Inverse and Eigenspace Decomposition Algorithms for Statistical Signal Processing

by
Fazal Noor

Department of Electrical Engineering
McGill University, Montréal

A
Thesis
submitted
to
the Faculty of Graduate Studies and Research
in
partial fulfillment of the requirements for the
Degree of Doctor of Philosophy

February, 1993

ISBN 0-315-87565-8

Inverse and Eigen Decomp Algorithms for Statistical Signal Processing

by
Fazal Noor

Department of Electrical Engineering
McGill University, Montréal

A
Thesis
submitted
to
the Faculty of Graduate Studies and Research
in
partial fulfillment of the requirements for the
Degree of Doctor of Philosophy

September, 1992

Abstract

Inverse and Eigenspace Decomposition Algorithms for Statistical Signal Processing

In this work, a number of advances are described which we feel lead to better understanding and solution of the eigenvalue and inverse eigenvalue problems for Hermitian Toeplitz matrices. First, a unitary matrix is derived which transforms a Hermitian Toeplitz matrix into a real Toeplitz plus Hankel matrix. Some properties of this transformation are also presented. Second, we solve the inverse eigenvalue problem for Hermitian Toeplitz matrices. Specifically, we present a method for the construction of a Hermitian Toeplitz matrix from an arbitrary set of real eigenvalues. The procedure utilizes the discrete Fourier transform to first construct a real symmetric negacyclic matrix from the specified eigenvalues. The algorithm presented is computationally efficient. Finally, we derive a new order recursive algorithm and modify Trench's algorithm, both for eigenvalue decomposition. The former development is of mathematical interest; whereas, the latter is clearly of practical interest. The modifications proposed to Trench's algorithm are to employ noncontiguous intervals and to include a procedure to detect multiple eigenvalues. The goals of the modification are to improve the rate of convergence. The modified algorithm presented utilizes three root searching techniques: the Pegasus method, the modified Rayleigh quotient iteration with bisection shifts (MRQI-B), and the MRQI with Pegasus shifts (MRQI-P). Simulation results are provided for large matrices of orders 50, 100, 200, and 500. Application of the algorithms to Pisarenko's harmonic decomposition, an important signal processing problem, is presented. Fortran programs of the modified method are also provided.

Resumé

Algorithmes pour Décomposition de l'Ensemble des Racines Propres et Racines Inverses dans le traitement des signaux aléatoires

Dans cet ouvrage, quelques avancements scientifiques sont décrits qui, nous croyons, mènent à une meilleure compréhension, et, de meilleures solutions aux problèmes des racines propres et racines inverses pour les matrices *Hermitian Toeplitz*.

En premier lieu, une matrice unitaire, est dérivée qui a pour but la transformation d'une matrice *Hermitian Toeplitz*; le résultat de cette transformation se traduit par la somme d'une matrice *Toeplitz* réelle à une matrice *Hankel* réelle. En deuxième lieu, nous allons résoudre le problème des racines inverses pour les matrices *Hermitian Toeplitz*. En particulier, nous présentons une méthode de réalisation des matrices *Hermitian Toeplitz* à partir d'un ensemble quelconque de racines propres et réelles. Cette procédure utilise une transformation de Fourier de valeurs discrètes pour réaliser, en premier lieu, une matrice réelle, symétrique et 'negacyclic'. L'algorithme présenté dans cette ouvrage est certe efficace au calcul. Enfin, pour la décomposition des racines propres nous allons en premier, dériver un nouvel algorithme d'ordre récurrent, et, ensuite, modifier l'algorithme de Trench; le premier cas est d'intérêt mathématique tandis que le suivant est clairement d'intérêt pratique. Les modifications à l'algorithme de Trench sont proposées pour utiliser des intervalles non-contigüe et pour inclure un procédé de détection des racines multiples. Les modifications sont appliquées dans le but d'améliorer l'allure de convergence de l'algorithme dans l'estimées des racines propres. L'algorithme ainsi modifier, utilise trois techniques de detection des racines, soit; la méthode de Pegasus, la méthode par itérations mitigées du quotient de Raleigh suivant un décalage par bisection (IMQR-B), et par IMQR suivant la méthode de décalage de Pegasus (IMQR-P). Les résultats des simulations de l'algorithme sont présentés pour des matrices d'ordre 50, 100, 200 et 500.

Nous présentons aussi, une application des algorithmes élaborés dans cette thèse au problème de décomposition harmonique des fréquences, développé par Pisarenko; ceci étant un problème important dans le traitement des signaux. Enfin, les détails du logiciel décrivant l'algorithme, codé en langage FORTRAN, sont présentés à la fin de cet ouvrage.

ACKNOWLEDGEMENTS

I am most grateful to my friend and teacher, Professor Salvatore D. Morgera, for his valuable guidance and assistance during the entire preparation of this thesis.

I would like to thank members of my supervisory committee, Professor P. Kabal and Professor S. McFee. I am very thankful and grateful to Professor W. Trench for providing me with a copy of his computer programs and Professor Y.H. Hu for very useful and fruitful discussions. Also, I thank Professor M.R. Soleymani for his encouragement and help.

I would like to thank all my colleagues for their encouragement and help from time to time.

I thank the secretaries of the Electrical Engineering Dept., the McGill Computer Centre for providing research facilities for my initial work, and the Information Networks and Systems Laboratory(INSL), where my final research work came to completion. I thank Mr. Salvatore Torrente, system manager of INSL, for his kind help whenever I needed it. Also, I would like to thank Mr. Albert Pang for his help.

Finally, I am most thankful for the encouragement and full support provided to me by my parents, Surraya Sultana and Ghulam R. Noor; to my lovely sisters, Sabeha, Aisha, Jasmin, and Afshan; and my supportive brothers, Faisal, Mohammed, Abrahim, and Ahmed.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The goal of signal processing is the extraction of information from signals contaminated with noise. There are various techniques for extracting the information, and the methods usually depend on the models used to represent the information embedded in a signal. Statistical models are employed to describe a signal since the behaviour of sources and mechanisms responsible for its generation and propagation are unpredictable. Signal processing of this sort is related to classical time series analysis, and, therefore, covariance matrices come to play a major role in many signal processing applications. In many cases in algorithm development, the main effort reduces to an analysis of the covariance matrices involved in order to extract and exploit underlying structure.

1.1 Covariance Matrices in Statistical Signal Processing

In the statistical signal processing area of high resolution spectrum estimation, which finds applications to array, radar, sonar, seismic, speech, and image processing, eigenvalue and eigenvector decomposition methods offer under appropriate conditions, an alternative solution to the classical method based on the Fourier transform. An important signal processing problem, for example, in array signal processing is that of resolving the directions of arrival of multiple plane waves reaching an array contaminated with additive background noise. In such a case, a series of snapshots are obtained by sampling the signal field at the sensors. Assume the signals to be narrowband and let the nth snapshot of the field received at the kth sensor (see Figure 1.1) be [1]-[4],

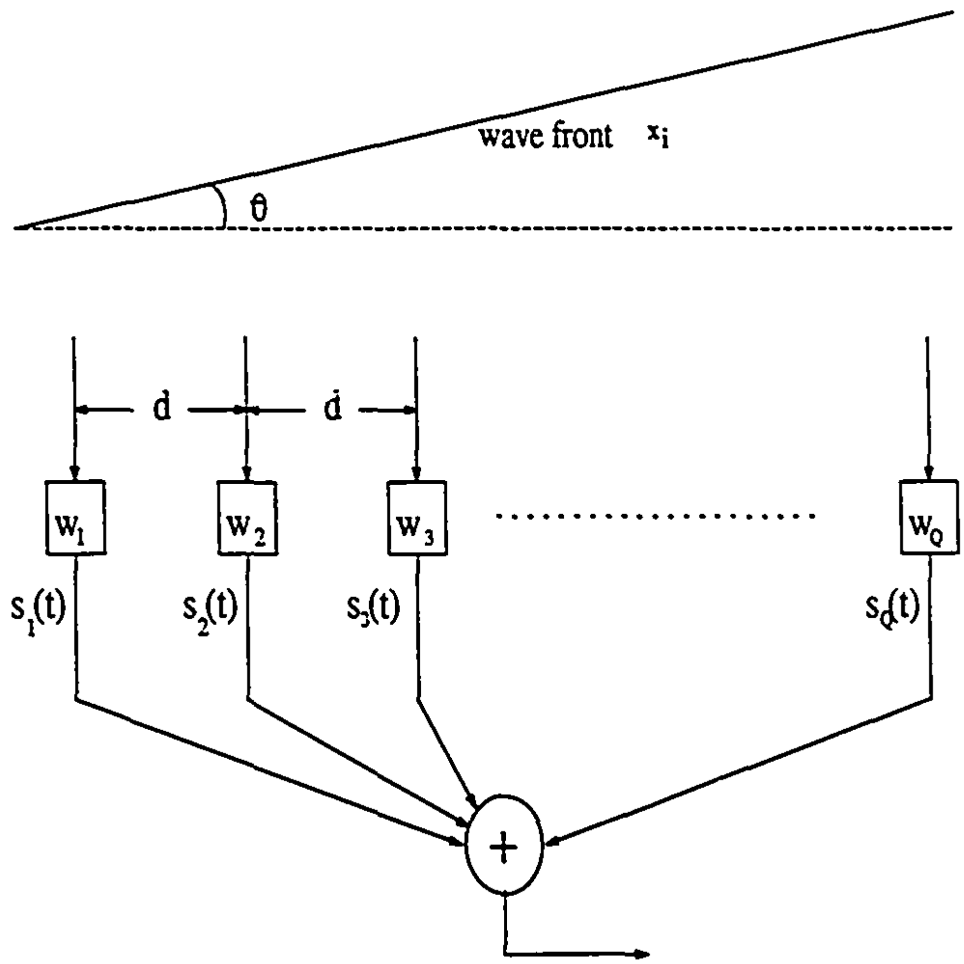

Figure 1.1: Linear array of sensors.

$$s_l(n) = \sum_{i=1}^{P} x_i(n)e^{-jlk_i} + z_l(n) \quad l = 1, ..., Q \tag{1.1}$$

where P is the number of plane waves, Q is the number of sensors, $x_i(n)$ is the amplitude of the ith narrowband wave, k_i are normalized wavenumbers, i.e. $k_i = \frac{2\pi d}{\lambda}\sin\theta_i$, d is the fixed distance between array sensors, λ is the spatial frequency, θ_i is an angle of incident of the ith wave impinging on the array, and $z_l(n)$ is the background noise.

Assume that the noise, $z_l(n)$, is spatially incoherent

$$E[\bar{z}_l(n)z_k(n)] = \sigma_z^2\delta_{lk} \tag{1.2}$$

and uncorrelated with the signal amplitudes, $x_i(n)$, i.e.,

$$E[\bar{z}_l(n)x_i(n)] = 0\,, \tag{1.3}$$

where the overbar denotes complex conjugation. Under the above conditions, it is possible to represent (1.1) in vector form as

$$\mathbf{s}_l(n) = \sum_{i=1}^{P} x_i(n)\bar{\mathbf{v}}_{k_i} + \mathbf{z}_l(n)\,, \tag{1.4}$$

where

$$\mathbf{v}_k = [e^{jk}\, e^{j2k}\, e^{j3k}\, \ldots\, e^{jQk}]^T \tag{1.5}$$

is the *phasing* or *steering vector*. Furthermore, assume that the sources are uncorrelated with each other. The signal field then has an autocorrelation matrix of order $(Q \times Q)$ of the following form

$$R = E[\mathbf{s}(n)\mathbf{s}^H(n)] = \sum_{i=1}^{P} \mathbf{v}_{k_i} D \mathbf{v}_{k_i}^H + \sigma_z^2 I\,, \tag{1.6}$$

where $D = E[\bar{x}_i(n)x_j(n)]$ is a diagonal matrix of order $(P \times P)$, and I is the identity matrix, and H denotes conjugate transpose. Rewriting (1.6) in matrix form, we have

$$R = VDV^H + \sigma_z^2 I\,, \tag{1.7}$$

where $V = [\mathbf{v}_{k_1}, \mathbf{v}_{k_2}, \cdots, \mathbf{v}_{k_P}]$ is the matrix consisting of P direction vectors. Since the signal and noise are assumed stationary, R is Hermitian and has a *Toeplitz* structure; matrices having this combined structure are called *Hermitian Toeplitz* matrices. In general, a matrix, C, of order n, is called Toeplitz if its elements $c_{ij} = c_{i-j}$ for all $i, j = 1, ..., n$;

is called symmetric Toeplitz if its elements $c_{ij} = c_{|i-j|}$ for all $i, j = 1, \ldots, n$; and is called Hermitian Toeplitz if its elements $\bar{c}_{-i} = c_i$ for all $i = 0, 1, \ldots, n-1$. Symmetric Toeplitz and Hermitian Toeplitz matrices are completely specified by their first row of elements.

It is well known that *eigenmethods* offer high resolution capabilities [1]-[4],[24]-[26] over conventional methods. The problem at hand reduces to the eigendecomposition of the covariance matrix

$$R\mathbf{q} = \lambda\mathbf{q} \tag{1.8}$$

in which the P largest eigenvalues of R correspond to the signal subspace and the remaining $(Q-P)$ minimum eigenvalues equal to σ_z^2 correspond to the noise subspace. Note that the eigenvectors in the noise subspace are not unique and any vector in the noise subspace evaluated on the unit circle

$$C(z) = \sum_{i=0}^{Q} q_i z^{-i} \tag{1.9}$$

will have P zeroes $z_i = e^{jk_i}$, for $i = 1, 2, \ldots, P$ at the desired wavenumber frequencies k_i and $(Q-P)$ other spurious zeros. This is easily verified, since if

$$R\mathbf{q} = \sigma_z^2\mathbf{q} \tag{1.10}$$

then

$$(VDV^H + \sigma_z^2 I)\mathbf{q} = \sigma_z^2\mathbf{q} \tag{1.11}$$

and, therefore, $V^H\mathbf{x} = \mathbf{0}$ since D is positive definite.

The information about the desired wavenumber frequencies k_i is obtained from an eigenvalue analysis of the covariance matrix, and,for this purpose, efficient methods are required to find the minimum eigenvalue [24]-[26]. In theory, the minimum eigenvalue has a multiplicity greater than unity; however, in practice, the minimum eigenvalue occurs as a cluster of eigenvalues having approximately the same value. For this reason, it is necessary to compute all clustered eigenvalues and take an average to better approximate the desired frequencies.

Once the eigendecomposition of this matrix is obtained, one might ask the question, is it possible to construct a Hermitian Toeplitz with these eigenvalues? This is a nonunique problem for Hermitian Toeplitz matrices and finds application in the area of array signal processing, particularly in the case of optimum beamforming for interference or jammer nulling.

In this work, we focus on inverse and eigenspace decomposition algorithms and their efficiency and accuracy for Hermitian Toeplitz covariance matrices. The subject of Toeplitz matrices is vast, as such matrices occur in a wide variety of other applications such as system identification, linear prediction, spectral estimation, and any problem in which the covariance matrix of a weakly stationary stochastic process arises. Readers further interested in applications are directed to references [1]-[5], while mathematically inclined readers might find [6] appealing.

Due to the Toeplitz structure, numerous properties have been presented in the literature [7]-[11]. One new property that we present is that a Hermitian Toeplitz matrix is *unitarily similar* to a real Toeplitz plus Hankel matrix. We study the effect of the unitary transform on the eigenvalues and eigenvectors of Hermitian Toeplitz matrices; on the eigenvalue relation between T, H, and $T + H$, where T and H denote the Toeplitz and Hankel factors, respectively; and on existing algorithms which solve a system of Hermitian Toeplitz equations. There exist algorithms which solve a real Toeplitz plus Hankel system of equations [12, 13]. We explore these algorithms in terms of their computational complexity. Furthermore, the unitary transform proves useful in obtaining a solution to the inverse eigenvalue problem for real symmetric matrices, once the solution to the inverse eigenvalue problem for Hermitian Toeplitz matrices is obtained.

Although the theoretical solution to the inverse eigenvalue problem for real symmetric Toeplitz matrices is unsolved [10, 15, 16], numerical solutions to the inverse eigenvalue problem for real symmetric Toeplitz matrices have been presented [16, 17]. We present a solution to the inverse eigenvalue problem in the case of Hermitian Toeplitz matrices.

On the other hand, there are numerous techniques for eigenvalue computation. Methods for eigenvalue computation of a general matrix require $O(n^3)$ operations [18, 19]; however, efficient algorithms exist [20, 21, 22] which exploit the Toeplitz structure to solve a system of linear equations and require $O(n^2)$ operations. A computational complexity defined as $M = O(\alpha n^2)$ implies that $(M/n^2) \rightarrow \alpha$ for large n [21]. Algorithms based on the Levinson recursion are presented and may be used to find the eigenvalues of Toeplitz matrices. We present methods that fall into two categories, order recursive and iterative. The order recursive methods presented utilize the deflation of polynomials and, hence, are sensitive to roundoff errors. On the other hand, Trench's iterative method and new methods based on modifications of Trench's iterative method are presented and are more viable for high order matrices. A further reduction in computational complexity may be achieved by using parallel methods [26]. Parallel methods use n processors and reduce the computing time by a factor of n.

1.2 Major Contributions

1. Discovery of a unitary matrix which transforms a Hermitian Toeplitz matrix into a real Toeplitz plus Hankel matrix. The importance of the unitary matrix is that it preserves structure. Some properties of this transformation are also presented.

2. Solution to the inverse eigenvalue problem for Hermitian Toeplitz matrices. A method is presented which shows that a negacyclic matrix of order $2n$ is equivalent to a Hermitian Toeplitz matrix of order n.

3. Derivation of a new order recursive algorithm for eigendecomposition. This algorithm is considered to be primarily of theoretical interest.

4. Modifications of Trench's iterative eigendecomposition algorithm for Hermitian Toeplitz matrices. The modifications include the use of noncontiguous intervals and the inclusion of the case of multiple eigenvalues. The modifications proposed are shown to have important consequences for efficiency when working with high order matrices.

1.3 Organization of the Thesis

The thesis is organized as follows. In Chapter 2, we present a unitary matrix which transforms a Hermitian Toeplitz matrix into a real Toeplitz plus Hankel matrix. Additional properties and consequences of this unitary transformation are also presented.

In Chapter 3, we present the inverse eigenvalue problem for Hermitian Toeplitz matrices. We describe a method that permits the construction of a Hermitian Toeplitz matrix with an arbitrary set of real eigenvalues. It is shown that a negacyclic real symmetric Toeplitz matrix of order $2n$ is equivalent to a Hermitian Toeplitz matrix of order n, thereby providing a simple solution to the inverse eigenproblem for Hermitian Toeplitz matrices.

In Chapter 4, we present two methods for solution of the eigenvalue problem. The methods presented fall into two categories, order recursive and iterative, with the latter being more numerically stable. In the iterative category, we present Trench's method and new methods based on modifications of Trench's method. The modifications involve maintaining tighter lower and upper bound noncontiguous intervals for each eigenvalue during the search mode and the inclusion of the multiple eigenvalue case. The modifications have important consequences for efficiency in terms of convergence and computational complexity when working with high order matrices. The algorithms may be applied to Pisarenko's harmonic decomposition and array processing problems of the type described earlier.

Chapter 5 summarizes the work and offers directions for further research in this interesting area.

Chapter 2

On a Unitary Transform for Hermitian Toeplitz Matrices

2.1 Introduction

It has been shown that Hermitian persymmetric [7] and centrohermitian [8] matrices are similar to a real symmetric matrix. The similarity transform reduction from the complex field to the real field results in savings in both computer time and storage in the calculation of the eigensystem of Hermitian persymmetric matrices [7]. A Hermitian Toeplitz matrix is a special form of a Hermitian persymmetric matrix and has a special structure (namely, Toeplitz) over the complex field. Applying the unitary similarity transform of [8] to a Hermitian Toeplitz matrix reduces it to a real symmetric matrix, but at the price of losing the special structure (Toeplitz) for which efficient algorithms exist [20, 21, 22].

In this chapter, we present a unitary matrix which transforms a Hermitian Toeplitz matrix into a real Toeplitz-plus-Hankel matrix of the *same* order. As a result of this, certain properties hold and are discussed. In fact, this unitary transform preserves the Toeplitz structure of the real part of the Hermitian Toeplitz matrix and transforms the imaginary part into a Hankel structure. It is a well known result that it is possible to convert a Hermitian Toeplitz system of order n into a block Toeplitz system of order $2n$ by equating real and imaginary parts, or, as in [12], by converting a Toeplitz-plus-Hankel structure to a block Toeplitz structure and then using a block-Levinson recursion method. On the other hand, there exists an algorithm that directly (i.e., without forming a block Toeplitz structure) solves a system of $T + H$ equations [13], where T and H denote the Toeplitz and Hankel factors, respectively. We present an efficient alternative to solving a special class of Toeplitz-plus-Hankel systems of equations for which the Toeplitz matrix is symmetric and the Hankel matrix is skew-centrosymmetric.

Definitions:

J is an exchange matrix with ones along the secondary diagonal and zeroes elsewhere. Note that $J = J^H = J^{-1}$, where H stands for complex conjugate transpose.

H is skew-centrosymmetric if $JHJ = -H$.

T is centrosymmetric if $JTJ = T$.

M is persymmetric if $JM^HJ = \bar{M}$. Note that Toeplitz matrices are persymmetric.

C is centrohermitian if $JCJ = \bar{C}$.

I is an identity matrix.

S is a symmetric matrix if $S^T = S$.

2.2 Mathematical Development

2.2.1 Unitary matrix

A Hermitian Toeplitz matrix C of even order n may be partitioned as

$$C = \begin{pmatrix} A & BJ \\ J\bar{B} & J\bar{A}J \end{pmatrix} \tag{2.1}$$

and split into real and imaginary parts

$$C = \begin{pmatrix} W & YJ \\ JY & JWJ \end{pmatrix} + j \begin{pmatrix} X & ZJ \\ -JZ & -JXJ \end{pmatrix}. \tag{2.2}$$

where $A = W + jX$; $BJ = YJ + jZJ$; W is Toeplitz and symmetric; Y, Z are symmetric; and X is Toeplitz and skew-symmetric. The matrices W, X, Y, Z are real and of order $n/2 \times n/2$. Then, a unitary transformation of the form

$$U = \frac{1}{2} \begin{pmatrix} (1-j)I & (1+j)J \\ (1+j)J & (1-j)I \end{pmatrix} \tag{2.3}$$

$$U^H = U^{-1} = \frac{1}{2} \begin{pmatrix} (1+j)I & (1-j)J \\ (1-j)J & (1+j)I \end{pmatrix} \tag{2.4}$$

will transform C into a real symmetric (but not centrosymmetric) matrix S which is the sum of real Toeplitz and Hankel matrices of special form, i.e.,

$$\begin{aligned} S = UCU^{-1} &= \begin{pmatrix} W & YJ \\ JY & W \end{pmatrix} + \begin{pmatrix} JZ & XJ \\ -JX & -ZJ \end{pmatrix} \\ &= \quad T \quad + \quad H, \end{aligned} \tag{2.5}$$

where T is Toeplitz, persymmetric symmetric (centrosymmetric) and H is Hankel, persymmetric skew-symmetic (skew-centrosymmetric). It is interesting to note that the above unitary transform U preserves the real part of C and transforms the imaginary part of C into a Hankel matrix as illustrated by the following example.

Example: Let

$$\begin{aligned} C &= \begin{pmatrix} 10 & 5+j2 & 4+j3 & 2+j \\ 5-j2 & 10 & 5+j2 & 4+j3 \\ 4-j3 & 5-j2 & 10 & 5+j2 \\ 2-j & 4-j3 & 5-j2 & 10 \end{pmatrix} \\ &= \begin{pmatrix} 10 & 5 & 4 & 2 \\ 5 & 10 & 5 & 4 \\ 4 & 5 & 10 & 5 \\ 2 & 4 & 5 & 10 \end{pmatrix} + j \begin{pmatrix} 0 & 2 & 3 & 1 \\ -2 & 0 & 2 & 3 \\ -3 & -2 & 0 & 2 \\ -1 & -3 & -2 & 0 \end{pmatrix}. \end{aligned}$$

Then, applying the above unitary transform U results in

$$S = \begin{pmatrix} 10 & 5 & 4 & 2 \\ 5 & 10 & 5 & 4 \\ 4 & 5 & 10 & 5 \\ 2 & 4 & 5 & 10 \end{pmatrix} + \begin{pmatrix} 1 & 3 & 2 & 0 \\ 3 & 2 & 0 & -2 \\ 2 & 0 & -2 & -3 \\ 0 & -2 & -3 & -1 \end{pmatrix}$$

$$= \begin{pmatrix} 11 & 8 & 6 & 2 \\ 8 & 12 & 5 & 2 \\ 6 & 5 & 8 & 2 \\ 2 & 2 & 2 & 9 \end{pmatrix}.$$

The trace of H is zero since it is skew-centrosymmetric. The antidiagonal consists of zeroes. Also note that the unitary transformation preserves the Toeplitz structure of the real part of C. For the imaginary part, the unitary transform has the effect of an exchange matrix with removal of the complex number j, that is, the result may be thought of as a postmultiplication of the imaginary part of C by J (i.e., $ImC \cdot J$) or as a premultiplication of ImC by $-J$ (i.e., $-J \cdot ImC$).

When the order n of C is odd, an analogous unitary transform exists with slight modification given as

$$U = \frac{1}{2}\begin{pmatrix} (1-j)I & 0 & (1+j)J \\ 0 & 2 & 0 \\ (1+j)J & 0 & (1-j)I \end{pmatrix} \tag{2.6}$$

and

$$U^H = U^{-1} = \frac{1}{2}\begin{pmatrix} (1+j)I & 0 & (1-j)J \\ 0 & 2 & 0 \\ (1-j)J & 0 & (1+j)I \end{pmatrix}. \tag{2.7}$$

Since the result of a multiplication of two unitary matrices is unitary. The above unitary matrices are obtained as

$$U = \frac{1}{2}\begin{pmatrix} I & I \\ J & -J \end{pmatrix}\begin{pmatrix} I & J \\ -jI & jJ \end{pmatrix} \tag{2.8}$$

for n of even order and as

$$U = \frac{1}{2}\begin{pmatrix} I & 0 & I \\ 0 & \sqrt{2} & 0 \\ J & 0 & -J \end{pmatrix}\begin{pmatrix} I & 0 & J \\ 0 & \sqrt{2} & 0 \\ -jI & 0 & jJ \end{pmatrix} \tag{2.9}$$

for n of odd order.

As a result of the previous discussion, if S is symmetric (but not centrosymmetric) such that $S + JSJ = T$, then S may be written as $S = T + H$. Also, if $S + JSJ = T$, then $S - JSJ = H$. Hence, S may be transformed into a Hermitian Toeplitz matrix.

2.2.2 Affect on eigenvalues and eigenvectors

The eigenvalues of the matrix C are invariant with respect to the unitary transformation UCU^{-1}. If λ is an eigenvalue of C and $\mathbf{v}$ is the associated eigenvector, then

$$C\mathbf{v} = \lambda\mathbf{v}. \tag{2.10}$$

Premultiplication by U results in

$$UC\mathbf{v} = \lambda U\mathbf{v} \tag{2.11}$$

which can also be written as

$$(UCU^{-1})U\mathbf{v} = \lambda U\mathbf{v}. \tag{2.12}$$

The eigenvectors are, therefore, premultiplied by U. Note that $\mathbf{v}$ has Hermitian symmetry (i.e., $\mathbf{v} = J\bar{\mathbf{v}}$). Now, let $\mathbf{v}$ be an eigenvector of C written as [7]

$$\mathbf{v} = \begin{pmatrix} \mathbf{x} \\ J\mathbf{x} \end{pmatrix} + j\begin{pmatrix} \mathbf{y} \\ -J\mathbf{y} \end{pmatrix}, \tag{2.13}$$

where $\mathbf{x}, \mathbf{y}$ are real vectors of dimension $n/2$. Then, $U\mathbf{v}$, the eigenvector of UCU^{-1}, is

$$U\mathbf{v} = \begin{pmatrix} \mathbf{x} \\ J\mathbf{x} \end{pmatrix} + \begin{pmatrix} \mathbf{y} \\ -J\mathbf{y} \end{pmatrix}. \tag{2.14}$$

For $\mathbf{v}$ of odd order, let $\mathbf{v}$ be an eigenvector of C written as,

$$\mathbf{v} = \begin{pmatrix} \mathbf{x} \\ \alpha \\ J\mathbf{x} \end{pmatrix} + j\begin{pmatrix} \mathbf{y} \\ 0 \\ -J\mathbf{y} \end{pmatrix}, \tag{2.15}$$

where $\mathbf{x}, \mathbf{y}$ are again real vectors of dimension $n/2$ and α is a real scalar. Then, $U\mathbf{v}$, the eigenvector of UCU^{-1}, is

$$U\mathbf{v} = \begin{pmatrix} \mathbf{x} \\ \alpha \\ J\mathbf{x} \end{pmatrix} + \begin{pmatrix} \mathbf{y} \\ 0 \\ -J\mathbf{y} \end{pmatrix}. \tag{2.16}$$

In other words, we can state by inspection that the real plus the imaginary part of v is an eigenvector of UCU^{-1}.

2.3 Eigenvalue relation between T, H, and $T+H$

Since T, H, and $T+H$ are $n \times n$ symmetric matrices, the eigenvalues of $T+H$ are bounded by

$$\lambda_k(T) + \lambda_1(H) \leq \lambda_k(T+H) \leq \lambda_k(T) + \lambda_n(H) \tag{2.17}$$

for $k = 1, 2, ..., n$ [18, 19]. The kth eigenvalue is denoted by $\lambda_k(\cdot)$ and $\lambda_1(\cdot) \leq \lambda_2(\cdot) \leq \cdots \leq \lambda_n(\cdot)$. The eigenvalues of an even and odd order skew-centrosymmetric H are $\{-\sigma_{n/2}, ..., -\sigma_1, \sigma_1, ..., \sigma_{n/2}\}$ and $\{-\sigma_{n-1/2}, ..., -\sigma_1, 0, \sigma_1, ..., \sigma_{n-1/2}\}$ for n even and odd, respectively. Since, for n even, the minimum eigenvalue is $\lambda_1(H) = -\sigma_{n/2}(H)$ and the maximum eigenvalue is $\lambda_n(H) = \sigma_{n/2}(H)$, the above relation may be written as

$$\lambda_k(T) - \sigma_{n/2}(H) \leq \lambda_k(T+H) \leq \lambda_k(T) + \sigma_{n/2}(H). \tag{2.18}$$

We may also write (2.17) as

$$\lambda_k(H) + \lambda_1(T) \leq \lambda_k(T+H) \leq \lambda_k(H) + \lambda_n(T). \tag{2.19}$$

The eigenvalues of T, H, and $T+H$ are tabulated in Table 2.1 for the matrix of the previous example. In Table 2.2, we tabulate the computed bounds (2.17) and (2.19) for the same example. The intersection of bounds obtained by (2.17) and (2.19) give a somewhat tighter bound on the eigenvalues of $T+H$. A closer look at the bounds in Table 2.2 reveals that a bound may also contain other eigenvalues. For instance, a bound on $\lambda_1(T+H)$, namely $[-1.17, 9.92]$, also contains the eigenvalues $\lambda_2(T+H)$ and $\lambda_3(T+H)$. The bound on $\lambda_3(T+H)$ obtained by (2.17) includes the eigenvalues $\lambda_1(T+H)$ and $\lambda_2(T+H)$, but the bound on $\lambda_3(T+H)$ obtained by (2.17) does *not*, however, include $\lambda_1(T+H)$ and $\lambda_2(T+H)$. As a result, the intersection of the two bounds does *not* contain other eigenvalues of $T+H$. Unfortunately, this does not always hold, since the intersection of the bounds (2.17) and (2.19) for $\lambda_2(T+H)$ also contains

Table 2.1: Eigenvalues of T, H, and T+H

Eigenvalue	T	H	T+H
λ_1	4.375	-5.553	2.869
λ_2	4.697	-0.402	4.643
λ_3	8.302	0.402	8.290
λ_4	22.624	5.553	24.196

Table 2.2: Intersection of bounds.

(2.17)	(2.19)	(2.17) $\cap$ (2.19)
$-1.17 \leq \lambda_1 \leq 9.92$	$-1.17 \leq \lambda_1 \leq 17.07$	$-1.17 \leq \lambda_1 \leq 9.92$
$-0.85 \leq \lambda_2 \leq 10.25$	$3.97 \leq \lambda_2 \leq 22.22$	$3.97 \leq \lambda_2 \leq 10.25$
$2.74 \leq \lambda_3 \leq 13.85$	$4.77 \leq \lambda_3 \leq 23.02$	$4.77 \leq \lambda_3 \leq 13.85$
$17.07 \leq \lambda_4 \leq 28.17$	$9.92 \leq \lambda_4 \leq 28.17$	$17.07 \leq \lambda_4 \leq 28.17$

$\lambda_3(T+H)$. We now state and prove a simple proposition regarding the eigenvalues of $T+H$.

Proposition: Consider the matrix sum $T+H$. If T is positive definite then the eigenvalues of $T+H$ are greater than the corresponding eigenvalues of H.

Proof: Since $\lambda_1(T) > 0$ for positive definite T, then, from (2.19), $\lambda_k(H)+\ \lambda_1(T) \leq \lambda_k(T+H)$: hence, the result obtains.

2.4 Effect on solving system of Hermitian Toeplitz equations

Efficient algorithms exist [20]-[23] which solve a system of Toeplitz equations given by

$$C\mathbf{v} = \mathbf{d} \tag{2.20}$$

for the vector $\mathbf{v}$. Now, if this equation is premultiplied by U, then $\mathbf{v}$ also satisfies

$$UC\mathbf{v} = U\mathbf{d}. \tag{2.21}$$

The change of variables $\mathbf{v} = U^{-1}\mathbf{q}$ and substitution in (2.21) results in

$$UCU^{-1}\mathbf{q} = U\mathbf{d}. \tag{2.22}$$

Consequently, the solution vector $\mathbf{v}$ of (2.20) can be obtained from the solution vector $\mathbf{q}$ of (2.22) or $\mathbf{q}$ can be obtained from $\mathbf{v}$.

Note that if a system of real equations is Toeplitz-plus-Hankel ($T + H$), where T is symmetric Toeplitz and H is skew-centrosymmetric Hankel, then the equations may be transformed into a Hermitian Toeplitz system and solved with $1.25n^2 + O(n)$ complex multiplies or $3.75n^2 + O(n)$ real multiplies [21]. This is a significant improvement in complexity over the approach of [12] which requires $12n^2 + O(n)$ real multiplies, and is slightly lower in complexity than the approach found in [13] which uses an entirely different development and requires $6n^2 + O(n)$ real multiples.

2.5 Discussion

In this chapter, we have shown that a constant unitary matrix exists which transforms a Hermitian Toeplitz matrix into a real Teoplitz-plus-Hankel structure. As a consequence of this property, some real symmetric matrices may be converted into Hermitian Toeplitz matrices and vice versa.

It is interesting to note that a Hermitian Toeplitz matrix may be thought of as a real Toeplitz matrix perturbed by a special Hankel (skew-centrosymmetric) matrix. Using perturbation theory, we showed the eigenvalue relation between T, H, and $T + H$. We stated and proved a simple proposition, namely, that the eigenvalues of $T + H$ are greater than the corresponding eigenvalues of H when T is positive definite. Those readers interested in this area may use the results of this chapter to further study the relation between the eigenvalues of the matrices T, H, and $T + H$.

Chapter 3

Inverse Eigenvalue Problem for Hermitian Toeplitz Matrices

3.1 Introduction

In this chapter, we are concerned with the inverse eigenvalue problem within the context of statistical signal processing and Hermitian Toeplitz covariance matrices associated with weakly stationary stochastic processes of complex form. Specifically, we present a method for the construction of a Hermitian Toeplitz matrix with an *arbitrary* set of real eigenvalues. The inverse eigenvalue problem treated is significantly simpler than the inverse eigenvalue problem encountered in the real weakly stationary stochastic process case when the covariance is real symmetric Toeplitz. The latter inverse eigenvalue problem is still unresolved for matrices of order greater than four [10, 15], although numerical procedures do exist [16, 17]. The reason for the relative difference in difficulty for the two inverse eigenvalue problems appears to be related to the fact that there are twice as many specifiable parameters in a Hermitian Toplitz matrix as there are in a real symmetric Toeplitz matrix.

The approach we take is to first construct an even order *negacyclic* real symmetric Toeplitz matrix having the desired eigenspectrum, where each eigenvalue, distinct or not, is repeated twice. The negacyclic matrix of order $2n$ so constructed, is then revealed to be the *real* matrix of a Hermitian Toeplitz matrix of order n which has the desired eigenspectrum. We provide a brief description of negacyclic matrices, describe the approach, and present an example.

3.2 Negacyclic matrices

Real negacyclic matrices are defined in Section 3.2.1 of [29] as circulant matrices having a change in sign for all elements below the main diagonal. A real symmetric negacyclic matrix, Q, of order m may be specified by the first row of elements, $\mathbf{q}^T = [q_0 q_1 \cdots q_{m-1}]$, where $q_{m-k} = -q_k$, $k = 0, 1, ..., m-1$, and the index $m-k$ is understood to be modulo m. It is seen, therefore, that real symmetric negacylic matrices are a subclass of real symmetric Toeplitz matrices.

The eigenspectrum, $\{\lambda_i : i = 0, 1, ..., m-1\}$, of a symmetric negacyclic matrix has elements which are given by the discrete Fourier transform (DFT) of $\hat{\mathbf{q}}^T = [q_0\, q_1\omega \cdots q_{m-1}\omega^{m-1}]$, where $\omega = e^{j\frac{\pi}{m}}$ [11, 29], i.e.,

$$\lambda_i = \sum_{k=0}^{m-1} q_k\, e^{j\frac{\pi}{m}k}\, e^{j\frac{2\pi}{m}ik}, \tag{3.1}$$

for $i = 0, ..., m-1$. For a symmetric negacyclic matrix of *even* order $m = 2n$, there are n eigenvalues given by

$$\begin{aligned}
\lambda_i &= q_0 + \sum_{k=1}^{n-1}[q_k e^{j\frac{\pi}{m}(2i+1)k} + q_{m-k}e^{j\frac{\pi}{m}(2i+1)(m-k)}] \\
&= q_0 + \sum_{k=1}^{n-1} q_k[e^{j\frac{\pi}{m}(2i+1)k} - e^{j\pi(2i+1)}e^{-j\frac{\pi}{m}(2i+1)k}] \\
&= q_0 + \sum_{k=1}^{n-1} q_k[e^{j\frac{\pi}{m}(2i+1)k} + e^{-j\frac{\pi}{m}(2i+1)k}] \\
&= q_0 + 2\sum_{k=1}^{n-1} q_k \cos\frac{\pi}{m}(2i+1)k
\end{aligned} \tag{3.2}$$

for $i = 0, ..., m-1$, which appear with multiplicity two; specifically, $\lambda_i = \lambda_{m-i-1}$, $i = 0, 1, ..., n-1$. Of course, the actual multiplicity may be higher, depending on whether the eigenvalues of (3.1) are distinct or not.

We now turn the situation around by observing that the vector of elements $\mathbf{q}$ of a negacyclic real symmetric Toeplitz matrix of order m may be obtained from a given set of n eigenvalues by use of the inverse DFT, viz.,

$$\begin{aligned}
q_k &= \frac{e^{-j\frac{\pi}{m}k}}{m}\sum_{i=0}^{n-1}[\lambda_i e^{-j\frac{2\pi}{m}ik} + \lambda_{m-i-1}e^{-j\frac{2\pi}{m}(m-i-1)k}] \\
&= \frac{e^{-j\frac{\pi}{m}k}}{m}\sum_{i=0}^{n-1}\lambda_i[e^{-j\frac{2\pi}{m}ik} + e^{-j2\pi k}e^{j\frac{2\pi}{m}ik}e^{j\frac{2\pi}{m}k}] \\
&= \frac{1}{m}\sum_{i=0}^{n-1}\lambda_i[e^{j\frac{\pi}{m}(2i+1)k} + e^{-j\frac{\pi}{m}(2i+1)k}] \\
&= \frac{1}{n}\sum_{i=0}^{n-1}\lambda_i\cos\frac{\pi}{m}(2i+1)k
\end{aligned} \tag{3.3}$$

for $k = 0, ..., n$. The DFT then becomes a simple vehicle for specifying the elements of Q given a set of eigenvalues $\{\lambda_i : i = 0, 1, ..., m-1\}$.

3.3 Relation to Hermitian Toeplitz matrices

The purpose of this section is to reveal the relationship that exists between symmetric negacyclic matrices of order m and Hermitian Toeplitz matrices of order n. Let

$$Q = \left(\begin{array}{ccccc|ccccc}
q_0 & q_1 & q_2 & q_3 & q_4 & 0 & -q_4 & -q_3 & -q_2 & -q_1 \\
q_1 & q_0 & q_1 & q_2 & q_3 & q_4 & 0 & -q_4 & -q_3 & -q_2 \\
q_2 & q_1 & q_0 & q_1 & q_2 & q_3 & q_4 & 0 & -q_4 & -q_3 \\
q_3 & q_2 & q_1 & q_0 & q_1 & q_2 & q_3 & q_4 & 0 & -q_4 \\
q_4 & q_3 & q_2 & q_1 & q_0 & q_1 & q_2 & q_3 & q_4 & 0 \\
- & - & - & - & - & - & - & - & - & - \\
0 & q_4 & q_3 & q_2 & q_1 & q_0 & q_1 & q_2 & q_3 & q_4 \\
-q_4 & 0 & q_4 & q_3 & q_2 & q_1 & q_0 & q_1 & q_2 & q_3 \\
-q_3 & -q_4 & 0 & q_4 & q_3 & q_2 & q_1 & q_0 & q_1 & q_2 \\
-q_2 & -q_3 & -q_4 & 0 & q_4 & q_3 & q_2 & q_1 & q_0 & q_1 \\
-q_1 & -q_2 & -q_3 & -q_4 & 0 & q_4 & q_3 & q_2 & q_1 & q_0
\end{array}\right) \tag{3.4}$$

be a negacyclic real symmetric Toeplitz matrix of order $m = 10$, partitioned into blocks of size $(n \times n)$, where $n = 5$. Note that the diagonal blocks are identical symmetric Toeplitz matrices and the off-diagonal blocks are *skew-symmetric* trace zero Toeplitz matrices which are negatives of one another. We denote the upper diagonal block as A and the lower off-diagonal block as B.

Recall that a Hermitian Toeplitz matrix C of order n is specified by its first row of elements, $\mathbf{c}^T = [c_0 c_1 \cdots c_{n-1}]$. To show that a real negacyclic matrix of order m may be represented as a Hermitian Toeplitz matrix of order n, we write the characteristic equation of (3.4) in the following form:

$$\begin{pmatrix} A & -B \\ B & A \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} = \lambda \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}, \tag{3.5}$$

where λ is an eigenvalue of Q and $[\mathbf{x}^T \mathbf{y}^T]^T$ is the corresponding eigenvector. It is clear that (3.5) is equivalent to the two characteristic equations resulting from the real and imaginary parts of

$$[A + jB](\mathbf{x} + j\mathbf{y}) = \lambda(\mathbf{x} + j\mathbf{y}). \tag{3.6}$$

Note that $C = A + jB$ is Hermitian Toeplitz and has the eigenvalue λ with $\mathbf{v} = \mathbf{x} + j\mathbf{y}$ as the associated eigenvector.

In summary, to construct a Hermitian Toeplitz matrix C with a given eigenspectrum, $\{\lambda_i : i = 0, 1, ..., n-1\}$, first compute q_k, $k = 0, 1, ..., n-1$, using (3.3), and then prescribe the elements of C by defining the elements of $\mathbf{c}$ as $c_i = q_i - jq_{n-i}$, for $i = 0, ..., n-1$. Note that permuting the given n eigenvalues produces many solutions to the inverse eigenvalue problem. In fact, there are $n!$ negacyclic matrices possible, and as many solutions, if the elements of the eigenspectrum are distinct.

3.4 Example

In this example, the problem is to construct a Hermitian Toeplitz matrix of order $n = 5$ having the eigenvalues $\lambda_0 = 1.0$, $\lambda_1 = 30.0$, $\lambda_2 = 50.0$, $\lambda_3 = 100.0$, and $\lambda_4 = 700.0$.

Using (3.3), we obtain the following elements q_k of the negacyclic matrix Q of order $m = 2n = 10$:

$$\begin{aligned} q_0 &= 176.20000000 & q_5 &= 0.00 \\ q_1 &= -141.18669451 & q_6 &= -q_4 \\ q_2 &= 95.38974075 & q_7 &= -q_3 \\ q_3 &= -68.85758704 & q_8 &= -q_2 \\ q_4 &= 32.28974075 & q_9 &= -q_1. \end{aligned}$$

The Hermitian Toeplitz matrix C, with elements written in terms of the q_k, is given by

$$C = \begin{pmatrix} q_0 & q_1 - jq_4 & q_2 - jq_3 & q_3 - jq_2 & q_4 - jq_1 \\ q_1 + jq_4 & q_0 & q_1 - jq_4 & q_2 - jq_3 & q_3 - jq_2 \\ q_2 + jq_3 & q_1 + jq_4 & q_0 & q_1 - jq_4 & q_2 - jq_3 \\ q_3 + jq_2 & q_2 + jq_3 & q_1 + jq_4 & q_0 & q_1 - jq_4 \\ q_4 + jq_1 & q_3 + jq_2 & q_2 + jq_3 & q_1 + jq_4 & q_0 \end{pmatrix}, \tag{3.7}$$

where we have used the Hermitian property, $c_{-i} = \bar{c}_i$, to fill in the elements below the main diagonal. The eigenvalues of C can now be found using one of the several numerical packages which are available, e.g., EISPACK. We chose to employ the modified method found in the next chapter which exclusively deals with Hermitian Toeplitz matrices. The eigenvalues found in this manner are

$$\begin{aligned} \lambda_0 &= 0.99999916 & \epsilon &= 8.4 \times 10^{-5} \\ \lambda_1 &= 30.00000508 & \epsilon &= 1.7 \times 10^{-5} \\ \lambda_2 &= 50.00000000 & \epsilon &= 0.0 \\ \lambda_3 &= 99.99998731 & \epsilon &= 1.2 \times 10^{-5} \\ \lambda_4 &= 699.99999319 & \epsilon &= 9.8 \times 10^{-5}, \end{aligned}$$

with the respective relative error, ϵ, also shown. The eigenvalues obtained are in excellent agreement with those found using EISPACK. As mentioned earlier, permutation of the originally specified 5 eigenvalues produces $5! = 120$ negacyclic matrices and Hermitian Toeplitz matrices. In this, and other examples, we have observed that some of the negacyclic matrices generated as a result of eigenvalue permutation will be related through a permutation of elements, while others generated will not and will have completely new element values.

3.5 Application to Array Signal Processing

The inverse eigenvalue problem for Hermitian Toeplitz matrices may find application to the area of array signal processing. As we have seen, the covariance matrix under the assumption of weakly stationary stochastic processes has a Hermitian Toeplitz structure.

Let the elements of the constructed Hermitian Toeplitz matrix be written as

$$\begin{aligned} c_k &= q_k - jq_{n-k} \\ &= \sum_{i=0}^{n-1} \lambda_i a_i e^{jkw_i} + \sum_{i=0}^{n-1} \lambda_i b_i e^{-jkw_i} \end{aligned} \tag{3.8}$$

where $a_i = 1 - (-1)^i$, $b_i = 1 + (-1)^i$, $w_i = \pi(2i+1)/2n$, and $\{\lambda_i\}$ is a given set of real numbers. Each of the two terms in (3.8) has the form given by Carathédory [1 ,p. 60], i.e.,

$$y_l = \sum_{k=1}^{P} \gamma_k e^{jlw_k} + \gamma_0 \delta_l \quad l = 1, 2, \cdots, N, \tag{3.9}$$

where the $(N+1)$ complex constants, $y_0, y_1, \cdots, y_N$, are not all zero and $\bar{y}_{-l} = y_l$. Under these conditions, there exists an integer P, $1 \leq P \leq N$, and certain real constants $\gamma_k > 0$ and ω_k for $k = 1, 2, \cdots, P$.

The correlation between the ith and jth sensor elements is,

$$r_{ij} = E[s_i(t)\bar{s}_j(t)] = \sum_{k=1}^{P} B_k e^{-j(d_i - d_j)\pi\cos\theta_k} + \sigma^2 \delta_{ij}, \tag{3.10}$$

where B_k represents the signal power of the kth source, d_i represent the distance between sensors, and θ_k represents the angle of incidence of the wave to the sensor elements. Comparing (3.10) and (3.9), Pillai [1], showed that $\{B_k\} \leftrightarrow \{\gamma_k\}$ and $\{\omega_k\} \leftrightarrow \{\pi\cos\theta_k\}$ and that the analogy is exact if the Q array elements are located in a way such that the differences $d_i - d_j = m$, $j \geq i$, for $i, j = 1, 2, \cdots, Q$ represent every integer in the set

$\{0, 1, 2, \cdots, N\}$, where $N \leq Q(Q-1)/2$. Then with Q array elements, there are $(N+1)$ autocorrelation lags

$$r(m) = r(j-i) = \sum_{k=1}^{P} B_k e^{jm\omega_k} + \sigma^2 \delta_m, \quad m = 1, 2, \cdots, N. \tag{3.11}$$

Constructing an analogy between (3.8) and (3.10), similar to that found in [1], the two terms in (3.8) suggest a certain array geometry (unknown) with $\{B_i\} \leftrightarrow \{\lambda_i\}$ and all the waves incident on the sensors have a precise angle such that $\{\omega_i\} \leftrightarrow \{\pi(2i+1)/2n\} \leftrightarrow \{\pi \cos\theta_i\}$ for the elements of the autocorrelation matrix be of the form shown in (3.8). We see that (3.8) may be thought of as two *shifted* linear arrays with the waves making unique angles to the sensors. This is a special case of a symmetric multipath environment [2 ,p. 288]. The inverse eigenvalue might be useful in a case in which some of the plane waves are the desired ones and the rest are interferers or jammers to be nulled. In this case assuming independence, the covariance matrix may be decomposed into a part due to the desired signals and a part due to the interference plus noise of the form

$$\mathcal{R} = \mathcal{R}_d + \mathcal{R}_n\,, \tag{3.12}$$

where $\mathcal{R}_d$ is due to the desired signals and $\mathcal{R}_n$ is due to the interference. In a special situation in which one knows the powers $\{B_i\} \leftrightarrow \{\lambda_i\}$ of the unwanted signals and assumes the interference is symmetric, it maybe possible to construct a matrix C_n which has a special structure designed to eliminate $\mathcal{R}_n$ and obtain the desired information from

$$\mathcal{R} = \mathcal{R}_d + (\mathcal{R}_n - C_n) \approx \mathcal{R}_d\,. \tag{3.13}$$

3.6 Discussion

A method was presented for solving the inverse eigenvalue problem for Hermitian Toeplitz matrices. The approach taken uses the fact that a Hermitian Toeplitz matrix of order n having the desired eigenspectrum can be constructed from the elements of a certain real symmetric negacyclic matrix of order $m = 2n$. The approach is computationally efficient and only requires an n-point DFT. Also note that using the unitary transform of the previous chapter on the constructed Hermitian Toeplitz matrix produces a solution to the inverse eigenvalue problem for real symmetric matrices.

The inverse eigenvalue problem for Hermitian Toeplitz matrices is relatively elementary since there are twice as many specifiable parameters in a Hermitian Toeplitz matrix as there are in a real symmetric Toeplitz matrix. An important and much more difficult problem is the inverse eigenvalue problem for real symmetric Toeplitz matrices. This problem remains unsolved for matrices of order n greater than 4 and a theoretical solution to it seems very challenging. However, numerical solutions for real symmetric Toeplitz matrices of any order n have been presented in [16, 17]. Using the unitary transform decribed on the constructed Hermitian Toeplitz matrix results in a real symmetric matrix, $S = T + H$. For example, for $n = 5$, S has the following form:

$$S = \begin{pmatrix} q_0 & q_1 & q_2 & q_3 & q_4 \\ q_1 & q_0 & q_1 & q_2 & q_3 \\ q_2 & q_1 & q_0 & q_1 & q_2 \\ q_3 & q_2 & q_1 & q_0 & q_1 \\ q_4 & q_3 & q_2 & q_1 & q_0 \end{pmatrix} + \begin{pmatrix} q_1 & q_2 & q_3 & q_4 & 0 \\ q_2 & q_3 & q_4 & 0 & -q_4 \\ q_3 & q_4 & 0 & -q_4 & -q_3 \\ q_4 & 0 & -q_4 & -q_3 & -q_2 \\ 0 & -q_4 & -q_3 & -q_2 & -q_1 \end{pmatrix}. \qquad (3.14)$$

Now, since the eigenvalues of $S = T + H$ and the elements q_i are known, then one would like to further study the eigenvalue relation between T, H, and $T+H$, and this may help in obtaining a better understanding of the existence question of whether a real Toeplitz matrix exists having arbitrary eigenvalues or not.

In array signal processing, the covariance matrix has a Hermitian Toeplitz structure under certain assumptions. It was shown that an analogy exists between (3.8) and (3.10) similar to that drawn by Pillai [1]. In comparing (3.8) and (3.10), we see that (3.8) may be thought of as two shifted linear arrays with the waves making unique angles to the sensors. It was explained in a special case of a symmetric multipath environment it is possible to eliminate the effect of the interference if their power are known.

Chapter 4

Recursive and Iterative Algorithms for Hermitian Toeplitz Marices

4.1 Introduction

In the previous chapter, the problem considered was the construction of a Hermitian Toeplitz matrix given an arbitrary set of real eigenvalues. In this chapter, we focus on the computation of the *complete* eigenspectrum for Hermitian Toeplitz and real Toeplitz matrices. In particular, the current trend is the investigation of methods which utilize not only the centrosymmetric structure, but also the Toeplitz structure in the design of new algorithms. We review some of the current approaches and algorithms available in the literature and see that these algorithms fall into two categories, order recursive and iterative. The order recursive algorithms of Wilkes and Hayes [30] and Morgera and Noor [31] are of interest since they demonstrate that the eigenvalues of an n-dimensional real symmetric or Hermitian Toeplitz matrix C_n may be obtained from the eigenvalues of its submatrices. Even though these algorithms suffer from certain numerical problems, the approaches, nevertheless, contain new results of some theoretical interest. Iterative methods, however, are more numerically stable than order recursive methods; this is principally due to the fact that characteristic polynomials are not formed, a computation which is historically known to increase the propagation of roundoff errors.

Work on iterative methods to determine the *smallest* eigenvalue of Toeplitz matrices has been reported by Cybenko and Loan [27] and Hu and Kung [26]. Recently, Trench [32] has proposed a method which represents an extension of [27] to determining the *complete* eigenspectrum of Hermitian Toeplitz matrices. If all the eigenvalues of a Hermitian Toeplitz matrix are required, then the standard procedures (which do not exploit the Toeplitz structure) given in [18, 19] are more efficient; however, if only a few are required, then the methods given in [26, 27, 32] are more efficient.

The chapter is organized as follows. Section 2 presents the mathematical development of the order recursive algorithms and provides an example of this category of algorithms. Section 3 presents an example and discussion of the order recursive algorithm. Section 4 is devoted to Trench's iterative approach and its modified version. The modifications to Trench's algorithm involve maintaining tighter lower and upper bound intervals for each eigenvalue during the search mode, and inclusion of the case of multiple eigenvalues. Simulation results are reported for Trench's method using the Pegasus method as a major root searching method, and the Modified method with three choices of root searching technique, namely, Pegasus, Modified Rayleigh Quotient Iteration with Bisection iterations (MRQI-B), and Modified Rayleigh Quotient Iteration with Pegasus iterations (MRQI-P). Extensive computer simulations are performed on constructed Hermitian Toeplitz matrices of orders 50, 100, 200, and 500. The modifications proposed have important consequences for efficiency when working with high order matrices. Section 5 provides some examples of the simulation results. Finally, in Section 6 we present an application of the algorithms to Pisarenko's harmonic decomposition.

4.2 Mathematical Development

The problem may be stated as follows: given a Hermitian Toeplitz matrix C_n of order n,

$$C_n = \begin{pmatrix} c_0 & \bar{c}_1 & \cdots & \bar{c}_{n-1} \\ c_1 & c_0 & \cdots & \bar{c}_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n-1} & c_{n-2} & \cdots & c_0 \end{pmatrix}, \tag{4.1}$$

where c_0 is real and $c_1, c_2, \ldots, c_{n-1}$ are complex, find the complete eigenspectrum. Since C_n is Hermitian, $\bar{c}_{-i} = c_i$, for $i = 0, 1, \ldots n-1$. The principal submatrix of C_n of order k is defined as $C_k = [c_{i-j} : 0 \leq i, j \leq k-1]$, for $k = 1, 2, \ldots, n$. Assuming C_k to be nonsingular, we may apply Levinson's recursion in order to obtain a set of reflection coefficients $\{\rho_k\}$ and a set of linear prediction coefficients $\{\phi_{ki}\}$. Now, let us consider the *shifted* system of normal equations,

$$(C_n - \lambda I_n)\Phi_n(\lambda) = [E_n(\lambda), 0, \ldots, 0]^T, \tag{4.2}$$

where

$$\Phi_n(\lambda) = \begin{pmatrix} 1 \\ \Phi_{n-1}(\lambda) \end{pmatrix}$$

The quantities $\Phi_n(\lambda)$ and $E_n(\lambda)$ are the *predictor vector* and the prediction error at the nth recursive step, recursively. The elements of $C_n - \lambda I_n$ are the same as those of C_n except that the main diagonal of C_n is replaced by $c_0 - \lambda$, where λ is treated as a continuous real variable. Levinson's recursion can be applied to $(C_{n-1} - \lambda I_{n-1})\Phi_{n-1} = [\bar{c}_1 \cdots \bar{c}_{n-1}]^T$ and is given by

$$\rho_k = -\frac{c_k + \sum_{i=1}^{k-1} \phi_{k-1,i} c_{k-i}}{D_k / D_{k-1}}, \qquad k = 1, 2, \ldots, n-1, \tag{4.3}$$

$$\phi_{kk} = \rho_k, \tag{4.4}$$

$$\phi_{ki} = \phi_{k-1,i} + \rho_k \bar{\phi}_{k-1,k-i}, \tag{4.5}$$

$$E_k = (1 - |\rho_k|^2) E_{k-1}, \tag{4.6}$$

$$E_{k-1} = D_k/D_{k-1}, \tag{4.7}$$

where the above quantities will all depend on λ.

In the sequel ((4.17)), will show that the reflection coefficient in terms of λ may be written as

$$\rho_{n-1}(\lambda) = \frac{\gamma_0\lambda^{n-2} + \gamma_1\lambda^{n-3} + \cdots + \gamma_{n-2}}{D_{n-1}(\lambda)} = \frac{N_{n-1}(\lambda)}{D_{n-1}(\lambda)}, \tag{4.8}$$

where $D_{n-1}(\lambda)$ is the characteristic polynomial of C_{n-1} and $\gamma_0, \gamma_1, \cdots, \gamma_{n-2}$ are *complex* coefficients. The values of λ for which $D_{n-1}(\lambda)$ equals zero are the eigenvalues of C_{n-1} and, at these values of λ, $|\rho_{n-1}(\lambda)|$ becomes infinite. Note as $|\rho_{n-2}(\lambda)|$ approaches unity, E_{n-2} approaches zero, which means that $|\rho_{n-1}(\lambda)|$ becomes infinite [30]. This can be verified by use of (4.7) and (4.17) and is left to the reader.

Setting $|\rho_{n-1}(\lambda)|^2 = 1$ in (4.8) and forming $P_{n-1}(\lambda) = D^2_{n-1}(\lambda) - |N_{n-1}(\lambda)|^2 = 0$ implies that there are $2n-2$ values of λ for which the resulting polynomial is zero. Out of the $2n-2$ values of λ, n values correspond to the eigenvalues of the matrix C_n, because at these values $D_n(\lambda)$ is zero and $|\rho_{n-1}(\lambda)| = 1$. The remaining $n-2$ values of λ at which $|\rho_{n-1}(\lambda)|^2 = 1$ correspond to the eigenvalues of the principal submatrix C_{n-2} and are denoted by μ_i, $i = 1, 2, \cdots, n-2$. At these eigenvalues, $C_{n-2} - \mu_i I$ will be singular, but $C_{n-1} - \mu_i I$ and $C_n - \mu_i I$ will be nonsingular. This is known as the *singular case* , for which the conventional formulation of Levinson's algorithm does not apply [35].

In the *singular case*, the reflection coefficients are related by

$$\rho_{n-1}(\lambda) = -\frac{\bar{\beta}_0(\lambda)}{\beta_0(\lambda)}\rho_r(\lambda), \tag{4.9}$$

where $r = n - 1 - 2l$ and is referred to as a *left-singular point*. The quantity $\beta_0(\lambda)$ is given by

$$\beta_0(\lambda) = \bar{c}_l\phi_{r0}(\lambda) + \bar{c}_{l+1}\phi_{r1}(\lambda) + \cdots + \bar{c}_{l+r}\phi_{rr}(\lambda), \tag{4.10}$$

where l is the Iohvidov index at point r [35]. Note that $\beta_0(\lambda)$ depends on the predictor vector and has a complex value in the Hermitian case. In the real symmetric case, $\beta_0(\lambda)$ is real, $l = 1$, and (4.9) reduces to the expression found in [30], i.e.,

$$\rho_{n-1}(\lambda) = -\rho_{n-3}(\lambda) \tag{4.11}$$

with the property that $\rho_k = \pm 1$ at the eigenvalues of the $(k-1)$ and $(k+1)$ order principal submatrices. The case of real symmetric Toeplitz matrices has been treated in [30], and we now present the order recursive algorithm for real symmetric Toeplitz matrices in pseudo-code form.

Order Recursive Algorithm - Real Symmetric Toeplitz matrices [30].

Step 1: (Initialization)

Given eigenvalues of submatrices C_{n-1} and C_{n-2} and values of reflection coefficients $\rho_{n-3}(\lambda)$ at eigenvalues of C_{n-2}.

Step 2: (Calculate reflection coefficient values)

Find the values of $\rho_{n-1}(\lambda)$ at the eigenvalues of C_{n-2} from the relation (4.9).

Step 3: (Solve)

$$\begin{pmatrix} \mu_1^{n-3} & \mu_1^{n-4} & \dots & 1 \\ \mu_2^{n-3} & \mu_2^{n-4} & \dots & 1 \\ \vdots & \vdots & \dots & \vdots \\ \mu_{n-2}^{n-3} & \mu_{n-2}^{n-4} & \dots & 1 \end{pmatrix} \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_{n-2} \end{pmatrix}$$

$$= \begin{pmatrix} \rho_{n-1}(\mu_1)D_{n-1}(\mu_1) - c_{n-1}\mu_1^{n-2} \\ \rho_{n-1}(\mu_2)D_{n-1}(\mu_2) - c_{n-1}\mu_2^{n-2} \\ \vdots \\ \rho_{n-1}(\mu_{n-2})D_{n-1}(\mu_{n-2}) - c_{n-1}\mu_{n-2}^{n-2} \end{pmatrix}$$

for γ_i, $i = 1, 2, ..., n-2$. Note that the above Vandermonde matrix can be efficiently inverted in $O(n^2)$ operations [18, 30]. The quantity $D_{n-1}(\lambda)$ is the characteristic equation of C_{n-1} and may be computed from the eigenvalues of C_{n-1}.

Step 4: (Form the two polynomials)

$$D_{n-1}(\lambda) \pm [c_{n-2}\lambda^{n-2} + \gamma_1\lambda^{n-3} + \ldots + \gamma_{n-2}] = 0.$$

Note, when the numerator of (4.17) is expanded $\gamma_0 = c_{n-2}$.

Step 5: (Obtain eigenvalues)

Deflate the polynomials by eigenvalues of C_{n-2}; the remaining eigenvalues will be those of C_n.

In the case of a Hermitian Toeplitz matrix, we use Levinson's algorithm to evaluate $\rho_k(\lambda)$. From (4.5), the predictor coefficients in terms of λ are

$$\begin{aligned}\phi_{k,i}(\lambda) &= \phi_{k-1,i}(\lambda) + \rho_k(\lambda)\bar{\phi}_{k-1,k-i}(\lambda) \\ &= \frac{N_{k-1,i}(\lambda)}{D_{k-1}(\lambda)} + \frac{N_k(\lambda)}{D_k(\lambda)}\frac{\bar{N}_{k-1,k-i}(\lambda)}{D_{k-1}(\lambda)},\end{aligned} \tag{4.12}$$

where N (with appropriate subscripting) is used to denote the numerator part of each component. The above equation may be written as

$$\phi_{k,i}(\lambda) = \frac{M_{k,i}(\lambda)}{D_{k-1}(\lambda)D_k(\lambda)}, \tag{4.13}$$

where

$$M_{k,i}(\lambda) = N_{k-1,i}(\lambda)D_k(\lambda) + N_k(\lambda)\bar{N}_{k-1,k-i}(\lambda). \tag{4.14}$$

The proof is given in the appendix. The numerator $M_{k,i}(\lambda)$ of (4.13) is divisible by $D_{k-1}(\lambda)$; therefore, (4.13) reduces to,

$$\phi_{k,i}(\lambda) = \frac{N_{k,i}(\lambda)}{D_k(\lambda)}. \tag{4.15}$$

Substituting (4.15) into (4.3) we obtain

$$\begin{aligned}\rho_k(\lambda) &= -\frac{D_{k-1}(\lambda)[c_k + \sum_{i=1}^{k-1}(N_{k-1,i}(\lambda)/D_{k-1}(\lambda))c_{k-i}]}{D_k(\lambda)} \\ &= -\frac{D_{k-1}(\lambda)c_k + \sum_{i=1}^{k-1} N_{k-1,i}(\lambda)c_{k-i}}{D_k(\lambda)}.\end{aligned} \tag{4.16}$$

Now, $\rho_k(\lambda)$ may be expressed as

$$\rho_k(\lambda) = (-1)^k \frac{\begin{vmatrix} c_1 & c_2 & \cdots & c_k \\ c_0-\lambda & c_1 & \cdots & c_{k-1} \\ \vdots & \vdots & \ddots & \vdots \\ \bar{c}_{k-2} & \bar{c}_{k-3} & \cdots & c_1 \end{vmatrix}}{\begin{vmatrix} c_0-\lambda & c_1 & \cdots & c_{k-1} \\ \bar{c}_1 & c_0-\lambda & \cdots & c_{k-2} \\ \vdots & \vdots & \ddots & \vdots \\ \bar{c}_{k-1} & \bar{c}_{k-2} & \cdots & c_0-\lambda \end{vmatrix}}, \qquad k = 1, 2, \ldots, n-1. \tag{4.17}$$

We see that $D_{k-1}(\lambda)$ is the minor of c_k and $N_{k-1,i}$ is the minor of c_{k-i}. In other words, the numerator of $\rho_k(\lambda)$ is the determinant expanded by the kth column. Note that the predictor coefficients are evaluated at the kth step of Levinson's algorithm. It turns out that the numerator of the predictor coefficients are the minors needed to evaluate the numerator of the reflection coefficient at step $k+1$. Once the reflection coefficient has been evaluated at the kth iteration, its magnitude squared is set equal to unity. The polynomial obtained is then deflated by the eigenvalues of D_{k-1} and reduces to D_{k+1}, the characteristic equation of the next larger principal submatrix. The eigenvalues are then determined. We now present the order recursive algorithm for Hermitian Toeplitz matrices in pseudo-code form.

Order Recursive Algorithm - Hermitian Toeplitz matrices

Step 1: (Initialization)

$$\begin{aligned} D_0(\lambda) &= 1 \\ D_1(\lambda) &= c_0 - \lambda \end{aligned}$$

For $k = 1, 2, \cdots, n-1$ DO

Step 2: (Calculate Numerator of Reflection Coefficient)

$$\begin{aligned} N_k(\lambda) &= -[D_{k-1}(\lambda)c_k + \sum_{i=1}^{k-1} N_{k-1,i}(\lambda)c_{k-i}] \\ N_{kk} &= N_k \end{aligned}$$

Step 3: (Set the magnitude squared of reflection coefficient to unity)

$$|\rho_k(\lambda)|^2 = |N_k(\lambda)/D_k(\lambda)|^2 = 1$$

to form

$$P_k(\lambda) = D_k^2(\lambda) - |N_k(\lambda)|^2 = 0$$

Step 4: (Deflate $P_k(\lambda)$)

This is a polynomial of degree $2k$. It is deflated by the the eigenvalues of $D_{k-1}(\lambda)$, the characteristic equation of C_{k-1}, and reduces to the characteristic equation, $D_{k+1}(\lambda)$, of the next larger principal submatrix.

Step 5: (Find the roots of $D_{k+1}(\lambda)$)

Due to the fact that the eigenvalues found from D_{k-1} and D_k interlace, or form a Sturmian chain [34], the bisection method is used here to find the roots of D_{k+1}. Other methods are possible.

Step 6: For $i = 1, 2, \cdots, k-1$ DO

Step 7: (Calculate Numerator of Predictor Coefficient)

$$M_{k,i}(\lambda) = N_{k-1,i}(\lambda)D_k(\lambda) + N_k(\lambda)\bar{N}_{k-1,k-i}(\lambda)$$

Step 8: (Deflate $M_{k,i}(\lambda)$ by eigenvalues of C_{n-1} to obtain $N_{k,i}(\lambda)$)

Store $N_{k,i}(\lambda)$.

If $i \leq k-1$ GO TO Step 6; Else, if $k \leq n-1$ GO TO Step 2; OTHERWISE EXIT.

Note there is a difference between the formulation of the polynomials at Step 4 of the order recursive algorithm for the real symmetric Toeplitz matrices case and Step 3 for the Hermitian Toeplitz case. The main difference is at Step 3 for the Hermitian Toeplitz case the polynomial is formed by setting the magnitude squared of the reflection coefficient to unity whereas in the real symmetric Toeplitz this is not the case.

4.3 Example and Discussion - Order Recursive Algorithms

We are given a Hermitian Toeplitz matrix of order $n = 8$ specified by its first row,

$$\mathbf{c_1}^T = [(10-\lambda),\ (5+j2),\ (4+j3),\ (2+j),\ (2+j3),\ (2+j2),\ (1+j2),\ (1+j)].$$

The recursion given above for Hermitian Toeplitz matrices is illustrated for $k = 2$. At **Step 2** of the recursion the numerator of the reflection coefficient is

$$\begin{aligned} N_2(\lambda) &= -[D_1(\lambda)c_2 + N_{1,1}(\lambda)c_1] \\ &= (4+j3)\lambda - (19+j10) \end{aligned}$$

and the numerator of $\phi_{22}(\lambda)$ is

$$N_{22}(\lambda) = N_2(\lambda).$$

Step 3: The magnitude squared of the reflection coefficient is set equal to unity, i.e.,

$$\begin{aligned}|\rho_2(\lambda)|^2 &= |N_2(\lambda)/D_2(\lambda)|^2 = 1 \\ &= \left|\frac{(4+j3)\lambda - (19+j10)}{\lambda^2 - 20\lambda + 71}\right|^2 = 1 \,.\end{aligned}$$

Squaring the denominator and numerator and subtracting, $P_2(\lambda)$ is obtained as

$$\begin{aligned}P_2(\lambda) &= D_2^2(\lambda) - |N_2(\lambda)|^2 = 0 \\ &= \lambda^4 - 40\lambda^3 + 517\lambda^2 - 2628\lambda + 4580 = 0 \,.\end{aligned}$$

Step 4: $P_2(\lambda)$ is deflated by the eigenvalue of C_1, which is 10, thereby reducing $P_2(\lambda)$ to the characteristic equation of C_3, i.e.,

$$D_3(\lambda) = \lambda^3 - 30\lambda^2 + 217\lambda - 458 \,.$$

Step 5: Using the bisection method, the eigenvalues are found to be

$$\begin{aligned}\lambda_1 &= 4.30683, \\ \lambda_2 &= 5.18595, \\ \lambda_3 &= 20.50756 \,.\end{aligned}$$

Steps 6 through 8 are performed to calculate the numerator of the predictor coefficient $\phi_{2,1}$

$$\begin{aligned}M_{2,1}(\lambda) &= N_{1,1}(\lambda)D_2(\lambda) + N_2(\lambda)\bar{N}_{1,1}(\lambda) \\ &= (5+j2)\lambda^2 - (74+j33)\lambda + (240+j130) \,.\end{aligned}$$

Deflating $M_{2,1}(\lambda)$ by 10, the eigenvalue of C_1, the quantity $N_{2,1}$ is obtained as

$$N_{2,1} = (5+j2)\lambda - (24+j13).$$

Table 4.1: Comparison between Eigenvalues.

Order of C	Order recursive algorithm	IMSL subroutine eigch
2	4.61483	4.61483
	15.38516	15.38516
3	4.30648	4.30647
	5.18595	5.18596
	20.50756	20.50756
4	2.86977	2.86975
	4.64324	4.64327
	8.29042	8.29040
	24.19656	24.19656
5	2.13382	2.13379
	3.74269	3.74274
	6.69760	6.69758
	9.65112	9.65111
	27.77475	27.77475

Table 4.1 cont'd: Comparison between Eigenvalues.

Order of C	Order recursive algorithm	IMSL subroutine eigch
6	1.80042	1.80037
	2.80100	2.80105
	5.86714	5.86717
	7.13220	7.13215
	11.38192	11.38191
	31.01730	31.01730
7	1.44659	1.44651
	2.25721	2.25731
	4.73446	4.73442
	6.74697	6.74706
	8.18895	8.18887
	12.61631	12.61631
	34.00949	34.00949
8	1.04472	1.04268
	2.14350	2.15049
	4.10511	4.08309
	5.02928	5.05140
	7.69648	7.68807
	9.24493	9.24837
	14.13424	14.13441
	36.60173	36.60173

The above procedure is repeated and results in the eigenvalues tabulated in Table 4.1. The first column of the table indicates the order of the matrix C_n and the second column shows the eigenvalues obtained by the algorithm of Section 2 for Hermitian Toeplitz matrices. The stopping tolerance employed in the bisection method is an eigenvalue precision of six digits. The eigenvalues shown in column three of the table are obtained from the IMSL subroutine EIGCH. The IMSL routine EIGCH, although not designed specifically for Hermitian Toeplitz matrices, is used as a benchmark for the comparison. Comparing the second and third columns, it is observed that the accuracy of the eigenvalues obtained by the order recursive algorithm are accurate to three digits until the order of C_n reaches seven, with the eigenvalues obtained for C_n of order eight no longer accurate to three digits. The reason for the loss of accuracy is due to the stopping tolerance of six digits employed in the bisection method and the propagation of roundoff errors inherent in the order recursive approach. In the next section, we present Trench's method and the modified Trench's method, both of which do not suffer from such numerical problems.

4.4 Trench's Method and Its Modified Version

Trench's method uses the Levinson-Durbin (L-D) algorithm for the shifted matrices $C_k - \lambda I_k$, $k = 1, 2, ..., n-1$, within an iterative root finding procedure to find the zeroes of the rational function (4.7). For details, the reader is referred to [32]; however, Trench's method basically relies on two key theorems which are consequences of Sylvester's law of inertia and the Cauchy theorem. For completeness, we state the two key theorems below; proofs may be found in [32].

Theorem 1. *If $C - \lambda I = LDU$ is the triangular factorization, then $Neg_m(\lambda)$, the number of negative elements $E_i(\lambda)$ in $D = diag\{E_m(\lambda), E_{m-1}(\lambda), \cdots, E_1(\lambda)\}$, equals the number of eigenvalues λ_i of C that are less than λ, provided λ is nondefective with respect to C_n. (A real number λ is nondefective with respect to C_n if it is not an eigenvalue of any of* *the principal submatrices C_k , $k = 1, 2, ..., n-1$).*

Theorem 2. *Assume that the real numbers α and β are nondefective with respect to C_n and that the interval (α,β) contains exactly one eigenvalue (with multiplicity one) of C_n. Also assume that neither α nor β is an eigenvalue of C_n. Then the interval (α,β) contains no eigenvalues of C_{n-1} if and only if $E_n(\alpha) > 0$ and $E_n(\beta) < 0$.*

From the above theorems, Trench's algorithm for finding the complete eigenspectrum of Hermitian Toeplitz matrices may be outlined as follows:

Trench's Algorithm - Hermitian Toeplitz matrices

Step 1-Select: Find the eigenvalues $\lambda_p, \lambda_{p+1}, \ldots, \lambda_q$, $1 \leq p < q \leq n$. Using trial and error, select an interval (a,b) by the bisection method such that $Neg_n(a) \leq p-1$ and $Neg_n(b) \geq q$.

FOR $i = p$ **TO** $q-1$

Step 2-Search: Search for the endpoint ξ_i not captured by trial and error such that the interval (ξ_{i-1}, ξ_i) will contain λ_i. This is again done by the bisection method and by keeping count of the negative signs of $\{E_1(\xi_i), E_2(\xi_i), \ldots, E_n(\xi_i)\}$. During this search process, keep capturing and storing the locations of other desired eigenvalues, while also retaining the values $E_n(\xi_i)$.

Step 3-Refine: Once the interval $\xi_{i-1} < \lambda_i < \xi_i$, is obtained:

(a) Set $\alpha = \xi_{i-1}$, $E_\alpha = E_n(\xi_{i-1})$ and $\beta = \xi_i$, $E_\beta = E_n(\xi_i)$.

(b) By trial and error, refine the interval (α,β) to (α',β') using bisection such that the following conditions both hold:

(i) $Neg_n(\alpha') = i-1$ and $Neg_n(\beta') = i$

(ii) $E_n(\alpha') > 0$ and $E_n(\beta') < 0$.

(c) Having refined the interval (α, β) to (α', β') by the bisection method in **Step 3(b)** above, switch to the Pegasus method to find λ_i.

NEXT i

END

Note that in the above algorithm, the L-D recursion is called for *each* iteration of the bisection and Pegasus methods. In Step 3, condition (i) by Theorem 1 assures that the chosen interval does not contain other eigenvalues of C_n. Condition (ii) by Theorem 2 assures that the refined interval (α', β') does not contain eigenvalues of C_{n-1}. The Pegasus method is a modification of the *Regula Falsi* method and is a more efficient zero finding method having an improved order of convergence [36, 37].

The first-level modifications we propose to Trench's method are to form tighter $L\xi_i$ (lower) and $U\xi_i$ (upper) bound intervals for each λ_i in the select and search steps and to extend the method to include the case of multiple eigenvalues. The modified algorithm is outlined as follows:

Modified Algorithm - Hermitian Toeplitz matrices

Step 1-Select: Find the eigenvalues $\lambda_p, \lambda_{p+1}, \ldots, \lambda_q$, $1 \leq p < q \leq n$. Using trial and error, select an interval (a,b) by the bisection method such that $Neg_n(a) \leq p-1$ and $Neg_n(b) \geq q$.

FOR $i = p$ **TO** $q-1$

Step 2-Search: Search for the endpoint $U\xi_i$ not captured by trial and error such that the interval ($L\xi_i, U\xi_i$) will contain λ_i. This is again done by the bisection method and by keeping count of the negative signs of $\{E_1(U\xi_i)$, $E_2(U\xi_i)$, ..., $E_n(U\xi_i)\}$. During this search process, keep tightening, capturing and storing the locations of other desired eigenvalues, while also retaining the values $E_n(L\xi_i)$, $E_n(U\xi_i)$, and $E_n(L\xi_{i+1})$. In the process, also detect, if any, the multiplicity m of multiple eigenvalues; (**IF** $|L\xi_i - U\xi_i| < Tol$ **Then** flagmultiple = true), where the value of Tol is 10^{-3}.

NEXT i

Step 3-Refine: Once all the intervals $L\xi_i < \lambda_i < U\xi_i$, $p \leq i \leq q$, are obtained:

FOR $j = p$ **TO** q

(a) Set $\alpha = L\xi_j$, $E_\alpha = E_n(L\xi_j)$ and $\beta = U\xi_j$, $E_\beta = E_n(U\xi_j)$.

(b) In case of multiple eigenvalues, set the matrix order n to $n-m+1$ and work with the submatrix C_{n-m+1}. By trial and error, refine the interval (α,β) to (α',β') using bisection such that the following conditions both hold:

(i) $Neg_n(\alpha') = j - 1$ and $Neg_n(\beta') = j$

(ii) $E_n(\alpha') > 0$ and $E_n(\beta') < 0$.

(c) Having refined the interval (α, β) to (α', β') by the bisection method in **Step 3(b)** above, switch to the MRQI-B or MRQI-P root finders to find λ_j.

NEXT j

END

Note that in the above modified algorithm, the L-D recursion is called for each iteration of the bisection shift and the Levinson recursion is called for each iteration of the MRQI-B or MRQI-P methods.

We discuss the former modification first. Assume that an interval (a, b) is given which encloses the eigenvalues $\lambda_p, \lambda_{p+1}, \ldots, \lambda_q$, and that we wish to find the intermediate points $\xi_p, \xi_{p+1}, \ldots, \xi_{q-1}$. As in Trench's procedure, we use the bisection method, $\gamma = (L\xi_r + U\xi_s)/2$, where r and s are integers such that $p \leq r < s \leq q$. The objective is to find $U\xi_r$. In the process of finding $U\xi_r$, other endpoints may be captured, e.g., γ_l, γ_j, γ_3, and γ_1, as shown in Figure 4.1. In Trench's search process for finding the intermediate points $\xi_p, \xi_{p+1}, \ldots, \xi_{q-1}$, an unnecessarily large number of calls to the L-D algorithm may result if we just let $\xi_k = \gamma$, for $r - 1 \leq k \leq s$. In the modified method, by using $L\xi_i$ and $U\xi_i$ for each λ_i, unnecessary calls to the L-D algorithm are reduced by storing the first selected γ_j as $L\xi_{k+1}$ and storing the last γ_l in $U\xi_k$, for $\gamma_l < \gamma_j$ and $k = Neg_n(\gamma_j) = Neg_n(\gamma_l)$, as depicted in Figure 4.1. Trench's method forms contigous intervals; as a result of this modification, ***noncontiguous*** intervals are formed for the bisection method.

Now, assume that the endpoints $\xi_p, \xi_{p+1}, \ldots, \xi_k$ have been found and that we wish to find ξ_{k+1}, indicated in Figure 4.1. Trench's method would use the interval (γ_l, γ_3) in the bisection method; whereas, the modified method would use the interval (γ_j, γ_3). Use of the tighter interval (γ_j, γ_3) would, in general, reduce the number of calls to the L-D algorithm. Although this modification may seem minor, it appears to have important consequences for efficiency when working with very high order matrices and,

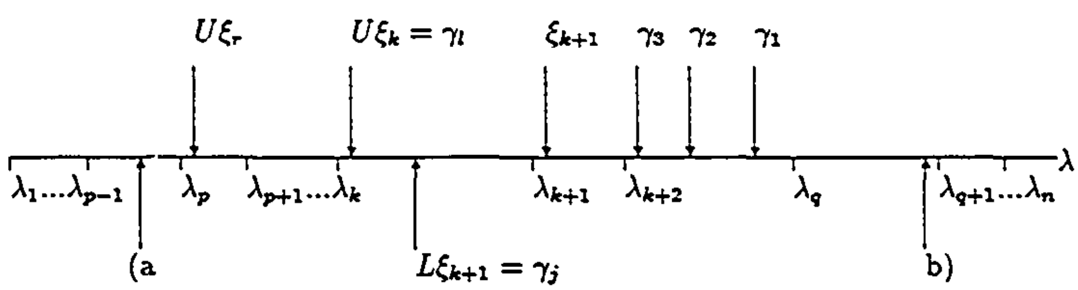

Figure 4.1: Interval (a,b) enclosing the desired eigenvalues $\lambda_p, \dots, \lambda_q$.

particularly so, when eigenvalues are not tightly clustered.

Next, in the multiple eigenvalue case, an eigenvalue λ_i with multiplicity m will have m linearly independent (nonunique) eigenvectors. Let the eigenvalues of C_{k-1} be $\gamma_j, j = 1, 2, \ldots, k-1$, and the eigenvalues of C_k be $\lambda_i, i = 1, 2, \ldots, k$. According to the Cauchy Interlace Theorem, the eigenvalues of C_{k-1} interlace those of C_k, i.e., $\lambda_1 \leq \gamma_1 \leq \lambda_2 \leq \gamma_2 \leq \cdots \leq \gamma_{k-1} \leq \lambda_k$. Cauchy's theorem implies that C_{k-1} must have an eigenvalue λ_i with multiplicity $m-1$, if C_k has an eigenvalue λ_i of multiplicity m.

An eigenvalue of C_n is obtained by varying λ; at the same time, there are $n(n-1)/2$ values (multiple values included) of λ for which the leading principal submatrices are singular. These values are the eigenvalues of submatrices for which, during the execution of the L-D algorithm, $|\rho_k(\lambda)|^2 = 1$ or $E_k(\lambda) = 0$, and for which the L-D algorithm will not proceed beyond this point. Now, in the multiple eigenvalue case, any interval (α, β) containing a multiple λ_i of C_n will certainly contain λ_i of C_{n-m+1} and condition (ii) in **Step 3** will not necessarily be true. Also, since $|\rho_k|^2 = 1$, the L-D algorithm will not proceed; this, is not, however, an obstacle to finding λ_i if the multiplicity m is known, because λ_i is then easily found by working with the submatrix C_{n-m+1}. In practice, true multiplicities are reflected as an eigenvalue *cluster*. The closeness of the eigenvalues in a cluster tends to cause all numerical procedures to lose efficiency, in the sense that considerable computational effort must be expended performing bisection shifts in search for eigenvalue interval endpoints. It is more appropriate to consider eigenvalues to be multiple when the condition, (**IF** $|L\xi_i - U\xi_i| < Tol$ **Then** flagmultiple = true), inserted in **Step 2** after the bisection shift, is satisfied. In our simulation studies, Tol was chosen to be 10^{-3}. Once the multiplicity m of λ_i is identified, then, according to Cauchy's Theorem, λ_i must also be an eigenvalue (with multiplicity one) of the principal submatrix C_{n-m+1}. Denote the eigenvector of C_{n-m+1} associated with λ_i by $\mathbf{q}_i$. Kung and Hu [41] have shown that the vector $\mathbf{q}_i$ suffices to characterize the m-dimensional subspace spanned by the eigenvectors $\mathbf{v}_{ij}$ of C_n associated with λ_i through the construction $\mathbf{v}_{ij} = \mathbf{Z}^{j-1}[\mathbf{q}_i^T 0 0 \cdots 0]^T$, where $\mathbf{Z}^{j-1}$ denotes a cyclic shift of $j-1$ elements, $j = 1, 2, \ldots, m$.

The second-level modification we consider is the use of the modified Rayleigh quotient iteration (MRQI) in place of the Pegasus method in the refine step. This modification is expected to improve convergence rate, as the MRQI has a cubic rate of convergence; whereas, the Pegasus method has a rate of convergence of 1.64 [36, 37]. The MRQI algorithm requires solution of the linear system of equations

$$(C - \mu_i I)\mathbf{y}_{i+1} = \mathbf{u}_i, \tag{4.18}$$

where μ_i is called the origin shift and $\mathbf{u}_i$ is a given normalized vector. The vector $\mathbf{y}_{i+1}$ may be solved for using the Levinson algorithm with $O(2n^2)$ complexity, or by parallel methods with a complexity of $O(n)$ with $O(n)$ processors [26, 40]. As μ_i approaches an eigenvalue λ_i, $\mathbf{y}_{i+1}$ approximates the associated eigenvector. The next origin shift μ_{i+1} is computed by the Rayleigh quotient,

$$\mu_{i+1} = \frac{\mathbf{y}_{i+1}^H C \mathbf{y}_{i+1}}{|\mathbf{y}_{i+1}|^2} = \frac{\mathbf{y}_{i+1}^H \mathbf{u}_i}{|\mathbf{y}_{i+1}|^2} + \mu_i, \tag{4.19}$$

where the superscript H denotes conjugate transpose.

In the event that the computed Rayleigh quotient falls outside the inclusion interval (α', β'), then a switch is made to the bisection method (note that we also report results obtained by replacing the bisection method by the Pegasus method). The Levinson-Durbin algorithm may also be used in combination with the Rayleigh quotient thereby leading to a quadratic rate of convergence [32].

4.5 Simulation Results

In this section, the performance of Trench's method using the Pegasus root finder and the Modified Trench's method simulated for three choices of root searching methods, namely, the Pegasus, the Modified Rayleigh Quotient Iteration with Bisection shifts (MRQI-B) and the MRQI-P (with Pegasus shifts) are presented. Moreover, we demonstrate the efficacy of the overall procedure in dealing with eigenvalue multiplicities.

First, we illustrate the modification of Trench's method, consider a Hermitian Toeplitz matrix of order $n = 10$ with the following first row of elements: [(50,0) (5,3) (1,3) (3,4)

Table 4.2: Results obtained by Trench's method.

(ξ_{i-1}, ξ_i)	$E_{10}(\xi_{i-1})E_{10}(\xi)$	(α', β')	No.it1	No.it2	λ_i
1.25-35.15	+ −	31.25-35.15	0	7	33.10
37.10-38.08	+ +	37.10-37.59	1	4	37.43
38.08-39.06	+ −	38.08-39.06	0	2	38.73
41.01-41.99	+ +	41.01-41.50	1	4	41.15
41.99-42.96	+ −	41.99-42.96	0	8	42.58
47.85-50.29 *	+ −	47.85-50.29	0	9	48.16
50.29-52.73 *	− −	50.90-51.51	2	4	51.27
52.73-62.50 *	− +	53.95-55.17	3	6	54.93
62.50-78.12	+ −	62.50-78.12	0	6	62.99
78.12-93.75	− −	85.93-93.75	1	7	89.60
No.it0=19	Total no. of iterations :	19 +	8 +	57	= 84

(1,1) (4,2) (4,9) (1,6) (3,4) (2,3)]. The interval ($a = 0, b = n \cdot c_0$) which contains all the eigenvalues, was chosen. Tables 4.2 and 4.3 show the number of iterations required by Trench's method and the modified method using noncontiguous intervals, respectively.

Table 4.2 shows the intervals (ξ_{i-1}, ξ_i) obtained by Trench's computer program and Table 4.3 shows the intervals $(L\xi_i, U\xi_i)$ obtained by the modified method. Note the different intervals indicated by the asterisks obtained by the two methods. The ± signs indicate whether the value of $E_{10}(\lambda)$ is either positive or negative. No.it0 corresponds to the total number of iterations required to obtain all the initial endpoints of the intervals. No.it1 corresponds to the number of iterations required to obtain the refined interval (α', β'). Note that no iterations are required to obtain (α', β') if conditions (i) and (ii) in **Step 3(b)** happen to be already satisfied. No.it2 corresponds to the number of iterations required to obtain λ_i by the root finder (Pegasus method). The stopping criteria for λ_i was $\mathbf{C_1}: |\zeta_j - \zeta_{j-1}| < .5(1.0 + \zeta_j)10^{-K}$ as in [32], where initially $\zeta_0 = \alpha'$, $\zeta_1 = \beta'$ and

Table 4.3: Results obtained by the Modified method.

$(L\xi_i, U\xi_i)$	$E_{10}(L\xi_i)E_{10}(U\xi_i)$	(α', β')	No.it1	No.it2	λ_i
31.25-35.15	+ −	31.25-35.15	0	7	33.10
37.10-38.08	+ +	37.10-37.59	1	4	37.43
38.08-39.06	+ −	38.08-39.06	0	2	38.73
41.01-41.99	+ +	41.01-41.50	1	4	41.15
41.99-42.96	+ −	41.99-42.96	0	8	42.58
46.87-50.78 *	+ +	46.87-48.82	1	3	48.16
50.78-54.68 *	+ +	50.78-52.73	1	4	51.27
54.68-62.50 *	+ +	54.68-58.59	1	4	54.93
62.50-78.12	+ −	62.50-78.12	0	6	62.99
78.12-93.75	− −	85.93-93.75	1	7	89.60
No.it0=18	Total no. of iterations : 18 + 6 + 49 = 73				

we chose $K = 6$ in our experiments. For this particular example, the total number of iterations required by the modified method was 73 and was 84 for Trench's method.

It was shown in Chapter 3, that a Hermitian Toeplitz matrix of order n may be constructed from a real symmetric negacyclic matrix of order $2n$. Using this relationship, we constructed Hermitian Toeplitz matrices of orders 50, 100, 200, and 500 using eigenvalue sets generated by **Pro-Matlab**'s random number generator. We then utilized the above algorithms to estimate the eigenvalues with results obtained shown in Tables 4.4, 4.5, and 4.6.

In these tables, Bi.it, Peg.it, and Ray.it denote the number of iterations required by the bisection, Pegasus, and Rayleigh quotient methods, respectively. Note that *two* termination criteria are used with the MRQI-B and MRQI-P root finders, $\mathbf{C}_1$ as above and $\mathbf{C}_2 : ||(C - \mu I)\mathbf{y}|| = 1/|\mathbf{y}| < 1000^{-2}$. The criterion $\mathbf{C}_1$ measures the accuracy of the eigenvalue estimate, while the criterion $\mathbf{C}_2$ measures the goodness of the eigenpair estimate $(\mu, \mathbf{y})$ as an approximation to the true eigenpair [42]. The MRQI-B and MRQI-P root finders terminate eigenvalue approximation if either of these conditions is satisfied. As a matter of interest, the average number of times, No.c1 and No.c2, that approximation is terminated based on criteria $\mathbf{C}_1$ and $\mathbf{C}_2$, respectively, is tabulated in Tables 4.5 and 4.6. In addition, the the average of the error $\epsilon = |\lambda_{exact} - \lambda_{approx.}|$ averaged over 100 trials per matrix order (i.e., for matrices of order 50, 100, 200, and 500) is shown in Figures 4.2 through 4.5 with the empirical mean (m) and standard deviation (std) of the average of the error shown for each method.

From Tables 4.5 and 4.6, we observe that use of the MRQI method in conjunction with Trench's procedure modified to utilize noncontiguous intervals results in an eigensolver having an improved convergence rate. From the Figures 4.2 through 4.5, we see that the MRQI-P procedure in some instances results in a slightly better accuracy for a reduced number of iterations than the MRQI-B procedure. Furthermore, from Tables 4.5 and 4.6, $\mathbf{C}_2$ is more often satisfied than $\mathbf{C}_1$, thereby indicating that the MRQI root finder delivers a good eigenpair estimate, rather than a good eigenvalue estimate. In Table 4.7, we summarize the performance of the algorithms in terms of the complexity, the convergence,

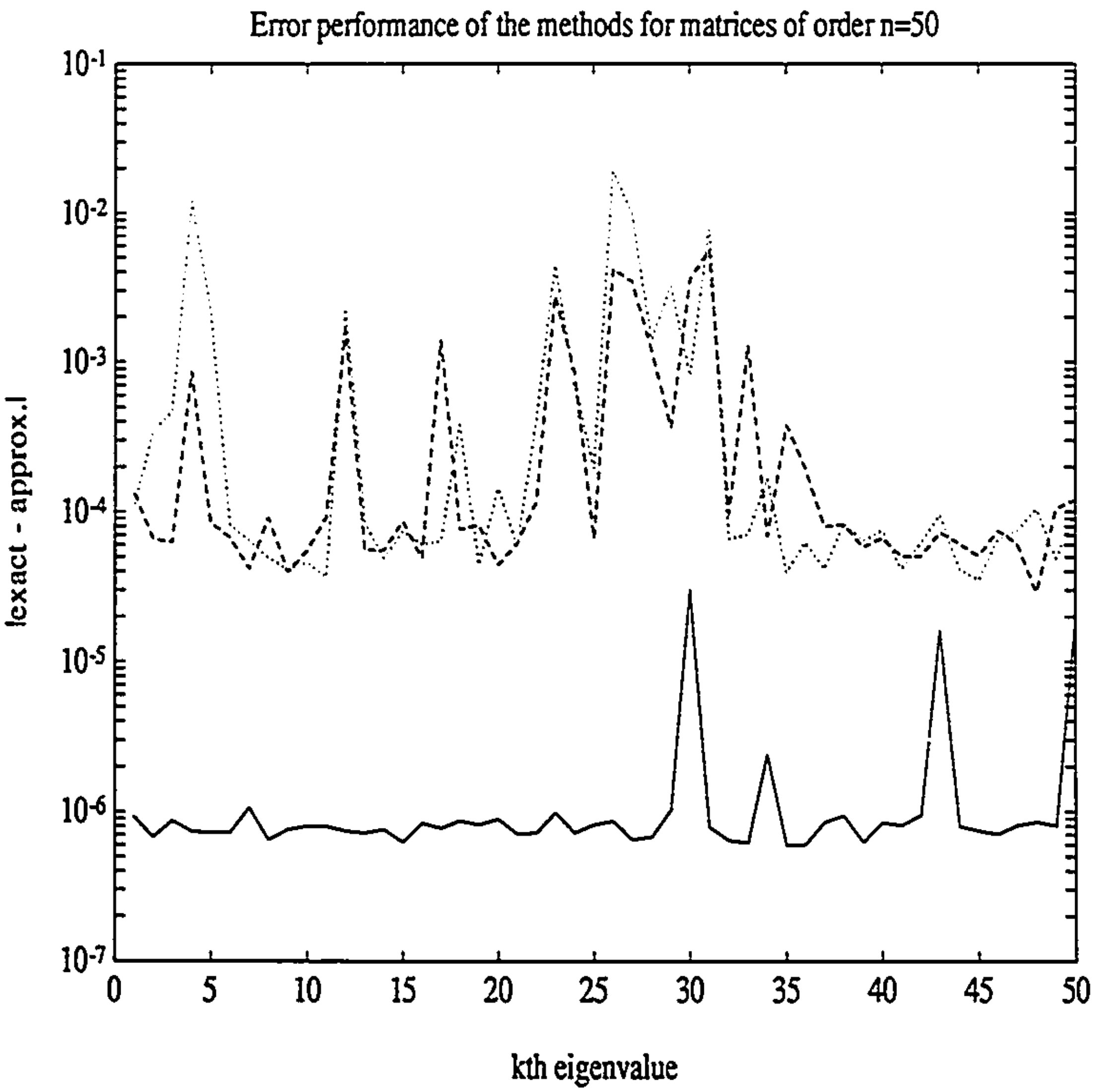

Figure 4.2: Average error averaged over 100 trials for matrices of order 50, Modified-Pegasus: solid m=2.056 $\times 10^{-06}$, std=5.209 $\times 10^{-06}$; Modified-MRQI-B : dashed m=6.070 $\times 10^{-04}$, std=1.219 $\times 10^{-03}$; Modified-MRQI-P : dotted m=1.358 $\times 10^{-03}$, std=3.548 $\times 10^{-03}$.

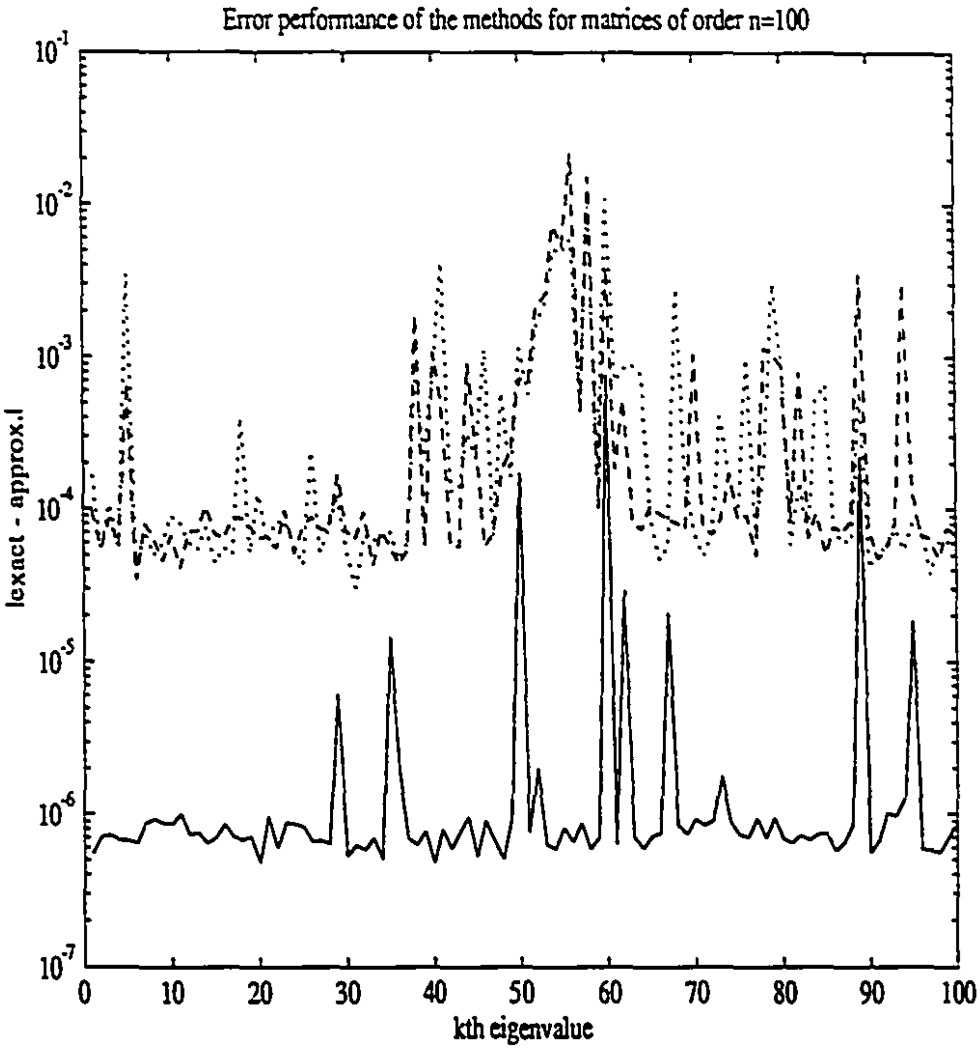

Figure 4.3: Average error averaged over 100 trials for matrices of order 100, Modified-Pegasus: solid m=1.308 $\times$ 10^{-05}, std=8.021 $\times$ 10^{-05}; Modified-MRQI-B : dashed m=8.342 $\times$ 10^{-04}, std=2.808 $\times$ 10^{-03}; Modified-MRQI-P : dotted m=7.323 $\times$ 10^{-04}, std=1.767 $\times$ 10^{-03}.

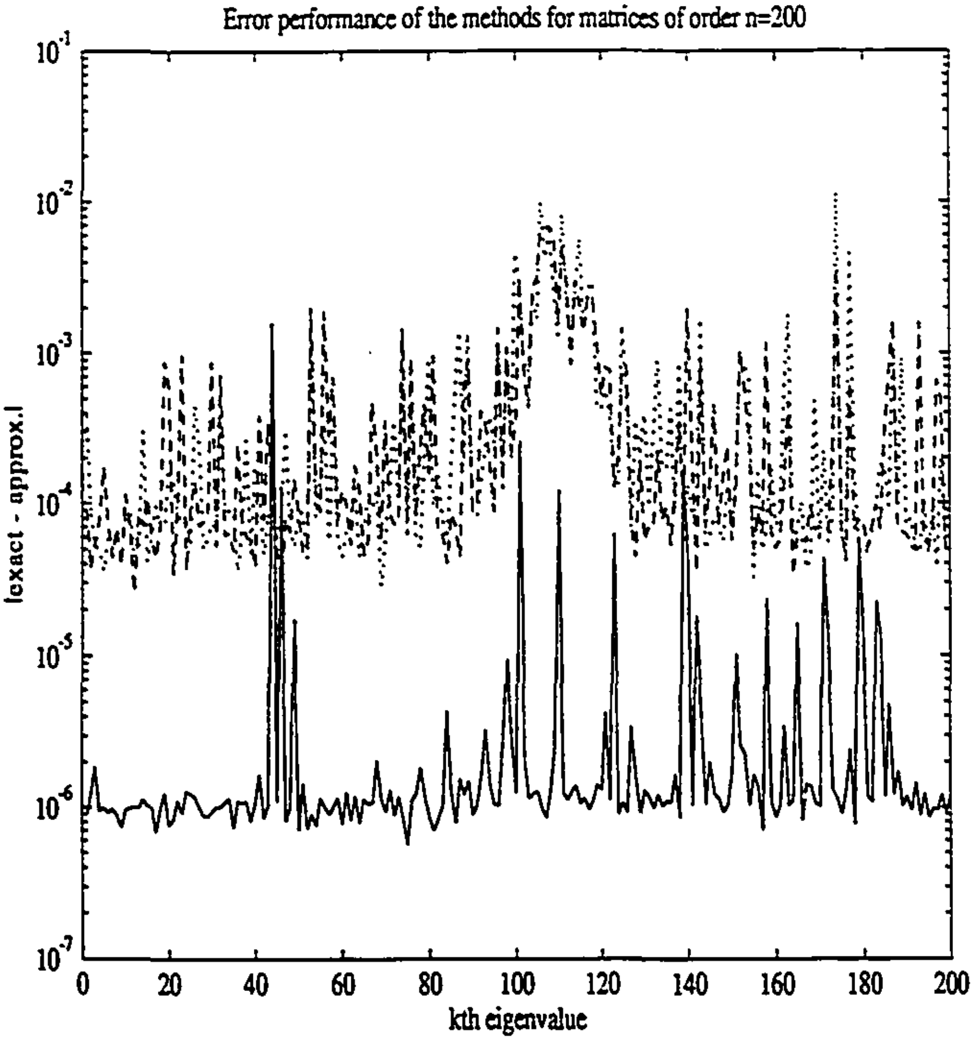

Figure 4.4: Average error averaged over 100 trials for matrices of order 200, Modified-Pegasus: solid m=1.428 $\times 10^{-05}$, std=1.152 $\times 10^{-04}$; Modified-MRQI-B : dashed m=5.479 $\times 10^{-04}$, std=1.114 $\times 10^{-03}$; Modified-MRQI-P : dotted m=6.055 $\times 10^{-04}$, std=1.484 $\times 10^{-03}$.

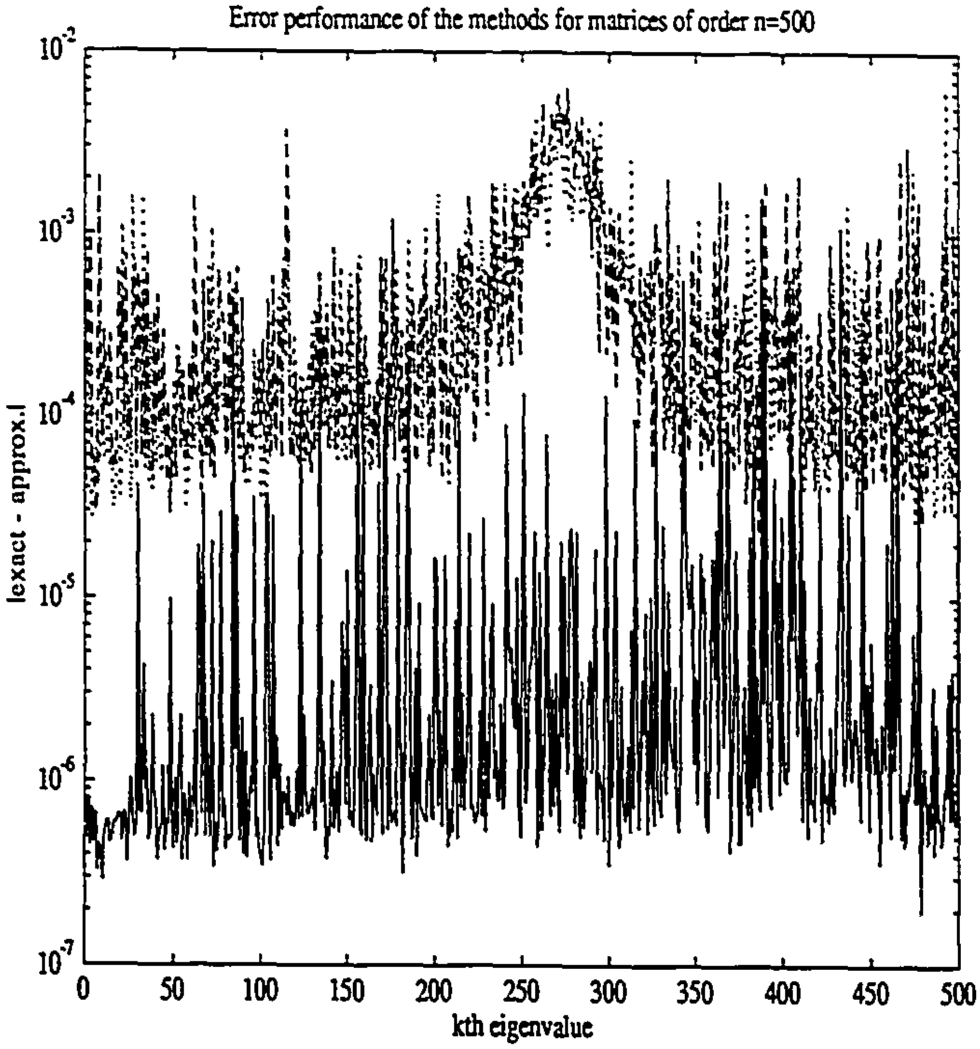

Figure 4.5: Average error averaged over 100 trials for matrices of order 500, Modified-Pegasus: solid m=2.194 $\times 10^{-05}$, std=1.080 $\times 10^{-04}$; Modified-MRQI-B : dashed m=5.660 $\times 10^{-04}$, std=9.416 $\times 10^{-04}$; Modified-MRQI-P : dotted m=4.856 $\times 10^{-04}$, std=7.734 $\times 10^{-04}$.

Table 4.4: Average no. of iterations for matrices with eigenvalues of random distribution.

Matrix Order N	Trench's method using Pegasus				Modified method using Pegasus			
	No.it0	Total of No.it1	Total of No.it2	Total	No.it0	Total of No.it1	Total of No.it2	Total
50	77.98	66.93	251.33	396.24	78.27	64.98	251.48	394.73
100	152.15	140.19	490.27	782.61	149.64	131.77	488.25	769.66
200	301.46	279.36	938.24	1519.06	295.32	264.73	937.60	1497.55
500	749.01	697.72	2221.28	3668.01	728.74	663.44	2220.68	3612.86

Table 4.5: Average no. of iterations for matrices with eigenvalues of random distribution.

Matrix Order N	Modified Method using MRQI-B					
	No.it0	No.it1	Bi.it	Ray.it	No.c1	No.c2
50	78.27	64.98	7.41	112.79	30.01	49.99
100	149.64	131.77	14.36	215.92	58.13	99.94
200	295.32	264.73	29.53	414.59	116.48	199.88
500	728.74	663.44	74.76	974.30	294.68	498.83

Table 4.6: Average no. of iterations for matrices with eigenvalues of random distribution.

Matrix Order N	Modified Method using MRQI-P					
	No.it0	No.it1	Peg.it	Ray.it	No.c1	No.c2
50	78.27	64.98	4.33	101.34	31.78	49.97
100	149.64	131.77	8.90	206.06	64.60	99.86
200	295.32	264.73	17.75	394.84	129.26	199.43
500	728.74	663.44	47.21	924.96	317.25	496.58

Table 4.7: Performance comparison of the algorithms.

Algorithms	Complexity	Convergence	Accuracy Criteria
Trench's/P	$O(n^2)$	1.64	C_1
Modified/P*	$O(n^2)$	1.64	C_1
Modified/MRQI-B*	$O(2n^2)$	cubic	C_2
Modified/MRQI-P*	$O(2n^2)$	cubic	C_2

* method modified for multiple eigenvalues

and the accuracy criteria used.

The L-D algorithm is used at all the steps of the Trench's and the modified algorithm using the Pegasus root finder. In the case of modified algorithm using the MRQI-B and MRQI-P root finders the Levinson algorithm is used at step 3c of the modified algorithm while the L-D algorithm is used in steps 1, 2, and 3b of the modified algorithm. Therefore, under the heading labeled complexity, we have tabulated the complexity of the L-D or the Levinson algorithm used per each shift of the root finding method, namely, the Pegasus, the MRQI-B, and the MRQI-P. From Table 4.7, note the tradeoff between complexity and convergence of the algorithms, the MRQI method has a cubic convergence rate but requires a Levinson recursion with a complexity of $O(2n^2)$ per iteration while the Pegasus method has a convergence rate of 1.64 and requires a L-D recursion with a complexity of $O(n^2)$ per iteration. Since root finding is an iterative procedure, it is not possible to give an exact operation count. Also, note the accuracy criteria most often satisfied in case of MRQI is C_2 indicating a good eigenpair estimate, rather than a good eigenvalue estimate.

4.6 Application to Pisarenko's Harmonic Decomposition

In this section, we illustrate Trench's method, the modified method, and the modified method further modified to include the case of multiple eigenvalues in the problem of Pisarenko's harmonic decomposition. In such an application, one usually deals with Toeplitz matrices with clustered eigenvalues. Consider a Hermitian Toeplitz matrix of order $(n+1)$ formed by the autocorrelation (model) given by

$$r(n) = \sigma^2\delta(n) + \sum_{k=1}^{L} A_k^2 e^{-j\omega_k n} \tag{4.20}$$

This sequence is formed from a random process of the form

$$s(t) = \sum_{k=1}^{L} A_k e^{(-j\omega_k t + \theta_k)} + w(t) \tag{4.21}$$

where L complex exponentials with frequencies ω_k and amplitudes A_k are added to complex white noise $w(t)$ with variance σ^2. The θ_k associated with the exponentials are random variables uniformly distributed over the interval $(-\pi, \pi]$.

In Pisarenko's problem, the number of signals L, amplitudes A_k, frequencies ω_k, and variance of noise σ^2 are unknown and have to be determined from the observed $r(n)$ [24]. Furthermore, the model order p is unknown and needs to be estimated *a priori*. The criterion for determining model order is the following[17]: if *exact* autocorrelations are known, then the model order is specified as that order for which the minimum eigenvalue does not change from one order to the next. On the other hand, if *estimated* autocorrelations are used, then the model order is specified as that order for which the minimum eigenvalue changes "little" from that for a model order of $(p-1)$. In order to determine the minimum eigenvalue, the above algorithms may be used; however, it may be more interesting to see how the algorithms behave in finding all the eigenvalues.

As a numerical example, suppose that the autocorrelation for $n = 9$ was measured and a Hermitian Toeplitz matrix of order 10 formed, specified by the first row,

(31.000000000000, 0.000000000000)
(−13.170368194580, −11.831966400146)
(7.000461101532, −3.994677305221)
(−18.831056594849, 6.159973621368)
(20.999992370605, 0.001805052533)
(−18.824028015137, −6.190902709960)
(6.998598575592, 4.015967369079)
(−13.179986953735, 11.803650856018)
(28.999967575073, 0.003980370983)
(−13.160694122314, −11.860273361206).

In fact, the matrix was formed from (4.20) by choosing the power of the noise as $\sigma^2 = 2.0$ and the number of distinct signals $L = 3$ with their amplitudes as $A_1 = 2$, $A_2 = 3$, $A_3 = 4$ and frequencies $\omega_1 = \pi/4$, $\omega_2 = \pi/2$, $\omega_3 = \pi$, respectively. Since $L = 3$, $n = 9$ was chosen for illustrative purposes (any value of n could have been chosen so long it is large enough to estimate the model order).

In practice only $r(n)$ is known, therefore, using the algorithms, the eigenvalues obtained are tabulated in Tables 4.8 and 4.9. From Tables 4.8 and 4.9, the minimum eigenvalue is seen to be approximately 2. Note that the number of distinct signals are $L = 10 - 7 = 3$. Having found the minimum eigenvalue, the corresponding eigenvector may be determined, and then, from the eigenvector, the frequencies ω_k and amplitudes A_k may be determined.

In Table 4.10, we illustrate the Modified method further modified to include the case of multiple eigenvalues. No.it2 corresponds to the number of iterations required by the root searching methods, namely, the Pegasus, the MRQI-B, and the MRQI-P. As shown above, an example of such a case is Pisarenko's harmonic decomposition. The tabulated results are for the above model with a matrix of order 14. The minimum eigenvalue occurred with a multiplicity of 11. It is to be noted that, in practice, the eigenvalues are seldom exactly equal and more likely to be close to each other. Although Trench's and the Modified algorithms are capable of handling the case of close eigenvalues, the amount

Table 4.8: Pisarenko's Harmonic Decomposition- Trench's method.

α'	β'	No.it1	Eigenvalue	No.it2
1.99999347	1.99999405	9	1.99999382	6
2.00000271	2.00000502	0	2.00000324	6
2.00000617	2.00000646	3	2.00000623	5
2.00000733	2.00000964	1	2.00000735	3
2.00001195	2.00001310	1	2.00001223	5
2.00001426	2.00001657	0	2.00001557	8
2.00002696	2.00002811	4	2.00002724	5
37.39064361	39.75001752	5	39.60024787	8
87.18750000	96.87500000	3	89.92255880	7
155.00000000	310.00000000	0	166.47712760	10
No.it0=31	Total no. of iterations : 31 + 26 + 63 = 120			

of computation, however, becomes unduly high in the search of the interval endpoints enclosing the eigenvalues. In such a case, we may assume close eigenvalues to be equal, if they do not differ by, say, 3 decimal places. Under such an assumption, observe from Table 4.10, that a significant reduction in the number of iterations required is possible.

4.7 Discussion

Since the order recursive algorithm presented here involves the formation and deflation of polynomials, it is liable to suffer roundoff errors and is not recommended for numerical computation. On the other hand, Trench's iterative method and the modified Trench's method do not appear to suffer from such numerical problems. The first modification was the placement of tighter upper and lower bounds about each element of the eigenspectrum, and when such placement is possible, the results are reduced computational complexity and improved convergence. The second modification to the algorithm was to

Table 4.9: Pisarenko's Harmonic Decomposition- Modified method.

α'	β'	No.it1	Eigenvalue	No.it2
1.99998885	1.99999116	7	1.99999094	6
2.00000271	2.00000386	1	2.00000301	5
2.00000502	2.00000617	1	2.00000509	5
2.00000733	2.00000964	1	2.00000847	10
2.00001195	2.00001426	1	2.00001201	4
2.00001657	2.00002580	0	2.00001677	8
2.00002696	2.00002811	3	2.00002716	5
38.75000000	77.50000000	0	39.60024821	7
87.18750000	96.87500000	3	89.92255879	7
155.00000000	310.00000000	0	166.47712951	10
No.it0=31	Total no. of iterations : 31 + 17 + 67 = 115			

Table 4.10: Multiple eigenvalues case.

Algorithms	No.it0	No.it1	No.it2
Modified/P	44	22	32
Modified/P*	21	3	24
Modified/MRQI-B*	21	3	15
Modified/MRQI-P*	21	3	14

* method modified for multiple eigenvalues

include a procedure for the case of multiple eigenvalues. The modified method with three choices of root searching techniques, namely the Pegasus, the MRQI-B, and the MRQI-P, was programmed and the simulation results presented. From the simulation results, since two termination criteria are required when using the MRQI methods and, since $\mathbf{C_2}$ is more often satisfied than $\mathbf{C_1}$, we conclude that the MRQI methods give a good eigenpair estimate, rather than a good eigenvalue estimate. We have also displayed in the table the interplay between accuracy, convergence rate, and computational complexity of the algorithms. In Trench's and the modified algorithm using the Pegasus root finder, the Levinson-Durbin algorithm is used; however, in the case of the modified algorithm using the MRQI-B and MRQI-P root finder, the Levinson algorithm is used.

4.8 Appendix: Proof of Eq.(4.13)

In this appendix, it is shown by induction that (4.13) in Section 2 holds. Starting with (4.12)

$$\begin{aligned}\phi_{k,i}(\lambda) &= \phi_{k-1,i}(\lambda) + \rho_k(\lambda)\bar{\phi}_{k-1,k-i}(\lambda) \\ &= \frac{N_{k-1,i}(\lambda)}{D_{k-1}(\lambda)} + \frac{N_k(\lambda)}{D_k(\lambda)}\frac{\bar{N}_{k-1,k-i}(\lambda)}{D_{k-1}(\lambda)}\,.\end{aligned} \tag{4.22}$$

we need to show that $D_{k-1}(\lambda)$ is a factor of the numerator, viz.,

$$N_{k-1,i}(\lambda)D_k(\lambda) + N_k(\lambda)\bar{N}_{k-1,k-i}(\lambda) \tag{4.23}$$

By definition, we have that

$$\phi_{k,i}(\lambda) = \frac{N_{k-1,i}(\lambda)}{D_{k-1}(\lambda)} - \frac{D_{k-1}(\lambda)c_k + \sum_{m=1}^{k-1} N_{k-1,m}(\lambda)c_{k-m}}{D_{k-1}(\lambda)c_0 + \sum_{m=1}^{k-1} \bar{N}_{k-1,m}(\lambda)c_m}\frac{\bar{N}_{k-1,k-i}(\lambda)}{D_{k-1}(\lambda)} \tag{4.24}$$

The above equation, after forming a common denominator, may be written as

$$\begin{aligned}\phi_{k,i}(\lambda) &= \frac{[c_0 N_{k-1,i}(\lambda) - c_k\bar{N}_{k-1,k-i}(\lambda)]D_{k-1}(\lambda)}{D_{k-1}(\lambda)D_k(\lambda)} \\ &+ \frac{\sum_{i=1}^{k-1}(c_m N_{k-1,i}(\lambda)\bar{N}_{k-1,m}(\lambda) - c_{k-m}\bar{N}_{k-1,k-i}(\lambda)N_{k-1,m}(\lambda))}{D_k(\lambda)D_{k-1}(\lambda)} \\ &.\end{aligned} \tag{4.25}$$

The numerator of the first term of (4.25) is easily seen to contain the factor $D_{k-1}(\lambda)$ found in the denominator. The fact that the numerator of the second term of (4.25) does as well can be verified by induction.

For $k = 2$ and $i = 1$

$$\begin{aligned}\phi_{2,1} &= \frac{c_0 D_1(\lambda)N_{1,1}(\lambda) + c_1[\bar{N}_{1,1}(\lambda)N_{1,1}(\lambda) - N_{1,1}(\lambda)\bar{N}_{1,1}(\lambda)] - c_2 D_1(\lambda)\bar{N}_{1,1}(\lambda)}{D_1(\lambda)D_2(\lambda)} \\ &= \frac{c_0 N_{1,1}(\lambda) - c_2\bar{N}_{1,1}(\lambda)}{D_2(\lambda)}\,,\end{aligned} \tag{4.26}$$

and for $k = 3$ and $i = 1$

$$\begin{aligned}\phi_{3,1} &= \frac{c_0 N_{2,1}(\lambda) - c_3 \bar{N}_{2,2}(\lambda)}{D_3(\lambda)} + \frac{c_1[\bar{N}_{2,1}(\lambda) N_{2,1}(\lambda) - N_{2,2}(\lambda)\bar{N}_{2,2}(\lambda)]}{D_2(\lambda) D_3(\lambda)} \\ &= \frac{c_0 N_{2,1}(\lambda) - c_3 \bar{N}_{2,2}(\lambda)}{D_3(\lambda)} + \frac{c_1[\bar{c}_1 c_1 - c_2 \bar{c}_2]}{D_3(\lambda)}. \end{aligned} \tag{4.27}$$

This inductive process can be carried on for different values of k and i. We conclude, therefore, that (4.13) is correct.

Chapter 5

Conclusions and Directions for Further Research

In conclusion, we have presented a unitary matrix which transforms a Hermitian Toeplitz matrix into a real Toeplitz plus Hankel matrix. The importance of the unitary transform presented is that it preserves structure. As a result, several remarkable properties were also presented. No extra memory space (compared to that for the Hermitian Toeplitz matrix) is required to store the elements of the $T+H$ structure. Second, we presented a solution to the inverse eigenvalue problem for Hermitian Toeplitz matrices. It was shown that a Hermitian Toeplitz matrix of order n may be obtained from a real symmetric negacyclic matrix of order $2n$. A solution to an inverse eigenvalue problem in the case for real symmetric matrices may also be obtained by first constructing a Hermitian Toeplitz matrix and then using the unitary transform presented in Chapter 2 to transform the constructed Hermitian Toeplitz matrix to a real symmetric matrix. The methods for the inverse eigenvalue problem for Hermitian Toeplitz matrices and for real symmetric matrices may be used to test and compare the performance of any eigenvalue decomposition algorithms specialized for Hermitian Toeplitz matrices or for general real symmetric matrices.

In statistical signal processing, when the stochastic processes of interest are weakly stationary, the covariance matrix has a special structure, namely, Hermitian Toeplitz. One goal of signal processing is to extract information contained in this covariance matrix. The main concern in the analysis of Hermitian Toeplitz matrices many times reduces to the solution of the eigenvalue problem. We have, therefore,derived new methods based on the Levinson and Levinson-Durbin recursions for the solution of the eigenvalue problem. The methods presented fall into two categories, order recursive and iterative. The order recursive algorithm was considered to be primarily of theoretical interest. In the iterative category, we presented Trench's method and new methods based on modifications of Trench's method. The modifications included the use of noncontiguous intervals and the inclusion of the case of multiple eigenvalues. The modifications were shown to have important consequences for efficiency in terms of convergence and computational complexity when working with high order matrices.

Theoretical solution to the inverse eigenvalue problem for real symmetric Toeplitz matrices remains unsolved. We believe a solution to the inverse eigenvalue problem for real symmetric Toeplitz matrices will probably lead to an additional number of interesting and computationally efficient algorithms.

5.1 Directions for further research

There are several paths open for those interested in further research in this area. Some of the most important ones are the following:

1. Further study of the eigenvalue relation between T, H, and $T + H$. See the discussion in Section 3.6.

2. Study the theoretical solution to the inverse eigenvalue problem for real symmetric Toeplitz matrices.

3. Further study for possible reduction of the computational complexity of the modified Trench's iterative eigendecomposition algorithm. One approach is to use parallel methods [26]. Another idea is use the unitary transform in conjunction with the

Hermitian Levinson algorithm to see whether further reduction in computational complexity is possible or not.

REFERENCES

[1] S.U. Pillai, *Array Signal Processing*, Springer-Verlag New York Inc., 1989.

[2] S. Haykin, J.H. Justice, N.L. Owsley, J.L. Yen, and A.C. Kak, *Array Signal Processing*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1985.

[3] R.A. Monzingo and T.W. Miller, *Introduction to Adaptive Arrays*, John Wiley and Sons, Inc., 1980.

[4] E. Nicolau and D. Zaharia, *Adaptive Arrays*, Editura Academiei, Bucharest, 1989.

[5] M.S. Lawrence, *Digital Spectral Analysis; with Applications*, Englewood Cliffs, N.J. Prentice-Hall, 1987.

[6] I.S. Iohvidov, *Hankel and Toeplitz Matrices and Forms*, Birkhauser, Boston, 1982.

[7] Marvin J. Goldstein, "Reduction of the Eigenproblem for Hermitian Persymmetric Matrices," *Mathematics of Computation*, vol.28, No. 125, Jan. 1974, pp. 237-238.

[8] Anna Lee, "Centrohermitian and Skew-Centrohermitian Matrices", *Linear Algebra Appl.*, 29, pp. 211-216, 1980.

[9] A. Cantoni and P. Butler, "Eigenvalues and eigenvectors of symmetric centrosymmetric matrices", *Linear Algebra Appl.*, vol. 13, pp. 275-288, 1976.

[10] G. Cybenko, "On the Eigenstructure of Toeplitz Matrices", *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 918-921, 1984.

[11] P. Delsarte and Y. V. Genin, " Spectral Properties of Finite Toeplitz Matrices", in Proc. 1983 Int. Symp Math. Theory of Networks and Syst., Beer Sheva, Israeal, pp. 194-213.

[12] G. A. Merchant and T. W. Parks, "Efficient Solution of a Toeplitz-plus-Hankel Coefficient Matrix System of Equations", *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-30, No. 1, pp. 40-44, February 1982.

[13] I. Gohberg and I. Koltracht, "Efficient Algorithm for Toeplitz plus Hankel Matrices", Integral Equations and Operator Theory, vol. 12, pp. 136-142, 1989.

[14] G. Heinig and K. Rost, *Fast Inversion of Toeplitz plus Hankel Matrices*, Wiss. Z.d. Techn. Hochsch. Karl-Marx-Stadt 27, H-1.

[15] W. F. Trench, "Spectral Evolution of a One Parameter Extension of a Real Symmetric Toeplitz Matrix", preprint.

[16] D.P. Laurie, "A Numerical Approach to the Inverse Toeplitz Eigenproblem", *SIAM J. Sci. Statist. Comput.*, 9, pp. 401-405, 1988.

[17] G. Feyh and C.T. Mullis, "Inverse Eigenvalue Problem for Real Symmetric Toeplitz Matrices", *Proc. ICASSP 1988*, New York City, U.S.A., pp. 1636-1637.

[18] E.H. Golub and C. Van Loan, *Matrix Computations*, John Hopkins, University Press, 1983.

[19] J.H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarenden Press, 1965.

[20] N. Levinson, "The Wiener rms(root mean square) error criterion in filter design and prediction", *J. Math. Phys.*, vol. 25, pp. 261-278.

[21] H. Krishna and S.D. Morgera, " The Levinson Recurrence and Fast Algorithms for Solving Toeplitz Systems of Linear Equations", *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, No. 6, pp. 839-848, June 1987.

[22] P. Delsarte and Y. Genin, " The Split Levinson Algorithm", *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pg. 470-478, 1986.

[23] Y. Bistritz, "A Circular Stability Test for General Polynomials," Systems and Control Letters, 7, pg. 89-97, 1986.

[24] V.F. Pisarenko, "The Retrieval of Harmonics from a Covariance Function," *Geophys. J. Roy. Astron. Soc.*, pp. 347-366, 1973.

[25] M.H. Hayes and M.A.Clements, "An Efficient Algorithm for Computing Pisarenko's Harmonic Decomposition Using Levinson's Recursion," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, No. 3, pp. 485-491, June 1986.

[26] Y.H. Hu and S.Y. Kung, "Toeplitz Eigensystem Solver," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-33, pp. 1264-1271, Oct. 1985.

[27] G. Cybenko and C. Van Loan, "Computing the Minimum Eigenvalue of a Symmetric Positive Definite Toeplitz Matrix," *SIAM J. Sci. Stat. Comput.*, 7, pp. 123-131, 1986.

[28] D.M. Wilkes, S.D. Morgera, F. Noor, and M.H. Hayes, "A Hermitian Toeplitz Matrix is Unitarily Similar to a real Toeplitz plus Hankel Matrix", *IEEE Transactions on Signal Processing*, vol. 39, No. 9, Sept. 1991, pp. 2146-2148.

[29] P. J. Davis, *Circulant Matrices*, New York, N.Y., 1979.

[30] D.M. Wilkes and M.H. Hayes, "An Eigenvalue Recursion for Toeplitz Matrices," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, No. 6, pp. 907-909, June 1987.

[31] S.D. Morgera and F. Noor, "An Eigenvalue Recursion for Hermitian Toeplitz Matrices," *ICASSP 88*, New York City, U.S.A. April 13, 1988, pp. 1647-1650.

[32] W.F. Trench, "Numerical Solution of the Eigenvalue Problem for Hermitian Toeplitz Matrices," *SIAM J. Matrix Anal. Appl.*, pp. 135-146, April 1989.

[33] A.A. Beex and M.P. Fargues, "Highly Parallel Recursive/Iterative Toeplitz Eigenspace Decomposition", *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-37, No. 11, pp. 1765-1768, November 1989.

[34] S.D. Morgera, "On the Reducibility of Finite Toeplitz Matrices-Applications in Speech Analysis and Pattern Recognition," *Signal Processing*, vol. 15, nos. 5,6. pp.425-443, Oct. 1982.

[35] P. Delsarte, Y.V. Genin, and Y.G. Kamp, "A Generalization of the Levinson Algorithm for Hermitian Toeplitz Matrices with Any Rank Profile," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-33, No. 4, pp.964-971, August 1985.

[36] A. Ralston and P. Rabinowitz, *A First Course in Numerical Analysis*, 2nd ed., McGraw-Hill, 1978.

[37] M. Dowell and P. Jaratt, "The "Pegasus" method for computing the root of an equation", *BIT 12*, pp. 503-508, 1972.

[38] F. Noor and S.D. Morgera, "Construction of a Hermitian Toeplitz matrix from an arbitrary set of real eigenvalues", vol. 40, No. 8, pp. 2093-2094, Aug. 1992.

[39] F. Noor and S.D. Morgera, "Recursive and Iterative algorithms for Hermitian Toeplitz matrices", accepted for publication in IEEE Transactions on Signal Processing.

[40] Y.H. Hu "Parallel Eigenvalue Decomposition for Toeplitz and related Matrices," *Proc. ICASSP '89*, Glascow, pp. 1107-1110.

[41] S.Y. Kung and Y.H. Hu, "Improved Pisarenko's spectrum estimate via SVD subspace approximation method," *Proc. 21-th CDC*, Orlando, FL, pp. 1312-1314, Dec. 1982.

[42] Y.H. Hu, I.C. Jou, and T.M. Parng, "Subspace approximation based covariance eigensystem solver," *IEEE Proc.*, vol. 134, No.2, April 1987, pp. 159-165.

Appendix A: Modified Method - Multiple Eigenvalue Case with the Pegasus Root Finder

```
PROGRAM MODMUL
C This algorithm is a Modified version of Trench's algorithm.
C The algorithm is modified to: 1) Use tighter interval endpoints
C enclosing the eigenvalues, 2) Handle multiple eigenvalues,
C if any, in the data matrix. This algorithm uses the Levinson-Durbin
C Algorithm with Modified Trench's method to determine the eigenvalues
C of a Hermitian Toeplitz matrix. Also uses the PEGASUS method as a root
C searching method. Store data in file called DATAH as below.
C 2           Example, first line should have N the order of matrix.
C 9.0 0.0     from line 2 write the elements of matrix.
C 8.0 7.0
C Results will appear in file called RESULTHM.
      COMPLEX*16 C(0:1000)
C C is the data matrix
```

```
      DOUBLE PRECISION CNR, CNI
C CNR, CNI ;real part, imaginary part
      COMMON /L1/ N, C
C N is the order of matrix
      COMMON /X1/ TRACE
C TRACE is sum of eigvalues if equals trace
      COMMON /C1/ KLEV
C KLEV counts calls to levson-durbin
      KLEV = 0
      OPEN (UNIT=11, FILE='datah',STATUS='OLD')
      OPEN (UNIT=12, FILE='resultm',STATUS='NEW')
      OPEN (UNIT=13, FILE='eigmul.dat',STATUS='NEW')
      OPEN (UNIT=14, FILE='vecmul.dat',STATUS='NEW')
      READ(11,*) N
      DO 1 INDEX=0, N-1
          READ(11,*) CNR, CNI
          C(INDEX) = CMPLX(CNR, CNI)
 1    CONTINUE
C Step 1. find interval (a,b)
      CALL SELECT
C Step 2. search for endpoints
      CALL SEARCH
C Step 3. refine interval and estimate eigenvalue
      CALL REFINE
      WRITE(12, 2) TRACE, KLEV
 2    FORMAT ('SUM OF EIGS =',F15.7 , ' TOTAL NO. OF LEV ITER= ', I7)
      STOP
      END
```

```
C***********************************************************************
      SUBROUTINE SELECT
C This routine selects the endpoints a and b of the (a,b) which
C contains the eigenvalues to be determined.
      COMPLEX*16 C(0:1000)
      DOUBLE PRECISION AA, BB, STORE, LE(1:1000), UE(1:1000)
      DOUBLE PRECISION DELLE(1:1000), DELUE(1:1000), DELTA(2)
      INTEGER N, IP, IQ, NEGAA, NEG, NEGBB
      COMMON /L1/ N, C
C STORE stores the element c0
      COMMON /Z1/ STORE
      COMMON /S1/ NEG, DELTA, MI
C LE, UE holds lower and upper bound endpoints
      COMMON /Z3/ LE, UE
C DELLE, DELUE retains the value En(Lξi) and En(Uξi)
      COMMON /EE3/ DELLE, DELUE
C IP, IQ choose any a,b to p=a and q=b to selectively find eigs
      COMMON /R1/ IP,IQ
C all the eigenvalues of positive definite matrix
      AA = 0.0
C are between 0.0 and n × c0
      BB = N*C(0)
C IOP, IOQ indicates eigenvalues between (a,b) are
C from 1 to N
      IOP = 1
      IOQ = N
C you can change ip and iq to selectively choose desired eigenvalues.
      IP = 1
```

```
      IQ = N
C two endpoints needed to enclose eigenvalues
      ITO = 0
      STORE = C(0)
C shift matrix C by an amount a
      C(0) = STORE - AA
C to determine the eigenvalue count of eigenvalues
      CALL LEVSON
C eigenvalue indicator
      NEGAA = NEG
C if count is true then retain the value
      IF ( NEGAA .LE. (IOP-1)) THEN
         LE(IOP) = AA
         DELLE(IOP) = DELTA(MI)
      ENDIF
C next shift matrix by and amount b
      C(0) = STORE - BB
      CALL LEVSON
      NEGBB = NEG
      IF ( NEGBB .GE. IOQ ) THEN
         UE(IOQ) = BB
         DELUE(IOQ) = DELTA(MI)
      ENDIF
C initialize the arrays Lξi and Uξi
      DO 1 MX=IOP, IOQ-1
         LE(MX+1)=-1.0
         UE(MX) = -1.0
  1   CONTINUE
```

```
C set inner endpoints
      IR = IP
      IS = IOQ
C use Lξi as lower point for bisect shift
      EL =LE(IOP)
C if endpoint not found then search
 2    IF ( UE(IR) .EQ. -1.0) THEN
         DO 3 IH=IR+1, IOQ
            IF ( UE(IH) .NE. -1.0 ) THEN
C mark the closest upper point to be used in bisect
               MARK = IH
               GO TO 4
            ENDIF
 3       CONTINUE
 4       EU = UE(MARK)
C bisection shift
 5       GAM = ( EL + EU ) * .5
         IF (ABS(EL-EU) .LE. 1.0e-6) THEN
            PRINT*, 'There is a multiple eigenvalue'
            RETURN
         ENDIF
         C(0) = STORE - GAM
         CALL LEVSON
C keep count of calls to L-D algorithm
         IT = IT + 1
         K = NEG
C endpoint is found
         IF ( K .EQ. IR ) THEN
```

```
C store the found endpoint
                    UE(K) = GAM
C store corresponding value
                    DELUE(K) = DELTA(MI)
                    LE(K+1) = GAM
                    DELLE(K+1) = DELTA(MI)
C next endpt of eigenvalue to be found
                    GOTO 6
C capture other endpoints or update endpoints so to tighten
               ELSE
                    IF (( K .EQ. (IR-1) ) .AND. (GAM .GE. LE(K+1))) THEN
C tighten upper endpoint
                         EL = GAM
                         LE(K+1) = GAM
                         DELLE(K+1) = DELTA(MI)
                         GOTO 5
                    ENDIF
C capture endpoint or update
                    IF ( (K .GT. IR) .AND. (K .LT. IS)) THEN
C endpoint is captured
                         IF (UE(K) .EQ. -1.0) THEN
                              UE(K) = GAM
                              DELUE(K) = DELTA(MI)
                              LE(K+1) = GAM
                              DELLE(K+1) = DELTA(MI)
                              EU = GAM
                         ELSE
C upper endpt updated
```

```
                  IF ( GAM .LT. UE(K) ) THEN
                      UE(K) = GAM
                      DELUE(K) = DELTA(MI)
                      EU = GAM
                  ENDIF
C lower endpt updated
                  IF (GAM .GT. LE(K+1) ) THEN
                      LE(K+1) = GAM
                      DELLE(K+1) = DELTA(MI)
                  ENDIF
              ENDIF
          GOTO 5
          ENDIF
          IF ( K .EQ. IS) THEN
              UE(K) = GAM
              DELUE(K) = DELTA(MI)
              EU = GAM
              GOTO 5
          ENDIF
          EL = GAM
          GOTO 5
      ENDIF
  ELSE
C one endpoint found go find the second one
      GOTO 6
  ENDIF
6 CONTINUE
  ITO = ITO + 1
```

```
C find second endpoint
      IF (ITO .LT. 2) THEN
          DO 7 IH=IQ, IR, -1
C using this endpoint for bisection shift
              IF ( LE(IH) .NE. -1.0 ) THEN
                  EL = LE(IH)
                  IR = IQ
                  GO TO 2
              ENDIF
 7        CONTINUE
          EL = LE(IR)
          IR = IQ
          GOTO 2
      ELSE
C both are found exit
          GOTO 8
      ENDIF
 8    CONTINUE
      RETURN
      END
C****************************************************************************
```

C search for the inner intervals $L\xi_i$, $U\xi_p$

```
      SUBROUTINE SEARCH
      COMPLEX*16 C(0:1000)
      DOUBLE PRECISION STORE, EL, EU, GAM, LE(1:1000), UE(1:1000)
      DOUBLE PRECISION DELLE(1:1000), DELUE(1:1000), DELTA(2)
      INTEGER N, IP, IQ, NEG
      COMMON /L1/ N, C
```

```
      COMMON /Z1/ STORE
      COMMON /S1/ NEG, DELTA, MI
      COMMON /Z3/ LE, UE
      COMMON /EE3/ DELLE, DELUE
      COMMON /R1/ IP, IQ
      COMMON /IREPEAT/ MREPEAT
      IT = 0
      IR = IP
      IS = IQ
      KEL = IR-1
      KUE = IS
      PRINT*, ' Enter tolerance for multiple eigenvalues: '
      READ*, TOLMUL
      MARK = IS
C terminate if all endpoints found
 1    IF ( IR .GT. (IS-1) ) THEN
          GOTO 8
C search for the endpoits
      ELSE
          IF ( UE(IR) .EQ. -1.0) THEN
              IF ( LE(IR) .NE. -1.0) THEN
                  EL = LE(IR)
                  SS = LE(IR)
                  SDELLE = DELLE(IR)
                  STOREL = DELLE(IR)
                  IXX = IR
              ENDIF
              IF ( LE(IR) .EQ. -1.0 ) THEN
```

```
            DO 2 LI=IR-1, 1, -1
                IF ( LE(LI) .NE. -1.0 ) THEN
                    EL = LE(LI)
                    SS = LE(LI)
                    SDELLE = DELLE(LI)
                    STOREL = DELLE(LI)
                    GOTO 3
                ENDIF
2           CONTINUE
        ENDIF
3       CONTINUE
        DO 4 IH=IR+1, IS
            IF ( UE(IH) .NE. -1.0 ) THEN
                MARK = IH
                GO TO 5
            ENDIF
4       CONTINUE
5       EU = UE(MARK)
6       GAM = ( EL + EU ) * .5
C multiple eigenvalue
        IF (ABS(EL-EU) .LE. TOLMUL) THEN
                LE(IXX) = SS
                DELLE(IXX) = SDELLE
                LE(IXX) = EL
                DELLE(IXX) = STOREL
                IR = IR + (KUE - KEL) - 1
                GOTO 7
        ENDIF
```

```
                  C(0) = STORE - GAM
                  CALL LEVSON
                  IT = IT + 1
                  K = NEG
                  IF ( MREPEAT .EQ. 1) THEN
C find next interval endpoint
                       GOTO 7
                  ENDIF
C endpoint found
                  IF ( K .EQ. IR ) THEN
                       UE(K) = GAM
                       DELUE(K) = DELTA(MI)
                       LE(K+1) = GAM
                       DELLE(K+1) = DELTA(MI)
                       GOTO 7
                  ENDIF
                  IF (( K .EQ. (IR-1) ) .AND. (GAM .GE. LE(K+1))) THEN
C update lower endpoint of interval
                       KEL = K
                       EL = GAM
                       LE(K+1) = GAM
                       DELLE(K+1) = DELTA(MI)
                       STOREL = DELTA(MI)
                       GOTO 6
                  ENDIF
C endpoint captured
                  IF ( (IR .LT. K) .AND. (K .LT. IS)) THEN
                       IF (UE(K) .EQ. -1.0) THEN
```

```
                UE(K) = GAM
                DELUE(K) = DELTA(MI)
                LE(K+1) = GAM
                DELLE(K+1) = DELTA(MI)
                EU = GAM
                KUE = K
            ELSE
C update upper endpoint
                IF ( GAM .LT. UE(K) ) THEN
                    UE(K) = GAM
                    DELUE(K) = DELTA(MI)
                    EU = GAM
                    KUE = K
                ENDIF
C update lower endpoint
                IF (GAM .GT. LE(K+1) ) THEN
                    LE(K+1) = GAM
                    DELLE(K+1) = DELTA(MI)
                ENDIF
            ENDIF
            GOTO 6
        ENDIF
        IF ( K .EQ. IS) THEN
            UE(K) = GAM
            DELUE(K) = DELTA(MI)
            EU = GAM
            KUE = K
            GOTO 6
```

```
                ENDIF
                IF ( K .LT. IR) THEN
                     EL = GAM
                     KEL = K
                     GOTO 6
                ENDIF
C exit to search for next endpoint
                GOTO 7
           ENDIF
      ENDIF
 7    CONTINUE
      IR = IR + 1
      GOTO 1
 8    CONTINUE
      RETURN
      END
C**********************************************************************
      SUBROUTINE LEVSON
      DOUBLE PRECISION DELTA(2)
      COMPLEX*16 C(0:1000), X(1000,2), SUM, EIGVX(1000)
      INTEGER N,M,J,MI,NEG
      COMMON /L1/ N, C
      COMMON /S1/ NEG, DELTA, MI
      COMMON /EIGV/ X
      COMMON /EVX/ EIGVX
      COMMON /C1/ KLEV
      COMMON /IREPEAT/ MREPEAT
      KLEV = KLEV + 1
```

```
      NEG = 0
      MREPEAT = 0
      X(1,1) = C(1)/C(0)
      DELTA(1) = C(0)
      IF ( DELTA(1) .LT. 0 ) THEN
         NEG = NEG + 1
      ENDIF
      SUM = (0.0,0.0)
      DO 1, M=2, N
         DELTA(2) = (1.0 - X(M-1,1) * CONJG(X(M-1,1)) ) * DELTA(1)
         IF ( DELTA(2) .LT. 0 ) THEN
            NEG = NEG + 1
         ENDIF
         IF (DELTA(2) .EQ. 0.0) THEN
            MREPEAT = 1
            RETURN
         ENDIF
         SUM = (0.0,0.0)
         DO 2 JM=1, M-1
            SUM = C(M-JM) * X(JM, 1) + SUM
2        CONTINUE
         X(M,2) = ( C(M) - SUM ) / DELTA(2)
         DO 3 J=1, M-1
            X(J,2) = X(J,1) - X(M,2) * CONJG( X(M-J,1) )
3        CONTINUE
         DO 4 L=1, M
            EIGVX(L)=X(L,1)
            X(L,1) = X(L,2)
```

```
4         CONTINUE
          DELTA(1) = DELTA(2)
1     CONTINUE
      MI = 2
      RETURN
      END
C************************************************************************
      SUBROUTINE REFINE
C After initial interval (a,b) is chosen by routine select subintervals for λi have
C been selected by subroutine select. Subroutine refine searches for interval
C (α',β') such that it does not contain eigenvalue of the submatrix C_{n-1}.
      COMPLEX*16 C(0:1000)
      DOUBLE PRECISION STORE, DELB, DELG, GAM, LE(1:1000), UE(1:1000)
      DOUBLE PRECISION DELLE(1:1000), DELUE(1:1000), SLIM1, SLIM2
      DOUBLE PRECISION DELTA(2), ALPHA, BETA, DELA
      INTEGER NEG, NEGNA, NEGNB, NEGNG
      COMMON /L1/ N, C
      COMMON /S1/ NEG, DELTA, MI
      COMMON /Z1/ STORE
      COMMON /Z3/ LE, UE
      COMMON /EE3/ DELLE, DELUE
      COMMON /T1/ SLIM1, SLIM2
      COMMON /T3/ DELB, DELA
      COMMON /IT1/ ITRTOT
      COMMON /R1/ IP, IQ
      COMMON /X1G/ GAM
      COMMON /MULC/MC
      ITRTOT = 0
```

```
      ITRATO = 0
      LIM = IQ
      C(0) = STORE
      I = IP
 1    MC = 0
      IF ( I .GT. LIM ) THEN
C all eigenvalues have been estimated. exit.
         GOTO 7
      ENDIF
      IF ( UE(I) .NE. -1.0 ) THEN
         MC = 1
         GOTO 3
C determine the no. of multiplicities
      ELSE
         ALPHA = LE(I)
         DELA = DELLE(I)
 2       IF ( UE(I) .EQ. -1.0 ) THEN
            I = I + 1
            MC = MC + 1
            GOTO 2
         ENDIF
         MC = MC + 1
         GOTO 4
      ENDIF
 3    ALPHA = LE(I)
      DELA = DELLE(I)
 4    BETA = UE(I)
      DELB = DELUE(I)
```

```
      NEGNA = I - 1
      NEGNB = I
      ITERA = 0
C search till 2 conditions are satified and call root to estimate the eigenvalue in this interval.
  5   IF ( (NEGNA .EQ. (I-1) ) .AND. (NEGNB .EQ. I ) ) THEN
         IF ( (DELA .GT. 0) .AND. (DELB .LT. 0 ) ) THEN
            SLIM1 = ALPHA
            SLIM2 = BETA
            CALL ROOT
            ITRATO = ITRATO + ITERA
            ITERA = 0
            I = I + 1
            GOTO 1
         ELSE
C use bisection shifts
            GOTO 6
         ENDIF
      ELSE
         GOTO 6
      ENDIF
  6   GAM = ( ALPHA + BETA ) * 0.5
      ITERA = ITERA + 1
      C(0) = STORE - GAM
      CALL LEVSON
      NEGNG = NEG
      DELG = DELTA(MI)
      IF ( NEGNG .LE. (I-1) ) THEN
         ALPHA = GAM
```

```
          NEGNA = NEGNG
          DELA = DELG
          GOTO 5
      ELSE
          BETA = GAM
          NEGNB = NEGNG
          DELB = DELG
          GOTO 5
      ENDIF
      I = I + 1
C next eigenvalues to be estimated
      GOTO 1
  7   CONTINUE
      RETURN
      END
C***********************************************************************
      SUBROUTINE ROOT
C After subroutine Eigen specifies the interval (α', β') which does not contain
C an eigenvalue of C_{n-1}. Root uses Pegasus method to find the eigenvalue in (α', β').
      COMPLEX*16 C(0:1000), X(1000,2), EIGVX(1000)
      DOUBLE PRECISION DX, DELX, DELS1, DELB, DELA, EPS, STORE, GAM
      DOUBLE PRECISION DELTA(2), DELS2, SLIM2, SLIM1, TOL, PX, PFX
      INTEGER N
      COMMON /L1/ N, C
      COMMON /S1/ NEG, DELTA, MI
      COMMON /T1/ SLIM1,SLIM2
      COMMON /T3/ DELB, DELA
      COMMON /Z1/ STORE
```

```
      COMMON /IT1/ ITRTOT
      COMMON /X1/ EIG
      COMMON /EIGV/ X
      COMMON /EVX/ EIGVX
      COMMON /X1G/ GAM
      COMMON /MULC/ MC
C can change this value to accuracy desired
      EPS = 1.0E-6
      ITR = 0
C maximum allowance for eigen estimate
      MAXITR = 30
C DX is the next shift
      DX = 0.0
C counts no. of retentions on side one.
      KX1 = 0
C counts no. of retentions on side two.
      KX2 = 0
      PX = GAM
      DELS1 = DELA
      DELS2 = DELB
      DELX = 1.0
 1    DX = ( SLIM2*DELS1 - SLIM1*DELS2)/(DELS1 - DELS2) ;next shift
      TOL = .5*(1.0+DABS(DX))*EPS
      IF ( DABS(DX-PX) .LT. TOL ) THEN
         TRACE = TRACE + DX
         WRITE(12, 2) DX, MC, ITR
 2       FORMAT (12X, F15.8, 2X, 'multiplicity of', I3, 10X, I4)
         ITRTOT= ITRTOT + ITR
```

```
            WRITE(14. 3) -1.00. 0.00
 3          FORMAT (1X. F25.8.6X. F25.8)
            DO 4 L=1. N-1
C write elements of the eigenvector
                WRITE(14. 5) EIGVX(L)
 5              FORMAT (1X. F25.8.6X. F25.8)
 4          CONTINUE
            RETURN
        ENDIF
C store DX shift as previous x
        PX = DX
        C(0) = STORE- DX
        CALL LEVSON
        DELX = DELTA(MI)
C count calls to L-D algorithm
        ITR = ITR + 1
C terminate estimate of eigenvalue
        IF (ITR .EQ. MAXITR ) THEN
            WRITE(12,6) DX,ITR
 6          FORMAT (15X, F15.8,26X, I4)
            TRACE = TRACE + DX
            ITRTOT= ITRTOT + ITR
            PRINT*, 'X= -1.0000 0.000'
            DO 7 L=1, N-1
C Print the eigenvectors
                PRINT*,'Eigvx2= ', EIGVX(L)
                PRINT*,'X= ', X(L,2)
 7          CONTINUE
```

```
            RETURN
        ENDIF
        IF ( (DELX * DELS1) .GT. 0 ) THEN
            PFX = DELS1
            DELS1 = DELX
            SLIM1 = DX
            KX2= KX2 + 1
            KX1= 0
C avoid retention of an endpoint by scaling down the function.
            IF ( KX2 .GT. 1 ) THEN
                DELS2 = (DELS2 * PFX) / (PFX + DELS1)
            ENDIF
            GOTO 1
        ENDIF
        IF ( (DELX * DELS2) .GT. 0 ) THEN
            PFX = DELS2
            DELS2 = DELX
            SLIM2 = DX
            KX1= KX1 + 1
            KX2= 0
C avoid retention of an endpoint by scaling down the function.
            IF ( KX1 .GT. 1 ) THEN
                DELS1 = (DELS1 * PFX) / (PFX + DELS2)
            ENDIF
            GOTO 1
        ENDIF
        RETURN
        END
```

Appendix B: Modified Method - Multiple Eigenvalue Case with the MRQI-B Root Finder

```
PROGRAM MODMUB
 C Note, replace the subroutine root in Appendix A by these 2 subroutines the
 C rest being the same. This algorithm is a Modified version of Trench's algorithm.
 C The algorithm is modified to: 1) Use tighter interval endpoints enlosing
 C the eigenvalues, 2) Handle the multiple eigenvalues, if any in the data matrix.
 C This algorithm uses the Levinson-Durbin Algorithm with Modified Trench's method
 C to determine the eigenvalues of a Hermitian Toeplitz matrix. Also uses the MRQI-B
 C method a root searching method with Levinson algorithm. Store your data in file
 C called DATAH as below.
 C 2          first line should have N the order of matrix.
 C 9.0 0.0    from line 2 write the elements of matrix.
 C 8.0 7.0
 C Results will appear in file called RESMBI.
C*************************************************************************
```

```
SUBROUTINE LEV
DOUBLE PRECISION DELTA(2)
COMPLEX*16 C(0:1000), X(1000,2), SUM, B(1000), Y(1000,2), SOM
COMPLEX*16 EIGVX(1000), EIGVY(1000)
INTEGER N,M,J,MI,NEG
COMMON /L1/ N,C
COMMON /S1/ NEG, DELTA, MI
COMMON /EIGV/ B, X, Y
COMMON /EVX/ EIGVX, EIGVY
COMMON /LEV2/ KLEV2
KLEV2 = KLEV2 + 1
NEG = 0
X(1,1) = C(1)/C(0)
Y(1,1) = B(1)/C(0)
DELTA(1) = REAL( C(0) )
IF ( DELTA(1) .LT. 0 ) THEN
   NEG = NEG + 1
ENDIF
SUM = (0.0,0.0)
SOM = (0.0,0.0)
DO 7, M=2, N
   DELTA(2) = (1.0 - X(M-1,1) * CONJG( X(M-1,1) ) ) * DELTA(1)
   IF ( DELTA(2) .LT. 0 ) THEN
      NEG = NEG + 1
   ENDIF
   SOM = (0.0,0.0)
   DO 1 JM=1, M-1
      SOM = C(M-JM) * Y(JM, 1) + SOM
```

```
1       CONTINUE
        Y(M,2) = ( B(M) - SUM ) / DELTA(2)
        DO 2 J=1, M-1
           Y(J,2) = Y(J,1) - Y(M,2) * CONJG( X(M-J,1) )
2       CONTINUE
        DO 3 L=1, M
           EIGVY(L)=Y(L,1)
           Y(L,1) = Y(L,2)
3       CONTINUE
        IF ( M .EQ. N ) THEN
           GOTO 7
        ENDIF
        SUM = (0.0,0.0)
        DO 4 JM=1, M-1
           SUM = C(M-JM) * X(JM, 1) + SUM
4       CONTINUE
        X(M,2) = ( C(M) - SUM ) / DELTA(2)
        DO 5 J=1, M-1
           X(J,2) = X(J,1) - X(M,2) * CONJG( X(M-J,1) )
5       CONTINUE
        DO 6 L=1, M
           EIGVX(L)=X(L,1)
           X(L,1) = X(L,2)
6       CONTINUE
        DELTA(1) = DELTA(2)
7     CONTINUE
      MI = 2
      RETURN
```

```
      END
C*************************************************************************
      SUBROUTINE ROOT
C After subroutine refine specifies the interval (α', β') which does not contain
C an eigenvalue of C_n - 1. Root uses MRQI-B method to find the eigenvalue in (α', β').
      COMPLEX*16 C(0:1000), X(1000,2), EIGVX(1000)
      COMPLEX*16 Y(1000,2), EIGVY(1000), B(1000)
      DOUBLE PRECISION DX, DELX, DELS1, DELB, DELA, EPS, STORE
      DOUBLE PRECISION DELTA(2), DELS2, SLIM1, SLIM2, TOL, PX, GAM
      DOUBLE PRECISION DX2, SOM, XNORM, YNORM, SUUM, SUMY
      INTEGER N
      COMMON /L1/ N, C
      COMMON /S1/ NEG, DELTA, MI
      COMMON /T1/ SLIM1, SLIM2
      COMMON /T3/ DELB, DELA
      COMMON /Z1/ STORE
      COMMON /IT1/ITRTOT, ITOTBI, ITOTIQ
      COMMON /X1/ TRACE
      COMMON /EIGV/ B, X, Y
      COMMON /EVX/ EIGVX, EIGVY
      COMMON /X1G/ GAM
      COMMON /MULC/ MC
      EPS = 1.0E-6
      ITR = 0
      TOL = 0
      PX = GAM
      DELS1 = DELA
      DELS2 = DELB
```

```
      ITQ = 0
      ITBIS = 0
      XNORM = 0.0
      DO 1 I=1, N-1
C compute the norm
         XNORM = XNORM + X(I,1)*CONJG( X(I,1) )
  1   CONTINUE
      XNORM = XNORM**.5
      B(1) = (-1.0,0.0) / XNORM
C normalize the vector
      DO 2 I=2, N
         B(I) = EIGVX(I-1)/ XNORM
  2   CONTINUE
C bisection shift
      DX = (SLIM1+SLIM2)*.5
C call lev to solve for y
  3   C(0) = STORE-DX
      CALL LEV
      DELX = DELTA(MI)
      ITR = ITR + 1
C use the bisect shift if true
      IF ((DELX*DELS1) .GT. 0) THEN
         SLIM1 = DX
         DELS1 = DELX
      ELSE
         SLIM2 = DX
         DELS2 = DELX
      ENDIF
```

```
C calculate numerator of rayleigh quotient
      SUUM = 0.0
      DO 4 I=1, N
         SUUM = SUUM + CONJG( Y(I,1) ) * B(I)
   4  CONTINUE
C calculate denominator of rayleigh quotient
      SUMY = 0.0
      DO 5 I=1, N
         SUMY = SUMY + CONJG( Y(I,1) ) * Y(I,1)
   5  CONTINUE
C rayleigh iteration
      DX2 = SUUM / SUMY + DX
C normalize the vector y
      YNORM = 0.0
      DO 6 I=1, N
         YNORM = YNORM + Y(I,1) * CONJG( Y(I,1) )
   6  CONTINUE
      YNORM = YNORM**.5
      DO 7 I=1, N
         B(I) = Y(I,1) / YNORM
   7  CONTINUE
      IF (( SLIM1 .LE. DX2) .AND. (DX2 .LE. SLIM2)) THEN
         WRITE(12, 8) DX2, ITQ
   8     FORMAT ('ray ', 15X, F15.8,26X, I4)
         TOL = .5*(1.0+DABS(DX))*EPS
         IF (DABS(DX-DX2) .LE. TOL ) THEN
            KCOND1 = KCOND1 + 1
         ENDIF
```

```
          IF (YNORM .GT. 1000 ) THEN
             KCOND2 = KCOND2 + 1
          ENDIF
C Estimate root found
          IF ( (DABS(DX-DX2) .LE. TOL ) .OR. (ITQ .EQ. 20)
     *       .OR. (YNORM .GT. 1000) ) THEN
             EIG = EIG + DX
             WRITE(12, 9) DX, MC, ITR
   9         FORMAT ('raylil', 8X, F18.8, ' X by ', I4 , 16X, I4)
             ITRTOT= ITRTOT + ITR
             GOTO 12
          ENDIF
          DX = DX2
       ELSE
  10      ITBIS = ITIS + 1
          DX = (SLIM1 + SLIM2) * .5
       ENDIF
       IF ( ( ( ABS(SLIM2 - SLIM1) )*.5 ) .LT. 1.0E-06 ) THEN
          EIG = EIG + DX
          WRITE(12, 11) DX,ITR
  11      FORMAT ('bisect',15X, F15.8,26X, I4)
          ITRTOT= ITRTOT + ITR
          GOTO 12
       ENDIF
       GO TO 3
  12   WRITE(12, 13) ITBIS, ITQ
  13   FORMAT ( 'bisect= ', I4 , 26X, 'ray qo=', I4)
C counter counts rayleigh iterations
```

```
      ITOTIQ = ITOTIQ+ITQ
C counter counts bisection iterations
      ITOTBI=ITOTBI+ITBIS
      RETURN
      END
```