *A new algorithm for solving Toeplitz systems of equations, Linear Algebra and its Appli., 88/89*, (1987), pp. 122-138.

[17]. P. Dewilde, A. Vieira and T. Kailath, *On a generalized Szego-Levinson realization algorithm for optimal linear predictors based on a network synthesis approach*, IEEE Trans. on Circuit and Systems, vol. CAS-25, No. 9, Sep, 1978.

[18]. P. Dyer and S. McReynolds, *The extension of square-root filtering to include process noise*, J. Opt.: Theory and Applications, 3, pp. 92-105, 1969.

[19]. L. Elden, *An algorithm for the regularization of ill-conditioned least squares problems*, SIAM J. Sci. Stat. Comput., vol 5, No. 1., (1984), pp. 237-254.

[20]. P. Eykhoff, *System identification*, John Wiley and Sons, New York, 1974

[21]. B. Friedlander, M. Morf, T. Kailath and L. Ljung, *New inversion formula for matrices classified in terms of their distance from Toeplitz matrices*, Linear Algebra and its Appl., 27 (1979), pp. 31-60.

[22]. I. Gohberg and A. Semencul, *On the inversion of finite Toeplitz matrices and their continuous analogs*, Mat. Issled., 2 (1972), pp. 201-233.

[23]. I. Gohberg and I. Fel'dman, *Convolution equations and projection methods for their solutions*, Translations of Mathematical Monographs, vol. 41, Amer. Math. Soc., 1974.

[24]. P. Gill, G. Golub, W. Murray and M. Saunders, *Methods for modifying matrix factorizations*, Math Comp., 28, pp. 505-535, 1974.

[25]. G. Golub and C. Van Loan, *Matrix computations*, Johns Hopkins Univ. Press, Maryland, 1983.

[26]. U. Grenander and G. Szego, *Toeplitz forms and their applications*, Chelsea Pub. Company, New York, 1955.

[27]. E. Hall, *Computer image processing and recognition*, Academic press, 1979.

[28]. G. Heinig and K. Rost, *Algebraic methods for Toeplitz-like matrices and operators*, Akademie-Verlag, Berlin, 1984.

[29]. T. Kailath, *Some Chandrasekhar-type algorithms for quadratic regulators*, Proc. IEEE Conf. on Decision and Control, New Orleans, Dec 1972.

[30]. T. Kailath, *Some new results and insights in linear least-squares estimation theory*, Proc. of the 1975 IEEE - USSR Joint Workshop on Information Theory, pp. 97-104, Moscow, Dec. 1975. Reprinted with corrections as App. I of [32].

[31]. T. Kailath, *Some new algorithms for recursive estimation in constant linear systems*, IEEE Trans on Information Theory, vol. IT-19, pp. 750-760, Nov., 1973.

[32]. T. Kailath, *Lectures on Wiener and Kalman filtering*, Springer-Verlag, New York, 1981.

[33]. T. Kailath, *A theorem of I. Schur and its impact on modern signal processing*, Operator Theory, Advances and Applications, Vol. 18, Birkhauser Verlag, Basel, 1986, pp. 9-30.

[34]. T. Kailath, *Signal processing applications of some moment problems*, Proceedings of Symposia in Applied Mathematics, vol. 37, 1987 pp. 71-109.

[35]. T. Kailath, A. Bruckstein and D. Morgan, Fast matrix factorizations via discrete transmission lines, *Linear Algebra Appl.* 75:1-25 (1986).

[36]. T. Kailath, S. Kung and M. Morf, Displacement ranks of matrices and linear equations, *J. Math. Anal. Appl.*, 68:395-407 (1979). See also *Bull. Amer. Math. Soc.*, 1:769-773 (1979).

[37]. T. Kailath, M. Morf and G. Sidhu, *Some new algorithms for recursive estimation in constant discrete-time linear systems*, Proc. 7th Princeton Symp. Information and System Sci., 1973

[38]. T. Kailath, M. Morf and G. Sidhu, *Some new algorithms for recursive estimation in constant discrete-time linear systems*, IEEE Trans. Automat. Contro., vol. AC-19, pp. 315-323, Aug. 1974.

[39]. T. Kailath, A. Vieira and M. Morf, Inverses of Toeplitz operators, innovations, and orthogonal polynomials, *SIAM Review*, 20(1):106-119 (1978).

[40]. R. Kashyap and A. Rao, *Dynamic stochastic models from empirical data*, Academic Press, New York, 1976.

[41]. S. Kung, *Multivariable and multidimensional systems: analysis and design*, Ph.D. Thesis, Stanford Univ., 1977.

[42]. H. Lev-Ari, *Nonstationary lattice-filter modeling*, Ph.D. Thesis, Stanford University, 1983.

[43]. H. Lev-Ari and T. Kailath, *Lattice filter parameterizations and modeling of nonstationary process*, IEEE Trans. Inform. Theory, IT-30 (1984), pp. 2-16.

[44]. H. Lev-Ari and T. Kailath, *Triangular factorization of structured Hermitian matrices*, Operator Theory, Advances and Applications, vol. 18, Birkhäuser, Boston, pp. 301-324, (1986).

[45]. H. Lev-Ari, Y. Bistritz and T. Kailath, *Generalized Bezoutians and families of efficient root-location procedures*, to appear, IEEE Trans. Cir. and Sys., 1989.

[46]. N. Levinson, *The Wiener RMS error criterion in filter design and prediction*, J. of Math. Phys. 25 (1947), 261-278.

[47]. S. Ljung and L. Ljung, *Fast numerical solution of Fredholm integral equations with stationary kernels*, Bit 22, (1982) pp. 54-72.

[48]. J. Massey, *Shift-register synthesis and BCH decoding*, IEEE Trans., Information Theory, IT-15, pp. 122-127 (1969).

[49]. M. Morf, *Doubling algorithms for Toeplitz and related equations*, Proceedings of the IEEE International Conf. on ASSP, Denver, (1980), pp. 954-959.

[50]. M. Morf and T. Kailath, *Square-root algorithms for linear least-squares estimation and control*, IEEE Trans. Auto Cont., vol-20, pp. 487-497, Aug. 1975.

[51]. B. Musicus, *Levinson and fast Cholesky algorithms for Toeplitz and almost Toeplitz matrices*, Report, Research Lab. of Electronics, MIT, Cambridge, MA, 1981.

[52]. D. Phillips, *A technique for the numerical solution of certain integral equations of the first kind*, J. Assoc. Comput. Mach., 9 (1962), pp. 84-06.

[53]. J. Phillips, *The triangular decomposition of Hankel matrices*, Math Comp., vol. 25, (1971) pp. 599-602.

[54]. J. Rissanen, *Algorithms for triangular decomposition of block Hankel and Toeplitz matrices with application to factoring positive matrix polynomial*, Math. Comput., vol. 27, Jan. (1973), pp. 147-154.

[55]. J. Le Roux and C. Gueguen, *A fixed point computation of partial correlation coefficients*, IEEE Trans. Acoustic Speech and Signal Processing, ASSP-25 (1977), pp. 257-259.

[56]. S. Rao and T. Kailath, *Orthogonal digital filters for VLSI implementation*, IEEE Trans. Circuit and System, CAS-31 (1984), pp. 933-945.

[57]. E. Robinson, *Time series analysis and applications*, Goose Pond Press, Houston, Texas, 1981.

[58]. H. Rutishauser, *Once again: The least squares problem*, Linear Alg. and its Appl., 1, pp. 471-478 (1968).

[59]. I. Schur, *On power series which are bounded in the interior of the unit circle. 1.* Operator Theory, Advances and Applications, vol. 18, Birkhauser, Boston, (1986), pp. 31-60.

[60]. V. Sobolev, *A Treatise on Radiative Transfer, Appendices*, Princeton, NJ. D. Van Nostrand Co., 1963.

[61]. Y. Sugiyama, M. Kasahara, S. Hirasawa and T. Namekawa *A method for solving key equations for decoding Goppa codes*, Int. J. on Control, 27 pp. 87-99 (1975).

[62]. G. Szego, *Orthogonal polynomial*, Amer. Math. Soc, Providence, Rhode Island, 1939.

[63]. W. Trench, *An algorithm for inversion of finite Toeplitz matrices*, J. of SIAM, vol. 12.3 (1964), pp. 515-522.

Chapter 2.

Generalized Displacement Structure for Block-Toeplitz, Toeplitz-block, and Toeplitz-derived Matrices

Abstract

The concept of displacement structure has been used to solve several problems connected with Toeplitz matrices and with matrices obtained in some way from Toeplitz matrices (e.g. by combinations of multiplication, inversion and factorization). Matrices of the latter type will be called Toeplitz-derived (or Toeplitz-like). In this chapter we shall introduce a generalized definition of displacement for block-Toeplitz and Toeplitz-block matrices. It will turn out that Toeplitz-derived matrices are perhaps best regarded as particular Schur complements obtained from suitably defined block matrices. The displacement structure will be used to obtain a generalized Schur algorithm for the fast triangular and orthogonal factorizations of all such matrices, and well structured fast solutions of the corresponding exact and overdetermined systems of linear equations.

1. Introduction.

In multichannel signal processing, system identification and image processing applications, one encounters various forms of structured matrices. One interesting family consists of matrices having a block-Toeplitz form

$$A_1 = \begin{bmatrix} B_0 & B_{-1} & \cdot & B_{-N+1} \\ B_1 & B_0 & \cdot & B_{-N+2} \\ \cdot & \cdot & \cdot & \cdot \\ B_{M-1} & B_{M-2} & \cdot & B_{-N+M} \end{bmatrix}, \quad B_i\text{: rectangular matrices,} \tag{1a}$$

or a Toeplitz-block form

$$A_2 = \begin{bmatrix} T_{1,1} & T_{1,2} & \cdot & T_{1,N} \\ T_{2,1} & T_{2,2} & \cdot & T_{2,N} \\ \cdot & \cdot & \cdot & \cdot \\ T_{M,1} & T_{M,2} & \cdot & T_{M,N} \end{bmatrix}, \quad T_{i,j}\text{: rectangular Toeplitz matrices} \tag{1b}$$

or often as Schur complements with respect to various entries in A_1 or A_2 (see the examples in Sec 4). Often we call the matrix A_1 an $M \times N$ *block-Toeplitz array*, and the matrix A_2 an $M \times N$ *Toeplitz-block array;* the matrices obtained as Schur complements are often not Toeplitz at all, but have been called near-Toeplitz, or close-to-Toeplitz, or Toeplitz-like or Toeplitz-derived matrices.

For such A, we shall show how to obtain fast triangular factorization $A = LU$, and fast QR factorization, $A = QR$, which among other things will also give us fast nicely structured methods for solving exactly-determined systems of equations,

$$A\,\mathbf{x} = \mathbf{b}, \quad A \in \mathbf{R}^{n \times n}, \quad A \text{ is strongly nonsingular} \tag{2}$$

and also over-determined systems of equations,

$$A\,\mathbf{x} = \mathbf{b}, \quad A \in \mathbf{R}^{m \times n}, \quad m \geq n, \quad A \text{ has full column rank.} \tag{3}$$

Our results will be based on a generalization of the concept of displacement structure used in earlier work (see e.g., [13-17]). Besides enabling us to solve several new problems, this generalized concept will also provide a new and simpler approach to many of the problems studied

in [13-17] and [4]. However, first we briefly review earlier approaches and results.

For a square block-Toeplitz matrix $A_1 \in \mathbf{R}^{n \times n}$, with square blocks $B_i \in \mathbf{R}^{r \times r}$, there exist several fast triangular factorization algorithms such as the Bareiss algorithm [1], the multichannel Levinson algorithm [18] and the Schur algorithm [13-14], [20], all of which require matrix (of the block size, $r \times r$) operations. Our approach will treat block-Toeplitz matrices in essentially the same way as *scalar* Toeplitz matrices, and in particular will use only elementary scalar operations; the absence of matrix operations such as inversion will simplify the design of dedicated hardware implementations. For a square Toeplitz-block matrix $A_2 \in \mathbf{R}^{n \times n}$ with $T_{ij} \in \mathbf{R}^{m_i \times n_j}$, the previous approaches were first to transform A_2 into a block-Toeplitz matrix by pre- and post-multiplication with permutation matrices, and then apply an algorithm for square block-Toeplitz matrix to get a row- and column- permuted triangular factorization of A_2; there is clearly a difficulty with this approach when $m_i \neq m_j$. Also the permuted matrix might not be strongly nonsingular. Our approach will not have this problem because it directly factorizes A_2 without permutations. Finally, for matrices obtained via Schur complementation, the concept of displacement structure (see e.g. [4], [14-17]) has been used to obtain a number of fast algorithms; in particular, several algorithms have recently appeared [2], [4], [8], [21] for the orthogonalization of scalar Toeplitz matrices; our new approach also provides a generalized unification of these algorithms.

Several illustrations and applications of our approach will be given in Sec 4. Our choice here was made in part to relate to examples and problems that are studied, generally in different-ways, in other chapters of this thesis.

Our generalized definition of displacement structure is presented in Sec 2. A correspondingly generalized Schur algorithm for matrix factorization is derived in Sec 3. As just mentioned, Sec 4 contains various applications. Finally some computational aspects are elaborated in Sec 5; in particular we may note the introduction of *spinors*, which include as special cases

the circular (Givens) and hyperbolic rotations as well as the well-known elementary (or elimination) matrices.

2. Displacement of Matrices.

Let $A \in \mathbf{R}^{m \times n}$ be a given matrix, and let F^f and F^b be *strictly* lower triangular matrices. The matrix

$$\nabla_{(F^f, F^b)} A \equiv A - F^f A F^{bT} \tag{4}$$

will be called the *displacement* of A with respect to the *displacement operators* $\{F^f, F^b\}$. Assume that

$$rank \nabla_{(F^f, F^b)} A = \alpha.$$

Any matrix pair $\{X, Y\}$ such that

$$\nabla_{(F^f, F^b)} A = XY^T, \quad X \equiv [\, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_\alpha \,], \quad Y \equiv [\, \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_\alpha \,]$$

will be called a *generator* of A (with respect to $\{F^f, F^b\}$). The number α will be called the *length* of the generator (with respect to $\{F^f, F^b\}$). A generator of A with the minimal possible length will be called a *minimal* generator. The length of the minimal generator of A (i.e., $\mathrm{rank}(\nabla_{(F^f, F^b)} A)$) will be called the *displacement rank* of A (with respect to $\{F^f, F^b\}$), and denoted as $\alpha_{(F^f, F^b)}(A)$.

If $\{X, Y\}$ is a generator of $A \in \mathbf{R}^{n \times n}$ with respect to $\{F^f, F^b\}$), then for any nonsingular matrix $S \in R^{\alpha \times \alpha}$, the matrix pair $\{XS, YS^{-T}\}$ is also a generator of A because

$$\nabla_{(F^f, F^b)} A = XY^T = XSS^{-1}Y^T. \tag{5}$$

Let $\{X, Y\}$ be a generator of a matrix with respect to strictly lower triangular displacement operators $\{F^f, F^b\}$. We say that a generator is *proper (with respect to the column j)* if, for a certain i, all the elements in the ith row of X and above, except for the element $[X]_{i,j}$, are zero, and all elements in the ith row of Y and above, except the element $[Y]_{i,j}$, are zero; for example,

$$X = \begin{bmatrix} & & & & & & \\ 0 & \cdot & 0 & * & 0 & \cdot & 0 \\ * & . & * & * & * & . & * \\ & & & & & & \\ * & . & * & * & * & . & * \end{bmatrix}, \quad Y = \begin{bmatrix} & & & & & & \\ 0 & \cdot & 0 & * & 0 & \cdot & 0 \\ * & . & * & * & * & . & * \\ & & & & & & \\ * & . & * & * & * & . & * \end{bmatrix}.$$

Often we shall denote a proper generator as $\{X_p, Y_p\}$. If $\{X, Y\}$ is not proper, then by choosing appropriate S, we can obtain a proper generator $\{XS, YS^{-T}\}$ under certain conditions on the matrix A (see Sec 5).

Note that the displacement of a symmetric matrix A can be written as $\nabla_{(F^f, F^b)}A = X\Sigma X^T$ where Σ is a diagonal matrix with 1 or -1 along the main diagonal; we shall say that A has a *symmetric generator*, $\{X, X\Sigma\}$.

As an example, for a square Toeplitz matrix $T = (t_{i-j}) \in \mathbf{R}^{n\times n}$

$$\nabla_{(Z_n, Z_n)}T = \begin{bmatrix} t_0 & t_{-1} & \cdot & t_{1-n} \\ t_1 & & & \\ \cdot & & \bigcirc & \\ t_{n-1} & & & \end{bmatrix} = X\Sigma X^T, \quad X = \begin{bmatrix} t_0 & 0 \\ t_1 & t_1 \\ \cdot & \cdot \\ t_{n-1} & t_{n-1} \end{bmatrix} / t_0^{1/2}, \quad \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

Choice of Displacement Operators.

Let $\{X, Y\}$ be a generator of length α of A with respect to F^f and F^b. If the matrix-vector multiplications $F^f\mathbf{v}$ and $F^b\mathbf{v}$ takes $f(n)$ operations, then our algorithm in Sec 3 will need $O(\alpha n f(n))$ operations. Therefore, our objective is to choose the "simplest" or sparse (to make $f(n)$ small) strictly lower triangular matrices F^f and F^b that also make α as small as possible. For a scalar $n \times n$ Toeplitz matrix, a natural choice of displacement operator is the simple $n \times n$ *shift matrix*, Z_n, with 1's along the first sub-diagonal, and 0's elsewhere.

For an $M \times N$ block-Toeplitz array with $r \times s$ blocks, the following choice of displacement operators gives the smallest α,

$$F^f = Z_{Mr}^M, \quad F^b = Z_{Ns}^N,$$

where Z_{kr}^{k} is a *block shift matrix*, i.e., a $k \times k$ array with $r \times r$ identity matrices on the 1st block subdiagonal and zeros elsewhere.

For block-Toeplitz or Toeplitz-block matrices, it is straight forward to obtain generators from the displacements by inspection (see Appendix 4 for closed forms), as we shall illustrate in several examples.

Example 2.1. *For the following block Toeplitz matrix A, called the "Hurwitz matrix", we can choose $F^f = Z^2$ and $F^b = Z$ to get a rank-2 displacement $\nabla_{(F^f, F^b)}A$,*

$$A = \begin{bmatrix} a_1 & a_3 & a_5 & a_7 & \cdot \\ a_0 & a_2 & a_4 & a_6 & \cdot \\ 0 & a_1 & a_3 & a_5 & \cdot \\ 0 & a_0 & a_2 & a_4 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix}, \quad \nabla_{(F^f, F^b)}A = \begin{bmatrix} a_1 & a_3 & a_5 & a_7 & \cdot \\ a_0 & a_2 & a_4 & a_6 & \cdot \\ & & & & \\ & \bigcirc & & & \\ & & & & \end{bmatrix}.$$

Note that $\nabla_{(F^f, F^b)}A = XY^T$, where

$$X = \begin{bmatrix} 1 & 0 & 0 & \cdot & \cdot \\ 0 & 1 & 0 & \cdot & \cdot \end{bmatrix}^T, \quad Y = \begin{bmatrix} a_1 & a_3 & a_5 & \cdot & \cdot \\ a_0 & a_2 & a_4 & \cdot & \cdot \end{bmatrix}^T. \qquad \square$$

For an $M \times N$ Toeplitz-block array with $m_i \times n_j$ Toeplitz matrices, we shall use the displacement operators,

$$F^f = \bigoplus_{i=1}^{M} Z_{m_i} \in \mathbf{R}^{m \times m}, \quad F^b = \bigoplus_{i=1}^{N} Z_{n_i} \in \mathbf{R}^{n \times n},$$

where $\bigoplus_{i}^{M}$ denotes the concatenated direct sum, i.e., $A \oplus B = \begin{bmatrix} A & O \\ O & B \end{bmatrix}$.

Example 2.2 *For the following Toeplitz-block matrix A, which arises in ARMA system identification problems [9], we can choose $F^f = Z_4$ and $F^b = Z_2 \oplus Z_3$ to obtain a rank-3 displacement,*

$$A = \begin{bmatrix} \beta_0 & \beta_{-1} & \gamma_0 & \gamma_{-1} & \gamma_{-2} \\ \beta_1 & \beta_0 & \gamma_1 & \gamma_0 & \gamma_{-1} \\ \beta_2 & \beta_1 & \gamma_2 & \gamma_1 & \gamma_0 \\ \beta_3 & \beta_2 & \gamma_3 & \gamma_2 & \gamma_1 \end{bmatrix}, \quad \nabla_{(F^f, F^b)} A = \begin{bmatrix} \beta_0 & \beta_{-1} & \gamma_0 & \gamma_{-1} & \gamma_{-2} \\ \beta_1 & & \gamma_1 & & \\ \beta_2 & & \gamma_2 & & \\ \beta_3 & & \gamma_3 & & \end{bmatrix}$$

Example 2.3. *Consider the matrix M,*

$$M = \begin{bmatrix} I & A & O \\ A^T & O & I \\ O & I & O \end{bmatrix}.$$

If A is an $M \times N$ block-Toeplitz array with $r \times s$ blocks, we could choose

$$F^f = F^b = Z_{Mr}^r \oplus Z_{Ns}^s \oplus Z_{Nr}.$$

If A is a Toeplitz-block array, for example, $A = \begin{bmatrix} T_1 \\ T_2 \end{bmatrix}$, where $T_1 \in \mathbf{R}^{m_1 \times n}$ and $T_2 \in \mathbf{R}^{m_2 \times n}$ then we could choose

$$F^f = F^b = Z_{m_1} \oplus Z_{m_2} \oplus Z_n \oplus Z_n.$$

Remark 2.1. One might check that the displacement operators for a block-Toeplitz matrix and a Toeplitz-block matrix are related by

$$Z_{kr}^r = P \cdot [\bigoplus_{i=1}^{r} Z_k] \cdot P,$$

where P is the permutation matrix that transforms the block-Toeplitz matrix into the Toeplitz-block matrix by pre and post-multiplication, vice versa.

3. Generalized Schur Algorithm and Partial Triangularization.

A fundamental method for triangular matrix factorization is the so-called *Schur reduction* process, which computes *Schur complements* of leading submatrices iteratively. Lev-Ari and Kailath [16-17] realized that the classical Schur algorithm amounts to Schur reduction, and gave several important generalizations including one for Hankel matrices. In the rest of this chapter, we shall further elaborate the ideas in [17].

Fast Schur Reduction using Displacement Structure.

Our fast algorithms will be based on the following theorem.

Theorem 3.1 *Let $\{X^{(1)}, Y^{(1)}\}$ be a generator of a rectangular matrix $A^{(1)} \in R^{m\times n}$ with respect to $\{F^f, F^b\}$. Also assume that $\{X^{(1)}, Y^{(1)}\}$ is proper with respect to a particular (pivoting) column, which we shall index as "pvt". If we denote the columns of $X^{(1)}$ and $Y^{(1)}$ by*

$$X_p^{(1)} = [\mathbf{x}_1^{(1)}, \cdots, \mathbf{x}_{pvt}^{(1)}, \cdots, \mathbf{x}_\alpha^{(1)}], \quad Y_p^{(1)} = [\mathbf{y}_1^{(1)}, \cdots, \mathbf{y}_{pvt}^{(1)}, \cdots, \mathbf{y}_\alpha^{(1)}],$$

then the matrix $A^{(2)}$ defined by

$$A^{(2)} \equiv A^{(1)} - \mathbf{x}_{pvt}^{(1)}\mathbf{y}_{pvt}^{(1)T}$$

has null first column and row, and has a generator $\{X^{(2)}, Y^{(2)}\}$, with respect to $\{F^f, F^b\}$ of the form

$$X^{(2)} = [\mathbf{x}_1^{(1)}, \cdot, \cdot, F^f\mathbf{x}_{pvt}^{(1)}, \cdots, \mathbf{x}_\alpha], \quad Y^{(2)} = [\mathbf{y}_1^{(1)}, \cdot, \cdot, F^b\mathbf{y}_{pvt}^{(1)}, \cdots, \mathbf{y}_\alpha^{(1)}].$$

Proof

$$\begin{aligned} A^{(2)} - F^f A^{(2)} F^{bT} &= [A^{(1)} - \mathbf{x}_{pvt}^{(1)}\mathbf{y}_{pvt}^{(1)T}] - F^f[A^{(1)} - \mathbf{x}_{pvt}^{(1)}\mathbf{y}_{pvt}^{(1)T}]F^{bT} \\ &= A^{(1)} - F^f A^{(1)} F^{bT} - \mathbf{x}_{pvt}^{(1)}\mathbf{y}_{pvt}^{(1)T} + F\mathbf{x}_{pvt}^{(1)}\mathbf{y}_{pvt}^{(1)T}F^{bT} \\ &= X^{(1)}Y^{(1)T} - \mathbf{x}_{pvt}^{(1)}\mathbf{y}_{pvt}^{(1)T} + F\mathbf{x}_{pvt}^{(1)}\mathbf{y}_{pvt}^{(1)T}F^{bT} \\ &= X^{(2)}Y^{(2)T} \end{aligned}$$

The first column and row of $A^{(2)}$ are null because

$$A^{(2)}\mathbf{e}_1 = [X^{(2)}Y^{(2)T}]\mathbf{e}_1 = 0, \quad \mathbf{e}_1^T A^{(2)} = \mathbf{e}_1^T[X^{(2)}Y^{(2)T}] = 0,$$

where we have used the facts that F^f and F^b are strictly lower triangular and $\{X^{(1)}, Y^{(1)}\}$ is proper. □

Before presenting the generalized Schur algorithm, we assume that we have a procedure called *MakeProper* that can convert a given generator $\{X, Y\}$ of $A \in \mathbf{R}^{m\times n}$ into a proper generator $\{X_p, Y_p\}$ (whenever A has a proper generator); this can be done with $O(\alpha m)$ operations as will be shown in Sec 5.

By applying the previous theorem using such a proper generator we can obtain a (possi-

bly non-proper) generator of $A^{(2)}$. By repeating this process, r times, we shall generate the matrices

$$
\begin{aligned}
A^{(r+1)} &= A^{(r)} - \mathbf{x}^{(r)}_{pvt_r}\mathbf{y}^{(r)T}_{pvt_r} \\
A^{(r)} &= A^{(r-1)} - \mathbf{x}^{(r-1)}_{pvt_{r-1}}\mathbf{y}^{(r-1)T}_{pvt_{r-1}} \\
&\quad \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \\
A^{(2)} &= A^{(1)} - \mathbf{x}^{(1)}_{pvt_1}\mathbf{y}^{(1)T}_{pvt_1}.
\end{aligned}
$$

It turns out that this process gives a *partial triangular factorization* of $A^{(1)}$; this follows by noting that

$$
\begin{aligned}
A^{(1)} &= \sum_{i=1}^{r} \mathbf{x}^{(i)}_{pvt_i}\mathbf{y}^{(i)T}_{pvt_i} + A^{(r+1)} \\
&= \begin{bmatrix} \mathbf{x}^{(1)}_{pvt_1} & \cdots & \mathbf{x}^{(r)}_{pvt_r} \end{bmatrix} \begin{bmatrix} \mathbf{y}^{(1)T}_{pvt_1} & & \\ & \ddots & \\ & & \mathbf{y}^{(r)T}_{pvt_r} \end{bmatrix} + A^{(r+1)}, \quad A^{(r)} = \begin{bmatrix} 0 & 0 \\ 0 & S^{(r+1)} \end{bmatrix}.
\end{aligned}
$$

Remark 3.1. If we define $A^{(1)} = \begin{bmatrix} B & D \\ C & E \end{bmatrix}$ then it is easy to check that $S^{(r+1)} = E - CB^{-1}D$. The matrix $S^{(r+1)}$ is called the *Schur complement* of B in $A^{(1)}$. Notice that the above process also gives a generator of $S^{(r+1)}$.

Remark 3.2. The above r step partial triangularization breaks down if and only if there is a singular leading principal submatrix of order less than or equal to r; we shall assume that this is not so.

The above procedure can be summarized in the following algorithm, which we shall call a *generalized Schur algorithm.*

Algorithm (Generalized Schur Algorithm)

Input: A generator $\{X, Y\}$ of $A \in \mathbf{R}^{m\times n}$ with respect to $\{F^f, F^b\}$.

Output: (i) Partial triangular factors $L \in \mathbf{R}^{m\times r}$ and $U \in \mathbf{R}^{r\times n}$ of A.

(ii) A generator $\{X, Y\}$ of the Schur complement of the $r \times r$ leading principal submatrix of A,

```
Procedure GeneralizedSchur
    begin
        for k := 1 to r do begin
            MakeProper;
            The kth column of L := x_pvt;     The kth row of U := y_pvt^T;
            Replace x_pvt with F^f x_pvt;     Replace y_pvt with F^b y_pvt;
        end
        return (L, U, {X, Y});
    end.
```

Note that the above procedure needs $O(\alpha m r)$ operations, where α is the length of the given generator, assuming that MakeProper takes $O(\alpha m)$ operations (see Sec 5).

Example 3.1 Triangularization of block-Toeplitz or Toeplitz-block matrices.

As trivial examples we can triangularize block-Toeplitz matrices or Toeplitz-block matrices simply by completing the above generalized Schur algorithm. Note that the multiplications $F^f \mathbf{x}_{pvt}$ and $F^b \mathbf{y}_{pvt}$ amount to shifting down "segments" of $\mathbf{x}_{pvt}$ and $\mathbf{y}_{pvt}$.

Example 3.2 Simultaneous Factorization of a Toeplitz matrix and its Inverse [4].

Consider the matrix

$$A = \begin{bmatrix} T & I \\ I & O \end{bmatrix}, \quad T = (t_{i-j}) \in R^{n \times n}, \quad t_0 = 1 \tag{6}$$

which has the following symmetric generator,

$$X = \begin{bmatrix} 1 & t_1 & \cdot & t_{n-1} & 1 & 0 & \cdot & 0 \\ 0 & t_1 & \cdot & t_{n-1} & 1 & 0 & \cdot & 0 \end{bmatrix}^T, \quad \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

After performing n steps of partial triangular factorization using the generalized Schur algorithm, we shall have

$$A = \begin{bmatrix} L \\ U \end{bmatrix} [\, L^T, U^T \,] + \begin{bmatrix} O & O \\ O & S \end{bmatrix}. \tag{7}$$

Now, one can check by comparing the entries of A in (6) and (7), that

$$T = LL^T, \quad T^{-1} = UU^T.$$

Remark 3.3 Recall that the classical Schur algorithm [20] gives only the factorization, $T = LL^T$, whereas the Levinson algorithm gives the factorization, $T^{-1} = UU^T$. If one only needs the factorization of T^{-1} the above method is slower than the Levinson algorithm [18]; a derivation of the Levinson algorithm from the generalized Schur algorithm can be found in Appendix 3.

Example 3.3 Orthogonalization of a fully windowed Toeplitz matrix.

Let $T = (t_{i-j}) \in \mathbf{R}^{m \times n}$, $m > n$ be a fully windowed Toeplitz matrix, i.e.,

$$t_{i-j} = 0, \quad \text{if } j > i, \quad \text{or} \quad i > m-n+j.$$

Then it is easy to check that $C \equiv T^T T$ is also an (unwindowed) Toeplitz matrix. Now consider the following matrix

$$A = \begin{bmatrix} T^T T & T^T \\ T & O \end{bmatrix}, \tag{8}$$

for which it can be checked that a generator of A is

$$X = \begin{bmatrix} c_0 & c_1 & \cdot & c_{n-1} & t_0 & t_1 & \cdot & t_{m-n} & 0 & \cdot & 0 \\ 0 & c_1 & \cdot & c_{n-1} & t_0 & t_1 & \cdot & t_{m-n} & 0 & \cdot & 0 \end{bmatrix}^T \in \mathbf{R}^{(m+n)\times 2}, \quad \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

After performing n steps of partial triangular factorization using the generalized Schur algorithm, we shall have

$$A = \begin{bmatrix} R^T \\ Q \end{bmatrix} [\, R, \; Q^T \,] + \begin{bmatrix} O & O \\ O & S \end{bmatrix}. \tag{9}$$

From (9), one can easily see that

$$T^T T = R^T R, \quad T = QR$$

so that Q is orthogonal because $R^T Q^T Q R = R^T R$.

Remark 3.4 Recall that the (fixed AR) lattice filter operates on a (data) sequence $\{t_i\}$ and performs orthogonalization to get the prediction errors without a knowledge of the covariance

matrix (i.e., $T^T T$) of the data sequence. The lattice filter is again a Levinson-like version of the above method.

4. Applications to non-Toeplitz matrices.

Now we shall consider various *extended* matrices M. By applying the generalized Schur algorithm in Sec 3 to judiciously chosen extended matrices, we can obtain interesting results including QR factorizations of block-Toeplitz or Toeplitz-block matrices.

Generators of extended matrices in this section can be also easily found by inspection. For closed forms of various generators, see Appendix 4.

A. QR factorization.

Let $A \in \mathbf{R}^{m \times n}$ be a block-Toeplitz or a Toeplitz-block matrix, and let us define the block matrix,

$$M \equiv \begin{bmatrix} -I & A & O \\ A^T & O & A^T \\ O & A & I \end{bmatrix}. \tag{10}$$

If we apply the generalized Schur algorithm to (10) then after the mth step we shall have a generator † of

$$\begin{bmatrix} A^T A & A^T \\ A & I \end{bmatrix}. \tag{11}$$

After another n-steps of partial triangularization, we shall have

$$\begin{bmatrix} A^T A & A^T \\ A & I \end{bmatrix} = \begin{bmatrix} R^T \\ Q \end{bmatrix} \cdot [\, R \;\; Q^T \,] + \begin{bmatrix} O & O \\ O & S \end{bmatrix}. \tag{12}$$

Now, one can check that the matrices Q and R in (13) satisfy

$$A = QR, \quad Q^T Q = I,$$

i.e., we obtained the QR factorization of A. This procedure will need $O(mn)$ flops.

† One can start with a generator of the extended matrix (11), as in Example 3.3. A closed-form expression for a generator of (11) for block-Toeplitz and Toeplitz-block matrix A can be found in Appendix.

If one wish to compute R^{-1} directly, then one can perform the $(m+n)$ steps of partial triangularization with the matrix,

$$M = \begin{bmatrix} -I & A & O \\ A^T & O & I \\ O & I & O \end{bmatrix}.$$

Note that

$$\begin{bmatrix} A^T A & I \\ I & O \end{bmatrix} = \begin{bmatrix} R^T \\ U \end{bmatrix} \cdot [\, R \;\; U^T \,] + \begin{bmatrix} O & O \\ O & S \end{bmatrix},$$

and therefore, $U = R^{-1}$ because $UR = I$.

B. Removing Forward Elimination in Square Systems.

If one's primary interest in the factorization is in solving a square system of equations,

$$A\mathbf{x} = \mathbf{b}, \tag{13}$$

then one might want to obtain the transformed right-side vector $\mathbf{y} \equiv L^{-1}\mathbf{b}$, during the course of the factorization process. This can be done by performing the following partial triangular factorization of the matrix M,

$$M \equiv \begin{bmatrix} A \\ -\mathbf{b}^T \end{bmatrix} = \begin{bmatrix} L \\ \mathbf{y}^T \end{bmatrix} \cdot L^T$$

whence the solution to (13) can be obtained by solving the triangular system of equations,

$$L^T\mathbf{x} = \mathbf{y}. \tag{14}$$

C. Removing Back-Substitution in Square Systems.

From a hardware implementation point of view, the back-substitution step in (14) can still be quite cumbersome [6]. This back-substitution process can also be eliminated by performing the partial factorization of the matrix,

$$M = \begin{bmatrix} A & -\mathbf{b} \\ I & 0 \end{bmatrix}.$$

Notice that the solution $\mathbf{x} = A^{-1}\mathbf{b}$ is the Schur complement of A in M. Therefore, after n steps

of partial triangularization, we shall have a generator of the solution, from which we can read out the solution; see [6] for details.

D. Solving Least Squares Problems without Back-substitution.

To solve the weighted least squares problem of minimizing

$$\|A_2(A_1\mathbf{x} - \mathbf{b})\|_2,$$

where A_1 and A_2 are block-Toeplitz or Toeplitz-block matrices, we form the matrix

$$M = \begin{bmatrix} -A_2 & A_1 & -\mathbf{b} \\ A_1^T & O & 0 \\ O & I & 0 \end{bmatrix}.$$

Now notice that the least squares solution,

$$\mathbf{x} = (A_1^T A_2^{-1} A_1)^{-1} A_2^T \mathbf{b} \tag{15}$$

is the Schur complement of the submatrix

$$\begin{bmatrix} -A_2 & A_1 \\ A_1^T & O \end{bmatrix}.$$

Therefore, after $m+n$ steps of the generalized Schur algorithm, we shall have a generator of the solution (15), from which the solution can be read out [6].

E. Regularization.

If the given Toeplitz least squares system is particularly ill-conditioned, it is meaningless to compute the exact (least squares) solution, since small perturbations of the matrix can cause very large perturbations in the solution. In such cases, we may solve the following regularized system [10], [19]

$$\begin{bmatrix} A \\ \eta I \end{bmatrix} \mathbf{x} = \begin{bmatrix} \mathbf{b} \\ 0 \end{bmatrix}$$

by partial triangularization of the matrix

$$M = \begin{bmatrix} \bigcirc & A & b \\ & \eta I & 0 \\ A^T \quad \eta I & O & 0 \\ \bigcirc & I & 0 \end{bmatrix}$$

After $m+2n$ steps of the generalized Schur algorithm, we shall have the solution. We may remark that this technique of regularization is known as the leakage method (adding white noise with variance η^2 to the data sample) in the signal processing literature (see e.g. [3]).

5. Construction of Proper Generators.

We shall present a method for constructing a proper generator using *spinors;* for a method using Householder matrices, see Appendix. A spinor $S_{(j|i)} \in \mathbf{R}^{\alpha\times\alpha}$ is defined as the identity matrix except for the following 4 entries,

$$[S_{(j|i)}]_{i,i} = c, \quad [S_{(j|i)}]_{i,j} = s_2, \quad [S_{(j|i)}]_{j,i} = -s_1, \quad [S_{(j|i)}]_{j,j} = c,$$

where $[A]_{i,j}$ denotes the (i, j)th element of the matrix A, and $c^2 + s_1 s_2 = 1$. The parameters $\{c, s_1, s_2\}$ will be called *Schur parameters.* Notice that the inverse of a spinor is also a spinor, viz., $S^{-1}_{(j|i)}$ is the identity matrix except for the following 4 entries,

$$[S^{-1}_{(j|i)}]_{i,i} = c, \quad [S^{-1}_{(j|i)}]_{i,j} = -s_2, \quad [S^{-1}_{(j|i)}]_{j,i} = s_1, \quad [S^{-1}_{(j|i)}]_{j,j} = c.$$

Let $\mathbf{x}^T \in \mathbf{R}^{1\times\alpha}$ and $\mathbf{y}^T \in \mathbf{R}^{1\times\alpha}$ be row vectors. Let c, s_1 and s_2 be chosen as

$$c = \left[\frac{x_i y_i}{x_i y_i + x_j y_j}\right]^{1/2}, \quad s_2 = -c\cdot\frac{x_j}{x_i}, \quad s_1 = -c\cdot\frac{y_j}{y_i},$$

and define $\mathbf{x}'$ and $\mathbf{y}'$ by

$$\mathbf{x}'^T = \mathbf{x}^T S_{(j|i)}, \qquad \mathbf{y}'^T = \mathbf{y}^T S^{-T}_{(j|i)}.$$

Then it is easy to check that $x_j' = y_j' = 0$, and $\mathbf{x}'^T\mathbf{y}' = \mathbf{x}^T\mathbf{y}$. We shall call the elements x_i and y_i *pivoting elements.* Therefore, by repeating this process we can *annihilate* all elements of $\mathbf{x}$ and $\mathbf{y}$ except the pivoting elements, resulting in

$$[\,0, \cdots 0, x_i, 0, \cdots, 0\,] = \mathbf{x}^T \prod_{j\neq i}^{\alpha} S_{(j|i)}, \qquad [\,0, \cdots 0, y_i, 0, \cdots, 0\,] = \mathbf{y}^T \prod_{j\neq i}^{\alpha} S^{-T}_{(j|i)}.$$

An arbitrary choice of pivoting element or an arbitrary *ordering* of annihilation, might result in $[1 + \frac{x_j y_j}{x_i y_i}] \leq 0$, for which real spinors do not exist. The following function returns an index "pvt", and two sets of indices, FIRST and NEXT; if we annihilate the elements in $\mathbf{x}^T$ and $\mathbf{y}^T$ whose indices are given in the set FIRST, pivoting with the element x_{pvt} and y_{pvt}, before the annihilation of the elements given in the set NEXT, then it is not hard to see that $[1 + \frac{x_j y_j}{x_{pvt} y_{pvt}}] \leq 0$.

```
Procedure FindOrdering
    begin
        Compute γ_i = x_i y_i for all 1 ≤ i ≤ α
        s := Σγ_i;
        Pset := { i | γ_i > 0 }; Nset := { i | γ_i < 0 }; Zset := { i | γ_i = 0 };
        if s > 0 then
            pvt := any i ∈ Pset;
            FIRST := Pset; NEXT := Nset;
        else if s < 0 then
            pvt := any i ∈ Nset;
            FIRST := Nset; NEXT := Pset;
        else /* Cannot rotate */
            return (s);
        Add Zset either to FIRST or NEXT;
        return (pvt, FIRST, NEXT)
    end
```

With FindOrdering, we can summarize the procedure for constructing proper generators.

```
Procedure MakeProper
    begin
        x^T := first non-zero row of X; y^T := first non-zero row of Y;
        FindOrdering;
        if s == 0 then
            return ("A has a singular minor");
        for each j ∈ FIRST, and then for each j ∈ NEXT
            Determine S_(j|pvt) to annihilate x_j and y_j;
            X := XS_(j|pvt); Y := YS^-T_(j|pvt)
        end;
        return ({X, Y})
    end;
```

Remark 5.1 The quantity $s = \sum \gamma_i$ that is used to find the annihilation ordering is the product of the diagonal elements $l_{k,k} \cdot u_{k,k}$ of the partial triangular matrices L and U obtained by the generalized Schur algorithm. Therefore, $s > 0$ for positive- definite symmetric matrices, and $s < 0$ for negative-definite symmetric matrices. Hence, for these matrices we can choose a single column as a pivoting column throughout the triangularization process.

Some Special Cases.

If we are given a symmetric generator of a symmetric matrix A, i.e., if $Y = X\Sigma$, then the updating of Y in the above procedure is redundant, because the updated $\{X', Y'\}$ after annihilating a row is still symmetric. To see this, let

$$\mathbf{x}^T = \mathbf{y}^T = [x_{pvt}, y_j].$$

Then the spinor that annihilates x_j will reduce to a *Givens rotation,*

$$G_{(j|pvt)} = \begin{bmatrix} c & -s \\ s & c \end{bmatrix}, \quad c^2 + s^2 = 1$$

On the other hand, if

$$\mathbf{u}^T = [u_{pvt}, u_j], \quad \mathbf{v}^T = [u_{pvt}, -u_j]$$

the spinor will become a *hyperbolic rotation,*

$$H_{(j|pvt)} = \begin{bmatrix} ch & -sh \\ -sh & ch \end{bmatrix}, \quad ch^2 - sh^2 = 1$$

Notice that Givens and hyperbolic rotations preserve the symmetry of the updated generator, i.e.,

$$YS^{-T} = Y' = X'\Sigma, \quad X' = XS, \quad S\text{: a Givens or hyperbolic rotation.}$$

As another special case of spinors, consider the two row vectors

$$\mathbf{u}^T = [u_{pvt}, u_j], \quad \mathbf{v}^T = [v_{pvt}, 0].$$

For this case, the spinor that annihilates u_j will reduce to the usual *elimination* matrix,

$$E_{(j|pvt)} = \begin{bmatrix} 1 & -\kappa_E \\ 0 & 1 \end{bmatrix}, \quad \kappa_E = \frac{u_j}{u_{pvt}}. \tag{16}$$

Example 5.1 *The celebrated "Routh procedure" [11] for stability testing is just a triangularization using our generalized Schur algorithm of the Hurwitz matrix of Example 2.1.*

Example 5.2 *Consider the following Toeplitz-block matrix called the "Sylvester matrix"* [11],

$$S = \begin{bmatrix} a_0 & & & & b_0 & & & \\ a_1 & a_0 & & & b_1 & b_0 & & \\ . & a_1 & . & & . & b_1 & . & \\ a_n & . & . & a_0 & b_m & . & . & b_0 \\ & a_n & . & a_1 & & b_m & . & b_1 \\ & & . & . & & & . & . \\ & & & a_n & & & & b_m \end{bmatrix} \in \mathbf{R}^{(m+n)\times(m+n)} \tag{17}$$

A nonsingular Sylvester matrix is always strongly nonsingular, and therefore, we can check whether a Sylvester matrix is singular or not by using the generalized Schur algorithm. The matrix S in (17) has a generator, with respect to $\{Z_{m+n}, Z_m \oplus Z_n\}$,

$$X = \begin{bmatrix} a_0 & a_1 & \cdot & a_n & 0 & 0 & \cdot & 0 \\ b_0 & b_1 & \cdot & \cdot & b_m & 0 & \cdot & 0 \end{bmatrix}^T, \quad Y = \begin{bmatrix} 1 & 0 & \cdot & 0 & 0 & & \cdot & 0 \\ 0 & & \cdot & 0 & 1 & 0 & \cdot & 0 \end{bmatrix}^T \tag{18}$$

When we apply the reduction technique to X and Y in (18), the spinors during the first n steps will be just the elimination matrices of (16).

Remark 5.2 For the triangular factorization of a block-Toeplitz matrix, one may use the *block-spinor,*

$$S = \begin{bmatrix} I & K_1 \\ -K_2^T & I \end{bmatrix} U^{-1}, \quad S^{-T} = \begin{bmatrix} I & K_2 \\ -K_1^T & I \end{bmatrix} L^{-T}, \quad LU = \begin{bmatrix} I + K_1 K_2^T & O \\ O & I + K_2^T K_1 \end{bmatrix}$$

to annihilate $X_{1,2}$, pivoting with $X_{1,1}$, by choosing $K = X_{1,1}^{-1} X_{2,1}$. However, the use of matrix inversion in the block rotation makes the control flow in most hardware implementations complicated, and therefore, the use of block rotations is often discouraged; our recommendation is to use the generalized Schur algorithm, which only operates on selected columns of scalars.

6. Concluding Remarks.

We have shown how to obtain triangular factorization and QR factorization of Toeplitz-block or block-Toeplitz matrices in $O(mn)$ flops. Our method is based on the displacement structure properties of matrices. We also presented some other applications of our algorithm.

We have generalized earlier definitions (see e.g. [14-17]) of the displacement for Toeplitz-like matrices and presented a correspondingly generalized Schur algorithm for their factorization. The extended definition allows us to handle block-Toeplitz and Toeplitz-block matrices and Schur complements with respect to the leading (block) entries of such matrices. Composite matrices obtained as products and inverses of Toeplitz matrices can be nicely handled by formulating them as Schur complements of entries in a suitably defined block-Toeplitz matrix. Some interesting examples were given in Sec 4; Several of them will be considered in other ways in other chapters in this thesis.

We also mention that displacement structure can also be introduced for Hankel and Hankel-like matrices (see e.g. [17]) and also Vandermonde-like matrices; analogs of the (generalized) definitions, algorithms and applications in this chapter have also been obtained for these matrices (see Chapter 4 or [5]).

APPENDIX 1.

Finding Generators of Matrices.

Once we have the displacement of a matrix, we can obtain a generator of the matrix by representing each pair of non-zero columns and rows that crosses at the main diagonal as a sum of two rank-one matrices. As an example, the displacement $\nabla_{(F^f, F^b)}$ in (10) can represented as follows;

$$\nabla_{(F^f, F^b)} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} \mathbf{e}_1^T + \mathbf{e}_1 \, [\, 0, \beta_{-1}, \gamma_0, \gamma_{-1}, \gamma_{-2} \,] + \begin{bmatrix} 0 \\ \gamma_1 \\ \gamma_2 \\ \gamma_3 \end{bmatrix} \mathbf{e}_3^T,$$

where $\mathbf{e}_i$ denotes the vector with 1 at the ith position and 0's elsewhere.

In general, the following procedure can be used to find a generator from the given displacement with $O(mn)$ flops.

Algorithm (Finding a generator)

Input: The displacement $\nabla_{(F^f, F^b)} A$
Output: A generator $\{X, Y\}$ of A

```
Procedure FindGenerator
    X := φ; Y := φ;
    while there is non-zero column or row
        for each pair of a column u and a row v^T that crosses in the i th position of the main
        diagonal of ∇_(F^f,F^b) A begin
            if u_i ≠ 0 then begin
                ũ := u/u_i^(1/2); ū := u except ū_i = 0;
                ṽ := v/u_i^(1/2); v̄ := v except v̄_i = 0;
            end
            else begin
                ũ := u except ũ_i = 1/2; ū := u except ū_i = −1/2;
                ṽ := v except ṽ_i = 1/2; v̄ := v except v̄_i = −1/2;
            end;
            X := [X, ũ, ū]; Y := [Y, ṽ, −v̄];
            Remove u and v;
        end;

        for each an unpaired i th column u begin
            X := [X, u]; Y := [Y, e_i];
            Remove u and v;
        end

        for each an unpaired j th row v^T begin
            X := [X, e_j]; Y := [Y, v];
            Remove u and v;
        end
        return {X, Y}
    end
```

APPENDIX 2.

Construction of a Proper Generator using Householder matrices.

The matrix E of the form,

$$E = I - 2\mathbf{u}\mathbf{v}^T, \quad \mathbf{u}^T\mathbf{v} = 1$$

will be called a *Householder matrix*. If $\mathbf{u} = \mathbf{v}$, then the matrix E reduces to the usual (orthogonal) Householder matrix. One can check that E has self-inverse, i.e.,

$$E = E^{-1}.$$

Let $\mathbf{x} \in \mathbf{R}^{m\times\alpha}$, $\mathbf{y} \in \mathbf{R}^{n\times\alpha}$ and $\mathbf{x}' \in \mathbf{R}^{m\times\alpha}$, $\mathbf{y}' \in \mathbf{R}^{n\times\alpha}$ are given vectors that satisfy

$$\mathbf{x}^T\mathbf{y} = \mathbf{x}'^T\mathbf{y}', \quad \mathbf{x}^T\mathbf{y}' = \mathbf{x}'^T\mathbf{y}.$$

For such vectors, we shall show how to find E such that

$$\mathbf{x}'^T = \mathbf{x}^T E, \quad \mathbf{y}'^T = \mathbf{y}^T E^T. \tag{A2.1}$$

To to this, first notice that (A2.1) can be re-written as

$$\mathbf{x}'^T = \mathbf{x}^T - 2\beta_1\mathbf{v}^T, \quad \mathbf{y}'^T = \mathbf{y}^T - 2\beta_2\mathbf{u}^T, \quad \beta_1 \equiv \mathbf{x}^T\mathbf{u}, \quad \beta_2 \equiv \mathbf{y}^T\mathbf{v}$$

so that

$$\mathbf{v} = [\mathbf{x} - \mathbf{x}']/2\beta_1, \quad \mathbf{u} = [\mathbf{y} - \mathbf{y}']/2\beta_2 \tag{A2.2}$$

Therefore,

$$\mathbf{y}^T\mathbf{v} = \beta_2 = \mathbf{y}^T[\mathbf{x} - \mathbf{x}']/2\beta_1, \quad \mathbf{x}^T\mathbf{u} = \beta_2 = \mathbf{x}^T[\mathbf{y} - \mathbf{y}']/2\beta_2$$

Hence, the Householder matrix E where $\mathbf{u}$ and $\mathbf{v}$, and therefore β_1 and β_2 are chosen such that

$$2\beta_1\beta_2 = \mathbf{y}^T[\mathbf{x} - \mathbf{x}'] = \mathbf{x}^T[\mathbf{y} - \mathbf{y}']$$

will satisfy (A2.2).

Let $\{X, Y\}$ be a non-proper generator, and let $\mathbf{x}^T$ and $\mathbf{y}^T$ denote the top-most non-zero rows of X and Y. To make $\{X, Y\}$ be proper, we choose a pivoting column, and post- multiply X and Y with E and E^T, respectively, so that

$$\mathbf{x}'^T = \mathbf{x}E = [0, \cdots, 0, x_{pvt}', 0, \cdots, 0]^T, \quad \mathbf{y}'^T = \mathbf{y}E^T = [0, \cdots, 0, y_{pvt}', 0, \cdots, 0]^T.$$

We summarize this procedure below.

```
Procedure MakeProper
    begin
        x^T := first non-zero row of X; y^T := first non-zero row of Y;
        s := x^T y;
        if s ≡ 0 then
            return ("Matrix has a singular minor");
        Choose an appropriate column as pivoting column;
        x_pvt' := [s·x_pvt/y_pvt]^1/2;    y_pvt' := x_pvt'·y_pvt/x_pvt;
        β1 := 1;    β2 := [x − x']^T y/2;
        u := [y − y']/2β2;    v := [x − x']/2;
        X := X − 2X u v^T;    Y := Y − 2Y v u^T;
        return ({X, Y});
    end.
```

APPENDIX 3.

1. Derivation of Levinson Algorithm from generalized Schur Algorithm

Let $X^{(k)}$ be the generator obtained after the kth step ($k \le n$) of partial triangularization, starting with the generator $X^{(1)}$;

$$X^{(k)} = \begin{bmatrix} 0 & \cdot & 0 & l_{k,k} & l_{k+1,k} & \cdot & l_{n,k} & u_{1,k} & \cdot & u_{k,k} & 0 & \cdot & 0 \\ 0 & \cdot & 0 & 0 & w_{k+1,k} & \cdot & w_{n,k} & u_{k,k} & \cdot & u_{1,k} & 0 & \cdot & 0 \end{bmatrix}^T. \tag{A3.1}$$

If we can obtain $l_{k,k}$ and $w_{k+1,k}$ from $u_{i,j}$'s in (A3.1), then we can compute the Schur parameters for the next hyperbolic rotation, and apply the rotation only to the bottom part of (A3.1);

$$\begin{bmatrix} u_{1,k} & \cdot & \cdot & u_{k,k} & 0 & \cdot & \cdot & 0 \\ u_{k,k} & \cdot & \cdot & u_{1,k} & 0 & \cdot & \cdot & 0 \end{bmatrix}.$$

First notice that

$$l_{k,k} = \prod_{i=1}^{k} (ch_i^2 - sh_i^2)^{1/2} = \prod_{i=1}^{k} (1 - k_i^2)^{1/2}, \quad k_i \equiv \frac{sh_i}{ch_i}. \tag{A3.2}$$

Also it is easy to check that

$$\begin{bmatrix} 1 & t_1 & & t_{n-1} \\ t_1 & 1 & \ddots & t_{n-2} \\ t_2 & t_1 & \ddots & \\ \cdot & \cdot & \ddots & \\ t_{n-1} & t_{n-2} & & 1 \end{bmatrix} \begin{bmatrix} 1 & u_{2,2} & & u_{n,n} \\ 0 & u_{1,2} & & u_{n-1,n} \\ & 0 & \ddots & \\ 0 & 0 & & u_{1,n} \end{bmatrix} = \begin{bmatrix} 1 & l_{2,2} & l_{3,3} & \cdot & l_{n-1,n-1} & l_{n,n} \\ w_{2,1} & 0 & 0 & \text{———} & 0 & 0 \\ w_{3,1} & w_{3,2} & 0 & & & \\ \cdot & \cdot & w_{4,3} & \ddots & & \vdots \\ \cdot & \cdot & \cdot & & 0 & \\ w_{n,1} & w_{n,2} & w_{n,3} & \cdot & w_{n,n-1} & 0 \end{bmatrix}.$$

Hence

$$w_{k+1,k} = [\, t_k, t_{k-1}, \cdots, 1 \,] \begin{bmatrix} u_{k,k} \\ u_{k-1,k} \\ \cdot \\ \cdot \\ u_{1,k} \end{bmatrix}, \tag{A3.3}$$

and therefore, $k_k = w_{k+1,k}/l_{k,k}$ can be computed by (A3.2) and (A3.3) yielding the Levinson algorithm.

2. Derivation of Lattice filtering Algorithm from generalized Schur Algorithm.

Let $X^{(k)}$ be the generator obtained after the kth step ($k \le n$) of partial triangularization, starting with the generator $X^{(1)}$;

$$X^{(k)} = \begin{bmatrix} 0 & \cdot & l_{k,k} & l_{k+1,k} & \cdot & l_{n,k} & b_{1,k} & \cdot & b_{m-n+k-1,k} & 0 & \cdot & 0 \\ 0 & \cdot & 0 & w_{k+1,k} & \cdot & w_{n,k} & f_{1,k} & \cdot & f_{m-n+k-1,k} & 0 & \cdot & 0 \end{bmatrix}^T,$$

where

$$[b_{1,1}, \cdots, b_{m-n,1}] = [f_{1,1}, \cdots, f_{m-n,1}] = [t_0, t_1, \cdots, t_{m-n}].$$

Again it is easy to see that

$$\begin{bmatrix} t_0 & t_1 & & & t_{n-1} & 0 & \\ 0 & t_0 & t_1 & & & t_{n-1} & \\ \cdot & 0 & \ddots & \ddots & & & \ddots \\ \cdot & \cdot & & t_0 & t_1 & & \\ 0 & 0 & & 0 & t_0 & & t_{n-1} \end{bmatrix} \begin{bmatrix} f_{1,1} & f_{1,2} & \cdots & f_{1,n} \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ f_{m-n,1} & \cdot & \cdots & \cdot \\ 0 & f_{m-n+1,1} & \cdots & \\ \cdot & \cdot & \cdots & \cdot \\ 0 & 0 & \cdots & f_{m,n} \end{bmatrix}$$

$$= \begin{bmatrix} 1 & l_{2,2} & l_{3,3} & \cdot & l_{n-1,n-1} & l_{n,n} \\ w_{2,1} & 0 & 0 & \text{——} & 0 & 0 \\ w_{3,1} & w_{3,2} & 0 & & & \vdots \\ \cdot & \cdot & w_{4,3} & \ddots & & \\ \cdot & \cdot & \cdot & & 0 & \\ w_{n,1} & w_{n,2} & w_{n,3} & \cdot & w_{n,n-1} & 0 \end{bmatrix}. \tag{A3.4}$$

Using (A3.4) and the fact that $T^T Q = R^T$, one can show that

the ith diagonal element of $Q^T S = -k_i$.

Therefore, we have an alternative way of computing the reflection coefficients,

$$k_k = \frac{\mathbf{f}_k^T \mathbf{b}_k}{\mathbf{b}_k^T \mathbf{b}_k}, \quad \mathbf{b}_k = [b_{1,k}, \cdots, b_{m-n+k-1,k}]^T, \quad \mathbf{f}_k = [f_{1,k}, \cdots, f_{m-n+k-1,k}]^T.$$

APPENDIX 4.

We shall give generators of some of important matrices explicitly. By using these generators, one would not need to use the procedure FindGenerator in Appendix 1, and for those matrices the operation count will reduce significantly.

Lemma A4-1 Generator for block Toeplitz matrices.

For the matrix A_1 in (1a), a generator $\{X, Y\}$ of A_1 with respect to $\{Z_{Nr}^r, Z_{Nr}^r\}$ is given by

$$X = \begin{bmatrix} I_r & B_1 B_0^{-1} & \cdots & B_{M-1} B_0^{-1} \\ O & B_1 B_0^{-1} & \cdots & B_{M-1} B_0^{-1} \end{bmatrix}^T, \quad Y = \begin{bmatrix} B_0 & B_{-1} & \cdots & B_{-N+1} \\ O & -B_{-1} & \cdots & -B_{-N+1} \end{bmatrix}^T.$$

Proof. By applying one step of (block) Gaussian elimination, we have

$$A - Z_{Nr}^r A Z_{Nr}^{rT} = \mathbf{x}_1 \mathbf{y}_1^T - K,$$

where $K = \mathbf{x}_1^{(1)} \mathbf{y}_1^{(1)T}$ is the Schur complement of A_0. □

Lemma A4-2 Generator for Toeplitz block matrices.

Let $A_2 \in R^{n \times n}$ be an $M \times N$ array of Toeplitz matrices, $T_{i,j} \in \mathbf{R}^{m_i \times n_j}$, where $\sum_{i=1}^{M} m_i = \sum_{i=1}^{N} n_i = n$. Let $\mathbf{a}_{i,j}$ denote the first column of $T_{i,j}$ and $\mathbf{b}_{i,j}^T$ denote the first row of $T_{i,j}$, with the first element equals to zero. Then a generator of A_2, $\{X, Y\}$ with respect to $\{\bigoplus_{i=1}^{M} Z_{m_i}, \bigoplus_{i=1}^{N} Z_{n_i}\}$ is given by

$$X = \begin{bmatrix} \mathbf{a}_{1,1} & \mathbf{a}_{1,2} & \cdot & \mathbf{a}_{1,N} & \mathbf{e}_1 & \cdot & \cdot & \cdot \\ \mathbf{a}_{2,1} & \mathbf{a}_{2,2} & \cdot & \mathbf{a}_{2,N} & \cdot & \mathbf{e}_1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \mathbf{a}_{M,1} & \mathbf{a}_{M,2} & \cdot & \mathbf{a}_{M,N} & \cdot & \cdot & \cdot & \mathbf{e}_1 \end{bmatrix}, \quad \mathbf{Y} = \begin{bmatrix} \mathbf{e}_1 & \cdot & \cdot & \cdot & \mathbf{b}_{1,1} & \mathbf{b}_{2,1} & \cdot & \mathbf{b}_{M,1} \\ \cdot & \mathbf{e}_1 & \cdot & \cdot & \mathbf{b}_{1,2} & \mathbf{b}_{2,2} & \cdot & \mathbf{b}_{M,2} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \mathbf{e}_1 & \mathbf{b}_{1,N} & \mathbf{b}_{2,N} & \cdot & \mathbf{b}_{M,N} \end{bmatrix}$$

Proof. The matrix $A - F_1 A F_2^T$ has non-zero elements only on the 1st row and 1st columns of each block. Among other possibilities, if one assigns vertical strips to $\mathbf{x}_i$ and the rest horizontal strips to $\mathbf{y}_i$, then the above generator results. □

Lemma A4-3 Generator for orthogonalization of block Toeplitz matrices.

If A_1 is an $M \times N$ Toeplitz array of square matrices $B_i \in \mathbf{R}^{r \times r}$ as in (1a), then a generator $\{X, X\Sigma\}$ of $M = \begin{bmatrix} A_1^T A_1 & A_1^T \\ A_1 & I \end{bmatrix}$, with respect to $F = Z_{Nr}^r \oplus Z_{Mr}^r$, where $\Sigma = I_{2r} \oplus -I_{2r}$, is given by $X = \begin{bmatrix} W \\ V \end{bmatrix}$ where

$$W = \begin{bmatrix} A_1^T \begin{bmatrix} B_0 S \\ B_1 S \\ \cdot \\ B_{M-1} S \end{bmatrix}, & \begin{matrix} O \\ B_{-1} \\ \cdot \\ B_{-N+1} \end{matrix} & A_1^T \begin{bmatrix} O \\ B_1 S \\ \cdot \\ B_{M-1} S \end{bmatrix}, & \begin{matrix} O \\ B_{M-1} \\ \cdot \\ B_{M-N+1} \end{matrix} \end{bmatrix}, \quad V = \begin{bmatrix} B_0 S & I & B_0 S & O \\ B_1 S & O & B_1 S & O \\ \cdot & \cdot & \cdot & \cdot \\ B_{M-1} S & O & B_{M-1} S & O \end{bmatrix}$$

where $S = [(\sum_{i=0}^{M-1} B_i^T B_i)^{-1}]^{1/2}$.

Proof. This is a straight-forward extensions of Lemma 2 in [4]. □

Lemma A4-4 Generator for Orthogonalization of Toeplitz-block Row

Let $A = [\, T_1, T_2, \ldots, T_N \,]$, where $T_j \in \mathbf{R}^{m \times n_j}$ are Toeplitz. Let $\mathbf{a}_j$ be the first column of T_j, B_j^T be the first row of T_j with the first element equals to zero, and $\mathbf{c}_j^T$ be the last row of T_j shifted right by one position. a generator $\{X, X\Sigma\}$ of $\begin{bmatrix} A^T A & A^T \\ A & I \end{bmatrix}$ with respect to $F = [\bigoplus_{i=1}^{N} Z_{n_i}] \oplus Z_m$, where $\Sigma = I_{N+1} \oplus -I_{N+1}$, is given by

$$X = \begin{bmatrix} \mathbf{w}_1 & \mathbf{w}_2 & \cdot & \mathbf{w}_N & \mathbf{w}_{N+1} & \mathbf{u}_1 & \mathbf{u}_2 & \cdot & \mathbf{u}_N & \mathbf{w}_{2N+2} \\ \mathbf{a}_1/\|\mathbf{a}_1\| & \mathbf{a}_2/\|\mathbf{a}_2\| & \cdot & \mathbf{a}_N/\|\mathbf{a}_N\| & \mathbf{e}_1 & \mathbf{a}_1/\|\mathbf{a}_1\| & \mathbf{a}_2/\|\mathbf{a}_2\| & \cdot & \mathbf{a}_N/\|\mathbf{a}_N\| & 0 \end{bmatrix}$$

where

$$\mathbf{w}_i = A^{(i-1)^T}\mathbf{a}_i/\|\mathbf{a}_i\|, \quad \mathbf{w}_{N+1} = [\mathbf{b}_1^T, \mathbf{b}_2^T, \ldots, \mathbf{b}_N^T]^T,$$
$$\mathbf{u}_i = \mathbf{w}_i^{(r_i)}, \quad \mathbf{w}_{2N+2} = [\mathbf{c}_1^T, \mathbf{c}_2^T, \ldots, \mathbf{c}_N^T]^T,$$

and $A^{(i)}$ is the matrix obtained after setting the first columns of all blocks up to the ith block by null vectors.

Proof. Notice that the 1st row of $A^TA - FA^TAF^T$ is A_1^TA, the n_1th row is $A_2^TA^{(1)}$, and the (n_1+n_2)th row is $A_3^TA^{(2)}$ and so on. Therefore, to each of above rows, pre-multiply corresponding columns crossing on the diagonal to get rank-one matrices, and apply Cholesky factorization to each rank-one matrix. The first Cholesky factors are $\mathbf{w}_i$, $1 \le i \le N$, and the Schur complements are $\mathbf{w}_i$, $N+2 \le i \le 2N+1$. The rest part of the matrix $A^TA - FA^TAF^T$ is rank-one, $\mathbf{w}_{N+1}^T\mathbf{w}_{2N+2}$.

For V, notice that $K_{2N+2}^T(\mathbf{w}_i, F) - K_{2N+2}^T(\mathbf{w}_{N+i+1}, F)$ has nonzero elements $\|A_i\|$ on the diagonal of the T_i block. Therefore,

$$K_{2N+2}(\mathbf{v}_i, Z_m)K_{2N+2}^T(\mathbf{w}_i, F) + K_{2N+2}(\mathbf{v}_{N+i+1}, Z_m)K_{2N+2}^T(\mathbf{w}_{N+i+1}, F)$$

forms the lower-triangular part of the T_i block. $K_{2N+2}(\mathbf{v}_{N+1}, Z_m)K_{2N+2}^T(\mathbf{w}_{N+1}, F)$ forms the rest upper triangular parts of all blocks. □

Lemma A4-5 Generator for Orthogonalization of stacked Matrices.

Let $A^T = [A_1^T, A_2^T, \ldots, A_M^T]$, where $A_i \in R^{m_i \times n}$. Let $\{X_i, X_i\Sigma_i\}$, be a generator of $M^{(i)}$ with respect to $F = F_2 \oplus F_{m_i}$, where $M^{(i)} = \begin{bmatrix} A_i^TA_i & A_i^T \\ A_i & I \end{bmatrix}$. Then the matrix $M = \begin{bmatrix} A^TA & A^T \\ A & I \end{bmatrix}$ has a generator $\{X, X\Sigma\}$, with respect to $\bar{F} = F_2 \oplus [\bigoplus_{i=1}^{M} F_{m_i}]$ where

$$X = \begin{bmatrix} W \\ V \end{bmatrix}, \quad W = [X_1, \ldots, X_M], \quad V = Y_1 \oplus \cdots \oplus Y_M.$$

Proof Because $A^T A = \sum A_i^T A_i$, we have

$$A^T A - F_2 A^T A F_2^T = \sum [\, A_i^T A_i - F_2 A_i^T A_i F_2^T \,] = W \Sigma W^T.$$

Also

$$A - F A F_2^T = \begin{bmatrix} A_1 - F_{m_1} A_1 F_2^T \\ \vdots \\ A_M - F_{m_M} A_M F_2^T \end{bmatrix} = V W^T. \qquad \square$$

Lemma A4-6 Generator for Orthogonalization of Toeplitz block Matrices.

Let A be an $M \times N$ array of Toeplitz matrices $T_{ij} \in \mathbf{R}^{m_i \times n_j}$. Then the matrix $\begin{bmatrix} A^T A & A^T \\ A & I \end{bmatrix}$ *has a generator $\{X, X\Sigma\}$ with respect to $F_1 = Z_{m_1} \oplus Z_{m_2} \oplus \cdots \oplus Z_{m_M}$ and $F_2 = Z_{n_1} \oplus Z_{n_2} \oplus \cdots \oplus Z_{n_N}$, where*

$$X = [W^T, V^T]^T, \quad W = [\, W_1, \ldots, W_M \,], \quad V = V_1 \oplus V_2 \oplus \cdots \oplus V_N,$$
$$A_i^T A_i - F_2 A_i^T A_i F_2^T = W_i \Sigma_i W_i^T, \quad A_i - Z_{m_i} A_i F_2^T = V_i W_i^T$$
$$A_i = [\, T_{i,1}, T_{i,2}, \ldots, T_{i,N} \,].$$

Proof. Immediate from Lemma A4-4 and A4-5. $\square$

REFERENCES

[1]. E. Bareiss, *Numerical solution of linear equations with Toeplitz and vector Toeplitz matrices*, Numer. Math., 13: (1969), 404-424.

[2]. A. Bojanczyk, R. Brent and F. de Hoog, *QR factorization of Toeplitz matrices*, Numer. Math., 49 (1986), pp. 81-94.

[3]. J. Cioffi, *Limited precision effects in adaptive filtering*, IEEE Trans, on Circuit and Systems, 1988.

[4]. J. Chun, T. Kailath and H. Lev-Ari, *Fast parallel algorithms for QR and triangular factorization*, SIAM J. Sci. Stat. Comput., Nov., vol. 8, No. 6, (1987), pp. 899-913.

[5]. J. Chun and T. Kailath, *Displacement structure for Hankel- and Vandermonde-like matrices*, Preprint, 1988.

[6]. J. Chun, V. Roychowdury and T. Kailath, *Systolic Array for Solving Toeplitz Systems of Equations*, SPIE's 32nd Conference on Advanced Algorithms and Architecture for Signal Processing III, San Diego, Aug., 1988.

[7]. G. Cybenko, *A generalized orthogonalization technique with applications to time series analysis and signal processing* Math. Comp. vol 40, No. 161, (1983), pp. 323-336.

[8]. G. Cybenko, *Fast Toeplitz orthogonalization using inner products*, SIAM J. Sci. Stat. Comput., (1987).

[9]. P. Eykhoff, *System identification*, John Wiley and Sons, New York, 1974

[10]. L. Elden, *An algorithm for the regularization of ill-conditioned least squares problems*, SIAM J. Sci. Stat. Comput., vol 5, No. 1., (1984), pp. 237-254.

[11]. F. Gantmacher *The theory of matrices*, vol. 2, Chelsea Publishing Comp., New York, 1960.

[12]. T. Kailath, *Linear Systems* Prentice-Hall, Englewood Cliffs, New Jersey, 1980.

[13]. T. Kailath, *Signal processing in the VLSI era*, VLSI and Modern Signal Processing, S. Kung, H. Whitehouse and T. Kailath, eds., Prentice-Hall, Englewood Cliffs, New Jersey, 1985.

[14]. T. Kailath, *Signal processing applications of some moment problems*, Proceedings of Symposia in Applied Mathematics, vol. 37, 1987 pp. 71-109.

[15]. T. Kailath, S. Kung and M. Morf, *Displacement ranks of matrices and linear equations*, J. Math. Anal. Appl., 68 (1979) pp. 395-407. See also Bull. Amer. Math. Soc., 1 (1979),

pp. 769-773.

[16]. H. Lev-Ari and T. Kailath, *Lattice filter parameterizations and modeling of nonstationary process*, IEEE Trans. Inform. Theory, IT-30 (1984), pp. 2-16.

[17]. H. Lev-Ari and T. Kailath, *Triangular factorization of structured Hermitian matrices*, Operator Theory, Advances and Applications, vol. 18, Birkhäuser, Boston, pp. 301-324, (1986).

[18]. N. Levinson, *The Wiener RMS error criterion in filter design and prediction*, J. of Math. Phys. 25 (1947), 261-278.

[19]. H. Rutishauser, *Once again: The least squares problem*, Linear Alg. and its Appli., 1. (1968), pp. 471-478.

[20]. I. Schur, *Über Potenzreihen, die im Innern des Einheitskreises beschrankt sind*, J. für die Reine und Angewandte Mathematik, 147 (1917), pp. 205-232. English translation appears in Operator Theory, Advances and Applications, vol. 18, Birkhäuser, Boston, pp. 31-60, (1986).

[21]. D. Sweet, *Fast Toeplitz orthogonalization*, Numer. Math., 43 (1984), pp. 1-21.

Chapter 3.

Generalization of Gohberg-Semencul Formulas

Abstract

Gohberg and Semencul gave some elegant formulas for the inverse of a Toeplitz matrix as a difference of products of lower and upper- triangular Toeplitz matrices. There are several algebraic and analytic proof of these formulas. Here we give a "constructive" proof for the Gohberg-Semencul formulas, under the assumption that the matrices are strongly non-singular, i.e., all leading minors are nonzero. This assumption is stronger than necessary, but it enables fast $O(n^2)$ constructions for the entries in the Gohberg-Semencul formulas. Our method also gives a natural generalization of the formula to matrices with displacement structure.

1. Introduction.

If T is a Toeplitz matrix,

$$\mathbf{T} = (c_{i-j}) \in R^{n\times n},$$

then one can easily show that T has the following *displacement representation* [5, 13-16],

$$c_0\mathbf{T}=\begin{bmatrix} c_0 & 0 & 0 \\ c_1 & c_0 & \\ \cdot & \cdot & \\ \cdot & \cdot & 0 \\ c_{n-1} & c_{n-2} & c_0 \end{bmatrix}\begin{bmatrix} c_0 & c_{-1} & c_{1-n} \\ 0 & c_0 & c_{2-n} \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & c_{-1} \\ 0 & 0 & c_0 \end{bmatrix}-\begin{bmatrix} 0 & 0 & 0 & 0 \\ c_1 & 0 & \cdot & \cdot \\ \cdot & c_1 & \cdot & \cdot \\ \cdot & \cdot & 0 & 0 \\ c_{n-1} & c_{n-2} & c_1 & 0 \end{bmatrix}\begin{bmatrix} 0 & c_{-1} & c_{-2} & c_{1-n} \\ 0 & 0 & c_{-1} & c_{2-n} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & c_{-1} \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{1}$$

It is an interesting fact [5] that $\mathbf{T}^{-1}$ also has a similar displacement representation. To give an explicit formula, we assume that the matrix T and its $(n-1)\times(n-1)$ leading principal submatrix are nonsingular. Let z and v denote the first and last columns of $\mathbf{T}^{-1}$, respectively. Then it turns out that we can write

$$z_1\mathbf{T}^{-1}=\begin{bmatrix} z_1 & 0 & 0 \\ z_2 & z_1 & \\ \cdot & \cdot & \\ \cdot & \cdot & 0 \\ z_n & z_{n-1} & z_1 \end{bmatrix}\begin{bmatrix} v_n & v_{n-1} & v_1 \\ 0 & v_n & v_2 \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & v_{n-1} \\ 0 & 0 & v_n \end{bmatrix}-\begin{bmatrix} 0 & 0 & 0 & 0 \\ v_1 & 0 & \cdot & \cdot \\ \cdot & v_1 & \cdot & \cdot \\ \cdot & \cdot & 0 & 0 \\ v_{n-1} & v_{n-2} & v_1 & 0 \end{bmatrix}\begin{bmatrix} 0 & z_n & z_{n-1} & z_1 \\ 0 & 0 & z_n & z_2 \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & z_n \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2}$$

It also holds that $z_1 = v_n$. This formula was first given by Gohberg and Semencul in 1972 [7, p. 86], [10] as one of a set of slightly different formulas of the type (2) for $\mathbf{T}^{-1}$. We shall describe a variant in Sec 2 (see Remark 2 in Sec 2). Here it is interesting to note that a simple inspection of (2) yields the following recursive formula for the elements of $\mathbf{T}^{-1}$,

$$z_1[\mathbf{T}^{-1}]_{i+1,j+1} = z_1[\mathbf{T}^{-1}]_{i,j} + z_{i+1}v_{n-j} - v_i z_{n-j+1}, \quad [\mathbf{A}]_{i,j} \equiv (i,j) \text{ element of } \mathbf{A}, \tag{3}$$

which was in fact first derived by Trench [21] in 1964. Its relationship to the Gohberg-Semencul formulas and connections with the Christoffel-Darboux formulas for orthogonal polynomials on the unit circle were made in [16].

As one might expect from the above discussion, there can be several ways of establishing Gohberg-Semencul formulas. But all presently known proofs involve a certain amount of algebraic and analytical manipulation. In this chapter we shall describe what may be regarded as a "constructive" proof for Gohberg-Semencul formulas [7, p. 86, p.89] under the rather restrictive condition of *strongly non-singular* T, i.e., T with all leading minors nonsingular. On the other hand, with this assumption, there are "fast" $O(n^2)$ algorithms for actually computing the Gohberg-Semencul expressions.

Our proof follows the so-called "array method" discussed in [4]. In the array method, the triangular factors of $\mathbf{T}^{-1}$ and T are *simultaneously* obtained via a sequence of elementary hyperbolic rotations applied to a certain "pre-array" of scalars. We show in this note that a slight modification of the pre-array leads to Gohberg-Semencul formulas. Our approach can be extended to obtain formulas of the Gohberg-Semencul type for "close-to-Toeplitz" matrices.

The present contribution grew out of our studies on constructive algorithms [4] for fast triangular and orthogonal factorizations of matrices given in a so-called displacement representation (see [14]). We discovered that one of these constructions not only gave a Gohberg-Semencul formula for the inverse of a Toeplitz matrix, but also indicated a natural method of extending the formulas to general non-Toeplitz matrices.

The *displacement* of a matrix $\mathbf{A} \in R^{n \times n}$ has been defined as [14]

$$\nabla \mathbf{A} \equiv \mathbf{A} - \mathbf{Z}_n \mathbf{A} \mathbf{Z}_n^T, \tag{4}$$

where $\mathbf{Z}_n$ is the $n \times n$ lower shift matrix with ones on the first subdiagonal and zeros everywhere else. Suppose that $\nabla \mathbf{A}$ is expressed as a sum of outer products,

$$\nabla \mathbf{A} = \sum_{i=1}^{\alpha} \mathbf{x}_i \mathbf{y}_i^T.$$

Then it is easy to see, using the nilpotency of $\mathbf{Z}_n$, that A can be written as a sum of products,

$$\mathbf{A} = \sum_{i=1}^{\alpha} \mathbf{L}(\mathbf{x}_i)\mathbf{L}^T(\mathbf{y}_i), \tag{5}$$

where as before L(x) denotes a lower triangular Toeplitz matrix with the first column x. The expression in (5) is called the *displacement representation* of A. For symmetric matrices we can refine this to the form,

$$A = \sum_{i=1}^{p} L(x_i)L^T(x_i) - \sum_{i=p+1}^{p+q} L(x_i)L^T(x_i). \tag{6}$$

For a simple example, consider a symmetric Toeplitz matrix, $T = (c_{i-j})$, $c_0 = 1$. Then we can see that

$$\nabla T = T - Z_n T Z_n^T = \begin{bmatrix} 1 & c_1 & \cdots & c_n \\ c_1 & & & \\ \vdots & & \bigcirc & \\ c_n & & & \end{bmatrix} = x_1 x_1^T - x_2 x_2^T,$$

where

$$x_1 = [\ 1, c_1, \cdots, c_n\ ]^T, \quad x_2 = [\ 0, c_1, \cdots, c_n\ ]^T.$$

It follows, as noted in (6), that

$$T = L(x_1)L^T(x_1) - L(x_2)L^T(x_2). \tag{7}$$

The fact that the Gohberg-Semencul formula (3) for T^{-1} has a similar form as (7) is not an accident. In fact, it was shown in [13, 14] that if the matrix A has the form (5), then $\bar{I}A^{-1}\bar{I}$ (where $\bar{I}$ is the matrix with ones on the reverse-diagonal and zeros elsewhere) has the representation,

$$\bar{I}A^{-1}\bar{I} = \sum_{i=1}^{\alpha} L(x_i)L^T(y_i), \quad \text{or} \quad A^{-1} = \sum_{i=1}^{\alpha} L^T(x_i)L(y_i) \tag{8}$$

with certain vectors $\{x_i, y_i\}$. (We may note that if we insist on a lower-upper type representation for A^{-1}, we can always obtain this but with $\alpha + 2$ terms in general). The formula (8) would be a generalization of the Gohberg-Semencul formula to arbitrary matrices, except that the proof in [14] did not show how to actually construct a displacement representation of $\bar{I}A^{-1}\bar{I}$ or A^{-1} from a displacement representation of A.

This construction will be supplied in this chapter for a large class of matrices, and in principle for all matrices. We shall first give a constructive proof of Gohberg-Semencul formula in Sec 2. In Sec 3 we shall generalize the earlier notion of displacement representation. Then we shall show that a certain partial triangularization procedure of the 2×2 block matrix,

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} \end{bmatrix}, \quad \mathbf{A}_{1,1} = \text{strongly nonsingular}, \tag{9}$$

easily yields a displacement representation of the so called Schur-complement of $\mathbf{A}_{1,1}$, i.e., a representation

$$\mathbf{A}_{2,2} - \mathbf{A}_{2,1}\mathbf{A}_{1,1}^{-1}\mathbf{A}_{1,2} = \sum_{i=1}^{p} \mathbf{L}(\mathbf{x}_i)\mathbf{L}^T(\mathbf{y}_i) - \sum_{i=p+1}^{p+q} \mathbf{L}(\mathbf{x}_i)\mathbf{L}^T(\mathbf{y}_i).$$

As an example, we shall obtain the Gohberg-Semencul formula for a Toeplitz matrix by applying the procedure to the block matrix

$$\mathbf{A} = \begin{bmatrix} \mathbf{T} & \mathbf{I} \\ \mathbf{I} & \mathbf{O} \end{bmatrix},$$

where the Schur complement of $\mathbf{T}$ is just $-\mathbf{T}^{-1}$. Then in Sec 4, we shall apply the procedure in Sec 3 to various matrices, e.g.,

$$\begin{bmatrix} \mathbf{T}^T\mathbf{T} & \mathbf{I} \\ \mathbf{I} & \mathbf{O} \end{bmatrix}, \quad \begin{bmatrix} \mathbf{T}_1 & \mathbf{T}_2 \\ \mathbf{T}_2^T & \mathbf{O} \end{bmatrix}, \quad \begin{bmatrix} \mathbf{T}^T\mathbf{T} & \mathbf{T}^T \\ \mathbf{T} & \mathbf{I} \end{bmatrix}, \quad \begin{bmatrix} \mathbf{T}^T\mathbf{T} & \mathbf{T}^T \\ \mathbf{I} & \mathbf{O} \end{bmatrix},$$

to obtain *generalized Gohberg-Semencul formulas*, or, equivalently, *displacement representations*, of the matrices,

$$(\mathbf{T}^T\mathbf{T})^{-1}, \quad \mathbf{T}_2^T\mathbf{T}_1^{-1}\mathbf{T}_2, \quad \mathbf{T}(\mathbf{T}^T\mathbf{T})^{-1}\mathbf{T}^T, \quad (\mathbf{T}^T\mathbf{T})^{-1}\mathbf{T}^T.$$

2. A Constructive Proof of two Gohberg-Semencul Formulas.

Let us consider a symmetric positive definite Toeplitz matrix $\mathbf{T} = (c_{i-j}) \in R^{n \times n}$, $c_0 = 1$. Later, we shall indicate simple modifications of the proof for nonsymmetric but strongly nonsingular Toeplitz matrices. As shown in Sec 1, $\mathbf{T}$ has the representation,

$$\mathbf{T} = \mathbf{L}(\mathbf{x}_1)\mathbf{L}^T(\mathbf{x}_1) - \mathbf{L}(\mathbf{x}_2)\mathbf{L}^T(\mathbf{x}_2).$$

Let us define the *pre-array* Δ_0, and a diagonal matrix J,

$$\Delta_0 \equiv \begin{bmatrix} L(x_1) & O_n & L(x_2) \\ I_n & O_n & I_n \end{bmatrix} \in R^{2n\times 3n}, \quad J \equiv \begin{bmatrix} I_n & & \\ & I_n & \\ & & -I_n \end{bmatrix} \in R^{3n\times 3n}, \tag{10}$$

where O_n and I_n denotes the $n \times n$ null matrix, and the $n \times n$ identity matrix, respectively. Suppose that we can find a *J-orthogonal matrix* $\Theta \in R^{3n\times 3n}$, viz., one satisfying $\Theta J \Theta^T = J$, such that the *post-array* $\Delta_0\Theta$ has the form

$$\tilde{\Delta} \equiv \Delta_0\Theta = \begin{bmatrix} L & O_n & O_n \\ U & L(y_1) & L(y_2) \end{bmatrix}, \tag{11}$$

where L is a lower triangular matrix, and $L(y_1)$, $L(y_2)$ are lower triangular Toeplitz matrices whose first columns are some y_1 and y_2, respectively, while U is not a priori constrained in anyway. Then, because of the J-orthogonality of Θ,

$$\Delta_0 J \Delta_0^T = \tilde{\Delta} J \tilde{\Delta}^T,$$

which yields the following identities

$$L(x_1)L^T(x_1) - L(x_2)L^T(x_2) = T = LL^T, \tag{12a}$$

$$L(x_1) - L(x_2) = I_n = LU^T, \tag{12b}$$

$$UU^T + L(y_1)L^T(y_1) - L(y_2)L^T(y_2) = O_n. \tag{12c}$$

From (12b), we see that $L^{-1} = U^T$ (therefore, U is upper triangular). Now (12a) shows that

$$T^{-1} = L^{-T}L^{-1} = UU^T. \tag{13a}$$

Therefore, (12c) implies that T^{-1} has the following displacement representation,

$$T^{-1} = L(y_2)L^T(y_2) - L(y_1)L^T(y_1). \tag{13b}$$

Next we shall show how to construct Θ so as to obtain a post-array $\tilde{\Delta}$ of the form (11). This can be done in several ways, but perhaps the simplest is to construct Θ as a product of elementary hyperbolic rotations, which are J-orthogonal. (We could also use hyperbolic Householder reflections.) Hyperbolic rotations, $H_{i,j}(\kappa)$ are defined as identity matrices except for the following four entries,

$$[\mathbf{H}_{i,j}(\kappa)]_{i,i} = [\mathbf{H}_{i,j}(\kappa)]_{j,j} = \frac{1}{(1-\kappa^2)^{1/2}}, \qquad [\mathbf{H}_{i,j}(\kappa)]_{i,j} = [\mathbf{H}_{i,j}(\kappa)]_{j,i} = \frac{-\kappa}{(1-\kappa^2)^{1/2}}. \tag{14}$$

We call the pair of indices (i,j), the *plane of rotation.* The matrix $\mathbf{H}_{i,j}(\kappa)$ is real and finite when $|\kappa| < 1$. In signal processing applications, κ is often called a *reflection coefficient.* Let $\mathbf{w}^T$ be a row vector with $|w_i| > |w_j|$. We can *annihilate* w_j, *pivoting with* w_i, by post-multiplying $\mathbf{w}^T$ with $\mathbf{H}_{i,j}(w_j/w_i)$,

$$[w_1 \cdots w_i \cdots w_j \cdots w_n]\mathbf{H}_{i,j}(w_j/w_i) = [w_1 \cdots w_i' \cdots w_{j-1}\ 0\ w_{j+1} \cdots w_n].$$

For a moment, let us assume that the magnitude of *pivoting elements* is always greater than that of *pivoted elements* and therefore, that $|\kappa| < 1$. A lemma given in the Appendix shows that this assumption is always valid for a positive definite T.

A Simple Example.

Our construction is perhaps best followed with a simple example. Thus, before we describe the general procedure, we shall illustrate the details with a 3×3 symmetric positive definite Toeplitz matrix,

$$\mathbf{T} = \begin{bmatrix} 1 & \alpha_1 & \alpha_2 \\ \alpha_1 & 1 & \alpha_1 \\ \alpha_2 & \alpha_1 & 1 \end{bmatrix}, \qquad \mathbf{x}_1 = [1, \alpha_1, \alpha_2]^T, \qquad \mathbf{x}_2 = [0, \alpha_1, \alpha_2]^T.$$

We post-multiply the pre-array,

$$\Delta_0 = \begin{bmatrix} \mathbf{L}(\mathbf{x}_1) & \mathbf{O}_n & \mathbf{L}(\mathbf{x}_2) \\ \mathbf{I}_n & \mathbf{O}_n & \mathbf{I}_n \end{bmatrix} \in R^{6\times 9}, \tag{15}$$

with hyperbolic rotations $\mathbf{H}_{2,7}(\kappa_1)$ and $\mathbf{H}_{3,8}(\kappa_1)$, where $\kappa_1 \equiv \alpha_1$, to annihilate the 1st sub-diagonal of $\mathbf{L}(\mathbf{x}_2)$ pivoting with the diagonal of $\mathbf{L}(\mathbf{x}_1)$ (see (16) below). To preserve the Toeplitz structure at the (2, 3) block, we also apply a "dummy" hyperbolic rotation $\mathbf{H}_{4,9}(\kappa_1)$. This will introduce a nonzero element β_4 at the lower left corner in the (2, 2) block. These steps are illustrated below.

$$\Delta_0 = \left[\begin{array}{ccc|c|ccc} 1 & & & & & & \\ \alpha_1 & 1 & & & \alpha_1 & & \\ \alpha_2 & \alpha_1 & 1 & & \alpha_2 & \alpha_1 & \\ \hline 1 & & & & 1 & & \\ & 1 & & & & 1 & \\ & & 1 & & & & 1 \end{array}\right] \rightarrow \underset{\Delta_0 \cdot H_{2,7}(\kappa_1)}{\left[\begin{array}{ccc|c|ccc} 1 & & & & & & \\ \alpha_1 & \beta_1 & & & & & \\ \alpha_2 & \beta_2 & 1 & & \beta_3 & \alpha_1 & \\ \hline 1 & \beta_4 & & & \beta_5 & & \\ & \beta_5 & & & \beta_4 & 1 & \\ & & 1 & & & & 1 \end{array}\right]} \tag{16a}$$

$$\underset{\Delta_0 \cdot H_{2,7}(\kappa_1) \cdot H_{3,8}(\kappa_1)}{\left[\begin{array}{ccc|c|ccc} 1 & & & & & & \\ \alpha_1 & \beta_1 & & & & & \\ \alpha_2 & \beta_2 & \beta_1 & & \beta_3 & & \\ \hline 1 & \beta_4 & & & \beta_5 & & \\ & \beta_5 & \beta_4 & & \beta_4 & \beta_5 & \\ & & \beta_5 & & & \beta_4 & 1 \end{array}\right]} \rightarrow \underset{\Delta_0 \cdot H_{2,7}(\kappa_1) \cdot H_{3,8}(\kappa_1) \cdot H_{4,9}(\kappa_1)}{\left[\begin{array}{ccc|c|ccc} 1 & & & & & & \\ \alpha_1 & \beta_1 & & & & & \\ \alpha_2 & \beta_2 & \beta_1 & & \beta_3 & & \\ \hline 1 & \beta_4 & & & \beta_5 & & \\ & \beta_5 & \beta_4 & & \beta_4 & \beta_5 & \\ & & \beta_5 & \beta_4 & & \beta_4 & \beta_5 \end{array}\right]} \equiv \Delta_1 \tag{16b}$$

Now we post-multiply Δ_1 with the hyperbolic rotation $H_{3,7}(\kappa_2)$, where $\kappa_2 \equiv \beta_3/\beta_1$, to annihilate the remaining element β_3 in the (1, 3) block of Δ_1. Again to preserve the Toeplitz structure in the (2, 3) block, we apply two dummy hyperbolic rotations, $H_{4,8}(\kappa_2)$ and $H_{5,9}(\kappa_2)$ to obtain $\bar{\Delta}$.

$$\underset{\Delta_1 \cdot H_{3,7}(\kappa_2)}{\left[\begin{array}{ccc|c|ccc} 1 & & & & & & \\ \alpha_1 & \beta_1 & & & & & \\ \alpha_2 & \beta_2 & \gamma_1 & & & & \\ \hline 1 & \beta_4 & \gamma_2 & & \gamma_4 & & \\ & \beta_5 & \gamma_3 & & \gamma_3 & \beta_5 & \\ & & \gamma_4 & \beta_4 & \gamma_2 & \beta_4 & \beta_5 \end{array}\right]} \rightarrow \underset{\Delta_1 \cdot H_{3,7}(\kappa_2) \cdot H_{4,8}(\kappa_2) \cdot H_{5,9}(\kappa_2)}{\left[\begin{array}{ccc|c|ccc} 1 & & & & & & \\ \alpha_1 & \beta_1 & & & & & \\ \alpha_2 & \beta_2 & \gamma_1 & & & & \\ \hline 1 & \beta_4 & \gamma_2 & & \gamma_4 & & \\ & \beta_5 & \gamma_3 & \gamma_2 & \gamma_3 & \gamma_4 & \\ & & \gamma_4 & \gamma_3 & \gamma_2 & \gamma_3 & \gamma_4 \end{array}\right]}. \tag{16c}$$

$$= \begin{bmatrix} \mathbf{L} & \mathbf{O}_n & \mathbf{O}_n \\ \mathbf{U} & \mathbf{L}(y_1) & \mathbf{L}(y_2) \end{bmatrix} = \bar{\Delta}. \tag{17}$$

Then as noted before (cf. (13b)), we have

$$\mathbf{T}^{-1} = \mathbf{L}(\mathbf{y}_2)\mathbf{L}^T(\mathbf{y}_2) - \mathbf{L}(\mathbf{y}_1)\mathbf{L}^T(\mathbf{y}_1).$$

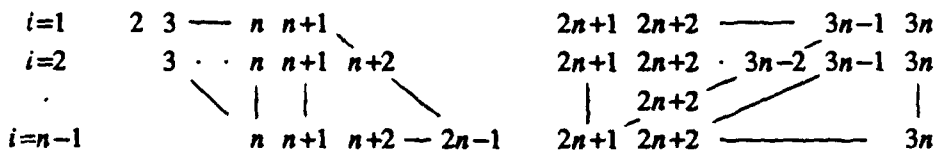
The general procedure.

In general, we successively annihilate the 1st, 2nd, $\cdots$, $(n-1)$th subdiagonals of the lower triangular matrix in the (1,3) block of Δ_0, by post-multiplying Δ_0 with a sequence of hyperbolic rotations. The procedure of annihilating the ith subdiagonal will be called the ith *sweep*, and the array obtained after the ith sweep will be denoted with Δ_i. In the ith sweep, we apply hyperbolic rotations on the following two sets of planes in Δ_{i-1},

$$A_i \equiv \{(i+1,2n+1), (i+2,2n+2), \cdots, (n,3n-i)\}, \tag{18a}$$

$$D_i \equiv \{(n+1,3n-i+1), (n+2,3n-i+2), \cdots, (n+i,3n)\}, \tag{18b}$$

where A_i and D_i stand for the sets of planes on which "annihilating" hyperbolic rotations and "dummy" hyperbolic rotations are applied, respectively. Thus, if we display (18) for each sweep, we have the pictorial representation:



pivoting columns pivoted columns

Within a given A_i (or D_i), any *ordering* of rotations can be chosen because the planes in (18a) (or (18b)) are "disjoint".

Note that Δ_0 has "Toeplitz structure" in the columns $(1, 2, \cdots, 2n)$ and $(2n+1, 2n+2, \cdots, 3n)$. Suppose that Δ_{i-1} has Toeplitz structure in the columns $(i, i+1, \cdots, 2n)$ and $(2n+1, 2n+2, \cdots, 3n)$. Then this Toeplitz structure allows us to choose hyperbolic rotations on the set A_i with identical reflection coefficients κ_i, to annihilate the ith subdiagonal elements in the (2,3) block of Δ_{i-1},

$$\Delta_{i-1}' \equiv \Delta_{i-1}\mathbf{H}_{A_i}(\kappa_i), \quad \mathbf{H}_{A_i}(\kappa_i) \equiv \mathbf{H}_{i+1,2n+1}(\kappa_i)\mathbf{H}_{i+2,2n+2}(\kappa_i) \cdots \mathbf{H}_{n,3n-i}(\kappa_i). \tag{19a}$$

Furthermore, by applying hyperbolic rotations on the set D_i with the same κ_i, we can keep the Toeplitz structure in the columns of $(i+1, i+2, \cdots, 2n)$ and $(2n+1, 2n+2, \cdots, 3n)$ in Δ_i,

$$\Delta_i \equiv \Delta_{i-1}\mathbf{H}_{D_i}(\kappa_i), \quad \mathbf{H}_{D_i}(\kappa_i) \equiv \mathbf{H}_{n+1,3n-i+1}(\kappa_i)\mathbf{H}_{n+2,3n-i+2}(\kappa_i)\cdots\mathbf{H}_{n+i,3n}(\kappa_i). \tag{19b}$$

In each sweep, the null (1,2) block is untouched, and the lower triangularity of the (1,1) block is maintained.

Hence, Δ_{n-1} will be the form of $\bar{\Delta}$ in (11), and will have Toeplitz structure in the columns of $(n, n+1, \cdots 2n)$ and $(2n+1, 2n+2, \cdots, 3n)$. Therefore, the diagonal elements of $\mathbf{L}(\mathbf{y}_1)$ will be zero and the first column of $\mathbf{L}(\mathbf{y}_1)$, (i.e., $\mathbf{y}_1$) will be identical to the last column of $\mathbf{U}$ shifted down by one position.

Now we shall show that the displacement representation (13b) obtained by the above construction is the (first) Gohberg-Semencul formula (2). Notice that

$$\mathbf{T}^{-1}\mathbf{e}_1 = [\mathbf{L}(\mathbf{y}_2)\mathbf{L}^T(\mathbf{y}_2) - \mathbf{L}(\mathbf{y}_1)\mathbf{L}^T(\mathbf{y}_1)]\mathbf{e}_1 = \mathbf{L}(\mathbf{y}_2)\mathbf{L}^T(\mathbf{y}_2)\mathbf{e}_1 = [\mathbf{L}(\mathbf{y}_2)]_{1,1}\mathbf{L}(\mathbf{y}_2)\mathbf{e}_1,$$
$$\mathbf{T}^{-1}\mathbf{e}_n = (\mathbf{U}\mathbf{U}^T)\mathbf{e}_n = [\mathbf{U}]_{n,n}\mathbf{U}\mathbf{e}_n,$$

where $\mathbf{e}_i$ denotes the vector with 1 at the ith position and 0 elsewhere. Therefore,

$$\text{First column of } \mathbf{L}(\mathbf{y}_2) = \mathbf{L}(\mathbf{y}_2)\mathbf{e}_1 = (\mathbf{T}^{-1}\mathbf{e}_1)/[\mathbf{L}(\mathbf{y}_2)]_{1,1} = z_1^{-1/2}\mathbf{z},$$
$$\text{First column of } \mathbf{L}(\mathbf{y}_1) = \mathbf{L}(\mathbf{y}_1)\mathbf{e}_1 = sd(\mathbf{U}\mathbf{e}_n) = sd(\mathbf{T}^{-1}\mathbf{e}_n)/[\mathbf{U}]_{n,n} = z_1^{-1/2}sd(\mathbf{v}),$$

where $sd(\mathbf{v})$ denotes the vector $\mathbf{v}$ shifted-down by one position, and we have used the easily verifiable fact that

$$z_1 = v_n = [\mathbf{L}(\mathbf{y}_2)]_{1,1}^2 = [\mathbf{U}]_{n,n}^2.$$

With these identifications, the displacement representation (13b) is exactly the Gohberg-Semencul formula (2).

Remark 1. The sequence of hyperbolic rotations, $\mathbf{H}_{A_i}(\kappa_i)$ (see (19) for the notation) introduces the ith superdiagonal in the (2,1) block, and the ith subdiagonal in the (2,3) block. On the other hand, $\mathbf{H}_{D_i}(\kappa_i)$ introduces the $(n-i)$th subdiagonal in the (2,2) block.

Remark 2. Readers may have noticed that our construction of $\tilde{\Delta}$, and therefore, the resulting displacement representation (13b) is not unique. This is because we can use *any* J-orthogonal transformation matrix Θ. In particular, we can apply extra hyperbolic rotations on D_n with *any* $|\kappa| < 1$, i.e., we can replace Δ_{n-1} by

$$\Delta_{n-1}\mathbf{H}_{D_n}(\kappa), \tag{20}$$

and still have the form (11). In fact, the second Gohberg-Semencul formula [7, p. 89] corresponds to the particular displacement representation (13b) obtained by choosing the reflection coefficient κ in (20) as the last reflection coefficient κ_n of any nonsingular $(n+1) \times (n+1)$ Toeplitz matrix whose $n \times n$ leading principal submatrix is $\mathbf{T}$.

Remark 3. Note that our construction of the Gohberg-Semencul formula needs $O(n^2)$ operations, because we need to keep track only two columns in Δ_i, and apply only $(n-1)$ different 2×2 hyperbolic rotations. Our construction here is closely related to the classical Schur algorithm [13, 18, 20], which has certain advantages for parallel computation. The first and last columns of $\mathbf{T}^{-1}$, which define the matrices $\mathbf{L}_1$ and $\mathbf{L}_2$ in the Gohberg-Semencul formula, can also be obtained with $O(n^2)$ operations by using the Levinson algorithm [5]. In fact, Trench [21] used this algorithm to obtain the "differential form" (3) of the Gohberg-Semencul formula.

3. Generlaized Displacement Representations and Schur Complements

First, we shall generalize the concept of displacement representation, and then give a fast triangularization algorithm for a certain 2×2 block matrix. During the triangularization procedure, displacement representations for the Schur complement of the (1, 1) block will naturally arise.

The following sum-of-products representation of a matrix $\mathbf{A} \in R^{m \times n}$ is called a (generalized) *displacement representation* of $\mathbf{A}$ with respect to the *displacement operators* $\{\mathbf{F}_1, \mathbf{F}_2\}$:

$$\mathbf{A} = \sum_{i=1}^{\alpha} \mathbf{K}_n(\mathbf{x}_i, \mathbf{F}_1)\mathbf{K}_n^T(\mathbf{y}_i, \mathbf{F}_2),$$

where $\mathbf{F}_1 \in R^{n\times n}$ and $\mathbf{F}_2 \in R^{n\times n}$ are nilpotent matrices of index less than or equal to n, i.e., $\mathbf{F}_1^n = \mathbf{F}_2^n = \mathbf{O}$, and $\mathbf{K}_n(\mathbf{x}_i, \mathbf{F}_1) \in R^{m\times n}$ and $\mathbf{K}_n(\mathbf{y}_i, \mathbf{F}_2) \in R^{n\times n}$ are the so called Krylov matrices,

$$\mathbf{K}_n(\mathbf{x}_i, \mathbf{F}_1) \equiv [\mathbf{x}_i, \mathbf{F}_1\mathbf{x}_i, \ldots, \mathbf{F}_1^{n-1}\mathbf{x}_i], \quad \mathbf{K}_n(\mathbf{y}_i, \mathbf{F}_2) \equiv [\mathbf{y}_i, \mathbf{F}_2\mathbf{y}_i, \ldots, \mathbf{F}_2^{n-1}\mathbf{y}_i].$$

The matrix pair, $\{ \mathbf{X}, \mathbf{Y} \}$, where $\mathbf{X} \equiv [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_\alpha]$ and $\mathbf{Y} \equiv [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_\alpha]$ is called a *generator* of $\mathbf{A}$ (with respect to $\{\mathbf{F}_1, \mathbf{F}_2\}$), and is denoted $G_\alpha(\mathbf{A}, \mathbf{F}_1, \mathbf{F}_2)$. The number α is called the *length* of the generator (with respect to $\{\mathbf{F}_1, \mathbf{F}_2\}$). A generator of $\mathbf{A}$ with the minimal possible length is called a *minimal* generator. The length of the minimal generator of $\mathbf{A}$ is called the *displacement rank* of $\mathbf{A}$ (with respect to $\{\mathbf{F}_1, \mathbf{F}_2\}$), and denoted as $\alpha(\mathbf{A}, \mathbf{F}_1, \mathbf{F}_2)$.

If the matrix $\mathbf{A}$ is symmetric, then $\mathbf{A}$ can be represented as

$$\mathbf{A} = \sum_{i=1}^{p} \mathbf{K}_n(\mathbf{x}_i, \mathbf{F})\mathbf{K}_n^*(\mathbf{x}_i, \mathbf{F}) - \sum_{i=p+1}^{p+q} \mathbf{K}_n(\mathbf{x}_i, \mathbf{F})\mathbf{K}_n^*(\mathbf{x}_i, \mathbf{F}), \tag{21}$$

and the generator $G_{p+q}(\mathbf{A}, \mathbf{F}, \mathbf{F})$ can be written as $\{ \mathbf{X}, \mathbf{X}\Sigma \}$, where $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_{p+q}]$ and $\Sigma \equiv \mathbf{I}_p \oplus -\mathbf{I}_q$.

A displacement representation of a matrix $\mathbf{A}$ can be obtained by using the following lemma, whose simple proof we shall omit.

Lemma. *For any* $\mathbf{A} \in R^{m\times n}$, *if* $\mathbf{F}_1$ *or* $\mathbf{F}_2$ *is nilpotent, there exists* $\alpha \le min(m, n)$ *such that*

$$\mathbf{A} = \sum_{i=1}^{\alpha} \mathbf{K}_n(\mathbf{x}_i, \mathbf{F}_1)\mathbf{K}_n^T(\mathbf{y}_i, \mathbf{F}_2) \qquad \textit{if and only if} \qquad \mathbf{A} - \mathbf{F}_1\mathbf{A}\mathbf{F}_2^T = \sum_{i=1}^{\alpha} \mathbf{x}_i\mathbf{y}_i^T.$$

In later developments an important role will be played by 2×2 block matrices (9). For simplicity, we shall first consider a symmetric matrix,

$$\mathbf{A} \equiv \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} \\ \mathbf{A}_{1,2}^T & \mathbf{A}_{2,2} \end{bmatrix} \in R^{(n+m)\times(n+m)}, \quad \mathbf{A}_{1,1} \in R^{n\times n}, \quad \mathbf{A}_{2,2} \in R^{m\times m}, \tag{22}$$

where $\mathbf{A}_{1,1}$ is a symmetric positive definite matrix, and $\mathbf{A}_{2,2}$ is a symmetric matrix.

For the matrix $\mathbf{A}$ in (22) we shall choose both displacement operators as

$$\mathbf{F} = \begin{bmatrix} \mathbf{Z}_n & \mathbf{O} \\ \mathbf{O} & \mathbf{Z}_m \end{bmatrix}.$$

With this choice, $\mathbf{K}_{n+m}(\mathbf{x}_i, \mathbf{F})$ in (10) will have the form

$$\mathbf{K}_{n+m}(\mathbf{x}_i, \mathbf{F}) \equiv \begin{bmatrix} \mathbf{L}(\mathbf{x}_{1,i}) & \mathbf{O} \\ \mathbf{L}(\mathbf{x}_{2,i}) & \mathbf{O} \end{bmatrix} \in R^{(n+m)\times(n+m)}, \quad \mathbf{L}(\mathbf{x}_{1,i}) \in R^{n\times n}, \quad \mathbf{L}(\mathbf{x}_{2,i}) \in R^{m\times m},$$

where $[\ \mathbf{x}_{1,i}^T, \mathbf{x}_{2,i}^T\ ] \equiv \mathbf{x}_i^T$, and $\mathbf{L}(\mathbf{x}_{1,i})$ and $\mathbf{L}(\mathbf{x}_{2,i})$ are lower triangular Toeplitz matrices whose first columns are $\mathbf{x}_{1,i}$ and $\mathbf{x}_{2,i}$, respectively. The $\mathbf{O}$'s denote rectangular null matrices of appropriate sizes.

Generators of Schur Complements.

We shall now show how to obtain the displacement representation (with respect to $\mathbf{Z}_m$ and $\mathbf{Z}_m$) of the matrix $\mathbf{A}_s$,

$$\mathbf{A}_s \equiv \mathbf{A}_{2,2} - \mathbf{A}_{1,2}^T \mathbf{A}_{1,1}^{-1} \mathbf{A}_{1,2},$$

which is the so-called Schur complement of $\mathbf{A}_{1,1}$.

1. Obtain a generator of $\mathbf{A}$, say $\{\ \mathbf{X}, \mathbf{X}\Sigma\ \}$, $\Sigma = \mathbf{I}_p \oplus -\mathbf{I}_q$.

2. Form the matrix Δ,

$$\Delta \equiv \Big[\ \mathbf{K}(\mathbf{x}_1, \mathbf{F}), \cdots \mathbf{K}(\mathbf{x}_p, \mathbf{F}), \mathbf{K}(\mathbf{x}_{p+1}, \mathbf{F}), \cdots, \mathbf{K}(\mathbf{x}_{p+q}, \mathbf{F})\ \Big] \tag{23a}$$

$$= \left[\begin{bmatrix} \mathbf{L}(\mathbf{x}_{1,1}) & \mathbf{O} \\ \mathbf{L}(\mathbf{x}_{2,1}) & \mathbf{O} \end{bmatrix} \cdots \begin{bmatrix} \mathbf{L}(\mathbf{x}_{1,p}) & \mathbf{O} \\ \mathbf{L}(\mathbf{x}_{2,p}) & \mathbf{O} \end{bmatrix} \begin{bmatrix} \mathbf{L}(\mathbf{x}_{1,p+1}) & \mathbf{O} \\ \mathbf{L}(\mathbf{x}_{2,p+1}) & \mathbf{O} \end{bmatrix} \cdots \begin{bmatrix} \mathbf{L}(\mathbf{x}_{1,p+q}) & \mathbf{O} \\ \mathbf{L}(\mathbf{x}_{2,p+q}) & \mathbf{O} \end{bmatrix} \right]. \tag{23b}$$

3. Post-multiply Δ by a *J-orthogonal* matrix Θ, where $\mathbf{J} \equiv \mathbf{I}_{(n+m)p} \oplus -\mathbf{I}_{(n+m)q}$ (i.e., the matrix Θ is such that $\Theta \mathbf{J} \Theta^T = \mathbf{J}$), that will transform Δ as follows (We shall show how to do this in the proof):

   (i) $\mathbf{K}(\mathbf{x}_1, \mathbf{F}) \rightarrow \begin{bmatrix} \mathbf{L} & \mathbf{O} \\ \mathbf{M} & \mathbf{L}_1 \end{bmatrix}$, $\mathbf{L}$ is lower-triangular; $\mathbf{L}_1$ is lower triangular Toeplitz

   (ii) $\mathbf{K}(\mathbf{x}_i, \mathbf{F}) \rightarrow \begin{bmatrix} \mathbf{O} & \mathbf{O} \\ \mathbf{L}_i & \mathbf{O} \end{bmatrix}$, $\mathbf{L}_i$ is lower triangular Toeplitz, $2 \le i \le p+q$.

   The result will be

$$\tilde{\Delta} \equiv \Delta\Theta = \left[\begin{bmatrix} \mathbf{L} & \mathbf{O} \\ \mathbf{M} & \mathbf{L}_1 \end{bmatrix} \begin{bmatrix} \mathbf{O} & \mathbf{O} \\ \mathbf{L}_2 & \mathbf{O} \end{bmatrix} \cdots \begin{bmatrix} \mathbf{O} & \mathbf{O} \\ \mathbf{L}_{p+q} & \mathbf{O} \end{bmatrix} \right]. \tag{24}$$

Then it turns out that

$$\mathbf{A}_{1,1} = \mathbf{L}\mathbf{L}^T, \tag{25a}$$

$$\mathbf{A}_{1,2}^T = \mathbf{M}\mathbf{L}^T, \tag{25b}$$

$$\mathbf{A}_s \equiv \mathbf{A}_{2,2} - \mathbf{A}_{1,2}^T\mathbf{A}_{1,1}^{-1}\mathbf{A}_{1,2} = \sum_{i=1}^{p}\mathbf{L}_i\mathbf{L}_i^T - \sum_{i=p+1}^{p+q}\mathbf{L}_i\mathbf{L}_i^T. \tag{25c}$$

Note that the generator of $\mathbf{A}_s$ has the same p and q as the given generator of $\mathbf{A}$ itself, a fact first discovered by Morf [19] and used by him to derive "divide-and-conquer"-type algorithms (see also [3]).

Proof The results in (25a, b) follow immediately by equating the (1, 1) and (2, 1) blocks on both sides of the equality,

$$\mathbf{A} = \Delta\mathbf{J}\Delta^T = \Delta\Theta\mathbf{J}\Theta^T\Delta^T = \tilde{\Delta}\mathbf{J}\tilde{\Delta}^T. \tag{26}$$

Furthermore, equating the (2, 2) blocks of (26) gives

$$\mathbf{A}_{2,2} = \mathbf{M}\mathbf{M}^T + \sum_{i=1}^{p}\mathbf{L}_i\mathbf{L}_i^T - \sum_{i=p+1}^{p+q}\mathbf{L}_i\mathbf{L}_i^T,$$

from which the equality (25c) follows by noting that

$$\mathbf{M}\mathbf{M}^T = \mathbf{A}_{1,2}^T\mathbf{L}^{-T}\mathbf{L}^{-1}\mathbf{A}_{1,2} = \mathbf{A}_{1,2}^T\mathbf{A}_{1,1}^{-1}\mathbf{A}_{1,2}.$$

The only thing left to do is to show how to construct Θ so as to obtain $\tilde{\Delta}$ of the form (24). This can be done in several ways, but perhaps the simplest and the most useful is to construct Θ as a product of $(p+q)(n+m) \times (p+q)(n+m)$ circular (or Givens) and hyperbolic rotations. Givens rotations $\mathbf{G}_{i,j}(\kappa)$ with *reflection coefficient* κ, are defined as identity matrices except for the following four entries,

$$[\mathbf{G}_{i,j}(\kappa)]_{i,i} = [\mathbf{G}_{i,j}(\kappa)]_{j,j} = \frac{1}{(1+|\kappa|^2)^{1/2}}, \quad [\mathbf{G}_{i,j}(\kappa)]_{i,j} = \frac{-\kappa}{(1+|\kappa|^2)^{1/2}}, \quad [\mathbf{G}_{i,j}(\kappa)]_{j,i} = \frac{\kappa}{(1+|\kappa|^2)^{1/2}}.$$

We annihilate $\{ \mathbf{L}(\mathbf{x}_{1,2}), \mathbf{L}(\mathbf{x}_{1,3}), \ldots, \mathbf{L}(\mathbf{x}_{1,p+q}) \}$ in (23b) with n *sweeps* (0th, 1st, $\cdots$, $(n-1)$st sweeps). The kth sweep annihilates the kth sub-diagonals of the $p-1$ matrices,

$\{ L(x_{1,2}), L(x_{1,3}) \ldots, L(x_{1,p}) \}$ with Givens rotations, and the kth sub-diagonals of the q matrices, $\{ L(x_{1,p+1}), L(x_{1,p+2}) \ldots, L(x_{1,p+q}) \}$ with hyperbolic rotations, pivoting with the diagonal elements in $L(x_{1,1})$ in both cases.

In the kth sweep, if $k > n-m$, then we apply 'dummy' rotations to the $(n-k+1)$st to the mth columns of $K(x_i, F)$, pivoting with the $(n+1)$st to $(m+k)$th columns of $K(x_1, F)$, in order to keep the Toeplitz structure in $\{ L(x_{2,2}), L(x_{2,3}) \ldots, L(x_{2,p+q}) \}$. This will introduce a non-zero lower triangular Toeplitz matrix in the (2, 2) block of $K(x_1, F)$. After the $(n-1)$st sweep, we shall have $\tilde{\Delta}$ in (24).

□

Operation Counts.

To annihilate n rows out of $n+m$ rows, we shall need approximately $4(p+q-1)\sum_{k=m}^{n+m} k = 2(p+q-1)\times(n^2+2nm+2m+n)$ multiplications. This will be less than the $O(n^3)$ multiplications needed to obtain the factors of a matrix $A_{1,1}$ and its inverse unless $p+q$ is nearly n. There are many interesting matrices (see Sec 4) for which $p+q \ll n$.

Example (A Gohberg-Semencul formula for Toeplitz matrices).

Let $T = (c_{i-j})$ be an $n \times n$ Hermitian positive definite Toeplitz matrix. Consider the matrix

$$A = \begin{bmatrix} T & I_n \\ I_n & O \end{bmatrix}. \tag{27}$$

For $F = Z_n \oplus Z_n$, it is easy to see that the displacement rank of A with respect to $\{F, F\}$ is two, and that a generator $G_2(A, F, F)$ is given by $\{X, X\Sigma\}$, where

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} \\ c_0^{-1/2} e_1 & c_0^{-1/2} e_1 \end{bmatrix}, \qquad \Sigma = 1 \oplus -1,$$

where

$$\mathbf{x}_{1,1} = c_0^{-1/2}\cdot[c_0, c_1, \cdots, c_{n-1}]^T, \quad \mathbf{x}_{1,2} = c_0^{-1/2}\cdot[0, c_1, \cdots, c_{n-1}]^T, \quad \mathbf{e}_1 = [\,1, 0, \cdots, 0\,]^T.$$

With this generator, the matrix Δ in (23) will have the form

$$\Delta = \begin{bmatrix} \mathbf{L}(\mathbf{x}_{1,1}) & \mathbf{O} & \mathbf{L}(\mathbf{x}_{1,2}) & \mathbf{O} \\ c_0^{-1/2}\mathbf{I}_n & \mathbf{O} & c_0^{-1/2}\mathbf{I}_n & \mathbf{O} \end{bmatrix}. \tag{28}$$

Now note that the matrix Δ in (28) is in the form same as (10). The procedure will triangularize the matrix Δ into a matrix $\tilde{\Delta}$ of the form (11),

$$\tilde{\Delta} = \begin{bmatrix} \mathbf{L} & \mathbf{O} & \mathbf{O} & \mathbf{O} \\ \mathbf{U} & \mathbf{L}_1 & \mathbf{L}_2 & \mathbf{O} \end{bmatrix}.$$

Since the Schur complement of $\mathbf{T}$ in $\mathbf{A}$ in (27) is $-\mathbf{T}^{-1}$, formula (25c) will yield

$$\mathbf{T}^{-1} = \mathbf{L}_2\mathbf{L}_2^T - \mathbf{L}_1\mathbf{L}_1^T.$$

Non Symmetric Matrices.

For non-symmetric rectangular matrices $\mathbf{A}$ one can formulate a similar procedure. We form the matrices Δ_1 and Δ_2 as

$$\Delta_1 \equiv \Big[\, \mathbf{K}(\mathbf{x}_1, \mathbf{F}_1), \cdots, \mathbf{K}(\mathbf{x}_\alpha, \mathbf{F}_1) \,\Big], \qquad \Delta_2 \equiv \Big[\, \mathbf{K}(\mathbf{y}_1, \mathbf{F}_2), \cdots, \mathbf{K}(\mathbf{y}_\alpha, \mathbf{F}_2) \,\Big], \tag{29}$$

and annihilate the elements in Δ_1 and Δ_2 by post-multiplying Δ_1 and Δ_2 with *spinor matrices*, instead of Givens and hyperbolic rotations. A spinor matrix is defined as the identity except for the following four entries,

$$[\mathbf{S}_{i,j}]_{i,i} = [\mathbf{S}_{i,j}]_{j,j} = \frac{1}{(1+\kappa_1\kappa_2)^{1/2}}, \quad [\mathbf{S}_{i,j}]_{i,j} = \frac{-\kappa_1}{(1+\kappa_1\kappa_2)^{1/2}}, \quad [\mathbf{S}_{i,j}]_{j,i} = \frac{\kappa_2}{(1+\kappa_1\kappa_2)^{1/2}}.$$

Because of the invariance property that

$$\mathbf{A} = \Delta_1\cdot\Delta_2^T = \Delta_1\mathbf{S}_{i,j}\cdot\mathbf{S}_{i,j}^{-1}\Delta_2^T,$$

a similar procedure to that in the symmetric case will give a displacement representation of $\mathbf{A}_s$,

$$\mathbf{A}_s \equiv \mathbf{A}_{2,2} - \mathbf{A}_{2,1}\mathbf{A}_{1,1}^{-1}\mathbf{A}_{1,2} = \sum_{i=1}^{\alpha}\mathbf{L}_i\mathbf{U}_i, \tag{30}$$

where $\mathbf{L}_i$ and $\mathbf{U}_i^T$ are lower triangular Toeplitz matrices.

4. Some Generalized Gohberg-Semencul Formulas.

The first step of the procedure to obtain a generator for the matrix $\mathbf{A}_s = \mathbf{A}_{2,2} - \mathbf{A}_{2,1}\mathbf{A}_{1,1}^{-1}\mathbf{A}_{1,2}$ is to find a generator of the matrix

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} \end{bmatrix}.$$

This can be done by using the lemma in Sec 3. In this section, we shall give generators of some interesting A's, from which the corresponding Gohberg-Semencul formulas for the matrices of interest will be evident.

Example 1 (Generator of inverse).

Let $\mathbf{B}$ be an $n \times n$ symmetric positive-definite matrix, with a known generator, $\{ \mathbf{W}, \mathbf{W}\Sigma \}$, $\Sigma = \mathbf{I}_p \oplus -\mathbf{I}_q$ with respect to $\{\mathbf{Z}_n, \mathbf{Z}_n\}$. Consider the matrix,

$$\mathbf{A} \equiv \begin{bmatrix} \mathbf{B} & \mathbf{I}_n \\ \mathbf{I}_n & \mathbf{O} \end{bmatrix}. \tag{31}$$

Then by using the lemma in Sec 3, it is easy to see that $\{ \mathbf{X}, \mathbf{X}\Sigma \}$, where

$$\mathbf{X} \equiv \begin{bmatrix} \mathbf{w}_1 & \cdots & \mathbf{w}_p & \mathbf{e}_1 & \mathbf{w}_{p+1} & \cdots & \mathbf{w}_{p+q} & \mathbf{e}_1 \\ 0 & \cdots & 0 & \mathbf{e}_1/2 & 0 & \cdots & 0 & -\mathbf{e}_1/2 \end{bmatrix}, \quad \Sigma \equiv \mathbf{I}_{p+1} \oplus -\mathbf{I}_{q+1} \tag{32}$$

is a generator of $\mathbf{A}$ with respect to $\{\mathbf{Z}_n \oplus \mathbf{Z}_n, \mathbf{Z}_n \oplus \mathbf{Z}_n\}$. The length of the generator of $\mathbf{B}^{-1}$ obtained with the above $\mathbf{A}$ will be $p+q+2$. However, if the given generator $G_{p+q}(\mathbf{B}, \mathbf{Z}_n, \mathbf{Z}_n)$ satisfies a condition called *admissibility* [18] then, as we shall see, we can obtain a generator of $\mathbf{B}^{-1}$ with a length less than $p+q+2$.

Example 2 (Inversion with admissible generators).

A generator for a Hermitian matrix $\mathbf{B} \in R^{n \times n}$, $\{ \mathbf{W}, \mathbf{W}\Sigma \}$, $\Sigma = \mathbf{I}_p \oplus -\mathbf{I}_q$ with respect to $\{\mathbf{Z}_n, \mathbf{Z}_n\}$ is called *admissible* if $\mathbf{e}_1 \in$ range $(\mathbf{W})$, i.e., if there is a linear combination of the columns of the generator that will give the unit vector.

Let $G_{p+q}(\mathbf{B}, \mathbf{Z}_n, \mathbf{Z}_n)$ be admissible, and

$$[\mu_1, \mu_2, \cdots, \mu_{p+q}]\mathbf{W}^T = \mathbf{e}_1^T.$$

Then it can be checked that the matrix,

$$\mathbf{A} \equiv \begin{bmatrix} \mathbf{B} & \mathbf{I}_n \\ \mathbf{I}_n & \eta\mathbf{I}_n \end{bmatrix}, \qquad \eta \equiv \sum_{i=1}^{p} |\mu_i|^2 - \sum_{i=p+1}^{p+q} |\mu_i|^2 \tag{33}$$

has a generator $\{ \mathbf{X}, \mathbf{X}\Sigma \}$, with respect to $\{\mathbf{Z}_n \oplus \mathbf{Z}_n, \mathbf{Z}_n \oplus \mathbf{Z}_n\}$, where

$$\mathbf{X} \equiv \begin{bmatrix} \mathbf{w}_1 & \cdots & \mathbf{w}_p & \mathbf{w}_{p+1} & \cdots & \mathbf{w}_{p+q} \\ \mu_1\mathbf{e}_1 & \cdots & \mu_p\mathbf{e}_1 & -\mu_{p+1}\mathbf{e}_1 & \cdots & -\mu_{p+q}\mathbf{e}_1 \end{bmatrix}, \qquad \Sigma \equiv \mathbf{I}_p \oplus -\mathbf{I}_q. \tag{34}$$

Since the Schur complement of $\mathbf{B}$ in (33) is $\eta\mathbf{I}_n - \mathbf{B}^{-1}$, we see that the generator of $\mathbf{B}^{-1}$ obtained with the $\mathbf{A}$ in (33) will have length $p+q+1$ if $\eta \neq 0$, or $p+q$ if $\eta = 0$, consistent with the results first obtained in [18].

For an example of a minimal admissible generator (besides generators of Toeplitz matrices), let us consider an $m \times n$ Toeplitz matrix $\mathbf{T} = (c_{i-j})$ with a full column rank. The matrix $\mathbf{T}^T\mathbf{T}$ has a minimal generator [4], $\{ \mathbf{W}, \mathbf{W}\Sigma \}$, $\Sigma = \mathbf{I}_2 \oplus -\mathbf{I}_2$, with respect to $\{\mathbf{Z}_n, \mathbf{Z}_n\}$, where

$$\mathbf{w}_1 = \mathbf{T}^T\mathbf{t}_1/\|\mathbf{t}_1\|_2, \quad \mathbf{w}_2 = \mathbf{t}_2, \quad \mathbf{w}_3 = \mathbf{Z}_n\mathbf{Z}_n^T\mathbf{w}_1, \quad \mathbf{w}_4 = \mathbf{Z}_n\mathbf{l}_1, \tag{35a}$$

$$\mathbf{t}_1 = [\, c_0, c_1, \cdots, c_{m-1} \,]^T, \ \mathbf{t}_2 = [\, 0, c_{-1}, \cdots, c_{1-n} \,]^T, \ \mathbf{l}_1 = [\, c_{m-1}, \cdots, c_{m-n} \,]^T, \tag{35b}$$

and $\|\cdot\|_2$ denotes the Euclidean norm. This generator of $\mathbf{T}^T\mathbf{T}$ is admissible since

$$[1/\|\mathbf{t}_1\|_2, 0, -1/\|\mathbf{t}_1\|_2, 0]\mathbf{W}^T = \mathbf{e}_1^T. \tag{36}$$

Therefore, the matrix $\mathbf{A}$ in (33) would have the form

$$\mathbf{A} \equiv \begin{bmatrix} \mathbf{T}^T\mathbf{T} & \mathbf{I}_n \\ \mathbf{I}_n & \mathbf{O} \end{bmatrix}, \tag{37}$$

and the procedure will give a generator of $(\mathbf{T}^T\mathbf{T})^{-1}$ of length 4. The displacement representation of $(\mathbf{T}^T\mathbf{T})^{-1}$ is useful in solving Toeplitz least squares problems ($m \geq n$).

Example 3 (Generator of $\mathbf{T}_1^T\mathbf{T}_2^{-1}\mathbf{T}_1$).

Let $\mathbf{T}_1 \in R^{n\times m}$ and $\mathbf{T}_2 = (c_{i-j}) \in R^{n\times n}$ be Toeplitz, and symmetric positive-definite Toeplitz, respectively. If we define

$$\mathbf{A} \equiv \begin{bmatrix} \mathbf{T}_2 & \mathbf{T}_1 \\ \mathbf{T}_1^T & \mathbf{O} \end{bmatrix} \tag{38}$$

then A has a generator $\{ \mathbf{X}, \mathbf{X}\Sigma \}$, $\Sigma \equiv \mathbf{I}_2 \oplus -\mathbf{I}_2$, with respect to $\{\mathbf{Z}_n \oplus \mathbf{Z}_m, \mathbf{Z}_n \oplus \mathbf{Z}_m\}$, where

$$\mathbf{X} \equiv \begin{bmatrix} \mathbf{v}_1 & \mathbf{u}_2 & \mathbf{v}_2 & \mathbf{u}_2 \\ \mathbf{u}_1 & \mathbf{e}_1/2 & \mathbf{u}_1 & -\mathbf{e}_1/2 \end{bmatrix},$$

$\mathbf{v}_1$ = the first column of $\mathbf{T}_2$ devided by $c_0^{1/2}$,

$\mathbf{u}_1$ = the first column of $\mathbf{T}_1^T$,

$\mathbf{v}_2$ = same as $\mathbf{v}_1$ with the first entry equals to zero,

$\mathbf{u}_2$ = the first column of $\mathbf{T}_1$ with the first entry equals to zero.

With the above A, we can obtain a generator of $\mathbf{T}_1^T\mathbf{T}_2^{-1}\mathbf{T}_1$ with length 4.

The displacement representation of $\mathbf{T}_1^T\mathbf{T}_2^{-1}\mathbf{T}_1$ is useful in solving weighted Toeplitz least squares problems ($n \geq m$), as arise in certain parametric time series identification problems.

Remark 1. One can obtain a generator of $\mathbf{T}_1^T\mathbf{T}_1$ by setting $\mathbf{T}_2$ in (38) with $\mathbf{I}_n$. This procedure will need $O(mn)$ computations, which is of the same order as evaluating the closed form expression in (35a).

Remark 2. It is interesting to note that $\alpha(\mathbf{T}_1^T\mathbf{T}_2^{-1}\mathbf{T}_1, \mathbf{Z}_m, \mathbf{Z}_m) \leq 4$, whereas $\alpha(\mathbf{T}_1^T\mathbf{T}_2\mathbf{T}_1, \mathbf{Z}_m, \mathbf{Z}_m) \leq 6$. The reason is that the first matrix can be identified as the Schur complement in a 2×2 block matrix, while to do so for $\mathbf{T}_1^T\mathbf{T}_2\mathbf{T}_1$ requires going to a 3×3 block matrix whose displacement rank can be 6.

Remark 3. Once one has a generator of $\mathbf{T}_1^T\mathbf{T}_2^{-1}\mathbf{T}_1$, one can obtain a generator of $(\mathbf{T}_1^T\mathbf{T}_2^{-1}\mathbf{T}_1)^{-1}$ using the matrix A in (31) and the generator in (32). This will give a generator of length 6. However, it turns out that minimal generators of $\mathbf{T}_1^T\mathbf{T}_2^{-1}\mathbf{T}_1$ are admissible (with $\eta = 0$), and the displacement rank of $(\mathbf{T}_1^T\mathbf{T}_2^{-1}\mathbf{T}_1)^{-1}$ is less than or equal to 4 (see Sec 5).

Example 4 (Generator of the projection operator).

Let T be an $m \times n$ Toeplitz matrix with full column rank. If we define

$$\mathbf{A} = \begin{bmatrix} \mathbf{T}^T\mathbf{T} & \mathbf{T}^T \\ \mathbf{T} & \mathbf{I}_m \end{bmatrix}, \tag{39}$$

then $\{ \mathbf{X}, \mathbf{X}\Sigma \}$, $\Sigma \equiv \mathbf{I}_2 \oplus -\mathbf{I}_2$, where

$$\mathbf{X} \equiv \begin{bmatrix} \mathbf{w}_1 & \mathbf{w}_2 & \mathbf{w}_3 & \mathbf{w}_4 \\ \mathbf{t}_1/\|\mathbf{t}_1\|_2 & \mathbf{e}_1 & \mathbf{t}_1/\|\mathbf{t}_1\|_2 & 0 \end{bmatrix}, \quad \mathbf{w}_i \text{ and } \mathbf{t}_1 \text{ are as in (35).} \tag{40}$$

is a generator of A with respect to $\{\mathbf{Z}_n \oplus \mathbf{Z}_m, \mathbf{Z}_n \oplus \mathbf{Z}_m\}$. By applying the procedure in Sec 3 to the above matrix A, we can obtain the generator (of length 4) of the projection operator $\mathbf{I}_m - \mathbf{T}(\mathbf{T}^T\mathbf{T})^{-1}\mathbf{T}^T$ on the ($(m-n)$-dimensional) kernel of $\mathbf{T}^T$. Also, in this case, the matrix M in (24) turns out to be the orthogonal basis of the range of T, because by (25a, b)

$$\mathbf{T}^T\mathbf{T} = \mathbf{L}\mathbf{L}^T, \quad \text{and} \quad \mathbf{T} = \mathbf{M}\mathbf{L}^T, \quad \text{which implies} \quad \mathbf{M}^T\mathbf{M} = \mathbf{I}_n, \quad \mathbf{M} \in R^{m\times n}.$$

In fact, we have the QR factorization of T (see also [4]).

Example 5 (Generator of the pseudo-inverse).

Let T be an $m \times n$ Toeplitz matrix with a full column rank. If we define

$$\mathbf{A} \equiv \begin{bmatrix} \mathbf{T}^T\mathbf{T} & \mathbf{T}^T \\ \mathbf{I}_n & \mathbf{O}_{n\times m} \end{bmatrix},$$

then by using the results in (36) and (40), we can see that A has a generator, $\{ \mathbf{X}, \mathbf{Y} \}$, with respect to $\{\mathbf{Z}_n \oplus \mathbf{Z}_n, \mathbf{Z}_n \oplus \mathbf{Z}_m\}$ where

$$\mathbf{X} \equiv \begin{bmatrix} \mathbf{w}_1 & \mathbf{w}_2 & \mathbf{w}_3 & \mathbf{w}_4 \\ \mathbf{e}_1/\|\mathbf{t}_1\|_2 & 0 & \mathbf{e}_1/\|\mathbf{t}_1\|_2 & 0 \end{bmatrix}, \quad \mathbf{Y} \equiv \begin{bmatrix} \mathbf{w}_1 & \mathbf{w}_2 & -\mathbf{w}_3 & -\mathbf{w}_4 \\ \mathbf{t}_1/\|\mathbf{t}_1\|_2 & \mathbf{e}_1 & -\mathbf{t}_1/\|\mathbf{t}_1\|_2 & 0 \end{bmatrix},$$

and $\mathbf{w}_i$ and $\mathbf{t}_1$ are as in (35). With the above generator, we can obtain a generator of the pseudo-inverse of T of length 4.

5. Concluding Remarks.

We have presented a constructive approach to the famous Gohberg-Semencul formula for

the inverse of a Toeplitz matrix, and obtained various generalizations of it. We note that formulas of Gohberg and Semencul type are closely related to the problem of finding displacement representations (generators) of various matrices.

For further development, we may mention that the procedure for 2×2 block matrices given in Sec 3 can be easily generalized to $N \times N$ block matrices. For instance, by considering 3×3 block matrices, one can obtain a generator of $[\mathbf{A}_{2,2} - \mathbf{A}_{1,2}^T \mathbf{A}_{1,1}^{-1} \mathbf{A}_{1,2}]^{-1}$. As an example, we can obtain a generator of $(\mathbf{T}_1^T \mathbf{T}_2^{-1} \mathbf{T}_1)^{-1}$ by working with the matrix

$$\mathbf{A} \equiv \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} \\ \mathbf{A}_{1,2}^T & \mathbf{O} \end{bmatrix}, \quad \mathbf{A}_{1,1} = \begin{bmatrix} \mathbf{T}_2 & \mathbf{T}_1 \\ \mathbf{T}_1^T & \mathbf{O} \end{bmatrix}, \quad \mathbf{A}_{1,2} = \begin{bmatrix} \mathbf{O} \\ \mathbf{I} \end{bmatrix}.$$

The length of the obtained generator will be 4. As another example, a generator of $\mathbf{T}_1^T \mathbf{T}_2 \mathbf{T}_1$ can be obtained by choosing

$$\mathbf{A}_{1,1} = \begin{bmatrix} \mathbf{T}_2 & \mathbf{I} \\ \mathbf{I} & \mathbf{O} \end{bmatrix}, \quad \mathbf{A}_{1,2} = \begin{bmatrix} \mathbf{O} \\ \mathbf{T}_1 \end{bmatrix}.$$

We also remark that divide-and-conquer versions of the procedure in Sec 3 can be readily obtained [3]. By using this approach to compute the displacement representation of $(\mathbf{T}^T \mathbf{T})^{-1}$, one can, for instance, obtain least-squares solutions for Toeplitz systems in $O(m \log^2 m)$ operations.

There exists an interesting relationship between the reflection coefficients of a Toeplitz matrix $\mathbf{T}$ and those of $\mathbf{T}^{-1}$ [15]. A constructive proof of this relationship can be found in [2].

Finally, we should note that several authors have explored the problem of fast inversion of various structured matrices by employing somewhat different, but related, definitions of displacement. We may mention the work of Heinig and Rost [12], Gohberg *et al* [9], [10], L. Lerer and M. Tismenetsky [17]. Some of the formulas therein are also generalizations of the Gohberg-Semencul formula. More work needs to be done to clarify the relationships between these different results and approaches.

APPENDIX

Lemma A1. *Let* T *be a symmetric Toeplitz matrix. Then the* $\{\kappa_i\}$ *defined by the construction in Sec 2 will be less than one in magnitude if and only if* T *is positive-definite.*

Proof. If $|\kappa_i| < 1$ for all $1 \le i \le n-1$, then we can complete the transformation to get the matrix $\tilde{\Delta}$ in (11), and $\mathbf{T} = \mathbf{L}\mathbf{L}^T$. Hence, T is positive-definite. Now, let us assume that T is positive definite, and $|\kappa_j| < 1$ for $1 \le j \le i-1$. After the $(i-1)$st sweep, the upper-half, Δ_{i-1}^U of the matrix Δ_{i-1} has the form,

$$\Delta_{i-1}^U = \begin{bmatrix} \mathbf{A} & \mathbf{O} & & \mathbf{O} & \mathbf{O} \\ \mathbf{B} & \mathbf{C} & \bigcirc & \mathbf{D} & \mathbf{O} \end{bmatrix}, \quad \Delta_{i-1}^U \mathbf{J} [\Delta_{i-1}^U]^T = \mathbf{T},$$

where A is a nonsingular lower triangular matrix, and C, D are lower triangular Toeplitz. Let c and d denote the diagonal elements of C and D, respectively. Suppose that $|c| \le |d|$, and therefore, $|\kappa_i| \ge 1$. Then T cannot be positive-definite, because

$$\lambda_{\min}(\mathbf{T}) \le \mathbf{s}^T \Delta_{i-1}^U \mathbf{J} [\Delta_{i-1}^U]^T \mathbf{s} = c^2 - d^2 \le 0,$$

where

$$\mathbf{s}^T = [-\mathbf{b}_1^T \mathbf{A}^{-1}, \mathbf{e}_1^T], \quad \mathbf{e}_1^T = [1, 0, \cdots, 0], \quad \mathbf{b}_1^T \equiv \text{the first row of } \mathbf{B},$$

which leads to a contradiction. □

REFERENCES

[1]. R. Brent, F. Gustavson and D. Yun, *Fast solution of Toeplitz systems of equations and computation of Pade approximants*, Journal of Algorithms, 1, (1980), pp. 259-295.

[2]. J. Chun and T. Kailath, *A constructive proof of the Gohberg-Semencul formula*, Linear Alg and its Appl., to appear, 1989.

[3]. J. Chun and T. Kailath, *Divide-and-conquer solutions for least-squares problems for matrices with displacement structure*, Proc. Sixth Army conf. on Applied Math. and

Comput., Boulder, CO, June, 1988.

[4]. J. Chun, T. Kailath and H. Lev-Ari, *Fast parallel algorithms for QR and triangular factorization*, SIAM J. Sci. Stat. Comput., vol. 8, No. 6, Nov., (1987), pp. 899-913.

[5]. B. Friedlander, M. Morf, T. Kailath and L. Ljung, *New inversion formula for matrices classified in terms of their distance from Toeplitz matrices*, Linear Algebra and its Appl., 27 (1979), pp. 31-60.

[6]. Ya. L. Geronimus, *Polynomials orthogonal on a circle and interval*, Pergamon press, New York, 1960.

[7]. I. Gohberg and I. Fel'dman, *Convolution equations and projection methods for their solutions*, Translations of Mathematical Monographs, vol. 41, Amer. Math. Soc., 1974.

[8]. I. Gohberg, T. Kailath and I. Koltracht, *Efficient solution of linear systems of equations with recursive structure*, Linear Algebra and its Appl., 80 (1986), pp 81-113.

[9]. I. Gohberg, T. Kailath, I. Koltracht and P. Lancaster, *Linear complexity parallel algorithms for linear systems of equations with recursive structure*, Linear Algebra and its Appl., 88 (1987), pp 271-315.

[10]. I. Gohberg and A. Semencul, *On the inversion of finite Toeplitz matrices and their continuous analogs*, Mat. Issled., 2 (1972), pp. 201-233.

[11]. U. Grenander and G. Szego *Toeplitz forms and their applications* 2nd ed., Chelsea publishing company, New York, 1984.

[12]. G. Heinig and K. Rost, Algebraic methods for Toeplitz-like matrices and operators, Akademie-Verlag, Berlin, 1984.

[13]. T. Kailath, A theorem of I. Schur and its impact on modern signal processing, in *I. Schur methods in operator theory and signal processing*, Operator Theory: Advances and Applications, vol. 18 (I. Gohberg, Ed.), Birkhaeuser Verlag Basel, 1986.

[14]. T. Kailath, S. Kung and M. Morf, *Displacement ranks of matrices and linear equations*, J. Math. Anal. Appl., 68 (1979) pp. 395-407. See also Bull. Amer. Math. Soc., 1 (1979), pp. 769-773.

[15]. T. Kailath and H. Lev-Ari, On mappings between covariance matrices and physical systems, *Contemporary Mathematics* 47 (1985).

[16]. T. Kailath, A. Vieira and M. Morf, *Inverses of Toeplitz operators, innovations, and orthogonal polynomial*, SIAM Review, vol. 20, No 1, Jan. (1978), pp. 106-119.

[17]. L. Lerer and M. Tismenetsky, *Generalized Bezoutian and matrix equations*, Linear Algebra and its Appl., 99 (1988), pp 123-160.

[18]. H. Lev-Ari, and T. Kailath, *Lattice filter parameterization and modeling of nonstationary process*, IEEE Trans. Inform. Theory, IT-30 (1984), pp. 2-16.

[19]. M. Morf, *Doubling algorithms for Toeplitz and related equations*, in Proceedings of the IEEE International Conf. on Acoustics, Speech and Signal Processing, Denver, (1980), pp. 954-959.

[20]. I. Schur, *Über Potenzreihen, die im Innern des Einheitskreises beschrankt sind*, J. für die Reine und Angewandte Mathematik, 147 (1917), pp. 205-232.

[21]. W. Trench, *An algorithm for inversion of finite Toeplitz matrices*, J. of SIAM, vol. 12.3 (1964), pp. 515-522.

Chapter 4.

Displacement Structure for Hankel, Vandermonde and Related (Derived) Matrices

Abstract

We introduce some generalized concepts of displacement structure for structured matrices obtained as products and inverses of Toeplitz, Hankel and Vandermonde matrices. The Toeplitz case has already been studied at some length, and the corresponding matrices have been called near-Toeplitz or Toeplitz-like or Toeplitz-derived. In this chapter, we shall focus mainly on Hankel- and Vandermonde-like matrices and in particular show how the appropriately defined displacement structure yields fast triangular and orthogonal factorization algorithms for such matrices.

1. Introduction.

Many signal processing problems require solving large systems of linear equations, either directly or via (weighted) least squares. The basic solution tools are *triangular factorization* and *QR factorization*. These factorizations require $O(n^3)$ or $O(mn^2)$ flops (floating point operations) for an $m \times n$ matrix, which can often be excessively large. Therefore, attention focuses on *structured matrices*, with an eye both to computational reductions and to implementability in special purpose (parallel) hardware. Structured matrices arise in various problems in coding theory, interpolation, control, signal processing and system theory.

Very common examples of structured matrices in the above areas are Toeplitz, Hankel and Vandermonde matrices:

$$T = (t_{i-j}) = \begin{bmatrix} t_0 & t_{-1} & \cdot & t_{-n+1} \\ t_1 & t_0 & \cdot & t_{-n+2} \\ \cdot & \cdot & \cdot & \cdot \\ t_{m-1} & t_{m-2} & \cdot & t_{-n+m} \end{bmatrix} \in \mathbf{R}^{m \times n}, \tag{1}$$

$$H = (h_{i+j-2}) = \begin{bmatrix} h_0 & h_1 & \cdot\ \cdot & h_{n-1} \\ h_1 & h_2 & \cdot\ \cdot & h_n \\ \cdot & \cdot & \cdot\ \cdot & \cdot \\ h_{n-1} & h_n & \cdot\ \cdot & h_{2n-2} \end{bmatrix} \in \mathbf{R}^{n \times n}, \tag{2}$$

$$V \equiv V(\mathbf{c},K) = \begin{bmatrix} — & \mathbf{c}^T & — \\ — & \mathbf{c}^T K & — \\ & \vdots & \\ — & \mathbf{c}^T K^{n-1} & — \end{bmatrix} \in \mathbf{R}^{n \times n}, \tag{3}$$

$$K = \mathrm{diag}(k_1, k_2, \cdots, k_n), \quad k_i \neq 0, \quad \mathbf{c} = [1, 1, \cdots, 1]^T.$$

These matrices have certain shift invariant properties: The (infinite) Toeplitz matrices are diagonally shift-invariant, Hankel matrices are "reverse diagonally" shift-invariant, and Vandermonde matrices are vertically shift-invariant apart from K.

However, it is easy to see that, for example, Toeplitz structure is not invariant under various frequently occurring operations such as multiplication, inversion, triangular and orthogonal factorization. What is often invariant is a suitably defined notion called the *displacement rank*. For example, let

$$\nabla A \equiv A - Z_n A Z_n^T, \quad A \in \mathbf{R}^{n \times n}, \tag{4}$$

where Z_n is the $n \times n$ lower-shift matrix with ones on the first subdiagonal and zeros everywhere else, and define the displacement rank of A by $rank(\nabla A)$. Then it can be shown that a Toeplitz matrix T and its inverse T^{-1} have the same displacement rank,

$$rank(\nabla T) = rank(\nabla T^{-1}) = 2. \tag{5a}$$

Moreover if T is Hermitian, then

$$inertia(\nabla T) = inertia(\nabla T^{-1}). \tag{5b}$$

Displacement structure as just defined has by now been studied in some detail (see [12] for a recent survey, and also [13], [17]), with many results for closely related definitions in [8] and [10] among others.

In this chapter, we shall briefly study an extended form of (4),

$$\nabla_{\{F^f, F^b\}} A \equiv A - F^f A F^{bT}, \tag{6}$$

and especially another definition

$$\Delta_{\{F^f, F^b\}} A \equiv F^f A - A F^{bT}, \tag{7}$$

where the matrices $\{F^f, F^b\}$ can be fairly general subject to certain restrictions described in Sec 2.

For reasons that will soon appear we shall call the matrices $\nabla_{\{F^f, F^b\}} A$ and $\Delta_{\{F^f, F^b\}} A$ the *Toeplitz displacement*, and the *Hankel displacement* of A *with respect to displacement operators* $\{F^f, F^b\}$. The rank of $\nabla_{\{F^f, F^b\}} A$ will be called the *Toeplitz displacement rank* of A (with respect to $\{F^f, F^b\}$), and denoted as $\alpha_{\{F^f, F^b\}} A$. The rank of $\Delta_{\{F^f, F^b\}} A$ will be called the *Hankel displacement rank* of A and denoted as $\beta_{\{F^f, F^b\}} A$.

We shall say, roughly, that a matrix is "structured" (e.g., "close-to-Toeplitz" or "close-to-Hankel"), it the matrix has a displacement rank (with respect to some $\{F^f, F^b\}$), independent of the size of matrix. The interesting fact, which enables fast algorithms for triangular and orthogonal matrix factorization and matrix inversion, is that such structure is inherited under inversion, multiplication and Schur complementation. This chapter will demonstrate this fact for various types of structured matrices.

First in Sec 2, we establish the claims just made about inversion, etc. Based on these results, we present a general factorization algorithm in Sec 3, which will be further specialized in later sections. Thus in Sec 4, we shall show how to compute the triangular factorization of Vandermonde matrices. Triangular factorizations of close-to-Hankel matrices will be presented in Sec 5, and QR factorizations of Vandermonde matrices in Sec 6. References to prior work on each of these applications can be found in the corresponding sections. This chapter complements our recent work on Toeplitz and close-to-Toeplitz matrices (see Chapter 2 or [5]). On a first reading, readers can skip all "Remarks" in this chapter without loss of continuity.

2. Some General Properties of Displacement Operators.

In this section, we shall examine useful choices for the *displacement operators* $\{F^f, F^b\}$ in the general definitions (6)-(7), and derive some results on Schur complements that will allow us to easily study the displacement structure of matrix products and inverses.

One of the criteria for choosing displacement operators is to make the corresponding displacement ranks of A as small as possible, because as will be seen presently the displacement rank determines the complexity of various operations on A. The special structures in the matrices (1)-(3), and the results (5) naturally suggest the shift matrix Z_n as an important candidate. In fact, for a Toeplitz matrix $T \in \mathbf{R}^{n\times n}$, a Hankel matrix $H \in \mathbf{R}^{n\times n}$ and a Vandermonde matrix $V = V(c,K) \in \mathbf{R}^{n\times n}$, we can readily see that

$$\nabla_{(Z_m,Z_n)}T = T - Z_m T Z_n^T = \begin{bmatrix} t_0 & t_1 & \cdot & \cdot & t_{n-1} \\ t_1 & & & & \\ \cdot & & \bigcirc & & \\ t_{m-1} & & & & \end{bmatrix} \quad \text{has rank 2,} \tag{8a}$$

$$\Delta_{(Z_n,Z_n)}H = Z_n H - H Z_n^T = \begin{bmatrix} 0 & -h_0 & -h_1 & \cdot & -h_{n-1} \\ h_0 & & & & \\ h_1 & & \bigcirc & & \\ \cdot & & & & \\ h_{n-1} & & & & \end{bmatrix} \quad \text{has rank 2,} \tag{8b}$$

$$\Delta_{(Z_n,K^{-1})}V = Z_n V - V K^{-1} = \begin{bmatrix} - & -c^T K^{-1} & - \\ & \bigcirc & \\ & & \end{bmatrix} \quad \text{has rank 1.} \tag{8c}$$

But how about non-Toeplitz, non-Hankel or non-Vandermonde matrices, but ones that are known to be the inverses of some Toeplitz, Hankel or Vandermonde matrices, respectively? The following lemma indicates that these matrices also have displacement structure, a fact first noted in [13] for Toeplitz matrices.

Lemma 2.1 Displacement rank of Inverses. *For any nonsingular matrix A,*

$$\alpha_{(F^f,F^b)}A = \alpha_{(F^{bT},F^{fT})}A^{-1}, \quad \beta_{(F^f,F^b)}A = \beta_{(F^{bT},F^{fT})}A^{-1}.$$

Proof.

$$\alpha_{(F^f,F^b)}A = rank[(A - F^f A F^{bT})A^{-1}] = rank[I - F^f A F^{bT} A^{-1}],$$
$$\alpha_{(F^{bT},F^{fT})}A^{-1} = rank[(A^{-1} - F^{bT} A^{-1} F^f)A] = rank[I - F^{bT} A^{-1} F^f A].$$

But the nonzero eigenvalues of $(F^f A)(F^{bT} A^{-1})$ and $(F^{bT} A^{-1})(F^f A)$ are identical. Therefore, $\alpha_{(F^f,F^b)}A = \alpha_{(F^{bT},F^{fT})}A^{-1}$. Next, note that

$$\beta_{(F^f,F^b)}A = rank[(F^f A - A F^{bT})A^{-1}] = rank[(F^f - A F^{bT} A^{-1})],$$
$$\beta_{(F^{bT},F^{fT})}A^{-1} = rank[A(F^{bT} A^{-1} - A^{-1} F^f)] = rank[A F^{bT} A^{-1} - F^f],$$

and therefore, $\beta_{(F^f,F^b)}A = \beta_{(F^{bT},F^{fT})}A^{-1}$. $\square$

In particular, we see immediately from Lemma 2.1 and (8) that

$$rank[T^{-1} - Z_n^T T^{-1} Z_n] = rank[T - Z_n T Z_n^T] = 2, \tag{9a}$$

$$rank[Z_n^T H^{-1} - H^{-1} Z_n] = rank[Z_n H - H Z_n^T] = 2, \tag{9b}$$

$$rank[K^{-1} V^{-1} - V^{-1} Z_n] = rank[Z_n V - V K^{-1}] = 1. \tag{9c}$$

Similar results can be obtained for matrix products. However, it will be useful to first consider the displacement properties of the so-called matrix Schur complements. The formation of Schur complements is the heart of triangularization procedures for matrices. Moreover we shall see that working with Schur complements of appropriately defined block matrices leads immediately to results on the displacement structure of matrix products (and inverses). The following lemma generalizes a result first given in [19].

Lemma 2.2 Displacement rank of Schur complements. *Let $\bar{S} \in \mathbf{R}^{(m-i)\times(n-i)}$ be the Schur complement of $A \in \mathbf{R}^{i\times i}$ in $M \in \mathbf{R}^{m\times n}$, i.e., let*

$$M \equiv \begin{bmatrix} A & B \\ C & D \end{bmatrix}, \quad S \equiv \begin{bmatrix} O & O \\ O & \bar{S} \end{bmatrix} \in \mathbf{R}^{m\times n}, \quad \bar{S} \equiv D - CA^{-1}B, \quad A\text{: nonsingular.}$$

If $F^f \in \mathbf{R}^{m\times m}$ and $F^b \in \mathbf{R}^{n\times n}$ are block lower triangular matrices, i.e.,

$$F^f = \begin{bmatrix} F_1^f & O \\ F_3^f & F_2^f \end{bmatrix}, \quad F^b = \begin{bmatrix} F_1^b & O \\ F_3^b & F_2^b \end{bmatrix}, \quad F_1^f \in \mathbf{R}^{i\times i}, \quad F_1^b \in \mathbf{R}^{i\times i}, \tag{10}$$

then

$$\alpha_{(F^f, F^b)} S = \alpha_{(F_2^f, F_2^b)} \bar{S} \le \alpha_{(F^f, F^b)} M, \tag{11a}$$

$$\beta_{(F^f, F^b)} S = \beta_{(F_2^f, F_2^b)} \bar{S} \le \beta_{(F^f, F^b)} M. \tag{11b}$$

Proof. It is not hard to check that M^{-1} has the form

$$M^{-1} = \begin{bmatrix} * & * \\ * & \bar{S}^{-1} \end{bmatrix}.$$

Therefore,

$$\alpha_{(F^f, F^b)} M = \alpha_{(F^{bT}, F^{fT})} M^{-1} \ge \alpha_{(F_2^{bT}, F_2^{fT})} \bar{S}^{-1} = \alpha_{(F_2^f, F_2^b)} \bar{S} = \alpha_{(F^f, F^b)} S, \tag{12a}$$

$$\beta_{(F^f,F^b)}M = \beta_{(F^{bT},F^{fT})}M^{-1} \geq \beta_{(F_2^{bT},F_2^{fT})}\bar{S}^{-1} = \beta_{(F_2^f,F_2^b)}\bar{S} = \beta_{(F^f,F^b)}S, \tag{12b}$$

where the first and third equalities in (12a) and (12b) follow from Lemma 2.1. The second inequalities follow from the block lower triangularity of F^f and F^b, and from the fact that a submatrix has smaller rank than the matrix. □

Remark 2.1. Without the triangularity assumption on F^f and F^b, the Schur complements may have larger displacement rank than the matrix itself. See Remark 2.4 below.

Applications.

Judicious use of Schur complements will allow us to easily derive the displacement properties of matrix inverses and matrix products. For example, we could alternatively have obtained the results in (9) as follows. Consider the *(extended)* matrices

$$M_1 \equiv \begin{bmatrix} T & I \\ I & O \end{bmatrix}, \quad M_2 \equiv \begin{bmatrix} H & I \\ I & O \end{bmatrix}, \quad M_3 \equiv \begin{bmatrix} V & I \\ I & O \end{bmatrix}, \tag{13a}$$

and define the displacement operators

$$F_1 \equiv \begin{bmatrix} Z_n^T & O \\ O & Z_n^T \end{bmatrix}, \quad F_2 \equiv \begin{bmatrix} Z_n & O \\ O & Z_n^T \end{bmatrix}, \quad F_3^f \equiv \begin{bmatrix} Z_n & O \\ O & K^{-1} \end{bmatrix}, \quad F_3^b \equiv \begin{bmatrix} K^{-1} & O \\ O & Z_n^T \end{bmatrix}. \tag{13b}$$

Then by using Lemma 2.2 one can check that

$$\alpha_{(Z_n^T,Z_n^T)}T^{-1} = \alpha_{(F_1,F_1)}M_1 = 2, \tag{13c}$$

$$\beta_{(Z_n^T,Z_n^T)}H^{-1} = \beta_{(F_2,F_2)}M_2 = 2, \tag{13d}$$

$$\beta_{(K^{-1},Z_n^T)}V^{-1} = \beta_{(F_3^f,F_3^b)}M_3 = 1. \tag{13e}$$

Lemma 2.2 is also useful in determining the displacement ranks of products of matrices. For example, let

$$M_1 \equiv \begin{bmatrix} I & T \\ T^T & O \end{bmatrix}, \quad F \equiv \begin{bmatrix} Z_n & O \\ O & Z_n \end{bmatrix}.$$

Then $-T^T T$ is the Schur complement of I in M_1, and therefore,

$$\alpha_{(F,F)}M_1 = 4 = \alpha_{(Z_n,Z_n)}T^T T. \tag{14a}$$

Similarly, let

$$M_2 \equiv \begin{bmatrix} I & H \\ H^T & O \end{bmatrix}, \quad F \equiv \begin{bmatrix} Z_n & O \\ O & Z_n^T \end{bmatrix}.$$

Then, we can see that $H^T H$ has low Hankel displacement with respect to $\{Z_n^T, Z_n^T\}$ (instead of $\{Z_n, Z_n\}$) because

$$\beta_{(F,F)} M_2 = 4 = \beta_{(Z_n^T, Z_n^T)} H^T H. \tag{14b}$$

Finally, for the product of Vandermonde matrices, consider the matrix

$$M_3 \equiv \begin{bmatrix} I & V^T \\ V & O \end{bmatrix}, \quad F \equiv \begin{bmatrix} K^{-1} & O \\ O & Z_n \end{bmatrix}.$$

Then we can see that VV^T has rank-2 Hankel displacement

$$\beta_{(F,F)} M_3 = 2 = \beta_{(Z_n, Z_n)} VV^T. \tag{14c}$$

This is not surprising because VV^T is, in fact, a Hankel matrix. Also some experiments will show that $V^T V$ does not seem to have a low-displacement rank with respect to "simple" displacement operators.

Remark 2.2. Referring to the result in (14b), we may note that not $H^T H$, but rather $H^T \bar{I} H$, where $\bar{I}$ is the *reverse-identity* (ones on the antidiagonal) matrix has low Hankel displacement rank with respect to the usual Hankel displacement operators $\{Z_n, Z_n\}$. To see this consider the following matrix M_2 and displacement operator;

$$M_2 \equiv \begin{bmatrix} \bar{I} & H \\ H^T & O \end{bmatrix}, \quad F \equiv \begin{bmatrix} Z_n & O \\ O & Z_n \end{bmatrix}. \tag{15}$$

Then

$$\beta_{(F,F)} M_2 = 4 = \beta_{(Z_n, Z_n)} H^T \bar{I} H.$$

Remark 2.3. By considering 3×3 or higher order block matrices, one can determine the displacement rank of other *composite* matrices. For instance, the matrix $\bar{I}(H_4 - H_3 H_1^{-1} H_2)^{-1} \bar{I}$, where H_i is a Hankel matrix, has Hankel displacement rank 4 with respect to $\{F, F\}$, where

$$F \equiv \begin{bmatrix} Z_n & O & O \\ O & Z_n & O \\ O & O & Z_n \end{bmatrix}.$$

Remark 2.4. One might have noticed that the Hankel matrix H also has a low *Toeplitz displacement* rank, viz, that

$$\nabla_{(Z_n, Z_n^T)} H = H - Z_n H Z_n, \quad \alpha_{(Z_n, Z_n^T)} H = 2.$$

The only difficulty of using this displacement in the context of the fast algorithm in Sec 3, is that the displacement operator Z_n^T is not a block lower triangular matrix. Therefore, the Schur complements of H can have larger displacement ranks than that of H. With a slight modification of Lemma 2.2, one can show that all Schur complements of a Hankel matrix with respect to this displacement operator have rank 3.

Remark 2.5. A displacement of a matrix A can *characterize* the matrix A if we can solve the equations (6) and (7) for A *uniquely* (the equations (6) and (7) are necessarily *consistent* in our context). It is well known that the (consistent) equation (6) uniquely determines the solution A, regardless of what the displacement operators $\{F^f, F^b\}$ are. However, the (consistent) equation (7) has non-unique solutions (see, e.g., [11], [15]) if (and only if) there exists a pair of eigenvalues $\lambda_i(F^f)$ and $\lambda_j(F^b)$ such that

$$\lambda_i(F^f) - \lambda_j(F^b) = 0. \tag{16}$$

This condition holds for most of the displacement operators for close-to-Hankel matrices that we are interested in, which is unfortunate because we are concerned how to find L and U such that $A = LU$ (in a fast way), given *only* the displacement of A. We shall circumvent this non-uniqueness problem by imposing an additional constraint (see Sec 3).

3. Fast Partial Triangular Factorization using Hankel Generators.

In the rest of this chapter, we shall only consider Hankel displacement structure.

Corresponding results on Toeplitz displacement can be found in [5]. Also it is important to mention at this point that we consider only *strongly nonsingular matrices* for triangular factorization, and *full column rank matrices* for QR factorization.

General Schur Reduction.

We note (see, e.g., [9], also [5], [17], [18], [23], [24]) that the standard triangular factorization procedure can be regarded as arising from the recursive computation of the Schur complements $\{S_i\}$ of the leading principal submatrices of a given matrix. Let $A_0 \equiv S_0 \equiv A \in \mathbf{R}^{n\times n}$, and define $\mathbf{l}_i \in \mathbf{R}^{n\times 1}$, $\mathbf{u}_i \in \mathbf{R}^{n\times 1}$, $d_i \neq 0$ and the *ith reduced matrix* A_i recursively by

$$A_{i-1} = \begin{bmatrix} | \\ \mathbf{l}_i \\ | \end{bmatrix} d_i \, [\, \text{---}\ \mathbf{u}_i^T\ \text{---}\,] + A_i, \qquad A_i \equiv \begin{bmatrix} O_{i\times i} & O_{i\times(n-i)} \\ O_{(n-i)\times i} & S_i \end{bmatrix}. \tag{17a}$$

Given A_{i-1}, we can determine the quantities in (17a) as

$$\mathbf{l}_i = A_{i-1}\mathbf{e}_i, \quad \mathbf{u}_i = A_{i-1}^T\mathbf{e}_i, \quad d_i = 1/[\mathbf{l}_i]_i = 1/[\mathbf{u}_i]_i, \tag{17b}$$

where $\mathbf{e}_i$ is the unit vector with one at the ith position and zero's elsewhere, and $[\mathbf{v}]_i$ denotes the ith element of the vector $\mathbf{v}$. The computation of the reduced matrix A_i from A_{i-1} will be called (one step) *Schur reduction* [9], [17]. Using r Schur reduction steps, we can obtain the *(r-step) partial triangular factorization,*

$$A = \sum_{i=1}^{r} \mathbf{l}_i d_i \mathbf{u}_i^T + A_r \tag{18}$$

$$= \begin{bmatrix} | & & | \\ \mathbf{l}_1 & \cdots & \mathbf{l}_r \\ | & & | \end{bmatrix} \begin{bmatrix} d_1 & & \\ & \ddots & \\ & & d_r \end{bmatrix} \begin{bmatrix} \text{---}\ \mathbf{u}_1^T\ \text{---} \\ \vdots \\ \text{---}\ \mathbf{u}_r^T\ \text{---} \end{bmatrix} + \begin{bmatrix} \bigcirc & \bigcirc \\ \bigcirc & S_r \end{bmatrix}$$

$$\equiv LDU^T + A_r.$$

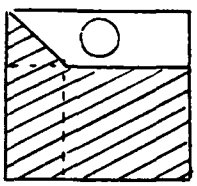
The "trapezoid" matrices L, U and the diagonal matrix D will be called *(r-step) partial triangular factors* of A. Notice that r Schur reduction steps take $O(rn^2)$ flops.

Schur Reduction using Hankel Displacements.

The displacement rank of a matrix A can be much smaller than the rank of the matrix A itself. Also if the chosen displacement operators $\{F^f, F^b\}$ for a given matrix $A \in \mathbf{R}^{n\times n}$ satisfy the condition in Lemma 2.2 *for all* $1 \le i \le r$, viz, that

the $r \times r$ leading principal submatrices of F^f and F^b are lower triangular, (19a)

or pictorially,

$F =$  (19b)

then the displacement ranks of the reduced matrices $(A_i\colon 0 \le i \le r)$ do not increase:

$$\beta_{(F^f,F^b)}A_i \le \beta_{(F^f,F^b)}A_{i-1}, \quad 1 \le i \le r. \tag{20}$$

Note that $\Delta_{(F^f,F^b)}A_i$ is determined only by $O(\beta n)$ parameters, whereas the matrix A_i itself needs $O(n^2)$ parameters. Therefore, we can hope that the Schur reduction procedure in (17) can be done more efficiently with $O(r\beta n)$ flops (instead of $O(rn^2)$ flops) if we successively compute $\Delta_{(F^f,F^b)}A_i$ rather than A_i. This is indeed the case as we shall see shortly. For the rest of this chapter, we shall restrict ourselves only to displacement operators that have the form (19), because (20) is a nice property to have.

Recalling the definition of displacement,

$$\Delta_{(F^f,F^b)}A_{i-1} = F^f A_{i-1} - A_{i-1}F^{bT}, \qquad F^f \text{ and } F^b \text{ have the form (19),}$$

we can check that

$$[\Delta_{(F^f,F^b)}A_{i-1}]\mathbf{e}_i = (F^f A_{i-1} - A_{i-1}F^{bT})\mathbf{e}_i = (F^f - [F^b]_{i,i}I)\mathbf{l}_i, \tag{21a}$$

$$[\Delta_{(F^f,F^b)}A_{i-1}]^T\mathbf{e}_i = (-F^b A_{i-1}^T + A_{i-1}^T F^{fT})\mathbf{e}_i = -(F^b - [F^f]_{i,i}I)\mathbf{u}_i, \tag{21b}$$

where $[F]_{i,j}$ denotes the (i,j)th element of F. Therefore, $\mathbf{l}_i$ and $\mathbf{u}_i$ can be obtained by

$$\mathbf{l}_i = (F^f - [F^b]_{i,i} I)^{\#} [\Delta_{(F^f, F^b)} A_{i-1}] \mathbf{e}_i, \quad \mathbf{u}_i = -(F^b - [F^f]_{i,i} I)^{\#} [\Delta_{(F^f, F^b)} A_{i-1}]^T \mathbf{e}_i, \tag{22}$$

where the superscript # denotes an *appropriate* generalized inverse. After determining $\mathbf{l}_i$, $\mathbf{u}_i$ and d_i, we can obtain the displacement of the reduced matrix A_i as

$$\begin{aligned} \Delta_{(F^f, F^b)} A_i &= F^f A_i - A_i F^{bT} && \text{(23a)} \\ &= F^f A_{i-1} - A_{i-1} F^{bT} - F^f \mathbf{l}_i d_i \mathbf{u}_i^T + \mathbf{l}_i d_i \mathbf{u}_i^T F^{bT} && \text{(23b)} \\ &= \Delta_{(F^f, F^b)} A_{i-1} - F^f \mathbf{l}_i d_i \mathbf{u}_i^T + \mathbf{l}_i d_i \mathbf{u}_i^T F^{bT}. && \text{(23c)} \end{aligned}$$

If $(F^f - [F^b]_{i,i} I)$ and $(F^b - [F^f]_{i,i} I)$ are singular, then there are many $\mathbf{l}_i$'s and $\mathbf{u}_i$'s that would satisfy (21). Let us consider separately the nonsingular and singular cases.

A. Nonsingular cases.

If $(F^f - [F^b]_{i,i} I)$ and $(F^b - [F^f]_{i,i} I)$ are nonsingular, i.e., if $[F^b]_{i,i}$ and $[F^f]_{i,i}$ are not eigenvalues of F^f and F^b, respectively, then $\mathbf{l}_i$, $\mathbf{u}_i$ and d_i and therefore A_i will be determined uniquely by taking the ordinary inverse in (22).

Example 3.1. *Let $F^f = Z_n$ and $F^b = K^{-1}$. These displacement operators are useful for the Vandermonde matrix $V = V(\mathbf{c}, K) \in \mathbf{R}^{n \times n}$ because they give the smallest displacement rank. Note that $(Z_n - [K^{-1}]_{i,i} I)$ and $(K^{-1} - [Z_n]_{i,i} I) = K^{-1}$ are nonsingular for $1 \le i \le n$.*

B. Singular cases.

If $(F^f - [F^b]_{i,i} I)$ and/or $(F^b - [F^f]_{i,i} I)$ are singular then the Schur reduction using (22) and (23) is ambiguous. Note that we are not completely free in choosing F^f and F^b, because the structure of a given matrix dictates appropriate F^f and F^b that give the smallest displacement rank. Instead, we shall overcome this difficulty by using the following two approaches. The key observation behind these approaches is the fact that only the projection of $\mathbf{l}_i$ and $\mathbf{u}_i$ lying in *kernel*$(F^f - [F^b]_{i,i} I)$ and *kernel*$(F^b - [F^f]_{i,i} I)$ are not uniquely determined.

If we have additional information about such ambiguous components of $\mathbf{l}_i$ and $\mathbf{u}_i$, then we can determine $\mathbf{l}_i$ and $\mathbf{u}_i$ correctly. We shall use this approach for the triangularization of

the inverse of Vandermonde matrices in Sec 4.

The other approach is to extend the matrix $A \in \mathbf{R}^{n\times n}$ to a larger one, say $\bar{A} \in \mathbf{R}^{m\times m}$, such that A is a leading principal submatrix of $\bar{A}$. Let $\bar{L} \in \mathbf{R}^{m\times n}$ and $\bar{U} \in \mathbf{R}^{n\times m}$ be n-step *partial* triangular factors of $\bar{A}$. Displacement operators $\bar{F}^f$ and $\bar{F}^b$ for $\bar{A}$ are chosen such that the following is true,

$$\bar{L} \perp kernel(\bar{F}^f - [\bar{F}^b]_{i,i} I) \quad \text{and} \quad \bar{U}^T \perp kernel(\bar{F}^b - [\bar{F}^f]_{i,i} I), \tag{24}$$

where $A \perp B$ denotes $A^T B = O$. Now, we can perform n-step partial triangularization of $\bar{A}$ using (22) and (23) unambiguously, because we can compute $\bar{\mathbf{l}}_i$ (the ith column of $\bar{L}$) and $\bar{\mathbf{u}}_i$ (the ith column of $\bar{U}^T$) by taking the Moore-Penrose inverse (pseudo-inverse) in (22). After finding the n-step *partial triangular factors* of $\bar{A} \in \mathbf{R}^{m\times m}$, we can obtain the *triangular factors* L and U of A simply by deleting the $m - n$ rows of $\bar{L}$ and $\bar{U}$.

Example 3.2. *Let $H \in \mathbf{R}^{n\times n}$ be a Hankel matrix in (2). For Hankel matrices, the (desirable) displacement operators are $\{Z_n, Z_n\}$. Note that $(Z_n - [Z_n]_{i,i} I) = Z_n$ is singular. However, if we define*

$$\bar{H}_1 = \begin{bmatrix} H & 0 \\ 0^T & 0 \end{bmatrix} \in \mathbf{R}^{(n+1)\times(n+1)}, \tag{25a}$$

then $\beta_{(Z_{n+1}, Z_{n+1})}\bar{H}_1$ is still small (in fact, 4), and partial triangular factors of H_1 and the displacement operators $\{Z_{n+1}, Z_{n+1}\}$ satisfy (24).

Example 3.3. *Let $H \in \mathbf{R}^{n\times n}$ be a Hankel matrix in (2). Define*

$$\bar{H}_2 = \begin{bmatrix} H & U_R \\ U_R & O \end{bmatrix} \in \mathbf{R}^{2n\times 2n}, \tag{25b}$$

where U_R is the "reverse upper triangular" Hankel matrix (with zero elements in the lower-right corner) such that $\bar{H}_2$ is Hankel. As an example, for a 3×3 Hankel matrix, U_R has the form,

$$H = \begin{bmatrix} h_0 & h_1 & h_2 \\ h_1 & h_2 & h_3 \\ h_2 & h_3 & h_4 \end{bmatrix}, \quad U_R = \begin{bmatrix} h_3 & h_4 & 0 \\ h_4 & 0 & \\ 0 & & \end{bmatrix}.$$

Now the partial triangular factors of $\overline{H}_2$ *and the displacement operators* $\{Z_{2n}, Z_{2n}\}$ *satisfy (24).*

Generators of Matrices.

For a given matrix $A \in \mathbf{R}^{m\times n}$, any matrix pair, $\{X, Y\}$ such that

$$\Delta_{(F^f, F^b)} A = XY^T, \quad X \equiv [\, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_\beta \,] \in \mathbf{R}^{m\times\beta}, \quad Y \equiv [\, \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_\beta \,] \in \mathbf{R}^{n\times\beta}$$

is called a *(Hankel) generator* of A *with respect to* $\{F^f, F^b\}$. The numbers β are called the *length* of the generator (with respect to $\{F^f, F^b\}$). A generator with respect to $\{F^f, F^b\}$ with its length equal to the displacement rank is called a *minimal* generator (with respect to $\{F^f, F^b\}$).

Example 3.2 (continued). Generator of $\overline{H}_1$. *The matrix* $\overline{H}_1$ *in (25a) has the displacement,*

$$\Delta_{(Z_{n+1}, Z_{n+1})} \overline{H}_1 = \begin{bmatrix} 0 & -h_0 & \cdot & -h_{n-2} & -h_{n-1} \\ h_0 & & & & -h_n \\ \cdot & & \bigcirc & & \cdot \\ h_{n-2} & & & & -h_{2n-2} \\ h_{n-1} & h_n & \cdot & h_{2n-2} & 0 \end{bmatrix},$$

and therefore has a generator, $\{X_1, Y_1\}$, *where*

$$X_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cdot & -h_n & h_0 \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & 0 & -h_{2n-2} & \cdot \\ 0 & 1 & 0 & h_{n-1} \end{bmatrix}, \quad Y_1 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ -h_0 & h_n & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & h_{2n-2} & 0 & \cdot \\ -h_{n-1} & 0 & 1 & 0 \end{bmatrix}. \tag{26a}$$

Example 3.3 (continued). Generator of $\overline{H}_2$. *The matrix* $\overline{H}_2$ *in (25b) has the displacement,*

$$\Delta_{(Z_{2n},Z_{2n})}\bar{H}_2 = \begin{bmatrix} 0 & -h_0 & \cdot\ \cdot & -h_{2n-2} \\ h_0 & & & \\ \cdot & & \bigcirc & \\ \cdot & & & \\ h_{2n-2} & & & \end{bmatrix},$$

and therefore has a generator, $\{X_2, Y_2\}$, where

$$X_2 = \begin{bmatrix} 1 & 0 & \cdot\ \cdot & 0 \\ 0 & h_0 & \cdot\ \cdot & h_{2n-2} \end{bmatrix}^T, \quad Y_2 = \begin{bmatrix} 0 & -h_0 & \cdot\ \cdot & -h_{2n-2} \\ 1 & 0 & \cdot\ \cdot & 0 \end{bmatrix}^T. \tag{26b}$$

Fast Schur Reduction using Hankel Generators.

Now, let

$$\Delta_{(F^f,F^b)}A_{i-1} = X^{(i-1)}Y^{(i-1)T}.$$

Notice that the matrix products involving $\Delta_{(F^f,F^b)}A_{i-1}$ in (22) can be done more efficiently as

$$[\Delta_{(F^f,F^b)}A_{i-1}]\mathbf{e}_i = [X^{(i-1)}[Y^{(i-1)T}\mathbf{e}_i]], \quad [\Delta_{(F^f,F^b)}A_{i-1}]^T\mathbf{e}_i = [Y^{(i-1)}[X^{(i-1)T}\mathbf{e}_i]], \tag{27a}$$

where matrix-vector products are performed in the sequence as shown with the square-brackets. Furthermore, a generator of A_i can be obtained as

$$X^{(i)} = [X^{(i-1)}, -F^f\mathbf{l}_i d_i, \mathbf{l}_i d_i], \quad Y^{(i)} = [Y^{(i-1)}, \mathbf{u}_i, F^b\mathbf{u}_i], \tag{27b}$$

because of (23c). Although the generator given in (27b) is not minimal, it is possible to delete the two redundant columns in $X^{(i)}$ and $Y^{(i)}$ in (27b) in an efficient way [16].

However, the above Schur reduction procedure is still not efficient, because of the matrix inversions required in (22), and the matrix-vector multiplications $F^f\mathbf{l}_i$ and $F^b\mathbf{u}_i$ in (27b). Nevertheless, for structured matrices A (e.g., Hankel, block-Hankel, Hankel-block, Vandermonde etc), displacement operators, F^f and F^b are extremely simple so that such operations are trivial.

4. Fast Triangular Factorization of Vandermonde Matrices.

The problem of finding the coefficients of the nth degree interpolating polynomial can be

formulated as a problem of solving an $(n+1) \times (n+1)$ Vandermonde matrix equation. Bjorck and Pereyra first noted [2] that the *divided-difference scheme* (which needs $O(n^2)$ flops) for finding *Newton's form* of the interpolating polynomials, in fact, solves the Vandermonde matrix equations. They also presented an algorithm that needs $O(n^2)$ flops for the factorization $V^{-1} = UL$, along with other extensions. Recently, Gohberg, Kailath and Koltracht [8] obtained the algorithm of Bjorck and Pereyra by a different route and gave different extensions. In this section, we shall present two fast algorithms for computing the factorizations $V = LU$ as well as $V^{-1} = UL$. We believe that our approach is more fundamental and provides richer insight.

Consider the Vandermonde matrix

$$V = V(\mathbf{c},K) = \begin{bmatrix} \text{---} & \mathbf{c}^T & \text{---} \\ \text{---} & \mathbf{c}^T K & \text{---} \\ & \vdots & \\ \text{---} & \mathbf{c}^T K^{n-1} & \text{---} \end{bmatrix} \in \mathbf{R}^{n\times n}, \tag{28a}$$

where

$$K = \mathrm{diag}(k_1, k_2, \cdots, k_n), \quad \mathbf{c}^T = [1, 1, \cdots, 1]. \tag{28b}$$

Note that

$$\Delta_{(Z_n,K^{-1})}V = \mathbf{x}^{(0)}\mathbf{y}^{(0)T}, \quad \mathbf{x}^{(0)} \equiv \mathbf{e}_1 \in \mathbf{R}^{n\times 1}, \quad \mathbf{y}^{(0)} \equiv -K^{-1}\mathbf{c} \in \mathbf{R}^{n\times 1}. \tag{29}$$

Therefore, V has a generator $\{\mathbf{x}^{(0)}, \mathbf{y}^{(0)}\}$ of length 1. Now the Schur reduction steps specialized for Vandermonde matrices can be summarized in the following theorem.

Theorem 4.1. *Let* $\Delta_{(Z_n,K^{-1})}V_{i-1} = \mathbf{x}^{(i-1)}\mathbf{y}^{(i-1)T}$. *Then*

$$\mathbf{l}_i = -k_i[\mathbf{y}^{(i-1)}]_i(I-k_iZ_n)^{-1}\mathbf{x}^{(i-1)}, \tag{30a}$$

$$\mathbf{u}_i = -K\mathbf{y}^{(i-1)}\mathbf{x}^{(i-1)T}\mathbf{e}_i = -K[\mathbf{x}^{(i-1)}]_i\mathbf{y}^{(i-1)}, \tag{30b}$$

$$d_i = 1/[\mathbf{l}_i]_i, \tag{30c}$$

and $\Delta_{(Z_n,K^{-1})}V_i = \mathbf{x}^{(i)}\mathbf{y}^{(i)T}$, *where*

$$\mathbf{x}^{(i)} = \left[[\mathbf{y}^{(i-1)}]_{i+1}\mathbf{x}^{(i-1)} - d_i[\mathbf{u}_i]_{i+1}Z_n\mathbf{l}_i + (d_i[\mathbf{u}_i]_{i+1}/k_{i+1})\mathbf{l}_i \right] / [\mathbf{x}^{(i)}]_{i+1}, \tag{31a}$$

$$\mathbf{y}^{(i)} = [\mathbf{x}^{(i-1)}]_{i+1}\mathbf{y}^{(i-1)} - d_i[\mathbf{l}_i]_i\mathbf{u}_i + d_i[\mathbf{l}_i]_{i+1}K^{-1}\mathbf{u}_i. \tag{31b}$$

Proof. (30) follows immediately from (22). From (27b), we have

$$X^{(i)}Y^{(i)T} = \mathbf{x}^{(i-1)}\mathbf{y}^{(i-1)T} - Z_n \mathbf{l}_i d_i \mathbf{u}_i^T + \mathbf{l}_i d_i \mathbf{u}_i^T K^{-1}. \tag{32}$$

The matrix $X^{(i)}Y^{(i)T}$, which is the displacement of the ith reduced matrix, has the null rows and columns from 1 to i. Because it is a rank-one matrix, a minimal generator of $X^{(i)}Y^{(i)T}$ can be obtained simply by taking the $(i+1)$st row and the $(i+1)$st column with an appropriate normalization;

$$\mathbf{x}^{(i)} = X^{(i)}Y^{(i)T}\mathbf{e}_{i+1}/[\mathbf{x}^{(i)}]_{i+1}, \quad \mathbf{y}^{(i)} = Y^{(i)}X^{(i)T}\mathbf{e}_{i+1}. \tag{33}$$

The generator $\{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}$ in (31) follows from (33), after inserting (32) for $X^{(i)}Y^{(i)T}$. □

Now we shall summarize the algorithm.

Algorithm 4.1. Fast Triangular Factorization of a Vandermonde Matrix

Input: A generator $\{\mathbf{x}^{(0)}, \mathbf{y}^{(0)}\}$ in (29) of $V \equiv V_0$;

Output: Triangular factorization, $V = LU$;

for $i := 1$ **to** n **do begin**

Compute $\mathbf{l}_i$, $\mathbf{u}_i$, d_i using (30); /* O(n) flops (see Remark 4.1 below) */

Obtain a generator of V_i using (31); /* O(n) flops */

end

$L := [\mathbf{l}_1, \mathbf{l}_2, \cdots, \mathbf{l}_n]$; $U^T := [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_n]$; $D := diag(d_1, d_2, \cdots, d_n)$;

return $(\{L, U, D\})$;

Remark 4.1. The matrix-vector multiplication, $\mathbf{p} \equiv (I - k_i Z_n)^{-1}\mathbf{x}^{(i-1)}$ in (30a) needs only $O(n)$ flops, because it is essentially the back-substitution procedure solving the bi-diagonal system, $(I - k_i Z_n)\mathbf{p} = \mathbf{x}^{(i-1)}$.

Remark 4.2. The above algorithm can be applied to any matrix V of the form (28a) with an arbitrary lower triangular matrix K and any vector $\mathbf{c}$ (rather than the $\mathbf{c}$ in (28b)). However, for such cases the algorithm may take greater than $O(n^2)$ flops.

Triangularization of the inverse of Vandermonde Matrices.

Consider the matrix,

$$M \equiv \begin{bmatrix} V & I \\ I & O \end{bmatrix}, \quad V \in \mathbf{R}^{n\times n}, \tag{34}$$

and its partial triangularization:

$$M = \begin{bmatrix} L_1 \\ U_2 \end{bmatrix} D \; [\, U_1 \, L_2 \,] + \begin{bmatrix} O & O \\ O & S \end{bmatrix}, \tag{35}$$

where U_2 and L_2^T are necessarily upper-triangular because M is banded. We can see that

$$V^{-1} = U_2 D L_2, \tag{36a}$$

because

$$V = L_1 D U_1, \quad I = U_2 D U_1, \quad I = L_1 D L_2. \tag{36b}$$

Therefore, we can obtain the factorization $V^{-1} = U_2 D L_2$ by the n-step partial-triangularization of the matrix M using the fast Schur reduction. To do so, we first need to find a generator of M with respect to appropriate displacement operators. Our choices of displacement operators are

$$F^f = \begin{bmatrix} Z_n & O \\ O & K^{-1} \end{bmatrix}, \quad F^b = \begin{bmatrix} K^{-1} & O \\ O & C_n^T \end{bmatrix}, \tag{37a}$$

where C_n is the *circular shift-down* matrix, i.e.,

$$C_n = \begin{bmatrix} 0 & & & 1 \\ 1 & 0 & & \\ & \ddots & \ddots & \\ & & 1 & 0 \end{bmatrix} \in \mathbf{R}^{n\times n}. \tag{37b}$$

Note that $\Delta_{(F^f, F^b)} M = \mathbf{x}^{(0)} \mathbf{y}^{(0)T}$ where

$$\mathbf{x}^{(0)} = \mathbf{e}_1 \in \mathbf{R}^{2n\times 1}, \quad \mathbf{y}^{(0)} = [-\mathbf{c}^T K^{-1}, -\mathbf{e}_n^T]^T \in \mathbf{R}^{2n\times 1}, \quad \mathbf{e}_n \in \mathbf{R}^{n\times 1}. \tag{38}$$

Note that F^f and F^b in (37a) have the form in (19). Also $(F^b - [F^f]_{i,i} I)$ is nonsingular for $1 \le i \le n$, and therefore, $\mathbf{u}_i$ can be obtained by taking the ordinary inverse in (22). However, $(F^f - [F^b]_{i,i} I)$ is singular for $1 \le i \le n$, and

$$kernel(F^f - [F^b]_{i,i}I) = span\begin{bmatrix} \mathbf{z} \\ \mathbf{e}_i \end{bmatrix}, \quad \mathbf{z} \in \mathbf{R}^{n\times 1}, \quad \mathbf{e}_i \in \mathbf{R}^{n\times 1}, \quad 1 \le i \le n, \tag{39}$$

where $\mathbf{z}$ is the null-vector, i.e., $[\mathbf{z}]_i = 0$ for all i. Therefore, if we take the pseudo-inverse in (22) for $\mathbf{l}_i$, then only the element $[\mathbf{l}_i]_{n+i}$ is not determined. However, note that this element $[\mathbf{l}_i]_{n+i}$ can be determined by other means. Namely,

$$[\mathbf{l}_i]_{n+i} = 1/(d_i \cdot [\mathbf{u}_i]_i),$$

because $U_2 = (DU_1)^{-1}$ from (36b).

Remark 4.3. The reason of using the F^b in (37a) rather than F_3^b in (13b) is to make $F^b - [F^f]_{i,i}I$ be nonsingular. We cannot, however, use C_n in the place of Z_n for F^f, because the resulting F^f would not have the form (19).

Now we shall summarize the n-step partial triangularization with the following theorem. The proof is similar to that for Theorem 4.1, and we shall omit it.

Theorem 4.2. *Let $\Delta_{(F^f,F^b)}M_{i-1} = \mathbf{x}^{(i-1)}\mathbf{y}^{(i-1)T}$, where $M_0 \equiv M$ and F^f, F^b are as in (34) and (37a), respectively. Then*

$$\mathbf{u}_i = -(F^b - [F^f]_{i,i}I)^{-1}\mathbf{y}^{(i-1)}\mathbf{x}^{(i-1)T}\mathbf{e}_i, \tag{40a}$$

$$\mathbf{l}_i = (F^f - [F^b]_{i,i}I)^{+}\mathbf{x}^{(i-1)}\mathbf{y}^{(i-1)T}\mathbf{e}_i + \mathbf{e}_{n+i}/(d_i \cdot [\mathbf{u}_i]_i), \tag{40b}$$

$$d_i = 1/[\mathbf{l}_i]_i, \tag{40c}$$

where A^+ denotes the pseudo-inverse of A. Also, $\Delta_{(F^f,F^b)}M_i = \mathbf{x}^{(i)}\mathbf{y}^{(i)T}$, where

$$\mathbf{x}^{(i)} = \left[[\mathbf{y}^{(i-1)}]_{i+1}\mathbf{x}^{(i-1)} - d_i[\mathbf{u}_i]_{i+1}F^f\mathbf{l}_i + (d_i[\mathbf{u}_i]_{i+1}/k_{i+1})\mathbf{l}_i \right] / [\mathbf{x}^{(i)}]_{i+1}, \tag{41a}$$

$$\mathbf{y}^{(i)} = [\mathbf{x}^{(i-1)}]_{i+1}\mathbf{y}^{(i-1)} - d_i[\mathbf{l}_i]_i\mathbf{u}_i + d_i[\mathbf{l}_i]_{i+1}F^b\mathbf{u}_i. \tag{41b}$$

Note that the computations of (40) take only $O(n)$ flops because

$$(F^b - [F^f]_{i,i}I)^{-1} = \begin{bmatrix} K & O \\ O & C_n \end{bmatrix}.$$

$$(F^f - [F^b]_{i,i}I)^{+} = \begin{bmatrix} -k_i(I - k_iZ_n)^{-1} & O \\ O & K^{(i)} \end{bmatrix}, \quad K^{(i)} \equiv K \text{ except } [K]_{i,i} = 0,$$

We shall summarize the algorithm.

Algorithm 4.2. Fast Triangular Factorization of V^{-1}.

Input: A generator $\{\mathbf{x}^{(0)}, \mathbf{y}^{(0)}\}$ in (38) of $M \equiv M_0 \in \mathbf{R}^{2n\times 2n}$;

Output: Triangular factorization, $V^{-1} = UL$;

for $i := 1$ **to** n **do begin**

Compute $\mathbf{l}_i, \mathbf{u}_i, d_i$ using (40); /* O(n) flops */

Obtain a generator of V_i using (41); /* O(n) flops */

end

$L := [\mathbf{l}_1, \mathbf{l}_2, \cdots, \mathbf{l}_n]$; $U^T := [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_n]$; $D := \mathrm{diag}(d_1, d_2, \cdots, d_n)$;

return (D and the bottom halfs of L, U^T);

5. Fast Triangular Factorization of close-to-Hankel Matrices.

In this section, we shall only consider *strictly* lower triangular displacement operators F^f and F^b, i.e., those with zeros on the main diagonal. It will be seen that the use of such displacement operators greatly simplifies finding minimal generators of Schur complements. Such displacement operators can be used for Hankel, Hankel block and block Hankel matrices.

Berlekamp [1], [19] (see also [3], [7]) was perhaps the first to describe a fast $O(n^2)$ algorithm (needs inner-product computations) for solving Hankel matrix equations; the closely related Berlekamp-Massey algorithm [19] is an algorithm of Phillips [21]. Rissanen [22] extended the results of Phillips to block-Hankel matrices; The Berlekamp-Massey algorithm involves certain inner-product computations, which is a bottle-neck for parallel evaluation. Recently, following earlier work of Kung [14] and Citron [7], Lev-Ari and Kailath [18] presented another fast algorithm that does not need inner-product computation. The results in this section can be regarded as an extension of the results of Lev-Ari and Kailath [18] to Hankel-block and block-Hankel matrices. Furthermore, we shall give a fast algorithm for com-

puting the triangular factorization of the inverse of Hankel matrices.

Let $\{X, Y\}$ be a Hankel generator of a matrix with respect to strictly lower triangular displacement operators $\{F^f, F^b\}$ *that also satisfy the condition (24).* (Otherwise, we assume that the matrix has been extended appropriately such that (24) holds.) We say that a Hankel generator is *proper* if, for a certain i, all the elements in the ith row of X and above, except for the element $[X]_{i,1}$, are zero, and all elements in the ith row of Y and above, except the element $[Y]_{i,\beta}$ are zero. Thus a proper generator has the form

$$X = [\mathbf{x}_1, \cdots, \mathbf{x}_\beta] = \begin{bmatrix} * & 0 & \text{—} & 0 \\ * & * & \text{—} & * \\ | & | & & | \\ * & * & \text{—} & * \end{bmatrix}, \quad Y = [\mathbf{y}_1, \cdots, \mathbf{y}_\beta] = \begin{bmatrix} 0 & \text{—} & 0 & * \\ * & \text{—} & * & * \\ | & & | & | \\ * & \text{—} & * & * \end{bmatrix}. \tag{42}$$

Before we show how to convert a non-proper generator to a proper one, we shall summarize the one-step Schur reduction with the following theorem. Often we shall denote a proper generator as $\{X_p, Y_p\}$ for clarity.

Theorem 5.1. *Let* $\Delta_{(F^f, F^b)} A_{i-1} = X_p^{(i-1)} Y_p^{(i-1)T}$, *where*

$$X_p^{(i-1)} = [\ \mathbf{x}_1^{(i-1)}, \mathbf{x}_2^{(i-1)}, \cdots, \mathbf{x}_\beta^{(i-1)}\ ], \quad Y_p^{(i-1)} = [\ \mathbf{y}_1^{(i-1)}, \mathbf{y}_2^{(i-1)}, \cdots, \mathbf{y}_\beta^{(i-1)}\ ].$$

Then

$$\mathbf{l}_i = [\mathbf{y}_\beta^{(i-1)}]_i (F^f)^+ \mathbf{x}_\beta^{(i-1)}, \quad \mathbf{u}_i = -[\mathbf{x}_1^{(i-1)}]_i (F^b)^+ \mathbf{y}_1^{(i-1)}, \quad d_i = 1/[\mathbf{l}_i]_i, \tag{43}$$

and $\Delta_{(F^f, F^b)} A_i = X^{(i)} Y^{(i)T}$, *where*

$$X^{(i)} \equiv [(\mathbf{x}_1^{(i-1)} - d_i [\mathbf{x}_1^{(i-1)}]_i \mathbf{l}_i), \mathbf{x}_2^{(i-1)}, \cdots, \mathbf{x}_\beta^{(i-1)}], \tag{44a}$$

$$Y^{(i)} \equiv [\mathbf{y}_1^{(i-1)}, \mathbf{y}_2^{(i-1)}, \cdots, (\mathbf{y}_\beta^{(i-1)} - d_i [\mathbf{y}_\beta^{(i-1)}]_i \mathbf{u}_i)]. \tag{44b}$$

Proof. (43) is immediate from (22). Using (43), we have

$$F^f \mathbf{l}_i d_i = [\mathbf{y}_\beta^{(i-1)}]_i d_i \mathbf{x}_\beta^{(i-1)}, \quad F^b \mathbf{u}_i = -[\mathbf{x}_1^{(i-1)}]_i \mathbf{y}_1^{(i-1)}.$$

Therefore, from (27b)

$$X^{(i)} = [X^{(i-1)}, -[\mathbf{y}_\beta^{(i-1)}]_i d_i \mathbf{x}_\beta^{(i-1)}, \mathbf{l}_i d_i], \quad Y^{(i)} = [Y^{(i-1)}, \mathbf{u}_i, -[\mathbf{x}_1^{(i-1)}]_i \mathbf{y}_1^{(i-1)}] \tag{45}$$

Now, (44) follows from (45). □

We can obtain the triangular factorization of close-to-Hankel matrices by applying Theorem 5.1 repeatedly as described below:

Algorithm 5.1. r-step Partial Triangular Factorization of Close-to-Hankel Matrices.

Input: A generator $\{X^{(0)}, Y^{(0)}\}$ of $A \equiv A_0 \in \mathbf{R}^{n\times n}$;

Output: Partial Triangular Factors $L \in \mathbf{R}^{n\times r}$ and $U \in \mathbf{R}^{r\times n}$;

for $i := 1$ **to** r **do begin**

Construct a proper generator of A_{i-1}; /* See below. */

Compute $\mathbf{l}_i$, $\mathbf{u}_i$, d_i using (43);

Obtain a generator of A_i by (44);

end

$L := [\mathbf{l}_1, \mathbf{l}_2, \cdots, \mathbf{l}_r]$; $U^T := [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_r]$; $D := \mathrm{diag}(d_1, d_2, \cdots, d_r)$;

return (L, U, D);

Example 5.1 Hankel matrices. *One can use Algorithm 5.1 to find the n-step partial triangularization of the extended matrices in (25) with any of the two generators in (26); both generators will need the same amount of computation. Triangular factors of Hankel matrices are obtained from the partial triangular factors of the extended matrices. Also note that $Z^+ = Z^T$.*

Example 5.2 Block-Hankel matrices. *The block Hankel matrix,*

$$H \equiv \begin{bmatrix} B_0 & B_1 & \cdots & B_{n-1} \\ B_1 & B_2 & \cdots & B_n \\ \cdot & \cdot & \cdots & \cdot \\ B_{n-1} & B_n & \cdots & B_{2n-2} \end{bmatrix} \in \mathbf{R}^{nb\times nb}, \quad B_i \in \mathbf{R}^{b\times b}, \tag{46}$$

has a low-rank displacement with respect to the block shift displacement operator Z_{nb}^b. However, this displacement operator and the block Hankel matrix (46) do not satisfy (24). We first add a block of null rows and a block of null columns to (46) to get the extended matrix

$\bar{H} \in \mathbf{R}^{(n+1)b\times(n+1)b}$. *Note that* $\bar{H}$ *also has low displacement rank with respect to* $\{Z^b_{(n+1)b}, Z^b_{(n+1)b}\}$. *One can easily find a generator of* $\bar{H}$ *with respect to these displacement operators. Now, we can find triangular factors of* H *from the* nb*-step partial triangular factors obtained by using the algorithm 5.1.*

Example 5.3 Hankel-Block Matrices. *Consider the following Hankel-block matrix* B *and its extended matrix* $\bar{B}$,

$$B \equiv \begin{bmatrix} H_1 & H_2 \\ H_3 & H_4 \end{bmatrix}, \quad \bar{B} \equiv \begin{bmatrix} B & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} \in \mathbf{R}^{(2n+1)\times(2n+1)}, \quad H_i \in \mathbf{R}^{n\times n}.$$

The matrix $\bar{B}$ *has displacement rank 6 with respect to* $\{Z_{2n+1}, Z_{2n+1}\}$ *We can obtain triangular factors of* B *from the* $2n$*-step partial triangular factors obtained by using the algorithm 5.1.*

Example 5.4 Inverse of Hankel Matrices. *Let* $H \in \mathbf{R}^{n\times n}$ *be a Hankel matrix. Define the matrices*

$$M \equiv \begin{bmatrix} H & \tilde{I} \\ \tilde{I} & O \end{bmatrix} \in \mathbf{R}^{2n\times 2n}, \quad \bar{M} \equiv \begin{bmatrix} M & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} \in \mathbf{R}^{(2n+1)\times(2n+1)}, \tag{47}$$

where $\tilde{I}$ *is the reverse identity (ones on the antidiagonal) matrix. The matrix* M *is a Hankel-block matrix of the form of* B *in Example 5.3, and the matrix* $\bar{M}$ *has displacement rank 4 with respect to* $\{Z_{2n+1}, Z_{2n+1}\}$. *We use the Algorithm 5.1 to get* n*-step partial triangular factorization of* $\bar{M}$,

$$\bar{M} = \begin{bmatrix} L_1 \\ C \\ \mathbf{0}^T \end{bmatrix} D [\, U_1 \; G \; \mathbf{0} \,] + \begin{bmatrix} O & O \\ O & S \end{bmatrix}$$

Note that

$$H^{-1} = U_2 D L_2, \quad U_2 \equiv \tilde{I}C, \quad L_2 \equiv G\tilde{I},$$

where U_2 *and* L_2^T *are upper-triangular, because*

$$H = L_1 D U_1, \quad \tilde{I} = C D U_1, \quad \tilde{I} = L_1 D G.$$

Example 5.5 Inverse of Hankel Matrices. *Instead of using the extended matrix $\overline{M}$ in (47), one can use the following extended matrix,*

$$\overline{M} \equiv \begin{bmatrix} \overline{H}_2 & 0 & I_{2n} \\ 0^T & 0 & 0^T \\ I_{2n} & 0 & O \end{bmatrix} \in \mathbf{R}^{(4n+1)\times(4n+1)}, \tag{48}$$

where $\overline{H}_2$ is as defined in (25b), and I_{2n} is the $2n \times 2n$ identity matrix. The matrix $\overline{M}$ in (48) has displacement rank 2 with respect to $\{F, F\}$, where $F = Z_{2n} \oplus Z_{2n+1}^T$. Also one can check that $\Delta_{(F,F)}\overline{M} = XY^T$, where

$$X = \begin{bmatrix} 1 & 0 & \cdot\cdot & 0 & 0 \\ 0 & h_0 & \cdot\cdot & h_{2n-2} & 1 \end{bmatrix}^T, \quad Y = \begin{bmatrix} 0 & -h_0 & \cdot\cdot & -h_{2n-2} & -1 \\ 1 & 0 & \cdot\cdot & 0 & 0 \end{bmatrix}^T. \tag{49}$$

Note that n-step partial triangular factors of $\overline{M}$ and the displacement operator F satisfy (24). Therefore, we can use the Algorithm 5.1 to get n-step partial triangular factorization of $\overline{M}$, and obtain triangular factors of H^{-1}.

Construction of Proper Generators.

The basic tool for constructing a proper (Hankel) generator is the use of *elimination matrices* $E_{i,j}(\eta)$, defined as the identity except for the element $[E_{i,j}(\eta)]_{i,j} = \eta$. Notice that $E_{i,j}^{-1}(\eta)$ is also different from the identity, except that $[E_{i,j}^{-1}(\eta)]_{i,j} = -\eta$. Let $\{X, Y\}$ be a non-proper generator of A. Without loss of generality we shall assume that $[X]_{i,1} \neq 0$. If not, we can always interchange (implicitly) columns of X and rows of Y^T to obtain such a generator of A. We can annihilate all elements in the ith row of X except the element $[X]_{i,1}$ by post-multiplying with the $n-1$ elimination matrices *pivoting with* the element $[X]_{i,1}$,

$$X\ E_{1,2}(\eta_2)E_{1,3}(\eta_3) \cdot\cdot E_{1,\beta}(\eta_\beta), \qquad Y\ E_{1,2}^{-T}(\eta_2)E_{1,3}^{-T}(\eta_3) \cdot\cdot E_{1,\beta}^{-T}(\eta_\beta),$$

where $\eta_k = -[X]_{i,k}/[X]_{i,1}$.

Again assuming that $[Y]_{i,\beta} \neq 0$, we can similarly annihilate all elements in the ith row of Y except the element $[Y]_{i,\beta}$ by post-multiplying with the $n-1$ elimination matrices;

$$Y\ E_{\beta-1,\beta}(\gamma_{\beta-1})E_{\beta-2,\beta}(\gamma_{\beta-2}) \cdots E_{1,\beta}(\gamma_1), \qquad X\ E^{-T}_{\beta-1,\beta}(\gamma_{\beta-1})E^{-T}_{\beta-2,\beta}(\gamma_{\beta-2}) \cdots E^{-T}_{1,\beta}(\gamma_1),$$

where $\gamma_k = -[Y]_{i,k}/[Y]_{i,\beta}$. Note that the last annihilation $E_{1,\beta}$ does not destroy the zero at $[X]_{i,\beta}$. This procedure will require $2n$ elimination matrices, and therefore $2\beta n$ flops.

If a matrix A is symmetric, then the matrix $\Delta_{(F,F)}A = XY^T$ is skew symmetric, and therefore, has the same number of positive and negative eigenvalues. Hence the *symmetric* displacement has the form

$$\Delta_{(F,F)}A = XPX^T, \quad P = \begin{bmatrix} O & -\bar{I}_\delta \\ \bar{I}_\delta & O \end{bmatrix},$$

where $\bar{I}_\delta$ is the $\delta \times \delta$ reverse identity matrix (Check the generators (26a) and (26b)). We call the generator $\{X, XP^T\}$ *skew symmetric*. With a skew symmetric generator $\{X, XP^T\}$ note that we only need to apply

$$X\ E_{1,2}(\eta_2)E_{1,3}(\eta_3) \cdots E_{1,\beta}(\eta_\beta),$$

to obtain a proper generator. Also, we only need to compute $\mathbf{l}_i$ in (43) and $X^{(i)}$ in (44a), and Algorithm 5.1 will give the Cholesky factorization $A = LDL^T$.

Remark 5.1. Algorithm 5.1 for finding the Cholesky factorization of the symmetric Hankel matrix in (2) by using the generator (26b) (choosing alternative pivoting elements) is identical to the Euclidean algorithm for finding

$$GCD(p(x), q(x)), \quad p(x) \equiv h_0x^{2n-2} + \cdots + h_{2n-3}x + h_{2n-2}, \quad q(x) \equiv x^{2n-1}.$$

We encourage readers to check this equivalence by using the 3×3 Hankel matrix,

$$H = \begin{bmatrix} 5 & 3 & 2 \\ 3 & 2 & 1 \\ 2 & 1 & 4 \end{bmatrix}.$$

Remark 5.2. Algorithm 5.1 reduces to the algorithm of Sugiyama *et. al.* [24] if we use the generator (49) to find the triangular factorization of the inverse of Hankel matrix. Also the

$O(n\log^2 n)$ algorithm of Brent *et. al.* [4] (see also [6]) for solving Toeplitz system of equations is closely related to the divide-and-conquer version of the algorithm by Sugiyama *et. al.* after permuting the rows of a Toeplitz matrix to make it Hankel. Furthermore, Berlekamp-Massey algorithm can be regarded as the "Levinson version" of the above procedure, so that one can work with only the bottom part of the algorithm applied to (49) (see [5]).

6. Fast QR Factorization of Vandermonde Matrices.

We shall show that the Algorithm 5.1 can be used for the QR factorization of the transpose of the Vandermonde matrix,

$$V^T = [\ \mathbf{c}, K\mathbf{c}, \cdots, K^{n-1}\mathbf{c}\ ], \quad K = \mathrm{diag}(\ k_1, k_2, \cdots, k_n).$$

First, notice that the matrix $VV^T \equiv H$ is a Hankel matrix,

$$VV^T = H = (h_{i+j-2}) = \begin{bmatrix} \mathbf{c}^T K\mathbf{c} & \mathbf{c}^T K^2\mathbf{c} & \cdot & \mathbf{c}^T K^n\mathbf{c} \\ \mathbf{c}^T K^2\mathbf{c} & \mathbf{c}^T K^3\mathbf{c} & \cdot & \mathbf{c}^T K^{n+1}\mathbf{c} \\ \cdot & \cdot & \cdot & \cdot \\ \mathbf{c}^T K^n\mathbf{c} & \mathbf{c}^T K^{n+1}\mathbf{c} & \cdot & \mathbf{c}^T K^{2n-1}\mathbf{c} \end{bmatrix} \in \mathbf{R}^{n\times n}.$$

Let us define

$$\bar{M}_1 \equiv \begin{bmatrix} \bar{H}_1 & \bar{V}_1 \\ \bar{V}_1^T & O \end{bmatrix} \in \mathbf{R}^{(2n+1)\times(2n+1)}, \tag{50}$$

where $\bar{H}_1$ is as (25a) and $\bar{V}_1^T = [\ V, K^n\mathbf{c}\ ]$. Note that

$$\beta_{(F_1,F_1)}\bar{M}_1 = 4, \quad F_1 \equiv Z_{n+1} \oplus K^{-1}$$
$$\Delta_{(F_1,F_1)}\bar{M}_1 = XPX^T, \tag{51a}$$

where

$$X = \begin{bmatrix} 1 & 0 & \cdot & \cdot & 0 & 0 & \cdot & \cdot & 0 \\ 0 & \cdot & \cdot & 0 & 1 & 0 & \cdot & \cdot & 0 \\ 0 & -h_n & \cdot & -h_{2n-2} & 0 & 0 & \cdot & \cdot & 0 \\ 0 & h_0 & \cdot & \cdot & h_{n-1} & k_1^{-1} & k_2^{-1} & \cdot & k_n^{-1} \end{bmatrix}^T, \quad P = \begin{bmatrix} & & & & & & & & -1 \\ & & & & & & & -1 & \\ & & & & & & 1 & & \\ & & & & & 1 & & & \end{bmatrix}. \tag{51b}$$

It is easy to check that the displacement operators $\{F_1, F_1\}$ and the n-step partial triangular factors of $\bar{M}_1$ satisfy (24). Therefore, we can use Algorithm 5.1 for the n-step partial

triangularization of

$$\overline{M}_1 = \begin{bmatrix} R^T \\ 0^T \\ Q \end{bmatrix} [\, R \;\, 0 \; Q^T \,] + \begin{bmatrix} O & O \\ O & S \end{bmatrix}. \tag{52}$$

Comparing (50) and (52), it is easy to see that

$$VV^T = R^T R, \quad V^T = QR, \quad Q^T Q = I.$$

Remark 6.1. Instead of using the matrix $\overline{M}_1$ in (50), one may use the extended matrix

$$\overline{M}_2 = \begin{bmatrix} \overline{H}_2 & \overline{V}_2 \\ \overline{V}_2^T & O \end{bmatrix} \in \mathbf{R}^{3n \times 3n},$$

where $\overline{H}_2$ is as in (25b), and

$$\overline{V}_2^T \equiv [\, V, W \,], \quad W \equiv K^n V.$$

For the matrix $\overline{M}_2$, one can check that

$$\beta_{(F_2,F_2)}\overline{M}_2 = 2, \quad F_2 \equiv Z_{2n} \oplus K^{-1},$$
$$\Delta_{(F_2,F_2)}\overline{M}_2 = XPX^T,$$

where

$$X = \begin{bmatrix} 1 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & 0 \\ 0 & h_0 & \cdot & \cdot & h_{2n-2} & k_1^{-1} & \cdot & k_n^{-1} \end{bmatrix}, \quad P \equiv \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}.$$

7. Concluding Remarks.

We introduced some generalized notion of displacement structure and developed some of their properties. The displacement structures associated with Toeplitz and close-to-Toeplitz matrices have been the most studied so far, with some new results in [5]. In this chapter we have focused on Hankel and close-to-Hankel matrices, and presented a general algorithm for triangular factorization of such matrices and their inverses. This general algorithm was also extended to obtain the triangular factorizations of Vandermonde and close-to-Vandermonde matrices and their inverses, and the QR factorizations of Vandermonde matrices and close-to-

Vandermonde matrices. (The QR factorization of Hankel matrices can be obtained via the QR factorization algorithm for Toeplitz matrices [5]). Relationships with all earlier algorithms for these problems have also been noted. We remark that Algorithm 5.1 can be easily implemented as a divide-and-conquer fashion. The approach taken in [6] can be used for this purpose.

REFERENCES

[1]. E. Berlekamp, *Algebraic coding theory*, McGraw-Hill, New York, 1968.

[2]. A. Bjorck and V. Pereyra, *Solution of Vandermonde systems of equations*, Math Comp., 24 (1980).

[3]. R. Blahut, *Theory and practice of error control codes*, Addison-Wesley, Reading, MA, 1983.

[4]. R. Brent, F. Gustavson and D. Yun *Fast Solution of Toeplitz Systems of Equations and Computation of Pade Approximants* Journal of Algorithms, 1, (1980), pp. 259-295.

[5]. J. Chun and T. Kailath, *Generalized displacement structure for block-Toeplitz, Toeplitz-block and Toeplitz-derived matrices*, Proc. NATO Advanced Study Inst. on Signal Processing, 1989.

[6]. J. Chun and T. Kailath, *Divide-and-conquer solutions for least-squares problems for matrices with displacement structure*, Proc. Sixth Army Conf. on Applied Math. and Comput., Boulder, CO, June, 1988.

[7]. T. Citron, *Algorithms and architectures for error correcting codes*, Ph.D. Thesis, Stanford Univ., August 1986.

[8]. I. Gohberg, T. Kailath and I. Koltracht *Efficient solution of linear systems of equations*

with recursive structure, Linear Algebra and its Appl., 80: pp. 81-113 (1986).

[9]. G. Golub and C. Van Loan, *Matrix computations*, Johns Hopkins Univ. Press, Maryland, 1983.

[10]. G. Heinig and K. Rost, *Algebraic methods for Toeplitz-like matrices and operators*, Akademie-Verlag, Berlin, 1984.

[11]. T. Kailath, *Linear Systems*, Prentice-Hall, Englewood Cliffs, New Jersey, 1980.

[12]. T. Kailath, *Signal processing applications of some moment problems*, Proceedings of Symposia in Applied Mathematics, vol. 37, 1987 pp. 71-109.

[13]. T. Kailath, S. Kung and M. Morf, *Displacement ranks of matrices and linear equations*, J. Math. Anal. Appl., 68 (1979) pp. 395-407. See also Bull. Amer. Math. Soc., 1 (1979), pp. 769-773.

[14]. S. Kung, *Multivariable and multidimensional systems: analysis and design*, Ph.D. Thesis, Stanford Univ., 1977.

[15]. P. Lancaster, *Theory of matrices*, Academic Press, New York, 1969.

[16]. H. Lev-Ari, Personal Communication, 1988.

[17]. H. Lev-Ari and T. Kailath, *Lattice filter parameterizations and modeling of nonstationary process*, IEEE Trans. Inform. Theory, IT-30 (1984), pp. 2-16.

[18]. H. Lev-Ari and T. Kailath, *Triangular factorization of structured Hermitian matrices*, Operator Theory, Advances and Applications, vol. 18, Birkhäuser, Boston, pp. 301-324, (1986).

[19]. J. Massey, *Shift-register synthesis and BCH decoding*, IEEE Trans., Information Theory, IT-15, pp. 122-127 (1969).

[20]. M. Morf, *Doubling algorithms for Toeplitz and related equations*, Proceedings of the IEEE International Conf. on ASSP, Denver, (1980), pp. 954-959.

[21]. J. Phillips, *The triangular decomposition of Hankel matrices*, Math Comp., vol. 25, (1971) pp. 599-602.

[22]. J. Rissanen, *Algorithms for triangular decomposition of block Hankel and Toeplitz matrices with application to factoring positive matrix polynomial*, Math. Comput., vol. 27, Jan. (1973), pp. 147-154.

[23]. I. Schur, *Über Potenzreihen, die im Innern des Einheitskreises beschrankt sind*, J. für die Reine und Angewandte Mathematik, 147 (1917), pp. 205-232.

[24]. I. Schur, *On power series which are bounded in the interior of the unit circle. 1.* (English translation of [23]), Operator Theory, Advances and Applications, vol. 18, Birkhauser, Boston, pp. 31-60, (1986).

[25]. Y. Sugiyama, M. Kasahara, S. Hirasawa and T. Namekawa *A method for solving key equations for decoding Goppa codes*, Int. J. on Control, 27 pp. 87-99 (1975).

Chapter 5.

Divide-and-Conquer Solutions of Least-Squares Problems for Matrices with Displacement Structure

Abstract

A divide-and-conquer implementation of a generalized Schur algorithm enables us to obtain (exact and) least-squares solutions of various block-Toeplitz or Toeplitz-block systems of equations with $O(\alpha^3 n \log^2 n)$ operations, where the displacement rank α is a small constant (typically between 2 to 4 for scalar near-Toeplitz matrices) independent of the size of matrices.

1. Introduction.

In recent years, there has been considerable research on fast algorithms for the solution of linear systems of equations with Toeplitz matrices. The Levinson and Schur algorithms allow solutions with $O(n^2)$ floating point operations (flops) for systems with $n \times n$ Toeplitz matrices.

In 1980. Brent *et al* [5] described a scheme for obtaining a solution with $O(n \log^2 n)$ flops. This was based on two ideas - the use of the Gohberg-Semencul formula [11], [12], [16] for the inverse of a Toeplitz matrix, and the use of divide-and-conquer (or doubling) techniques for computing (generators of) the Gohberg-Semencul formula.

Let x and y denote the first and last columns of $T^{-1} \in \mathbf{R}^{n \times n}$. Then if the first component of x, say x_1, is nonzero, Gohberg and Semencul [12] showed that we could write

$$T^{-1} = \frac{1}{x_1}[L(\mathbf{x})L^T(\tilde{I}_n \mathbf{y}) - L(Z_n \mathbf{y})L^T(Z_n \tilde{I}_n \mathbf{x})], \quad x_1 \neq 0, \tag{1}$$

where $\tilde{I}_n$ is the *reverse-identity matrix*, Z_n is the *shift matrix*,

$$\tilde{I}_n \equiv \begin{bmatrix} & & & 1 \\ & & \iddots & \\ & 1 & & \\ 1 & & & \end{bmatrix}, \quad Z_n \equiv \begin{bmatrix} 0 & & & \\ 1 & 0 & & \\ & 1 & \ddots & \\ & & 1 & 0 \end{bmatrix},$$

and

$L(\mathbf{v})$ = a lower-triangular Toeplitz matrix with first column v.

The significance of (1) in the present application is that the product of a vector and a lower- or upper-triangular Toeplitz matrix is equivalent to the convolution of two vectors, which can be done using $O(n \log n)$ flops (see, e.g. [4]).

Brent *et al* used a divide-and-conquer scheme for a certain Euclidean algorithm to factorize row-permuted Toeplitz matrices (i.e., Hankel matrices), and to obtain the vectors {x, y} of the Gohberg-Semencul formula with $O(n \log^2 n)$ flops [See [9] for the connection of Brent *et al's* approach to other related results]. Later Bitmead and Anderson [3] and Morf [20] used

another approach based on the displacement-rank properties of matrix Schur complements, to obtain similar results; while this approach allows for generalization to non-Toeplitz matrices, the hidden coefficient in their proposed $O(n\log^2 n)$ constructions turned out to be extremely large (see Sexton *et al* [24]). Later Musicus [21], de Hoog [10], Ammar and Gragg [2] used a more direct approach based on a combination of the Schur and Levinson algorithms to obtain better coefficients; in particular, Ammar and Gragg made a detailed study and claimed an operation count of $8n\log^2 n$ flops. With this count, the new (called superfast in [2]) method for solving (exactly determined) Toeplitz systems is faster than the one based on the Levinson algorithm whenever $n > 256$. We should mention here that Schur-algorithm-based methods are natural in the context of transmission-line and layered-earth models, so it is not a surprise that similar techniques were also conceived in those fields - see Choate [7], McClary [19] and Bruckstein and Kailath [6]. A good source for background on the Levinson and Schur algorithms, transmission line models, displacement representations as mentioned and used in the present chapter may be [13].

The method we have taken in this chapter is in the spirit of the generalized Schur algorithm [8]. Our algorithm can be applied to non-Toeplitz matrices, and does not have the drawback of the large coefficient in the methods of Bitmead and Anderson [3] or Morf [20]. Furthermore, we can readily handle matrices such as $(T^T T)^{-1}$ and $(T^T T)^{-1}T^T$, where T may be a near-Toeplitz matrix or a rectangular block-Toeplitz matrix, or a Toeplitz-block matrix; in particular, therefore, we can also obtain the *least-squares* solutions of over-determined Toeplitz and near-Toeplitz systems with $O(n\log^2 n)$ flops. Our algorithm is closely related to the algorithm of Musicus [21]. However, our presentation is conceptually much simpler (especially for the non-Toeplitz cases treated in [21]) than previous approaches; in particular, we do not use the relationship between the Schur algorithm and Levinson algorithms needed in [2], [10], [21].

An outline of our approach is the following. For a matrix E,

$$E = \begin{bmatrix} E_{1,1} & E_{1,2} \\ E_{2,1} & E_{2,2} \end{bmatrix}, \qquad E_{1,1}, \text{ nonsingular,}$$

the Schur complement of $E_{1,1}$ in E is

$$S \equiv E_{1,1} - E_{2,1}E_{1,1}^{-1}E_{1,2}.$$

Notice that matrices such as

$$S_1 \equiv T^{-1}, \qquad S_2 \equiv (T^T T)^{-1}, \qquad S_3 \equiv (T^T T)^{-1}T^T \tag{2}$$

can be identified as the Schur complements of the following *extended matrices*,

$$E_1 = \begin{bmatrix} T & I \\ -I & O \end{bmatrix}, \quad E_2 = \begin{bmatrix} T^T T & I \\ -I & O \end{bmatrix}, \quad E_3 = \begin{bmatrix} T^T T & T^T \\ -I & O \end{bmatrix}. \tag{3}$$

Now the matrices E in (3) have the following (generalized) *displacement representation*, for suitably chosen matrices $\{F^f, F^b\}$,

$$E = \sum_{i=1}^{\alpha} K(\mathbf{x}_i, F^f)K^T(\mathbf{y}_i, F^b),$$

where $K(\mathbf{x}_i, F^f)$ and $K(\mathbf{y}_i, F^b)$ are lower triangular matrices whose j columns are $(F^f)^{(j-1)}\mathbf{x}_i$ and $(F^b)^{(j-1)}\mathbf{y}_i$, respectively. The smallest possible number α is called the *displacement rank* of E with respect to $\{F^f, F^b\}$. For an example, let T be an $m \times n$ scalar Toeplitz matrix, with $m \geq n$. Then the matrix E_2 has displacement rank 4 with respect to $\{F, F\}$, where $F = \begin{bmatrix} Z_n & O \\ O & Z_n \end{bmatrix}$, and has a displacement representation [14],

$$E_2 = \sum_{i=1}^{2} K(\mathbf{y}_i, F)K^T(\mathbf{x}_i, F) - \sum_{i=3}^{4} K(\mathbf{y}_i, F)K^T(\mathbf{x}_i, F), \quad \mathbf{y}_i \equiv \begin{bmatrix} I_n & O \\ O & -I_n \end{bmatrix}\mathbf{x}_i. \tag{4a}$$

If we define $\mathbf{x}_i^T \equiv [\mathbf{w}_i^T, \mathbf{v}_i^T]$, note that the matrix $K(\mathbf{x}_i, F)$ in (4a) has the form

$$\begin{bmatrix} L(\mathbf{w}_i) & O \\ L(\mathbf{v}_i) & O \end{bmatrix} \in \mathbf{R}^{2n \times 2n}, \quad O \in \mathbf{R}^{n \times n}, \tag{4b}$$

where $L(\mathbf{w}_i)$ and $L(\mathbf{v}_i)$ are lower triangular Toeplitz matrices with first columns $\mathbf{w}_i$ and $\mathbf{v}_i$.

Given a displacement representation of E, we use a certain *generalized Schur algorithm* (see Sec 2) to successively compute displacement representations of the Schur complements of

all the leading principal submatrices in E. For the above example, n steps of the generalized Schur algorithm will yield

$$\begin{bmatrix} O & O \\ O & (T^T T)^{-1} \end{bmatrix} = \sum_{i=1}^{2} K(\mathbf{u}_i, F) K^T(\mathbf{u}_i, F) - \sum_{i=3}^{4} K(\mathbf{u}_i, F) K^T(\mathbf{u}_i, F),$$

where the top n elements of $\mathbf{u}_i$ are zero. Therefore, if we denote the bottom n elements of $\mathbf{u}_i$ as $\mathbf{u}_{2,i}$, we can have the displacement representation

$$(T^T T)^{-1} = \sum_{i=1}^{2} L(\mathbf{u}_{2,i}) L^T(\mathbf{u}_{2,i}) - \sum_{i=3}^{4} L(\mathbf{u}_{2,i}) L^T(\mathbf{u}_{2,i}).$$

Now, the generalized Schur algorithm, which is a two-term polynomial recursion, can be implemented in a divide-and-conquer fashion with $O(\alpha^3 f(n) \log n)$ flops, where $f(n)$ denotes the number of operations for the multiplication of two polynomials. Therefore, if the multiplication of two polynomials is done again by divide-and-conquer, i.e., by using fast convolution algorithms, then the overall computation requires $O(\alpha^3 n \log^2 n)$ flops. We remark that the factor α^3 can be reduced to α if several convolutions can be performed in parallel. Once we have a displacement representation of the desired Schur complement S, the matrix-vector multiplication, $S\mathbf{b}$, can be done with $O(\alpha n \log n)$ flops using fast convolutions. As an example, we can obtain the least squares solution for the Toeplitz system,

$$T\mathbf{x} = \mathbf{b}, \quad T \in \mathbf{R}^{m \times n}, \quad m \geq n$$

as follows:

(i) Form $T^T\mathbf{b}$ using 2 fast convolutions,

(ii) Obtain a displacement representation of $(T^T T)^{-1}$ using the divide-and-conquer version of the generalized Schur algorithm,

(iii) Form $(T^T T)^{-1}(T^T\mathbf{b})$ using 8 fast convolutions.

If we had obtained the displacement representation of $(T^T T)^{-1} T^T$ directly (using E_3), then step (i) above would not be needed.

2. Generalized Schur Algorithm.

After a brief review of basic concepts and definitions, we shall describe the generalized Schur algorithm of references [8] and [14], but in a polynomial form important for the divide-and-conquer implementations. We shall need to recall some definitions and basic properties.

Generators of Matrices.

Let F^f and F^b be nilpotent matrices. The matrix

$$\nabla_{(F^f, F^b)} A \equiv A - F^f A F^{bT}$$

is called the *displacement* of A with respect to the *displacement operators* $\{F^f, F^b\}$. Define the (F^f, F^b)-displacement rank of A as $\mathrm{rank}[\nabla_{(F^f, F^b)} A]$. Any matrix pair $\{X, Y\}$ such that

$$\nabla_{(F^f, F^b)} A = XY^T, \quad X \equiv [\, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_\alpha \,], \quad Y \equiv [\, \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_\alpha \,] \tag{5}$$

is called a *(vector form) generator* of A with respect to $\{F^f, F^b\}$. The generator will be said to have *length* α. If the length α is equal to the displacement rank of A, we say that the generator is *minimal*. A generator such as $Y = X\Sigma$, where Σ is a diagonal matrix with 1 or -1 along the diagonal, is called a *symmetric generator*.

The following Lemma [14], [15] establishes the connection between generators and displacement representations.

Lemma. Let E be an $m \times n$ matrix. If F^f and F^b are nilpotent, then the equation $\nabla_{(F^f, F^b)} E = \sum_1^\alpha \mathbf{x}_i \mathbf{y}_i^T$ has the unique solution $E = \sum_1^\alpha K(\mathbf{x}_i, F^f) K^T(\mathbf{y}_i, F^b)$, where $K(\mathbf{x}_i, F^f) \equiv [\mathbf{x}_i, F^f \mathbf{x}_i, \cdots, F^{f(n-1)} \mathbf{x}_i]$ and $K(\mathbf{y}_i, F^b) \equiv [\mathbf{y}_i, F^b \mathbf{y}_i, \cdots, F^{b(n-1)} \mathbf{y}_i]$.

Choice of Displacement Operators.

The generalized Schur algorithm operates with generators, and needs $O(\alpha mn)$ flops for sequential implementation and $O(\alpha^3 n \log^2 n)$ for divide-and-conquer implementation. Therefore, for a given matrix A, we should try to choose the displacement operators that give the smallest

α. If the matrix A is an $n \times n$ Toeplitz matrix, the appropriate displacement operator F is Z_n, an $n \times n$ shift matrix. If A has some near-Toeplitz structure, then F would have forms such as

$$F = Z_n \oplus Z_m, \quad F = \bigoplus_{i=1}^{n} Z_{n_i}, \quad F = Z_n^{\beta},$$

where $\oplus$ denotes the *direct sum*, $Z_n \oplus Z_m \equiv \begin{bmatrix} Z_n & O \\ O & Z_m \end{bmatrix}$, and $\bigoplus_{i=1}^{n}$ denotes the concatenated direct sum.

Example 1. Let $T = (t_{i-j})$ be an $m \times n$ pre- and post-windowed scalar Toeplitz matrix, i.e., $t_{ij} = 0$ if $j > i$ or $i > m - n + j$ with $m > n$. Then it is easy to check that the matrix $C = (c_{i-j}) \equiv T^T T$ is also a (unwindowed) Toeplitz matrix, and with respect to $\{Z_n \oplus Z_n, Z_n \oplus Z_m\}$, E_3 in (3) has a generator $\{X, Y\}$ of length 2, where

$$\begin{aligned}
\mathbf{x}_1 &= [c_0, c_1, \cdots, c_n, -1, 0, \cdots, 0]^T / c_0^{1/2}, \\
\mathbf{x}_2 &= [0, c_1, \cdots, c_n, -1, 0, \cdots, 0]^T / c_0^{1/2}, \\
\mathbf{y}_1 &= [c_0, c_1, \cdots, c_n, t_0, t_1, \cdots t_{m-n}, 0, \cdots 0]^T / c_0^{1/2}, \\
\mathbf{y}_2 &= -[0, c_1, \cdots, c_n, t_0, t_1, \cdots t_{m-n}, 0, \cdots 0]^T / c_0^{1/2}.
\end{aligned}$$

□

Example 2. If T is a Toeplitz-block matrix, i.e.,

$$T = \begin{bmatrix} T_{1,1} & T_{1,2} & \cdot & T_{1,N} \\ T_{2,1} & T_{2,2} & \cdot & T_{2,N} \\ \cdot & \cdot & \cdot & \cdot \\ T_{M,1} & T_{M,2} & \cdot & T_{M,N} \end{bmatrix} \in \mathbf{R}^{m \times n}, \quad T_{ij} = \text{scalar } m_i \times n_j \text{ Toeplitz matrix}, \tag{6}$$

then for the matrices E in (3), we choose [8], [14] the following displacement operators

$$E_1: \quad F^f = [\bigoplus_{i=1}^{M} Z_{m_i}] \oplus F_1, \quad F^b = [\bigoplus_{i=1}^{N} Z_{n_i}] \oplus F_1, \quad m = n, \tag{7a}$$

$$E_2: \quad F^f = [\bigoplus_{i=1}^{N} Z_{n_i}] \oplus F_1, \quad F^b = [\bigoplus_{i=1}^{N} Z_{n_i}] \oplus F_1, \quad m = n, \tag{7b}$$

$$E_3: \quad F^f = [\bigoplus_{i=1}^{N} Z_{n_i}] \oplus F_1, \quad F^b = [\bigoplus_{i=1}^{N} Z_{n_i}] \oplus [\bigoplus_{i=1}^{M} Z_{m_i}], \tag{7c}$$

where F_1 can be either Z_n or $\bigoplus_{i=1}^{N} Z_{n_i}$. However, for the divide-and-conquer implementation, we

prefer to choose $\bigoplus_{i=1}^{N} Z_{n_i}$; see the Remark in Sec 4.

Example 3. On the other hand, if the matrix T in (3) is a block-Toeplitz matrix with $\beta \times \beta$ blocks,

$$T = \begin{bmatrix} B_0 & B_{-1} & \cdot & B_{-N+1} \\ B_1 & B_0 & \cdot & B_{-N+2} \\ \cdot & \cdot & \cdot & \cdot \\ B_{M-1} & B_{M-2} & \cdot & B_{-N+M} \end{bmatrix} \in \mathbf{R}^{m \times n}, \quad B_k \in \mathbf{R}^{\beta \times \beta}, \quad m \equiv M\beta, \quad n \equiv N\beta. \tag{8}$$

then for the extended matrices E, we should choose [8] the displacement operators

$$F^f = Z_n^\beta \oplus Z_n^\beta, \qquad F^b = Z_n^\beta \oplus Z_m^\beta, \tag{9}$$

where, for E_1 we assumed that T is a square $n \times n$ matrix.

Generators of the above and other extended block-Toeplitz or Toeplitz-block matrices can be found in [8] and [14].

Polynomial Form of Generators.

In general, the displacement operators F^f and F^b for both extended block-Toeplitz matrices and extended Toeplitz-block matrices have the form,

$$F = \bigoplus_{i=1}^{N} Z_{n_i}^\beta, \quad n \equiv \sum_{i=1}^{N} n_i. \tag{10}$$

We shall say that the displacement operator F in (10) has N *sections*. One of the key operations in generalized Schur algorithms is matrix-vector multiplication, $F\mathbf{v}$, i.e, a *sectioned shift* operation. With the polynomial representation of vectors, the shift operation has a nice algebraic expression. For a given vector $\mathbf{v}$, let $v(z)$ denote the polynomial whose coefficient for the term z^i is the i+1st component of the vector, i.e.,

$$\mathbf{v} = [v_0, v_1, v_2, \cdots, v_{n-1}]^T \quad \rightarrow \quad v(z) = v_0 + v_1 z + v_2 z^2 + \cdots v_{n-1} z^{n-1}. \tag{11}$$

Then,

$$Z_n \mathbf{v} \equiv \mathbf{v}' = [0, v_0, v_1, \cdots, v_{n-2}]^T \quad \rightarrow \quad v(z)z \mod z^n.$$

In general, for the matrix whose displacement operator is the F in (10), let us define integers $\{\delta_i\}$ by

$$\delta_i = \sum_{k=1}^{i} n_k, \quad \delta_1 < \delta_2 < \cdots < \delta_N.$$

Let $v(z)$ and $\theta(z)$ be polynomials of degree less than or equal to $n-1$, and define the degree at most (n_i-1) polynomial, $v_i(z)$, by

$$v(z) = v_1(z) + z^{\delta_1} v_2(z) + z^{\delta_2} v_3(z) + \cdots + z^{\delta_{N-1}} v_{n_N}(z). \tag{12a}$$

Given two polynomials $v(z)$ and $\theta(z)$, and the displacement operator F in (10), the *(polynomial form) displacement operator* $\otimes_F$ is defined by the following operation,

$$v(z) \otimes_F \theta(z) \equiv r(z) \equiv r_1(z) + z^{\delta_1} r_2(z) + z^{\delta_2} r_3(z) + \cdots + z^{\delta_{N-1}} r_{n_N}(z), \tag{12b}$$

where

$$r_i(z) \equiv v_i(z)\theta(z^{\beta}) \mod z^{n_i}, \tag{12c}$$

i.e., $r_i(z)$ is the polynomial $v_i(z)\theta(z^{\beta})$ after chopping off the higher degree terms, so that $r_i(z)$ has the degree at most $(n_i - 1)$.

Let

$$X = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_\alpha], \quad Y = [\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_\alpha]$$

be a generator of a matrix A with respect to certain $\{F^f, F^b\}$, and let

$$\mathbf{x}_i \to x_i(z), \quad \mathbf{y}_i \to y_i(w).$$

Then we call the pair of polynomial vectors, $\{X(z), Y(w)\}$, where

$$X(z) \equiv [\, x_1(z), x_2(z), \cdots, x_\alpha(z) \,], \quad Y(w) \equiv [\, y_1(w), y_2(w), \cdots, y_\alpha(w) \,],$$

a (polynomial form) generator of A, with respect to *(polynomial from) displacement operator* $\{\otimes_{F^f}, \otimes_{F^b}\}$.

Example 1 (Continued). The matrix E_3 in (3) has a generator $\{X(z), Y(w)\}$ with respect to $\{\otimes_{F^f}, \otimes_{F^b}\}$, where $F^f = Z_n \oplus Z_n$, $F^b = Z_n \oplus Z_m$, and

$$x_1(z) = [c_0 + c_1 z + \cdots + c_n z^n - z^{n+1}]c_0^{-1/2},$$
$$x_2(z) = [c_1 z + cz^2 + \cdots + c_n z^n - z^{n+1}]c_0^{-1/2},$$
$$y_1(w) = [c_0 + c_1 w + \cdots + c_n w^n + t_0 w^{n+1} + t_1 w^{n+2} + \cdots + t_{m-n} w^{m+1}]c_0^{-1/2},$$
$$y_2(w) = -[c_1 w + \cdots + c_n w^n + t_0 w^{n+1} + t_1 w^{n+2} + \cdots + t_{m-n} w^{m+1}]c_0^{-1/2}.$$

Also notice that

$$x_1(z) \otimes_{F'} z = [c_0 z + c_1 z^2 + \cdots + c_{n-1} z^n - z^{n+2}]c_0^{-1/2},$$
$$y_1(w) \otimes_{F^b} w =$$
$$= [c_0 w + c_1 w^2 + \cdots + c_{n-1} w^n + t_0 w^{n+2} + t_1 w^{n+3} + \cdots + t_{m-n-1} w^{m+1}]c_0^{-1/2}. \qquad \square$$

Next we note that for given vectors $\mathbf{a}$ and $\mathbf{b}$ such that $\mathbf{a}^T\mathbf{b} \neq 0$, we can always find [8] matrices Θ and Ψ such that

$$\mathbf{a}^T\Theta = [a_1', 0, 0, \cdots, 0], \quad \mathbf{b}^T\Psi = [b_1', 0, 0, \cdots, 0], \quad \Theta \cdot \Psi^T = I, \tag{13}$$

and therefore, $\mathbf{a}^T\mathbf{b} = a_1' b_1'$. We define polynomial matrices $\Theta(z)$ and $\Psi(w)$ by

$$\Theta(z) \equiv \Theta \begin{bmatrix} z & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}, \quad \Psi(w) \equiv \Psi \begin{bmatrix} w & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}. \tag{14}$$

We remark also that if $\mathbf{a} = \mathbf{b}$, then $\Psi(w) = \Theta(w)$, and if $\mathbf{b} = \Sigma\mathbf{a}$, where $\Sigma \equiv I_p \oplus -I_q$, then $\Psi(w) = \Theta(w)\Sigma$, so that we only need to find, and post-multiply by, $\Theta(z)$.

Generalized Schur Algorithm

Let a matrix E have a generator $\{X_0(z), Y_0(w)\}$ with respect to $\{\otimes_{F'}, \otimes_{F^b}\}$. and define E_{ij} by

$$E = \begin{bmatrix} E_{1,1} & E_{1,2} \\ E_{2,1} & E_{2,2} \end{bmatrix} \in \mathbf{R}^{m \times n},$$

where $E_{1,1}$ is a $k \times k$ strongly nonsingular matrix, i.e., the one with all nonsingular leading submatrices. The k-step generalized Schur algorithm [8], [14] presented below in polynomial form gives a generator of the matrix,

$$\begin{bmatrix} O & O \\ O & S \end{bmatrix}, \quad S \equiv E_{2,2} - E_{2,1}E_{1,1}^{-1}E_{1,2} \in \mathbf{R}^{(m-k)\times(n-k)},$$

with respect to $\{\otimes_{F^f}, \otimes_{F^b}\}$, or equivalently, a generator of S with respect to $\{\otimes_{\bar{F}^f}, \otimes_{\bar{F}^b}\}$, where $\bar{F}^f$ and $\bar{F}^b$ denote the trailing square submatrices of size $(m-k)$ and $(n-k)$ of F^f and F^b, respectively.

Algorithm (k-step Generalized Schur Algorithm)

Input: Generator of E, $\{X_0(z), Y_0(w)\}$; displacement operator $\{\otimes_{F^f}, \otimes_{F^b}\}$;

Number of steps k.

Output: Generator of S $\{X_k(z), Y_k(w)\}$

Procedure GeneralizedSchur

begin

for $i := 0$ **to** $k-1$ **do begin**

$\mathbf{a}^T := [z^{-i}X_i(z)]_{z=0}$;

$\mathbf{b}^T := [z^{-i}Y_i(z)]_{z=0}$;

Find $\Theta_i(z)$ and $\Psi_i(w)$ to transform $\mathbf{a}^T$ and $\mathbf{b}^T$ such as (13);

$X_{i+1}(z) = X_i(z)\otimes_{F_f}\Theta_i(z)$; $\quad Y_{i+1}(w) = Y_i(w)\otimes_{F_b}\Psi_i(w)$

end

return $\{X_k(z), Y_k(w)\}$

end

Remark. The polynomial vectors, $X_i(z)$ and $Y_i(w)$, have degrees $m-1$ and $n-1$ respectively, for all i. Each step eliminates the non-zero lowest degree term, and therefore the terms of $X_i(z)$ and $Y_i(w)$ whose degrees are less than z^i and w^i are zeros.

By applying the generalized Schur algorithm, one can obtain generators, or equivalently displacement representations, for various interesting Schur complements.

3. Divide-and-Conquer Implementation.

The (sequential) k-step generalized Schur algorithm in Sec 2 can also be implemented efficiently using divide-and-conquer approach. We shall only explain how to find $X_k(z)$; essentially the same argument applies for $Y_k(w)$.

Let us define $\Theta_{p:q}(z)$ and $X_{p:q}(z)$ by

$$\Theta_{p:q}(z) \equiv \Theta_p(z)\Theta_{p+1}(z)\cdots\Theta_q(z),$$
$$X_{p:q}(z) \equiv X_{0:q}(z)\otimes_{F'}\Theta_{0:p-1}(z), \quad X_{0:q}(z) \equiv X_0(z) \mod z^{q+1},$$

where $0 \le p \le q$. The polynomial matrix $\Theta_{p:q}(z)$ has a degree $q-p+1$. The polynomial vector $X_{p:q}(z)$ has degree q, and is obtained by dropping from $X_p(z)$ all terms of degree higher than z^q. Also note the useful properties,

$$[x(z)\otimes_F\theta_1(z)]\otimes_F\theta_2(z) = x(z)\otimes_F[\theta_1(z)\theta_2(z)],$$
$$[x_1(z)+x_2(z)]\otimes_F\theta(z) = [x_1(z)\otimes_F\theta(z)] + [x_2(z)\otimes_F\theta(z)].$$

These properties and the fact that $\Theta_{p:q}(z)$ is completely determined by $X_{p:q}(z)$ allow a divide-and-conquer implementation of the generalized Schur algorithm.

Given $X_{p:q}(z)$, we can compute $\Theta_{p:q}(z)$ as follows. If $p = q$, then we are successful, and compute $\Theta_{p:p}(z) = \Theta_p(z)$. Otherwise, we choose an "appropriate" (see Sec 4) *division point* r such that $p < r < q$, and try to solve the smaller sub-problem of finding $\Theta_{p:r-1}(z)$, given $X_{p:r-1}(z)$. Once we know $\Theta_{p:r-1}(z)$, we can compute $X_{r:q}(z)$ by

$$X_{r:q}(z) = X_{0:q}(z)\otimes_{F'}\Theta_{0:r-1}(z) = [X_{0:q}(z)\otimes_{F'}\Theta_{0:p-1}(z)]\otimes_{F'}\Theta_{p:r-1}(z) \tag{15a}$$
$$= X_{p:q}(z)\otimes_{F'}\Theta_{p:r-1}(z). \tag{15b}$$

Now we again try to find $\Theta_{r:q}(z)$ given $X_{r:q}(z)$. After we obtain $\Theta_{r:q}(z)$, we can combine the two results, $\Theta_{p:r-1}(z)$ and $\Theta_{r:q}(z)$, by multiplication,

$$\Theta_{p:q}(z) = \Theta_{p:r-1}(z)\Theta_{r:q}(z). \tag{16}$$

Programming details of the above *recursive generalized Schur algorithm* are shown in the Appendix.

The previous recursive description can be visualized nonrecursively using *trees* (see Fig 1 and 2). Each node in the tree is annotated with the *rules:* "find", "apply" and "combine",

$$f_{p:p} : \text{ Find } \Theta_{p:p}(z),$$
$$a_{p:q} : X_{r:q}(z) := X_{p:q}(z) \otimes_F \Theta_{p:r-1}(z),$$
$$c_{p:q} : \Theta_{p:q}(z) := \Theta_{p:r-1}(z)\Theta_{r:q}(z).$$

We traverse the tree in *post-order* (i.e., follow the order labeled on each node of the tree), and evaluate the rules.

Now, we shall consider two examples in detail.

Example 4. Pseudo-Inverse of pre and post windowed Toeplitz Matrices.

Consider the matrix E_3 in Example 1, where

$$T^T T = \begin{bmatrix} 16 & 8 & 4 & 1 \\ 8 & 16 & 8 & 4 \\ 4 & 8 & 16 & 8 \\ 1 & 4 & 8 & 16 \end{bmatrix}, \quad T^T = \begin{bmatrix} 3 & 2 & 1 & 1 & -1 & 0 & 0 & 0 \\ 0 & 3 & 2 & 1 & 1 & -1 & 0 & 0 \\ 0 & 0 & 3 & 2 & 1 & 1 & -1 & 0 \\ 0 & 0 & 0 & 3 & 2 & 1 & 1 & -1 \end{bmatrix}.$$

It is desired to find a displacement representation of $(T^T T)^{-1} T^T$. This can be done by the 4-step recursive generalized Schur algorithm. The input to the algorithm is a generator $\{X_0(z), Y_0(w)\}$ of

$$E_3 = \begin{bmatrix} T^T T & T^T \\ -I & O \end{bmatrix},$$

with respect to $\{\otimes_{F^f}, \otimes_{F^b}\}$, where $F^f = Z_n \oplus Z_n$, $F^b = Z_n \oplus Z_m$. The output, $\{X_4(z), Y_4(w)\}$ is a generator of $(T^T T)^{-1} T^T$, with respect to $\{\otimes_{Z_n}, \otimes_{Z_m}\}$. The computational sequence is illustrated in Fig 1, where it is assumed that the division points were chosen successively by 2, 1 and 3.

(1). $f_{0:0}$: $\Theta_{0:0}(z) = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} z & \\ & 1 \end{bmatrix}$ because $X_{0:0}(z) = [4, 0]$

(2). $a_{0:1}$: $X_{1:1}(z) = X_{0:1}(z) \otimes_{F^f} \Theta_{0:0}(z) = [4 + 2z, 2z] \otimes_{F^f} \Theta_{0:0}(z) = [4z, -2z]$

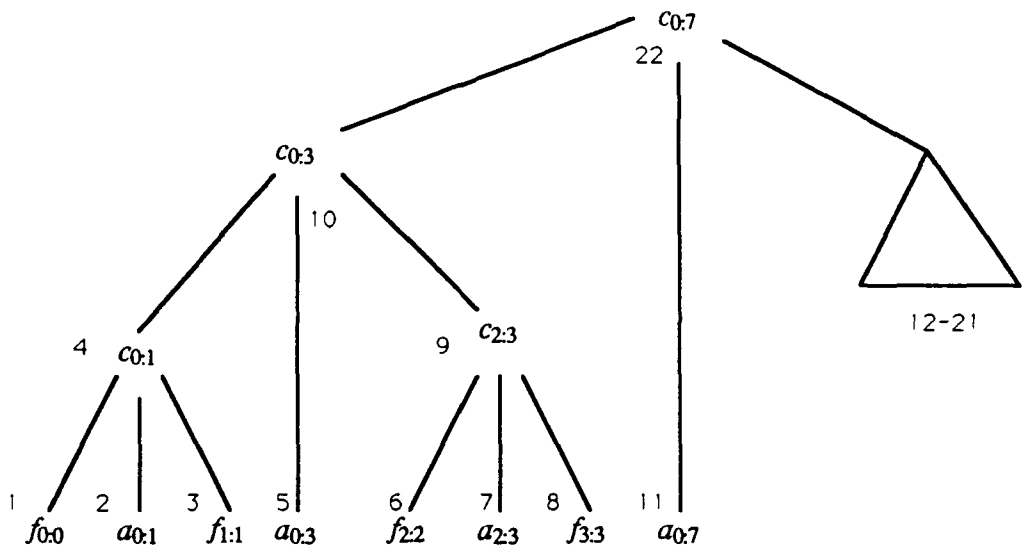


Fig 1. Sequence of Computations for Example 4.

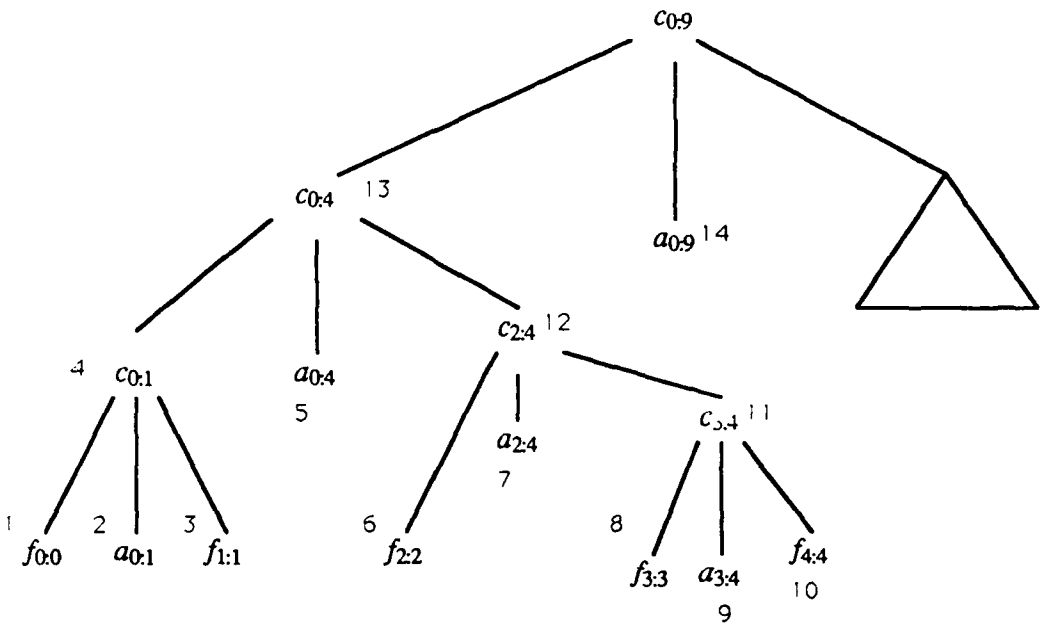


Fig 2. Sequence of Computations for Example 5.

(3). $f_{1:1}$: $\Theta_{1:1}(z) = \frac{2}{\sqrt{3}} \cdot \begin{bmatrix} 1 & +1/2 \\ -1/2 & -1 \end{bmatrix} \begin{bmatrix} z & \\ & 1 \end{bmatrix}$

(4). $c_{0:1}$: $\Theta_{0:1}(z) = \Theta_{0:0}(z)\Theta_{1:1}(z) = \frac{2}{\sqrt{3}} \cdot \begin{bmatrix} z^2 & -z/2 \\ -z/2 & 1 \end{bmatrix}$

(5). $a_{0:3}$: $X_{2:3}(z) = X_{0:3}(z) \otimes_{F/} \Theta_{0:1}(z) = \frac{2}{\sqrt{3}} \cdot [3z^2 + 3z^3/2, -z^3/4]$

(6). $f_{2:2}$: $\Theta_{2:2}(z) = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} z & \\ & 1 \end{bmatrix}$ because $X_{2:2}(z) = \frac{2}{\sqrt{3}} \cdot [3z^2, 0]$

(7). $a_{2:3}$: $X_{3:3}(z) = X_{2:3}(z) \otimes_{F/} \Theta_{2:2}(z) = \frac{2}{\sqrt{3}} \cdot [3z^3, z^3/4]$

(8). $f_{3:3}$: $\Theta_{3:3}(z) = \frac{12}{\sqrt{143}} \cdot \begin{bmatrix} 1 & 1/12 \\ -1/12 & -1 \end{bmatrix} \begin{bmatrix} z & \\ & 1 \end{bmatrix}$

(9). $c_{2:3}$: $\Theta_{2:3}(z) = \Theta_{2:2}(z)\Theta_{3:3}(z) = \frac{12}{\sqrt{143}} \cdot \begin{bmatrix} z & z/12 \\ 1/12 & 1 \end{bmatrix} \begin{bmatrix} z & \\ & 1 \end{bmatrix}$

(10). $c_{0:3}$: $\Theta_{0:3}(z) = \Theta_{0:1}(z)\Theta_{2:3}(z) = \frac{24}{\sqrt{3}\sqrt{143}} \cdot \begin{bmatrix} z^4 - z^2/24 & z^3/12 - z/12 \\ -z^3/12 + z/12 & -z^2/24 + 1 \end{bmatrix}$

(11). $a_{0:7}$: $X_{4:7}(z) = [4+2z+z^2+z^3/4-z^4/4,\ 2z+z^2+z^3/4-z^4/4] \otimes_{F/} \Theta_{0:3}(z)$

$$= [(4+2z+z^2+z^3/4,\ 2z+z^2+z^3/4) - z^4(1/4,\ 1/4)] \otimes_{F/} \Theta_{0:3}(z)$$

$$= -z^4[(1/4,\ 1/4)\Theta_{0:3}(z) \bmod z^4]$$

$$= -\frac{6z^4}{\sqrt{3}\sqrt{143}}[z/12 - z^2/24 - z^3/2,\ 1 - z/2 - z^2/24 + z^3/12]$$

Because $T^T T$ is symmetric, $\Psi_{0:3}(w) = \Theta_{0:3}(w)\Sigma$, where $\Sigma = 1 \oplus -1$, and therefore,

$$Y_{4:13}(w) = [(4+2z+z^2+z^3/4) + z^4(3/4+z/2+z^2/4-z^4/4),$$
$$(2z+z^2+z^3/4) + z^4(3/4+z/2+z^2/4+z^3/4-z^4/4)] \otimes_{F^b} \Theta_{0:3}(w)\Sigma$$
$$= \frac{z^4 6}{\sqrt{3}\sqrt{143}}[1/4z + z^2/24 - 3z^3/2 + 49z^4/24 + 11z^5/8 + 13z^6/24 + 3z^7/2,$$
$$-3 - z/2 + z^2/8 - 2z^3/3 + 11z^4/8 - 13/24z^5 - z^6/8 - z^7/12].$$

Therefore,

$$(T^T T)^{-1} T^T = \gamma^2[L(\mathbf{x}_1)L^T(\mathbf{y}_1) + L(\mathbf{x}_2)L^T(\mathbf{y}_2)], \quad \gamma = \frac{6}{\sqrt{3}\sqrt{143}},$$

where $L(\mathbf{x}_i)$ and $L(\mathbf{y}_i)$ are the lower triangular Toeplitz matrices whose first columns are $\mathbf{x}_i$ and $\mathbf{y}_i$, respectively, and

$$\mathbf{x}_1 = [0, -1/12, 1/24, 1/2]^T,$$

$$\mathbf{x}_2 = [-1,\ 1/2,\ 1/24,\ -1/12]^T,$$
$$\mathbf{y}_1 = [0,\ 1/4,\ 1/24,\ -3/2,\ 49/24,\ 11/8,\ 13/24,\ 3/2]^T,$$
$$\mathbf{y}_2 = [-3,\ -1/2,\ 1/8,\ -2/3,\ 11/8,\ -13/24,\ -1/8,\ 1/12]^T. \qquad \square$$

Remark 1. For a symmetric generator of length 2 with $\beta = 1$, the 2×2 polynomial matrix $\Theta(z)$ in (14) can have the form (hyperbolic reflection)

$$\Theta_i(z) = \begin{bmatrix} ch_i z & sh_i \\ -sh_i z & -ch_i \end{bmatrix}, \qquad ch_i^2 - sh_i^2 = 1.$$

Let

$$\Theta_{p:q}(z) \equiv \Theta_p(z)\Theta_{p+1}(z) \cdots \Theta_q(z) \equiv \begin{bmatrix} \Theta_{1,1}(z) & \Theta_{1,2}(z) \\ \Theta_{2,1}(z) & \Theta_{2,2}(z) \end{bmatrix}.$$

Then, by induction, one can easily prove that

$$z^{q-p+1}\Theta_{1,1}(z^{-1}) = (-1)^{q-p+1}\Theta_{2,2}(z), \quad z^{q-p+1}\Theta_{1,2}(z^{-1}) = (-1)^{q-p+1}\Theta_{2,1}(z).$$

Therefore, we need to compute and store only two entries of $\Theta_{p:q}(z)$.

Remark 2. For an unwindowed scalar Toeplitz matrix, the matrix E_2 in (3) has displacement rank 4, whereas the matrix E_3 has displacement rank 5. Therefore, when we solve Toeplitz least squares problems, it is more efficient to find a displacement representation of $(T^T T)^{-1}$ rather than of $(T^T T)^{-1} T^T$. With the notation in (4), the matrix E_2 for an unwindowed scalar Toeplitz matrix $T = (t_{i-j}) \in \mathbf{R}^{m \times n}$ $(m \geq n)$ has a generator [14],

$$\mathbf{w}_1 = T^T \mathbf{t}_1 / \|\mathbf{t}_1\|, \quad \mathbf{w}_2 = \mathbf{t}_2, \quad \mathbf{w}_3 = Z_n Z_n^T \mathbf{w}_1, \quad \mathbf{w}_4 = Z_n \mathbf{l},$$
$$\mathbf{t}_1 \equiv [t_0, t_1, \cdots, t_{m-1}]^T, \quad \mathbf{t}_2 \equiv [0, t_{-1}, \cdots, t_{1-n}]^T, \quad \mathbf{l} \equiv [t_{m-1}, \cdots, t_{m-n}]^T,$$
$$\mathbf{v}_1 = \mathbf{v}_3 = \mathbf{e}_1 / \|\mathbf{t}_1\|, \quad \mathbf{v}_2 = \mathbf{v}_4 = 0,$$

where $\|\cdot\|$ denotes the Euclidean norm, and $\mathbf{e}_1$ is the vector with 1 in the first position, and zeros elsewhere.

Example 5. Displacement Representation for the inverse of a Sylvester Matrix.

Let T denote the following Sylvester matrix,

$$T \equiv \begin{bmatrix} 2 & 0 & 0 & 1 & 0 \\ 1 & 2 & 0 & 2 & 1 \\ 3 & 1 & 2 & 1 & 2 \\ 0 & 3 & 1 & 1 & 1 \\ 0 & 0 & 3 & 0 & 1 \end{bmatrix} \tag{17}$$

and suppose that it is desired to obtain a displacement representation of T^{-1}. Then the appropriate extended matrix is

$$E_1 = \begin{bmatrix} T & I \\ -I & O \end{bmatrix}, \tag{18}$$

and it is easy to see that the following $\{X_0(z), Y_0(w)\}$ is a generator of E_1 with respect to $\{\otimes_{F^f}, \otimes_{F^b}\}$, where $F^f = Z_5 \oplus Z_5$, $F^b = Z_3 \oplus Z_2 \oplus Z_5$;

$$X_0(z) \equiv [x_1(z), x_2(z), x_3(z)], \quad Y_0(w) \equiv [y_1(w), y_2(w), y_3(w)]$$

$$x_1(z) = 2 + z + 3z^2 - z^5, \quad x_2(z) = 1 + 2z + z^2 + z^3 - z^8, \quad x_3(z) = 1, \tag{19a}$$

$$y_1(w) = 1, \quad y_2(w) = w^3, \quad y_3(w) = w^5 \tag{19b}$$

Now the 5-step recursive generalized Schur algorithm gives a desired generator of T^{-1}, with respect to $\{Z_5, Z_5\}$, and a possible computational sequence is shown in Fig 2, where the division points are chosen successively as 2, 1, 3 and 4.

(1). $f_{0:0}$: $\Theta_{0:0}(z) = \begin{bmatrix} z & -1/2 & -1/2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, $\Psi_{0:0}(w) = \begin{bmatrix} w & 0 & 0 \\ w/2 & 1 & 0 \\ w/2 & 0 & 1 \end{bmatrix}$

(2). $a_{0:1}$: $X_{1:1}(z) = [2z, 3z/2, -z/2]$, $Y_{1:1}(w) = [w, 0, 0]$

(3). $f_{1:1}$: $\Theta_{1:1}(z) = \begin{bmatrix} z & -3/4 & -1/4 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, $\Psi_{1:1}(w) = \begin{bmatrix} w & 0 & 0 \\ 3w/4 & 1 & 0 \\ -w/4 & 0 & 1 \end{bmatrix}$

(4). $c_{0:1}$: $\Theta_{0:1}(z) = \begin{bmatrix} z^2 & -3z/4-1/2 & z/4-1/2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, $\Psi_{0:1}(w) = \begin{bmatrix} w^2 & 0 & 0 \\ w^2/2+3w/4 & 1 & 0 \\ w^2/2-w/4 & 0 & 1 \end{bmatrix}$

(5). $a_{0:4}$: $X_{2:4}(z) = [2z^2+z^3+3z^4, -5z^2/4-5z^3/4, -5z^2/4+3z^3/4]$

$$\begin{aligned} Y_{2:4}(w) &= Y_{0:4}(w) \otimes_{F^b} \Psi_{0:1}(w) \\ &= [(1, 0, 0)\Psi_{0:1}(w) \bmod w^3] + w^3[(0, w, 0)\Psi_{0:1}(w) \bmod w^2] \\ &= [w^2+3w^4/4, w^3, 0] \end{aligned}$$

(6). $f_{2:2}$: $\Theta_{2:2}(z) = \begin{bmatrix} z & 5/8 & 5/8 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ $\Psi_{2:2}(w) = \begin{bmatrix} w & 0 & 0 \\ -5w/8 & 1 & 0 \\ -5w/8 & 0 & 1 \end{bmatrix}$

(7). $a_{2:4}$: $X_{3:4}(z) = [2z^3+z^4, -5z^3/8+15z^4/8, 11z^3/8+15z^4/8]$,

$$Y_{3:4}(w) = Y_{2:4}(w) \otimes_{F^b} \Psi_{2:2}(w)$$
$$= [(w^2, 0, 0)\Psi_{2:2}(w) \mod w^3] + w^3[(3w/4, 1, 0)\Psi_{2:2}(w) \mod w^2]$$
$$= [-5w^4/8, w^3, 0]$$

(8). $f_{3:3}$: $\Theta_{3:3}(z) = \begin{bmatrix} 0 & 1 & 0 \\ z & 16/5 & 11/5 \\ 0 & 0 & 1 \end{bmatrix}$, $\Psi_{3:3}(w) = \begin{bmatrix} -16w/5 & 1 & 0 \\ w & 0 & 0 \\ -11w/5 & 0 & 1 \end{bmatrix}$

(9). $a_{3:4}$: $X_{4:4}(z) = [-5z^4/8, 7z^4, 6z^4]$, $Y_{4:4}(w) = [w^4, -5w^4/8, 0]$

(10). $c_{4:4}$: $\Theta_{4:4}(z) = \begin{bmatrix} z/(2\sqrt{2}) & 28/(5\sqrt{2}) & 6/5 \\ -5z/(16\sqrt{2}) & 1/(2\sqrt{2}) & -3/4 \\ 0 & 0 & 1 \end{bmatrix}$ $\Psi_{4:4}(w) = \begin{bmatrix} w/(2\sqrt{2}) & 5/(16\sqrt{2}) & 0 \\ -28w/(5\sqrt{2}) & 1/(2\sqrt{2}) & 0 \\ -12\sqrt{2}w/5 & 0 & 1 \end{bmatrix}$

After evaluating, $c_{3:4}$, $c_{2:4}$ and $c_{0:4}$, we obtain $\Theta_{0:4}(z)$ and $\Psi_{0:4}(w)$, and finally

(14). $a_{0:9}$: $X_{0:9}(z) = [x_1(z), x_2(z), x_3(z)] \otimes_{F^l} \Theta_{0:4}(z)$

$$= z^5[(-1, -z^3, 0) \otimes_{F^l} \Theta_{0:4}(z)]$$
$$= z^5[(-1, -z^3, 0)\Theta_{0:4}(z) \mod z^5] = z^5[u_1(z), u_2(z), u_3(z)],$$

where

$$u_1(z) = -z/(2\sqrt{2}) - z^2/(2\sqrt{2}) + z^3/\sqrt{2} + z^4/\sqrt{2}$$
$$u_2(z) = 4/(5\sqrt{2}) + 4z/\sqrt{2} + 16z^2/(5\sqrt{2}) - 28z^3/(5\sqrt{2}) - 28z^4/(5\sqrt{2})$$
$$u_3(z) = 2/5 + z/5 + 2z^2/5 + z^3/5 - 6z^4/5.$$

$$Y_{0:9}(w) = [y_1(w), y_2(w), y_3(w)] \otimes_{F^b} \Psi_{0:4}(w)$$
$$= w^5[(0, 0, 1) \otimes_{F^b} \Psi_{0:4}(w)] = w^5[v_1(w), v_2(w), v_3(w)],$$

where

$$v_1(w) = -12\sqrt{2}w/5 + 12w^2/(5\sqrt{2}) + 12w^3/(5\sqrt{2}) - 12w^4/(5\sqrt{2}),$$
$$v_2(w) = -w/\sqrt{2} + w^2/(2\sqrt{2}) + w^3/(2\sqrt{2}) - w^4/(2\sqrt{2}),$$
$$v_3(w) = 1.$$

Therefore,

$$T^{-1} = L(\mathbf{u}_1)L^T(\mathbf{v}_1) + L(\mathbf{u}_2)L^T(\mathbf{v}_2) + L(\mathbf{u}_3)L^T(\mathbf{v}_3),$$

where $\mathbf{u}_i$ and $\mathbf{v}_i$ are the vectors whose jth component is the coefficient of z^{j-1} and w^{j-1} of $u_i(z)$ and $v_i(w)$, respectively.

□

Remark 1. If we had chosen the displacement operator $F^f = Z_5 \oplus Z_3 \oplus Z_2$, $F^b = Z_3 \oplus Z_2 \oplus Z_5$ for the matrix T in (17) we would have the same generator (19) for E_1, but the obtained generator of T^{-1} would be the one with respect to $\{Z_3 \oplus Z_2, Z_5\}$ rather than with respect to $\{Z_5, Z_5\}$. The displacement ranks of T^{-1} with respect to both displacement operators are 2, but the above procedure gives non-minimal generators of length 3.

Remark 2, The following extended matrix

$$\begin{bmatrix} T & \mathbf{b} \\ -I & \mathbf{0} \end{bmatrix}, \quad T = \text{Sylvester matrix} \tag{20}$$

also has a displacement rank of 3. One could as well obtain the solution $T^{-1}\mathbf{b}$ directly by applying the recursive generalized Schur algorithm to (20); the last column of X, where $\{X, \mathbf{y}\}$ is the computed generator of $T^{-1}\mathbf{b}$ with respect to $\{Z_n, 1\}$, can be shown to be the solution $T^{-1}\mathbf{b}$.

4. Polynomial Products with Fast Convolutions.

The product of two polynomials of degree d_1 and d_2 can be performed efficiently using $d \equiv d_1+d_2+1$ point fast cyclic convolution algorithms [4]. Among others, fast Fourier transformations (FFT's) can be used for convolutions, and Ammar and Gragg [2] carefully examined the use of FFT's for a doubling algorithm for square Toeplitz systems of equations. We shall only consider the subtle complications that arise in the recursive generalized Schur algorithm in this chapter.

The polynomial matrix-matrix product of (16) needs α^3 of $q-p$ point cyclic convolutions. The polynomial vector-matrix product of (15b) has α^2 of scalar polynomial products of the form, $x(z) \otimes_{F^f} \theta(z)$, where $x(z)$ is a polynomial with nonzero terms of $z^p, z^{p+1}, \cdots, z^q$. Let us assume that

$$0 < \delta_1 < \cdots < \delta_l \le p < \delta_{l+1} < \cdots < \delta_s \le r < \delta_{s+1} < \cdots < \delta_t \le q < \delta_{t+1} < \cdots < \delta_N.$$

Then

$$x'(z) \equiv x(z) \otimes_{F^f} \theta(z) \tag{21a}$$

$$= [z^{\delta_l} x_{l+1}(z) + z^{\delta_{l+1}} x_{l+2}(z) + \cdots + z^{\delta_s} x_{s+1}(z) + \cdots + z^{\delta_t} x_{t+1}(z)] \otimes_{F^f} \theta(z) \tag{21b}$$

$$= [z^{\delta_l} x_{l+1}(z) + \cdots + z^{\delta_{s-1}} x_s(z)] \otimes_{F^f} \theta(z) \tag{22a}$$

$$+ z^{\delta_s} [x_{s+1}(z)\theta(z^{\beta}) \mod z^{n_{s+1}}] \tag{22b}$$

$$+ z^{\delta_{s+1}} [x_{s+2}(z)\theta(z^{\beta}) \mod z^{n_{s+2}}] \tag{22c}$$

$$\cdots\cdot$$

$$+ z^{\delta_t} [x_{t+1}(z)\theta(z^{\beta}) \mod z^{n_{t+1}}]. \tag{22d}$$

The terms in (22a) do not need to be computed because these terms will be summed to zeros after adding all the partial sums in the vector-matrix multiplication of (15b). Recall that $x_i(z)$ has degree n_i, and $\theta(z^{\beta})$ has degree $\beta^{(q-p+1)}$. Therefore, the product $x_i(z)\theta(z^{\beta})$ from (22b) to (22d) can be performed by

$2n_i+1$	point cyclic convolutions	if degree$[\theta(z^{\beta})] \geq$ degree$[x_i(z)]$,
$n_i+\beta^{(q-p+1)}+1$	point cyclic convolutions	if degree$[\theta(z^{\beta})] <$ degree$[x_i(z)]$.

Remark. Notice that two d/2 point convolutions take $cd\log(d/2)$ flops if one d point convolution takes $cd\log d$ flops. Therefore, the polynomial product (21) is more efficient for the displacement operator F^f with more sections, because such displacement operators break a long convolution into many smaller convolutions. Therefore, for a given matrix we prefer to choose a displacement operator with as many sections as possible, while keeping the displacement rank minimal. Also we remark that the first and last terms (22b) and (22d) need smaller point convolutions.

If the dimensions of the matrix are powers of 2, then we can always choose the center division point, $r = \lceil (p+q)/2 \rceil$. This *balanced division* (or doubling) gives the least number of computations, in general. For this case, let $\eta \equiv p-q$, and $T(\eta)$ denote the number of computations for one recursion. Then

$$T(\eta) \le 2T(\eta/2) + W(\eta), \qquad W(\eta) \equiv O(\alpha^3 \eta \log\eta),$$

and therefore, one can show [1] that the k-step recursion takes

$$T(k) \le O(\alpha^3 k \log^2 k).$$

However, in most cases the doubling is not possible, and for such circumstances, the desirable choice of r is such that $r-p$ and $q-r+1$ are highly composite numbers (so that fast convolution algorithms can be applied efficiently), as well as r is close to $(q-p)/2$ (so as to achieve balancing).

Matrix-Vector Products using Displacement Representation.

The final step of finding solutions for linear equations is the matrix-vector multiplication $S\mathbf{b}$, given a displacement representation of $S \in \mathbf{R}^{m\times n}$,

$$S = \sum_{i=1}^{\alpha} K(\mathbf{x}_i, F^f) K^T(\mathbf{y}_i, F^b), \tag{23}$$

where the length α is a multiple of the block size β, $\alpha = \beta\delta$, say, and

$$F^f = \bigoplus_{i=1}^{M} Z_{m_i}^{\beta}, \quad F^b = \bigoplus_{i=1}^{N} Z_{n_i}^{\beta}, \quad m = \sum_{i=1}^{M} m_i, \quad n = \sum_{i=1}^{N} n_i.$$

The expression in (23) can be rewritten in the *block displacement form*

$$S = \sum_{i=1}^{\delta} K_\beta(X_i, F^f) K_\beta^T(Y_i, F^b), \quad X_i \in \mathbf{R}^{m\times\beta}, \quad Y_i \in \mathbf{R}^{n\times\beta}, \tag{24}$$

where

$$K_\beta(X_i, F^f) = [X_i, F^f X_i, F^{f2} X_i, \cdots F^{f[(m/\beta)-1]} X_i] \in \mathbf{R}^{m\times n} \tag{25a}$$

$$K_\beta(Y_i, F^b) = [Y_i, F^b Y_i, F^{b2} Y_i, \cdots F^{b[(n/\beta)-1]} Y_i] \in \mathbf{R}^{n\times n}. \tag{25b}$$

Furthermore, because F^f and F^b have M and N sections, respectively, (25a) and (25b) have the forms

$$K_\beta(X_i, F^f) = \begin{bmatrix} K_\beta(X_{1,i}, Z_{m_1}^{\beta}) & O \\ K_\beta(X_{2,i}, Z_{m_2}^{\beta}) & O \\ \cdot & \cdot \\ K_\beta(X_{M,i}, Z_{m_M}^{\beta}) & O \end{bmatrix}, \quad K_\beta(Y_i, F^b) = \begin{bmatrix} K_\beta(Y_{1,i}, Z_{n_1}^{\beta}) & O \\ K_\beta(Y_{2,i}, Z_{n_2}^{\beta}) & O \\ \cdot & \cdot \\ K_\beta(Y_{N,i}, Z_{n_N}^{\beta}) & O \end{bmatrix},$$

where $K_\beta(X, Z^\beta)$ is the block lower triangular Toeplitz matrix with the first column block X. The matrix O denotes a null matrix of appropriate size such that $K_\beta(X_i, F^f)$ and $K_\beta(Y_i, F^b)$ are $m \times n$ and $n \times n$ matrices, respectively.

To see how to use convolutions for the product,

$$K_\beta(X_i, F^f)K_\beta^T(Y_i, F^b)\mathbf{b},$$

it is enough to consider matrix-vector multiplications of the form $K_\beta(X, Z^\beta)\mathbf{b}$. Note that $K_\beta(X, Z^\beta)\mathbf{b}$ can be expressed as sum of β products of scalar lower triangular Toeplitz matrix and vectors. As an example,

$$\begin{bmatrix} a_0 & c_0 & & \\ a_1 & c_1 & & \\ a_2 & c_2 & a_0 & c_0 \\ a_3 & c_3 & a_1 & c_1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} a_0 & & & \\ a_1 & a_0 & & \\ a_2 & a_1 & a_0 & \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ 0 \\ b_2 \\ 0 \end{bmatrix} + \begin{bmatrix} c_0 & & & \\ c_1 & c_0 & & \\ c_2 & c_1 & c_0 & \\ c_3 & c_2 & c_1 & c_0 \end{bmatrix} \begin{bmatrix} b_1 \\ 0 \\ b_3 \\ 0 \end{bmatrix}. \tag{26}$$

The multiplications in the right sides of (26) can be done by fast convolutions, and therefore, so can the multiplication $S\mathbf{b}$.

5. Concluding Remarks.

We have presented $O(\alpha^3 n \log^2 n)$ algorithms for the determination of exact and least squares solutions of linear systems with matrices having (generalized) displacement rank α. Such algorithms for exact solutions have been studied by several authors, most recently by Ammar and Gragg [2] for Toeplitz systems. They also made a very close study of the implementation of the convolution operation in an attempt to obtain the smallest coefficient; we have not attempted so close an analysis for the more general algorithm in this chapter. Nor have we attempted a numerical error analysis of the algorithm; nevertheless one might hope that numerical refinements devised for the Schur algorithm (see e.g., Koltracht and Lancaster [17]) may be carried over to the divide-and-conquer framework as well.

APPENDIX

We shall summarize the explanation in Sec 3 using a Pascal-like recursive procedure. First, note that the polynomial $\Theta_{p:q}(z)$ (and $\Psi_{p:q}(z)$) has $q-p+2$ terms. The first column of $\Theta_{p:q}(z)$ has terms ranging from degree z to z^{q-p+1}, and the other columns have terms from 1 to z^{q-p}. Hence, by shifting the first column by one position, we can store $\Theta_{p:q}(z)$ and $\Psi_{p:q}(z)$ in the array "Poly" from p to q slots inclusive:

```
Poly: array [1..α, 1..α, 0..MAX-1] of record
    θ: coefficients;
    ψ: coefficients
end;
```

The computation of $\Theta_{p:q}(z)$ is sequential, i.e., once we compute $\Theta_{p:q}(z)$, we do not need to keep $\Theta_{p:r-1}(z)$, and therefore, the array "Poly" can be kept as a single global variable.

The polynomial vector $X_{p:q}(z)$ has $q-p+1$ terms, and therefore, can be stored in an array type GENERATORS:

```
type
    GENERATORS = array [1..α, 0..MAX-1] of record
        x: coefficient;
        y: coefficient
    end;
```

However, $X_{p:q}(z)$ cannot be kept as a global variable, and local copies should be maintained until we compute $X_{r:q}(z)$.

Now we can describe the recursive generalized Schur algorithm as follows.

Algorithm (Recursive k-step Generalized Schur Algorithm).

Input: Generator of E, $\{X_0(z), Y_0(w)\}$; displacement operator $\{\otimes_{F^f}, \otimes_{F^b}\}$;

Number of steps, k.

Output: Generator of S, $\{X_k(z), Y_k(w)\}$;

```
procedure RecursiveSchur
    var
        G, LowerG: GENERATORS;
    begin
        Find(0, k-1, G);
        Apply(0, k, n, G, LowerG);
        return (LowerG)
    end
```

The procedure Find(p, q, G) computes $\Theta_{p:q}(z)$, and $\Psi_{p:q}(w)$ given $\{X_{p:q}(z), Y_{p:q}(w)\}$, and the procedure Apply(p, r, q, G, LowerG) returns LowerG = $\{X_{r:q}(z), Y_{r:q}(w)\}$ given G = $\{X_{p:q}(z), Y_{p:q}(w)\}$

```
procedure Find(p, q: index; G: GENERATORS);
    var
        r : index;
        G, LowerG: GENERATORS;
    begin
        if p = q  then begin
            Compute Θ_{p:p}(z) and Ψ_{p:p}(w);
            return
        end
```

```
        r := appropriate integer close to ⌈(p+q)/2⌉;
        Find(p. r-1, G);
        Apply(p, r, q, G, LowerG);
        Find(r, q, LowerG);
        (* Use fast convolution for polynomial products *)
        Θ_{p:q}(z) := Θ_{p:r-1}(z)Θ_{r,q}(z);
        Ψ_{p:q}(w) := Ψ_{p:r-1}(w)Ψ_{r,q}(w)
    end

procedure Apply(p, r, q: index; G: GENERATORS; var LowerG: GENERATORS);
    begin
        (* Use fast convolution for polynomial products *)
        X_{r:q}(z) := X_{p:q}(z) ⊗_{F'} Θ_{p:r-1}(z);
        Y_{r:q}(w) := Y_{p:q}(w) ⊗_{F''} Ψ_{p:r-1}(w);
        LowerG := {X_{r:q}(z), Y_{r:q}(w)}
        Free the storage of {X_{p:q}(z), Y_{p:q}(w)};
        return (LowerG);
    end
```

REFERENCES

[1]. A. Aho, J. Hopcroft and J. Ullman, *The design and analysis of computer algorithms*, Addison-Wesley, Reading, MA, 1974. p. 305.

[2]. G. Ammar and W. Gragg, *Superfast solution of real positive definite Toeplitz systems*, SIAM J. Matrix Anal., Appl., Vol. 9, No. 1, Jan, (1988), pp. 61-76.

[3]. G. Bitmead and B. D. O. Anderson, *Asymptotically fast solution of Toeplitz and related systems of linear equations*, Linear Algebra and its Appli., 34 (1980), pp. 103-116.

[4]. R. Blahut, *Fast algorithms for digital signal processing*, Addison-Wesley, Reading, MA, 1985.

[5]. R. Brent, F. Gustavson and D. Yun *Fast Solution of Toeplitz Systems of Equations and Computation of Pade Approximants* Journal of Algorithms, 1, (1980), pp. 259-295

[6]. A. Bruckstein and T. Kailath, *Doing inverse scattering the fast(est) way*, Technical Report, Stanford University, CA 94305, July, 1985.

[7]. W. Choate, *A fast algorithm for normal incidence seismograms*, Geophysics, vol. 47, No. 2, Feb., (1982), pp. 196-202.

[8]. J. Chun and T. Kailath, *Unified approach for matrix factorization using generalized Schur algorithm*, Pre-print, Stanford University, CA 94305, 1988.

[9]. J. Chun and T. Kailath, *Fast Triangularization and Orthogonalization of Hankel and Vandermonde Matrices*, Pre-print, Stanford University, CA 94305, 1988.

[10]. F. De Hoog, *A new algorithm for solving Toeplitz systems of equations, Linear Algebra and its Appli., 88/89*, (1987), pp. 122-138.

[11]. I. Gohberg and I. Fel'dman, *Convolution equations and projection methods for their solutions*, Translations of Mathematical Monographs, vol. 41, Amer. Math. Soc., 1974.

[12]. I. Gohberg and A. Semencul, *On the inversion of finite Toeplitz matrices and their continuous analogs*, Mat. Issled., 2 (1972), pp. 201-233.

[13]. T. Kailath, *Signal processing applications of some moment problems*, Proceedings of Symposia in Applied Mathematics, vol. 37, 1987 pp. 71-109.

[14]. T. Kailath and J. Chun, *Generalized Gohberg-Semencul formulas for matrix inversion*, Proc. Symp. on Operator Theory and Applications, Calgary, Canada, Aug. 1988.

[15]. T. Kailath, S. Kung and M. Morf, *Displacement ranks of matrices and linear equations*, J. Math. Anal. Appl., 68 (1979) pp. 395-407. See also Bull. Amer. Math. Soc., 1 (1979), pp. 769-773.

[16]. T. Kailath, A. Vieira and M. Morf, *Inverses of Toeplitz operators, innovations, and orthogonal polynomial*, SIAM Review, vol. 20, No 1, Jan. (1978), pp. 106-119.

[17]. I. Koltracht and P. Lancaster, *Threshold algorithms for the prediction of reflection coefficients in a layered medium*, Geophysics, vol. 53, No. 7, Jul., (1988), pp. 908-919.

[18]. H. Lev-Ari and T. Kailath, *Triangular factorization of structured Hermitian matrices*, Operator Theory, Advances and Applications, vol. 18, Birkhauser, Boston, (1986), pp. 301-324.

[19]. W. McClary, *Fast seismic inversion*, Geophysics, vol. 48, No. 10, Oct., (1983), pp. 1371-1372.

[20]. M. Morf, *Doubling algorithms for Toeplitz and related equations*, Proceedings of the IEEE International Conf. on ASSP, Denver, (1980), pp. 954-959.

[21]. B. Musicus, *Levinson and fast Cholesky algorithms for Toeplitz and almost Toeplitz matrices*, Report, Research Lab. of Electronics, MIT, Cambridge, MA, 1981.

[22]. I. Schur, *Über Potenzreihen, die im Innern des Einheitskreises beschrankt sind*, J. für die Reine und Angewandte Mathematik, 147 (1917), pp. 205-232.

[23]. I. Schur, *On power series which are bounded in the interior of the unit circle. I.* (English translation of [22]), Operator Theory, Advances and Applications, vol. 18, Birkhauser, Boston, (1986), pp. 31-60.

[24]. H. Sexton, M. Shensa and J. Speiser, *Remarks on a displacement-rank inversion method for Toeplitz systems*, Linear Algebra and its Appli., 45, (1982), pp. 127-130.

Chapter 6.

Concluding Remarks.

After a summary of the main features of our approach to fast algorithms, we shall briefly note some areas for further investigation.

1. Summary of Results.

We introduced two different *displacements* of a matrix $A \in \mathbf{R}^{m \times n}$ defined as

$$\nabla_{(F^f, F^b)} A \equiv A - F^f A F^{bT}, \tag{1}$$

or

$$\Delta_{(F^f, F^b)} A \equiv F^f A - A F^{bT}, \tag{2}$$

where F^f and F^b are chosen matrices. We call the matrices $\nabla_{(F^f, F^b)}$ and $\Delta_{(F^f, F^b)}$ the *Toeplitz and Hankel displacements* of A with respect to the *displacement operators* $\{F^f, F^b\}$, respectively. We say that the matrix A is *structured* if $\nabla_{(F^f, F^b)} A$ has low rank (close-to-Toeplitz) or $\Delta_{(F^f, F^b)} A$ has low rank (close-to-Hankel), where "low" is with reference to the dimensions of A. The ranks of the displacements $\nabla_{(F^f, F^b)} A$ and $\Delta_{(F^f, F^b)} A$ are called the *displacement ranks* of the matrix A. The computational complexity (as well as the space complexity) of fast algorithms is proportional to the displacement rank of the matrix. Therefore, the displacement operators should be chosen so that the displacements have ranks as low as possible.

Invariance of Displacement rank under Schur complementations.

A basic property is the following. Let the block matrix

$$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

have Hankel (Toeplitz) displacement rank α with respect to lower triangular displacement operators, $\{F^f, F^b\}$. Then the matrix,

$$\begin{bmatrix} O & O \\ O & S \end{bmatrix}, \quad S \equiv D - CA^{-1}B, \quad \text{the Schur complement of } M \text{ with respect to } A,$$

has the same Hankel (Toeplitz) displacement rank α with respect to $\{F^f, F^b\}$. This result has many consequences. For example, by considering the matrices

$$\begin{bmatrix} A & I \\ I & O \end{bmatrix}, \quad \begin{bmatrix} I & A \\ A^T & O \end{bmatrix}, \quad \begin{bmatrix} \tilde{I} & A \\ A^T & O \end{bmatrix}$$

we can see that the inverse A^{-1} and the product A^TA also have low displacement rank if A has low displacement rank.

The displacement structure of a matrix is captured by its *generators viz.*, a matrix pair X, Y that satisfies the displacement equations

$$\nabla_{(F^f, F^b)}A = XY^T, \quad \text{or} \quad \Delta_{(F^f, F^b)}A = XY^T, \quad X \in \mathbf{R}^{m \times \alpha}, \quad Y \in \mathbf{R}^{n \times \alpha}.$$

The inverses and Schur complements of structured matrices can be obtained by operating on their generators. Doing so requires $O(\alpha mn)$ computations, where α is the displacement rank of A, whereas working with the matrix A itself requires $O(mn^2)$ or $O(m^2n)$ computations.

We described such algorithms for QR factorization, inversion, regularization, and solution of least squares problems. The key to obtaining these results is a combination of (efficient) algorithms for successive Schur-complementation applied to certain judiciously chosen "composite" (block) matrices.

We shall briefly outline the resulting generalized Schur algorithms.

Generalized Schur Algorithms.

To efficiently compute the Schur complement $D - CA^{-1}B$ of the matrix

$$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

we first need to obtain a *proper generator* for M, i.e., a generator of the form,

$$X = \begin{bmatrix} * & 0 & \cdots & 0 \\ * & * & \cdots & * \\ \vdots & \vdots & & \vdots \\ * & * & \cdots & * \end{bmatrix}, \quad Y = \begin{bmatrix} * & 0 & \cdots & 0 \\ * & * & \cdots & * \\ \vdots & \vdots & & \vdots \\ * & * & \cdots & * \end{bmatrix}, \tag{3}$$

for the Toeplitz displacement (Chapter 2), and

$$X = \begin{bmatrix} * & 0 & \cdots & 0 \\ * & * & \cdots & * \\ \vdots & \vdots & & \vdots \\ * & * & \cdots & * \end{bmatrix}, \quad Y = \begin{bmatrix} 0 & \cdots & 0 & * \\ * & \cdots & * & * \\ \vdots & & \vdots & \vdots \\ * & \cdots & * & * \end{bmatrix}, \tag{4}$$

for the Hankel displacement (Chapter 4). A non-proper generator of A can be converted to a proper one in several ways. One is by applying the following matrices,

$$S_{ij} = \begin{bmatrix} 1 & & & \\ & c & s_2 & \\ & -s_1 & c & \\ & & & 1 \end{bmatrix}, \quad c^2 + s_1 s_2 = 1 \tag{5}$$

which can be used to null out different entries by appropriate choices of $\{c, s_1, s_2\}$. Proper generators (3) or (4) can be obtained by using the matrix (5), or its special cases: Givens rotations, hyperbolic rotations and elementary matrices. By post-multiplying X and Y with a sequence of appropriate matrices S_{ij}, we can transform X and Y to proper form with $O(\alpha m)$ computations.

The next step is to modify one column of X and Y. Repeating the same step (i.e.,

transformation to proper form followed by a modifications of the column) r times, where r is the size of the square block A in M produces the generator of the Schur complement $D - CA^{-1}B$.

Displacement Representation of Composite Matrices (Chapter 3).

The above algorithms can be applied to the block matrices

$$A = \begin{bmatrix} T & I \\ I & O \end{bmatrix}, \quad \begin{bmatrix} T^T T & I \\ I & O \end{bmatrix}, \quad \begin{bmatrix} T_1 & T_2 \\ T_2^T & O \end{bmatrix}, \quad \begin{bmatrix} T^T T & T^T \\ T & I \end{bmatrix}, \quad \begin{bmatrix} T^T T & T^T \\ I & O \end{bmatrix},$$

to obtain *generalized Gohberg-Semencul formulas* or, equivalently, *displacement representations*, of the matrices,

$$T^{-1}, \quad (T^T T)^{-1}, \quad T_2^T T_1^{-1} T_2, \quad T(T^T T)^{-1} T^T, \quad (T^T T)^{-1} T^T.$$

Fast Matrix Factorizations and Solutions of Linear Equations (Chapters 2 and 4).

Another use of the generalized Schur algorithm is to obtain fast solutions for various equations of structured matrices such as Toeplitz and close-to-Toeplitz matrices, Hankel and close-to-Hankel matrices, and Vandermonde matrices.

Divide-and-Conquer Implementations (Chapter 5).

It turns out that the generalized Schur algorithm can be easily implemented in divide-and-conquer fashion.

2. Some Known and Unknown Numerical Properties of the fast Algorithms.

In this section, we shall present a rather casual description of the numerical properties of the fast algorithms considered in this thesis.

2.1 Levinson Algorithm.

The Levinson algorithm was analyzed by Cybenko [6]. Bunch [4] clarified Cybenko's work by introducing the concept of three different numerical stabilities.

Stability: *An algorithm for solving linear equations is stable for a class of matrices* $\mathbf{M}$ *if for each* $A \in \mathbf{M}$ *and for each* $\mathbf{b}$ *the computed solution* $\hat{\mathbf{x}}$ *to* $A\mathbf{x} = \mathbf{b}$ *satisfies* $\hat{A}\hat{\mathbf{x}} = \hat{\mathbf{b}}$ *for some* $\hat{A}$ *and* $\hat{\mathbf{b}}$, *where* $\hat{A}$ *is close to* A *and* $\hat{\mathbf{b}}$ *is close to* $\mathbf{b}$.

Strong stability: *An algorithm for solving linear equations is strongly stable for a class of matrices* $\mathbf{M}$ *if for each* $A \in \mathbf{M}$ *and for each* $\mathbf{b}$ *the computed solution* $\hat{\mathbf{x}}$ *to* $A\mathbf{x} = \mathbf{b}$ *satisfies* $\hat{A}\hat{\mathbf{x}} = \hat{\mathbf{b}}$, *where* $\hat{A} \in \mathbf{M}$ *and* $\hat{A}$ *and* $\hat{\mathbf{b}}$ *are close to* A *and* $\mathbf{b}$, *respectively.*

Weak stability: *An algorithm for solving linear equations is weakly stable for a class of matrices* $\mathbf{M}$ *if for each well-conditioned* $A \in \mathbf{M}$ *and for each* $\mathbf{b}$ *the computed solution* $\hat{\mathbf{x}}$ *to* $A\mathbf{x} = \mathbf{b}$ *is such that* $\|\mathbf{x}-\hat{\mathbf{x}}\|/\|\mathbf{x}\|$ *is small.*

According to the above definitions, strong stability implies stability and stability implies weak stability.

Sometimes it is hard to prove that an algorithm is stable. Bunch pointed out that Cybenko only proved that the Levinson algorithm is weakly stable. To see this let us consider Cybenko's result and the classical perturbation theorem (see e.g. [18]).

Cybenko's result: *The computed solution* $\hat{\mathbf{x}}$ *to* $T\mathbf{x} = \mathbf{b}$ *will always have a small residual,* $\mathbf{r} \equiv T\hat{\mathbf{x}} - \mathbf{b}$ *for "well-conditioned" symmetric positive Toeplitz matrices.*

Perturbation Theorem: *If* $A\mathbf{x} = \mathbf{b}$ *and* $\tilde{A}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}$, *where* A *is nonsingular, and if* $\|A-\tilde{A}\|\cdot\|A^{-1}\| < 1$, *then* $\tilde{A}$ *is nonsingular and*

$$\frac{\|\mathbf{x}-\tilde{\mathbf{x}}\|}{\|\mathbf{x}\|} \le \frac{\kappa(A)}{1 - \kappa(A)\dfrac{\|A-\tilde{A}\|}{\|A\|}} \left[\frac{\|A-\tilde{A}\|}{\|A\|} + \frac{\|\mathbf{b}-\tilde{\mathbf{b}}\|}{\|\mathbf{b}\|} \right].$$

From Cybenko's result and the above theorem, it is easy to see that the Levinson algorithm is weakly stable, because

$$\frac{\|\hat{\mathbf{x}}-\mathbf{x}\|}{\|\mathbf{x}\|} \le \kappa(T)\frac{\|\mathbf{r}\|}{\|\mathbf{b}\|}.$$

If Cybenko had shown that $\mathbf{r} = T\hat{\mathbf{x}} - \mathbf{b}$ is small for *all* symmetric positive Toeplitz matrices (including ill-conditioned matrix) then he would have proven that the Levinson algorithm is stable. It is unknown [4] whether the Levinson algorithm for symmetric positive-definite Toeplitz matrices is stable (in Bunch's sense) or not.

2.2 Schur Algorithm.

Numerical properties of the (generalized) Schur algorithm are also not fully understood yet. However, it would not be a wild conjecture that the Schur algorithm is also (at least) weakly stable because the Schur algorithm and the Levinson algorithm are closely related (see Chapter 2). To compare the Schur algorithm with the Cholesky algorithm, we generated the ill-conditioned positive-definite Toeplitz matrix (See Appendix for how to generate ill-conditioned positive-definite Toeplitz matrices),

$$T = \begin{bmatrix} 1 & .99 & .999602 & .98922 & .99847 \\ .99 & 1 & .99 & .999602 & .98922 \\ .999602 & .99 & 1 & .99 & .999602 \\ .98922 & .999602 & .99 & 1 & .99 \\ .99847 & .98922 & .999602 & .99 & 1 \end{bmatrix}, \quad \kappa(T) \approx 2.9 \times 10^7.$$

Let $\bar{L}_c$ and $\bar{L}_s$ denote the lower triangular factors of T computed with single precision arithmetic by the Cholesky algorithm and the Schur algorithm, respectively. Our simulation results show that

$$\| T - \bar{L}_c \bar{L}_c^T \|_2 = 7.5 \times 10^{-8}, \quad \| T - \bar{L}_s \bar{L}_s^T \|_2 = 8.9 \times 10^{-8}. \tag{6}$$

Therefore, we cannot exclude the possibility that the Schur algorithm is even stable in Bunch's sense (see [10] also). Moreover certain "thresholding strategy" in which "small" reflection coefficients are set equal to zero can substantially improve the numerical behavior of the Schur algorithm [3], [13].

2.3 Troubles associated with large Reflection Coefficients.

Cybenko obtained posteriori bounds on the condition number of Toeplitz matrices in terms of the reflection coefficients [6] (see [13] for a more recent result). For close-to-Toeplitz matrices large reflection coefficients indicate a large condition number of the matrices.

It is well-known [2], [17] that hyperbolic rotations with large reflection coefficients give numerically poor results. To see this consider a single 2×2 hyperbolic rotation;

$$[\mathbf{a}', \mathbf{b}'] = [\mathbf{a}, \mathbf{b}]\cdot H, \qquad H = \begin{bmatrix} ch & -sh \\ -sh & ch \end{bmatrix}, \tag{7}$$

where

$$\mathbf{a} = [a_1, a_2, \cdots, a_n]^T, \qquad \mathbf{b} = [b_1, b_2, \cdots, b_n]^T, \qquad |a_1| > |b_1|,$$
$$\mathbf{a}' = [a_1', a_2', \cdots, a_n']^T, \quad \mathbf{b}' = [0, b_2', \cdots, b_n']^T.$$

Let us slightly perturb the 'data' $\mathbf{a}$ and $\mathbf{b}$ by μ_1 and μ_2, and find the size of η_i, the resulting relative perturbation in the 'results':

$$[\mathbf{a}'(1+\eta_1), \mathbf{b}'(1+\eta_2)] = [\mathbf{a}(1+\mu_1), \mathbf{b}(1+\mu_2)]\cdot H.$$

Then clearly

$$[\eta_1\mathbf{a}', \eta_2\mathbf{b}'] = [\mathbf{a}, \mathbf{b}] \begin{bmatrix} ch\,\mu_1 & -sh\,\mu_1 \\ -sh\,\mu_2 & ch\,\mu_2 \end{bmatrix}. \tag{8}$$

Therefore

$$\|[\eta_1\mathbf{a}', \eta_2\mathbf{b}']\|_F \le [2(ch^2 + sh^2)\cdot\mu^2]^{1/2}\cdot\|[\mathbf{a}, \mathbf{b}]\|_F = 2^{1/2}\mu\cdot\left(\frac{1+k^2}{1-k^2}\right)^{1/2}\cdot\|[\mathbf{a}, \mathbf{b}]\|_F, \tag{9}$$

where $\mu = \max(|\mu_1|, |\mu_2|)$, and k is the reflection coefficient, $k = sh/ch = b_1/a_1$. From (9), one can see that a hyperbolic rotation gives an inaccurate result if the reflection coefficient k of H is close to ± 1, i.e., a_1' is close to zero (which is not surprising because of the large difference in the two eigenvalues of H).

To see the numerical difficulty associated with large reflection coefficients, we generated a positive-definite Toeplitz matrix with the following reflection coefficients (see Appendix),

$$K = [0,\ 0.99999,\ -0.99999,\ 0.99999,\ -0.99999].$$

The resulting matrix has condition number $\kappa(T) = 6.6 \times 10^{17}$. When we applied the Schur algorithm with double precision arithmetic to re-compute the reflection coefficients, we obtained

$$\bar{K} = [0,\ 0.99999,\ -0.99998999999710,\ 0.99998958118522,\ -0.94450034224923]$$

which was quite different from the "true" reflection coefficients. For some problems (e.g., orthogonal filter synthesis [16]), what we need is only the reflection coefficients rather than the factorization.

2.4 Ill-conditioned Matrices can have small Reflection Coefficients.

A ill-conditioned matrix does not always have large reflection coefficients. To see this let us define the two matrices

$$T = \begin{bmatrix} 1 & 0.5 & 0.2 \\ 0.5 & 1 & 0.5 \\ 0.2 & 0.5 & 1 \end{bmatrix}, \quad S = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 10^5 & 0 & 1 \end{bmatrix}. \tag{10}$$

One can check that

$$A \equiv STS^T = \begin{bmatrix} 1 & 0.5 & 10^5+0.2 \\ 0.5 & 1 & 5\times10^4+0.5 \\ 10^5+0.2 & 5\times10^4+0.5 & 10000040001 \end{bmatrix} \equiv LL^T,$$

where

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 0.5 & \sqrt{0.75} & 0 \\ 10^5+0.2 & \dfrac{0.4}{\sqrt{0.75}} & \sqrt{\dfrac{56}{75}} \end{bmatrix}.$$

The condition numbers of T and A are

$$\kappa(T) = 4.7, \quad \kappa(A) = 1.5 \times 10^{20}.$$

The matrix A also has displacement rank 2 because S is lower triangular Toeplitz (see Chapter 3 or [12]). Furthermore, the matrices A and T have the same reflection coefficients.

$$K = [0,\ 1/2,\ -1/15]. \tag{11}$$

Let $\bar{L}_s$ and $\bar{L}_c$ denote the lower triangular factors of A computed with double-precision arithmetic by the Schur algorithm and the Cholesky algorithm, respectively. Our simulation experiment gives rather surprising result:

$$\|L - \bar{L}_s\| = 2.2 \times 10^{-16}, \quad \|L - \bar{L}_c\| = 6.9 \times 10^{-7}, \tag{12a}$$

$$\|A - \bar{L}_s \bar{L}_s^T\| = 2.2 \times 10^{-16}, \quad \|A - \bar{L}_c \bar{L}_c^T\| = 1.1 \times 10^{-16}. \tag{12b}$$

As far as the bounds in (12a) are concerned the Schur algorithm performs far better than the Cholesky algorithm (beware of the matrix products STS^T, however).

2.5 Generalized Schur Algorithm and Sweet's Algorithm.

Sweet's algorithm [19] behaves quite differently from the Schur-type fast QR factorization algorithms. Let Q_w and R_w be the matrices computed by Sweet's algorithm, and let Q_s and R_s be the matrices computed by our algorithm. The simulation result by Luk and Qiao [15] for the following matrix,

$$T = \frac{1}{27}\begin{bmatrix} 27 & 9 & 3 & -23+t \\ 9 & 27 & 9 & 3 \\ 3 & 9 & 27 & 9 \\ -23+t & 3 & 9 & 27 \end{bmatrix}, \quad t = 10^{-7} \tag{13}$$

shows that

$$\|Q_w R_w - T\|_F / \|T\|_F = 5.9\times10^{-17}, \quad \|Q_w^T Q_w - I\|_F / \|I\|_F = 2.6\times10^{-17}.$$

However for the same matrix, our algorithm gives a poor result:

$$\|Q_s R_s - T\|_F / \|T\| = 3\times10^{-16}, \quad \|Q_s^T Q_s - I\|_F / \|I\|_F = 0.48.$$

The inaccurate Q_s might be understood along the lines that (i) $T^T T$ is ill-conditioned, (ii) the last reflection coefficient is very close to one and (iii) our fast QR factorization algorithm, in fact, computes the partial triangularization (Chap 2) of

$$\begin{bmatrix} T^T T & T^T \\ T & I \end{bmatrix}.$$

Although Sweet's algorithm works well for the ill-conditioned matrix (13), Luk and Qiao have shown that Sweet's algorithm did badly for the following well-conditioned matrix ($\kappa(T) = 5.6$),

$$T = \frac{1}{24}\begin{bmatrix} 8 & 4 & 2 & 1-t \\ 4 & 8 & 4 & 2 \\ 2 & 4 & 8 & 4 \\ 1-t & 2 & 4 & 8 \end{bmatrix}, \quad t = 10^{-7}. \tag{14}$$

The simulation result with our fast algorithm for the matrix T gives a very accurate result probably because $T^T T$ does not have large reflection coefficients.

2.6 Numerically Stable Fast Algorithm for Indefinite Matrices.

For indefinite matrices, the leading principal submatrices can be (close-to) singular. It is easy to find a well-conditioned indefinite matrix for which the Schur or Levinson algorithm perform badly. There are some previous works [5] that might be useful for finding a numerically stable generalized Schur algorithm.

3. Other Open Research Problems

3.1 Indefinite Structured Matrices.

There are no doubt other special algorithms that can be put into array form in the way we have described. In particular, we might mention algorithms for determining the root distribution of polynomials by studying the inertia of certain related matrices called Bezoutians (see e.g. [14]). One feature of such matrices is that they may not be strongly regular (strongly nonsingular) in the sense that not all leading minors may be nonzero. This has been an often tacit assumption in most of this thesis. Solution methods are known for some of these problems, especially in the Hankel case, and it would be interesting to examine them from our point of view.

3.2 Array form of the RLS Algorithms.

Another area of exploration would be to see how to obtain array forms for the various fast lattice and transversal filter RLS algorithms.

3.3 Accelerating the convergence of the Schur Algorithm.

The Schur algorithm is also widely used for the spectral factorization (see e.g. [9]). For such applications, it would be useful to find a way to accelerate the convergence of the algorithm.

3.4 Doubly Structured Matrices.

In many signal processing applications, we encounter so-called "doubly structured" matrices (e.g. block-Toeplitz matrix with Toeplitz blocks). All known fast algorithms do not fully utilize this additional structure.

3.5 Low displacement rank Decompositions.

Let A and B have displacement ranks α_1 and α_2, respectively. By considering the matrix

$$\begin{bmatrix} I & B \\ A & O \end{bmatrix}$$

one can check that the matrix AB has displacement rank less than or equal to $\alpha_1 + \alpha_2 + 1$. For a given matrix M with displacement rank α, is it possible to find matrices M_1 and M_2, whose displacement ranks are close to $\alpha/2$, such that $M = M_1 M_2$?

APPENDIX

Given a sequence of reflection coefficients

$$K = [k_0, k_1, k_2, \cdots, k_n], \quad k_0 = 0,$$

we can generate [7], [11] a symmetric positive-definite Toeplitz matrix using the transmission line shown in Fig 1. We excite the quiescent transmission line (i.e. zero initial condition) with

the impulse sequence

$$[1, 0, 0, \cdots].$$

Then the output sequence (see Fig 1),

$$[c_0, c_1, c_2, \cdots], \quad c_0 \equiv 1$$

gives the desired Toeplitz matrix

$$T = (c_{i-j}).$$

Because only orthogonal rotations are used, this procedure is numerically stable no matter how large the reflection coefficients are.

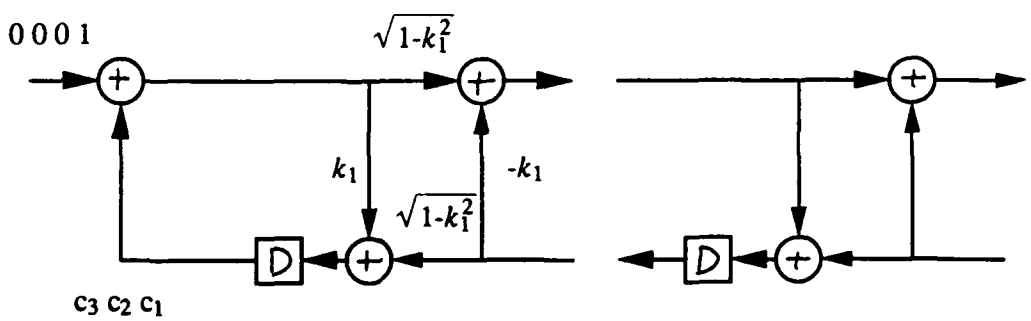


Fig 1. Generating Toeplitz Matrix given Reflection Coefficients.

REFERENCES

[1]. A. Bojanczyk, R. Brent and F. de Hoog, *QR factorization of Toeplitz matrices*, Numer. Math., 49 (1986), pp. 81-94.

[2]. A. Bojanczyk, R. Brent, P. van Dooren and F. de Hoog, *A note on downdating the Cholesky factorization*, SIAM J. Sci. Stat. Comput., vol. 8, May (1987), pp. 210-221.

[3]. A. Bruckstein, I. Koltracht and T. Kailath, *Inverse scattering with noisy data*, SIAM J. on Sci. Stat. Comput., 7, Vol. 4, pp. 1331-1349, Oct. 1986.

[4]. J. Bunch, *The weak and strong stability of algorithms in numerical linear algebra*, Linear Algebra and its Applications, Vol. 88/89, 49-66.

[5]. J. Bunch and B. Parlett, *Direct methods for solving symmetric indefinite systems of linear equations*, SIAM J. Numerical Analysis, (8), 639-655.

[6]. G. Cybenko, *The numerical stability of the Levinson-Durbin algorithm for Toeplitz systems of equations*, SIAM J. Sci. and Stat. Comp., Vol. 1, pp. 303-319, 1980

[7]. B. Friedlander, *Lattice filters for adaptive processing*, Proc. IEEE, Vol. 70, No. 8, Aug., pp. 829-867, 1982.

[8]. T. Georgiou, *On a Schur-algorithm based approach to spectral factorization: State-space formulae*, System and Control Letters 10 pp. 123-129, 1988.

[9]. M. Gover and S. Barnett, *Inversion of Toeplitz matrices which are not strongly non-singular*, IMA Journal of Numerical Anal., No. 5, pp. 101-110, 1985.

[10]. I. Ipsen, *Proc. NATO Advanced Study Inst. on Signal Processing*, Leuven, Belgium, 1989.

[11]. T. Kailath, A. Bruckstein and D. Morgan, Fast matrix factorizations via discrete transmission lines, *Linear Algebra Appl.* 75:1-25 (1986).

[12]. T. Kailath and H. Lev-Ari, On mappings between covariance matrices and physical systems, *Contemporary Mathematics* 47 (1985).

[13]. I. Koltracht and P. Lancaster, *Threshold algorithms for the prediction of reflection coefficients in a layered medium*, Geophysics, vol. 53, No. 7, Jul., (1988), pp. 908-919.

[14]. H. Lev-Ari, Y. Bistritz and T. Kailath, *Generalized Bezoutians and families of efficient root-location procedures*, to appear in IEEE Trans. Circ. Syst., 1989.

[15]. F. Luk and S. Qiao, *A fast but unstable orthogonal triangularization technique for Toeplitz matrices*, Lin. Alg. and Appl. (1987)

[16]. S. Rao and T. Kailath, *Orthogonal digital filters for VLSI implementation*, IEEE Trans. Circuit and System, CAS-31 (1984), pp. 933-945.

[17]. G. Stewart, *Some topics in numerical analysis*, Oak Ridge National Lab, ORNL-4303, 1968.

[18]. G. Stewart, *Intro. to Matrix Comp.*, Academic Press, New York, 1973

[19]. D. Sweet, *The use of pivoting to improve the numerical performance of Toeplitz solvers*, SPIE Conf., San Diego, 1986.