'95 KACC (1995. 10. 23 ~ 25)

COMPLETE ALGORITHM TO LOCATE DESIRED MULTIPLE OR CLUSTERED EIGENPAIRS

°Chang Wan Jeon*, Hyoung Joong Kim**, and Jang Gyu Lee***

*Automatic Control Research Center, Seoul Nat'l Univ., Kwanak Ku, Seoul 151-742, KOREA
Tel:+82-2-880-7317; Fax:+82-2-885-6620; E-mail;jcw@asrignc3.snu.ac.kr

**Dept. of Control and Instrumentation Eng., Kangwon Nat'l University
Tel:+82-361-50-6343; Fax:+82-361-242-2059; E-mail;khj@cc.kangwon.ac.kr

*** Automatic Control Research Center, Seoul Nat'l Univ., Kwanak Ku, Seoul 151-742, KOREA
Tel:+82-2-880-7308; Fax:+82-2-885-6620; E-mail;jgl@asrignc3.snu.ac.kr

Abstract In this paper, two algorithms for computing multiple or clustered eigenvalues are proposed. The algorithms can be applied to all kinds of Hermitian matrix unlike the existing algorithm. Characteristics of the proposed algorithms is examined by MATLAB simulations.

Keywords Multiple eigenvalue, Eigenvector

1. INTRODUCTION

Remarkable progress has been made on numerical solution of the eigenvalue problem during the last decade [1]-[7]. Most of previous works have concentrated on the distinct eigenvalues. However, many matrices have multiple eigenvalues and/or clustered eigenvalues. Although those algorithms can be applicable to a matrix having clustered eigenvalues, the closeness of the eigenvalues in a cluster tends to cause all the numerical procedure to lose efficiency, in the sense that considerable computational effort must be expended performing bisection shifts in search of eigenvalue interval endpoints. Furthermore, if the desired eigenvalue has multiplicity greater than one, all of them might not work. Thus, unless a matrix is known to have distinct eigenvalues and spaced sufficiently apart, experts advise not to use the existing algorithms. Recently, Noor and Morgera [8] have proposed a method claiming that it works well on multiple and/or clustered eigenvalues. Their method is a valuable one. However, their algorithm needs to be sophisticated; their algorithm may fail to a certain set of Hermitian Toeplitz matrices.

In this paper, we propose two algorithms which are applicable to all kinds of Hermitian matrices for computing multiple or clustered eigenvalues. The first one treats the multiple and/or clustered eigenvalues as a multriple one and works with the original matrix. The whole bag of tricks of the second method is based on the interlacing theorem of Cauchy. It employs the reduced matrix instead of the original matrtrix. People may misunderstand the interlacing theorem. The pitfall is stressed by introducing an example. The main contribution of the proposed algorithms is their generality. That is, these methods work well on any Hermitian matrix. Moreover, we show the features of them through extensive simulations.

2. INCLUSION INTERVAL

Most of the previous researches locate the inclusion interval first and then locate the desired eigenpair. The inclusion interval, denoted by $[b_l, b_u]$, contains the desired eigenvalue, where b_l and b_u are the lower and upper bound of the interval. Such an inclusion interval can be obtained from either the *LDU* factorization [2][3] or Levinson-Durbin algorithm [1][4].

An inclusion interval is obtained from the *LDU* factorization of the matrix $A - x_k I$ of the form

$$A - x_k I = LDL^* \qquad (1)$$

where A is a symmetric or Hermitian matrix, where L is a lower triangular matrix with 1's along their diagonals, $D = diag[d_1 \ d_2 \ \cdots \ d_n]$ is a diagonal matrix, and L^* is a complex conjugate transpose of L. For a given x_k, a constant function $m(x_k)$, which is called eigenvalue distribution and denotes the number of negative entries of D, implies that A has $m(x_k)$ eigenvalues that are strictly less than x_k. Thus, for the i-th eigenvalue, a valid inclusion interval must fulfill the conditions $m(b_l) = i - 1$ and $m(b_u) = i$. Until these two conditions are satisfied, the *LDU* factorizations have to be done repeatedly with different x_k. An *LDU* factorization requires $O\left(\frac{2}{3}n^3\right)$ operations.

The Levinson-Durbin algorithm can be used to locate the inclusion interval for a real symmetric or Hermitian Toeplitz matrix. The computational complexity of the Levinson-Durbin algorithm is $O(n^2)$. Consider the following Levinson-Durbin algorithm for a symmetric Toeplitz matrix $T = (t_{ij})$ where $t_{ij} = t_{|i-j|}$.

<u>Levinson-Durbin algorithm</u>

Initialization:

$\phi_{11}(x_k) = t_1/(t_0 - x_k)$,

$e_0 = t_0 - x_k$.

Compute:

For $2 \le k \le n-1$.

$e_k(x_k) = e_{k-1}(x_k)\left(1 - \left|\phi_{k-1,k-1}(x_k)\right|^2\right)$,

$$\phi_{kk}(x_k) = \frac{1}{e_k(x_k)}\left[t_k - \sum_{i=1}^{k-1}\phi_{i,k-1}(x_k)t_{k-i}\right], \quad (2)$$

$\phi_{ik}(x_k) = \phi_{i,k-1}(x_k) - \phi_{kk}(x_k)\overline{\phi}_{k-i,k-1}(x_k), \quad 1 \le i \le k-1$.

The $\phi_{ij}(x_k)$ satisfies the Yule-Walker equation.

$$\begin{bmatrix} t_0 & t_1 & \cdots & t_{j-1} \\ t_1 & t_0 & \cdots & t_{j-2} \\ \vdots & \vdots & \ddots & \vdots \\ t_{j-1} & t_{j-2} & \cdots & t_0 \end{bmatrix}\begin{bmatrix} \phi_{1j} \\ \phi_{2j} \\ \vdots \\ \phi_{jj} \end{bmatrix} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_j \end{bmatrix} \quad (3)$$

for $j = 1, \cdots, n-1$. Cybenko has shown that if $k \ge 2$ and

$$L_k(x_k) = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ -\phi_{1,k-1}(x_k) & 1 & \cdots & 0 & 0 \\ -\phi_{2,k-1}(x_k) & -\phi_{1,k-2}(x_k) & \cdots & 1 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -\phi_{k-1,k-1}(x_k) & -\phi_{k-2,k-2}(x_k) & \cdots & -\phi_{1,1}(x_k) & 1 \end{bmatrix}, \quad (4)$$

then

$$L_k^*(x_k)(T_k - x_k I_k)L_k(x_k) = diag\left(e_k(x_k),\ e_{k-1}(x_k),\ \cdots\ e_1(x_k)\right). \quad (5)$$

Because of Sylvester's law of inertia, this implies that the number of eigenvalues of T_k less than x_k equals the number of negative values in $diag\left(e_k(x_k),\ e_{k-1}(x_k),\ \cdots\ e_1(x_k)\right)$, provided that x_k is nondefective with respect to T_k. In order to ensure that any other eigenvalues of T_n or any other eigenvalues of T_{n-1} is not contained in the interval, we must refine the inclusion interval (b_l, b_u) to (b_l', b_u') by bisection and trial and error method until the following conditions hold:

(i) $m(b_l') = i-1$ and $m(b_u') = i$

(ii) $e_n(b_l) > 0$ and $e_n(b_u) < 0$

Those two methods, the *LDU* factorization and the Levinson-Durbin algorithm, tend to lose efficiency when the eigenvalues are placed very closely around the desired eigenvalue. Moreover they may not work when there exist multiple desired eigenvalues that are all identical. Of course, when all the eigenvalues are distinct and separated apart, then those methods work well.

3. MAIN RESULTS

Failure or inefficiency in computing multiple or clustered eigenvalues results from performing bisection shifts in search of eigenvalue inclusion interval (b_l^i, b_u^i) of the *i*-th eigenvalue. For multiple eigenvalue case, the conditions of existing algorithms described in the previous section are not hold, which should be satisfied to obtain proper inclusion interval. Consequently, the existing algorithms fail in computing the multiple eigenvalues.

On the other hand, although the conditions might hold for clustered eigenvalues, a great number of computations are required, which tends to cause all the algorithms to lose efficiency. To overcome this problem, two approaches are proposed in the next subsections.

3.1 The first approach

One method is to define the bound, $|b_u^i - b_l^i| < Lim$, to stop the inclusion interval finding procedure and computing an eigenvalue contained the interval. This method can compute the multiple eigenvalue accurately. In the case of the clustered eigenvalue, only one of the eigenvalues contained in the inclusion interval is calculated and the error might be proportional to the value of *Lim*. However, it is appropriate to consider eigenvalues contained in the interval to be multiple, since true multiplicities are reflected as an eigenvalue cluster in practice. To efficiently compute an eigenvalue in the interval containing multiple or clustered eigenvalues, we employ the GMRQI-JKL which is an efficient algorithm to compute any desired eigenvalue in a given interval [7]. Based on the above discussion, the algorithm is summarized below. *m(a)* in the algorithm denotes the number of eigenvalues of the matrix that are less than *a*.

Algorithm I

```
    Step 1-Select: Find the eigenvalues $\lambda_p, \lambda_{p+1}, \cdots, \lambda_q$, $1 \le p \le q \le n$.
Using trial and error, select an interval (a,b) by bisection such that $m(a) \le p-1$, $m(b) \ge q$.
    i=p
    while $i \le q$
    Step 2-Search: Search for the endpoint $b_{u_i}$ not captured by trial and error such that $(b_{l_i}, b_{u_i})$ contains $\lambda_i$. This is done by bisection via Levinson-Durbin algorithm or LDU decomposition. In the process, also detect, if any, the multiplicity m of multiple eigenvalues using the condition, $|b_{u_i} - b_{l_i}| < Lim$.
    Step 3-Locate: Once all the inclusion intervals are obtained:
        a) Switch to the GMRQI-JKL method to find $\lambda_i$.
        b) $i = i + m$
    End While Loop
```

Remark 1. In this algorithm, it is needless to refine the interval to assure the interval does not contain the eigenvalue of the principal submatrices, provided the middle point of the interval is not an eigenvalue of the principal submatrices, which is impossible in practical application. Actually, termination for this reason is never occurred in our simulation.

3.2 The second approach

Apart from the first approach, we solve the problem based on the Cauchy's eigenvalue interlacing theorem. According to the Cauchy's theorem, we can see that there exists an eigenvalue of the principal submatrix of order *n-m+1*, C_{n-m+1}, in the given interval, if the Hermitian matrix C_n of order *n* has an eigenvalue λ_i with multiplicity *m*. Therefore, we can find λ_i easily by working with the submatrix C_{n-m+1} if the multiplicity is known. The multiplicity can be found by employing the bound *Lim* similar with the first approach. A similar idea of this approach has been reported in [8]. However, their algorithm is not general, in the sense that their algorithm does not cover all kinds of Hermitian Toeplitz matrix. They asserted that *"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 with multiplicity m."* They proposed an algorithm to compute a multiple eigenvalue based on the above assertion. According to the assertion, their algorithm employed the matrix C_{n-m+1} in order to compute the multiple eigenvalues with multiplicity *m* of the matrix C_n. Unfortunately, the assertion is not true. If the assertion is true, the eigenvalue with multiplicity *m* in the matrix

C_n must have multiplicity one in the reduced-order matrix C_{n-m+1}. However, some kinds of matrix do not follow the assertion. Consider the following matrix C

$$C = \begin{bmatrix} 201 & 0 & 0 & 0 & 0 & 0 & 10 & 10 & 100 & 100 \\ 0 & 201 & 0 & 0 & 0 & 0 & 0 & 10 & 10 & 100 \\ 0 & 0 & 201 & 0 & 0 & 0 & 0 & 0 & 10 & 10 \\ 0 & 0 & 0 & 201 & 0 & 0 & 0 & 0 & 0 & 10 \\ 0 & 0 & 0 & 0 & 201 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 201 & 0 & 0 & 0 & 0 \\ 10 & 0 & 0 & 0 & 0 & 0 & 201 & 0 & 0 & 0 \\ 10 & 10 & 0 & 0 & 0 & 0 & 0 & 201 & 0 & 0 \\ 100 & 10 & 10 & 0 & 0 & 0 & 0 & 0 & 201 & 0 \\ 100 & 100 & 10 & 10 & 0 & 0 & 0 & 0 & 0 & 201 \end{bmatrix}.$$

The eigenvalues of the matrix are 34.7671, 145.6496, 199.4267, 200.3092, 201.0000, 201.0000, 201.6908, 202.5733, 256.3504, 367.2329. Multiplicity of the eigenvalue 201.0000 is 2. According to their assertion, the 9×9 principal submatrix C_9 has an eigenvalue 201.0000 with multiplicity 1. However the submatrix has eigenvalues 98.9341, 191.9798, 199.9138, 201.0000, 201.0000, 201.0000, 202.0862, 210.0202, 303.0659 and the multiplicity of the eigenvalue 201,0000 is 3. Note that the multiplicity of the eigenvalue is increased and the conditions of the *Step 3* in the algorithm of Noor and Morgera are not satisfied. In this case, their algorithm always fails. In our algorithm, we check the multiplicity, denoted by p, of the submatrix C_{n-m+1} and work with the submatrix $C_{n-m-p+2}$. This process is repeated until the multiple eigenvalues are not detected. Consequently, the Algorithm II overcomes the drawback of the Noor and Morgera's algorithm. Ours proposed as the second approach can be considered as a generalized version of the Noor and Morgera's algorithm. The algorithm is summarized as follows.

Algorithm II

```
Step 1-Select: Find the eigenvalues $\lambda_p, \lambda_{p+1}, \cdots, \lambda_q$, $1 \le p \le q \le n$.
Using trial and error, select an interval $(a,b)$ by bisection such that $m(a) \le p-1$, $m(b) \ge q$.
  $i=p$
  while $i \le q$
  Step 2-Search: Search for the endpoint $b_u$ not captured by trial and error such that $(b_l, b_u)$ contains $\lambda_i$. This is done by bisection via Levinson-Durbin algorithm or LDU decomposition.
  In the process, also detect, if any, the multiplicity $m$ of multiple eigenvalues using the condition, If $|b_u - b_l| < Lim$, Then flagmultiple=true.
  Step 3-Refine: Once all the intervals $b_l < \lambda_i < b_u$, $p \le i \le q$, are obtained:
    a) Set $\alpha = b_l$, $E_\alpha = E_n(b_l)$ and $\beta = b_u$, $E_\beta = E_n(b_u)$.
    b) For eigenvalue with multiplicity $m (1 \le m \le n)$,
       Repeat until flagmultiple=false
          Set the matrix order $n$ to $n-m+1$
          If $n=1$, then $\lambda_k = A_1$, $i \le k \le i+m-1$, and go to d).
          Refine the interval $(\alpha,\beta)$ to $(\alpha',\beta')$ with the submatrix
            $C_{n-m+1}$ such that the following condition (1) or (2)
            holds.
            Condition (1) - i) $m(\alpha') = i-1$ and $m(\beta') = i$
                           ii) $e_n(\alpha') > 0$ and $e_n(\beta') < 0$
            Condition (2) - $|\beta' - \alpha'| < Lim$
          If Condition (1) holds, then flagmultiple=false
                        else $m = m(\beta') - m(\alpha')$
       End Repeat Loop
    c) Switch to the MRQI [2][3] method to find $\lambda_i$.
    d) $i=i+m$
  End While Loop
```

4. SIMULATION RESULTS

Needless to say, the proposed algorithms work well on the matrix C introduced in section 3. Simulation result on the matrix C is summarized in Table 1. Table 1 shows that both methods identify multiple eigenvalues well. Note that the Noor and Morgera's algorithm always fails in that case since the reduced-order submatrix still has multiple eigenvalues. Algorithm I locates the true eigenvalues 100 percent correctly down to four places of decimals in this example. Algorithm II locates three eigenvalues slightly erroneously. However, it is not always the case with numerical accuracy. Through study on the accuracy is given in the next simulation.

On the other hand, in case that some eigenvalues of a matrix are clustered, we overcome the problem by employing the bound, denoted by *Lim*, in the algorithm and by considering eigenvalues contained in the interval (a,b) to be multiple when the condition $|b-a| < Lim$ is satisfied. As the second simulation, we examine the characteristic of the proposed algorithms for various value of *Lim*. As the first step, we construct a Hermitian Toeplitz matrix of order 100 having eigenvalues λ_k, $k = 1,2,\cdots,100$, where $\lambda_{k+1} = 1.2023\lambda_k$ and $\lambda_0 = 10^{-8}$. Thus, the matrix has 100 distinct eigenvalues theoretically. However, it is an example of clustered eigenvalue problem. Table 2 shows the distribution of the number of eigenvalues. Next, we apply the proposed algorithms to the matrix for various value of *Lim*. The simulation results are tabulated in Table 3 and 4. An important conclusion can be drawn from the tables; there is a computational burden-accuracy tradeoff. Therefore we must consider a permissible error bound to select a value of *Lim*. Figure 1 and 2 show the absolute error $\varepsilon = |\lambda_{true} - \lambda_{est}|$ of Algorithm I and II for various value of *Lim*, respectively. Note that the smaller is the magnitude of the eigenvalues, the greater is the effect of the *Lim*. For large eigenvalues, the variation of the *Lim* has little influence on the absolute error. Figure 3 shows the comparison results between the two algorithms on the number of operations and the average eigenvalue error. The Algorithm I has better accuracy as the bound *Lim* increased while the Algorithm II as the bound decreased. In the aspect of computational complexity, the Algorithm II has better performance for all value of *Lim*.

5. CONCLUSIONS

In this paper, we have proposed two algorithms for computing multiple or clustered eigenvalues. The algorithms overcame the drawback of the existing algorithm. The proposed algorithms have a tradeoff characteristic between accuracy and computational burden. A possible direction for future research would be to develop an algorithm which gives more accurate results with less computational effort for multiple or clustered eigenvalues.

REFERENCES

[1] G. Cybenko and C. Van. Loan, "Computing the minimum eigenvalue of a symmetric positive definite Toeplitz matrix," *SIAM J. Sci. Stat. Comput.*, vol. 7, pp. 123-131, 1986.

[2] Y. H. Hu and S. Y. Kung, "Toeplitz eigensystem solver," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-33, no. 4, pp. 1264-1271, Oct. 1985.

[3] Y. H. Hu, "Parallel eigenvalue decomposition for Toeplitz and related matrices," in *Proc. ICASSP '89,* Glasgow, Scotland, pp. 1107-1110, 1989.

[4] W. F. Trench, " Numerical solution of the eigenvalue problem for Hermitian Toeplitz matrices, " *SIAM J. Matrix Anal. Appl.*, vol. 10, no. 2, pp. 135-146, Apr. 1989.

[5] C. W. Jeon, H. J. Kim, and J. G. Lee, "Improvement of the Rayleigh quotient iteration method," *Proc. of the Korean Institute of Electrical Engineers Conference,* Seoul, Korea, pp. 319-321, Nov. 18, 1994.

[6] C. W. Jeon, H. J. Kim, and J. G. Lee, "A New Extreme Eigensystem Solver," *Proc. of the Control and Instrumentation Engineering Conference,* Chuncheon, Korea, pp. 131-135, April 29, 1995.

[7] C. W. Jeon, H. J. Kim, and J. G. Lee, "Efficient Algorithms for Computing eigenpairs for Hermitian Matrices," *Proc. of the Korean Institute of Electrical Engineers Conference,* Taejon, Korea, July 20-22, 1995.

[8] F. Noor and S. D. Morgera, " Recursive and iterative algorithms for computing eigenvalues of Hermitian Toeplitz matrices," *IEEE Trans. Signal Processing,* vol. 41, no.3, pp. 1272-1280, March 1993.

Table 1. Simulation results on matrix *C* by the generalized algorithm.

Eigen-values	True	Estimated	
		Algorithm I	Algorithm II
λ_1	34.7671	34.7671	34.7694
λ_2	145.6496	145.6496	145.6561
λ_3	199.4267	199.4267	199.4508
λ_4	200.3092	200.3092	200.2999
λ_5	201.0000	201.0000	201.0000
λ_6	201.0000	201.0000	201.0000
λ_7	201.6908	201.6908	201.6808
λ_8	202.5733	202.5733	202.5784
λ_9	256.3504	256.3504	256.3334
λ_{10}	367.2329	367.2329	367.2329

Table 2. Distribution of the number of eigenvalue of a matrix of order 100

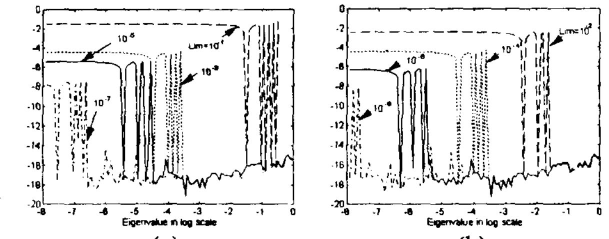
interval	0-10^{-7}	10^{-7}-10^{-6}	10^{-6}-10^{-5}	10^{-5}-10^{-4}	10^{-4}-10^{-3}	10^{-3}-10^{-2}	10^{-2}-10^{-1}	10^{-1}-1	amount
number	11	12	13	12	13	13	13	13	100

Table 3. Simulation results for various *Lim* on a matrix of order of 100 having distribution of the number of eigenvalue as shown in Table 2 via Algorithm I (unit:Mflops).

Lim	Number of Operations	Mean of Absolute Error
10^{-1}	93	6.3337e-003
10^{-2}	240	8.1147e-004
10^{-3}	341	1.8982e-004
10^{-4}	532	8.5131e-005
10^{-5}	665	7.0447e-005
10^{-6}	794	7.2647e-005
10^{-7}	976	6.9786e-005
10^{-8}	1,077	6.8064e-005

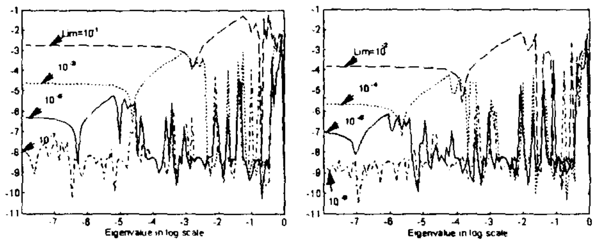
Table 4. Simulation results for various *Lim* on a matrix of order of 100 having distribution of the number of eigenvalue as shown in Table 2 via Algorithm II (unit:Mflops).

Lim	Number of Operations	Mean of Absolute Error
10^{-1}	28	2.7059e-002
10^{-2}	88	2.5904e-003
10^{-3}	150	2.5307e-004
10^{-4}	206	1.4598e-005
10^{-5}	249	1.1611e-006
10^{-6}	279	9.8427e-008
10^{-7}	320	2.1498e-009
10^{-8}	347	8.5081e-011



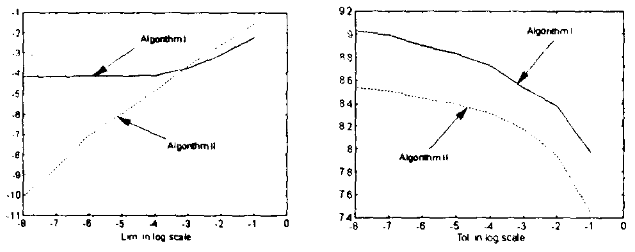
(a) (b)

Figure 1. Error performance of the Algorithm I for various *Lim*. (a) $Lim=10^{-1}$ (dash), $Lim=10^{-3}$ (dot), $Lim=10^{-5}$ (solid), $Lim=10^{-7}$(dash-dot). (b) $Lim=10^{-2}$ (dash), $Lim=10^{-4}$ (dot), $Lim=10^{-6}$ (solid), $Lim=10^{-8}$(dash-dot).



(a) (b)

Figure 2. Error performance of the Algorithm II for various *Lim*. (a) $Lim=10^{-1}$ (dash), $Lim=10^{-3}$ (dot), $Lim=10^{-5}$ (solid), $Lim=10^{-7}$(dash-dot). (b) $Lim=10^{-2}$ (dash), $Lim=10^{-4}$ (dot), $Lim=10^{-6}$ (solid), $Lim=10^{-8}$(dash-dot).



(a) (b)

Figure 3. Comparison results between the two algorithms. (a) Average eigenvalue error. (b) Computational complexity.