Univ of Texas, Dept of Computer Sciences Technical Report # TR-03-49

GENERALIZED FINITE ALGORITHMS FOR CONSTRUCTING HERMITIAN MATRICES WITH PRESCRIBED DIAGONAL AND SPECTRUM

INDERJIT S. DHILLON, ROBERT W. HEATH JR., MÁTYÁS A. SUSTIK AND JOEL A. TROPP

Abstract. In this paper, we present new algorithms that can replace the diagonal entries of a Hermitian matrix by any set of diagonal entries that majorize the original set without altering the eigenvalues of the matrix. They perform this feat by applying a sequence of $(N-1)$ or fewer plane rotations, where N is the dimension of the matrix. Both the Bendel-Mickey and the Chan-Li algorithms are special cases of the proposed procedures. Using the fact that a positive semi-definite matrix can always be factored as X^*X, we also provide more efficient versions of the algorithms that can directly construct factors with specified singular values and column norms. We conclude with some open problems related to the construction of Hermitian matrices with joint diagonal and spectral properties.

1. Introduction

It is sometimes of interest to construct a collection of Hermitian matrices that have specified diagonal elements and eigenvalues. When all the eigenvalues are non-negative, the problem is essentially equivalent to constructing a collection of rectangular matrices with specified column norms and singular values. In particular, if a rectangular matrix X has requirements on its singular values and squared column norms, the Hermitian matrix X^*X has a corresponding requirement on its eigenvalues and diagonal entries.

A specific example of this problem is to construct Hermitian matrices with unit diagonal and prescribed non-negative eigenvalues [3]. Such matrices are called correlation matrices — Davies and Higham discuss several applications that require such matrices, ranging from the generation of test matrices for eigenvalue solvers to the design of statistical experiments [3, 7, 9]. A related matrix construction problem has also arisen in connection with wireless communications. It turns out that $d \times N$ matrices, $d < N$, with d identical non-zero singular values and with prescribed column norms satisfy a certain "sum capacity" bound and "minimum squared correlation" property that is important in wireless applications. These matrices only exist if a majorization condition holds, as discussed in Section 2.1. For an introduction to squared correlation, see the Appendix. Refer to [20, 22, 23] for a thorough treatment of how the problem arises in wireless applications.

Two finite step techniques, the Bendel-Mickey [3] and Chan-Li [4] algorithms, are available for special cases. Both algorithms apply a sequence of plane rotations to an initial matrix that change its diagonal entries while preserving its spectrum. The Chan-Li algorithm starts with the diagonal matrix of eigenvalues and can reach a real, symmetric matrix with a specified majorizing diagonal. On the other hand, the Bendel-Mickey algorithm can start with an arbitrary Hermitian matrix and transform it to a Hermitian matrix with *equal* diagonal entries.

In this paper, we present new algorithms that generalize the Chan-Li and Bendel-Mickey procedures so that we can start with an arbitrary Hermitian matrix and change its diagonal entries to specified values while retaining its original spectrum. The only requirement is that the new diagonal

---

Date: November 20, 2003.

I. Dhillon and M. Sustik are with the Dept. of Comp. Sci., The University of Texas at Austin, Austin, TX 78712 USA, {inderjit|sustik}@cs.utexas.edu.

R. Heath is with the Dept. of Elect. and Comp. Engr., The University of Texas at Austin, Austin, TX 78712 USA, rheath@ece.utexas.edu.

J. Tropp is with the Inst. for Comp. Engr. and Sci. (ICES), The University of Texas, Austin, TX 78712 USA, jtropp@ices.utexas.edu.

elements majorize (in essence, average) the original ones. Thus our generalized algorithms permit us to construct a class of Hermitian matrices satisfying spectral and diagonal constraints that is richer than the collection yielded by older algorithms.

We now give a brief outline of the paper. In §2, we provide the necessary background and summarize previous results. We present our generalized algorithms in §3, and §4 contains some numerical examples. The Appendix discusses total squared correlation.

2. Background and Related Work

2.1. **Majorization.** The majorization relation is a partial ordering on vectors that appears in a striking number of apparently unrelated contexts. Lorenz originally developed the ordering for econometrics, where he used it to compare the equitability of income distributions [2]. An intuitive definition is that one vector majorizes another if the former has "more average" entries than the latter. Let us make this notion precise.

Definition 1. Let $\boldsymbol{a}$ be a real, N-dimensional vector, and denote its k-th smallest component by $a_{(k)}$. This number is called the k-th *order statistic* of $\boldsymbol{a}$.

Definition 2. Let $\boldsymbol{a}$ and $\boldsymbol{z}$ be real N-dimensional vectors, and suppose that their order statistics satisfy the following relationships.

$$\begin{aligned} a_{(1)} &\leq z_{(1)}, \\ a_{(1)} + a_{(2)} &\leq z_{(1)} + z_{(2)}, \\ &\vdots \\ a_{(1)} + a_{(2)} + \cdots + a_{(N-1)} &\leq z_{(1)} + z_{(2)} + \cdots + z_{(N-1)}, \qquad \text{and also} \\ a_{(1)} + a_{(2)} + \cdots + a_{(N)} &= z_{(1)} + z_{(2)} + \cdots + z_{(N)}. \end{aligned}$$

Then we say that $\boldsymbol{z}$ *majorizes* $\boldsymbol{a}$, and we write $\boldsymbol{z} \succcurlyeq \boldsymbol{a}$. If each of the inequalities is strict, then $\boldsymbol{z}$ *strictly majorizes* $\boldsymbol{a}$, and we write $\boldsymbol{z} \succ \boldsymbol{a}$.

It is not hard to verify that the majorization relation is reflexive, anti-symmetric, and transitive, so it defines a partial ordering on $\mathbb{R}^N$. An equivalent definition is that $\boldsymbol{z} \succcurlyeq \boldsymbol{a}$ if and only if $\boldsymbol{z} = M\,\boldsymbol{a}$ for some doubly-stochastic matrix M. Birkhoff's Theorem states that the collection of doubly-stochastic matrices of size N is identical with the convex hull of the permutation matrices having size N. It follows that those vectors which majorize a fixed vector form a compact, convex set. See [12, 16] for more details.

Majorization plays a role on our stage because it defines the precise relationship between the diagonal entries and eigenvalues of a Hermitian matrix.

Theorem 3 (Schur-Horn [12])**.** *The diagonal entries of a Hermitian matrix majorize its eigenvalues. Conversely, if $\boldsymbol{a} \succcurlyeq \boldsymbol{\lambda}$, then there exists a Hermitian matrix with diagonal entries listed by $\boldsymbol{a}$ and eigenvalues listed by $\boldsymbol{\lambda}$.*

I. Schur demonstrated the necessity of the majorization condition in 1923, and A. Horn proved the converse some thirty years later [12]. Horn's original proof is quite complicated, and a small cottage industry has grown up to produce simpler, more constructive arguments. See, for example, [4,6,15]. A comprehensive reference on majorization is Marshall and Olkin's monograph [16].

2.2. **Some Posets.** First, we define some concepts related to partial orderings, and then we develop some new partial orderings on Hermitian matrices that are closely related to the matrix construction problem.

Definition 4. A set S equipped with a partial ordering $\succcurlyeq$ is called a *poset*. It is denoted as $(S, \succcurlyeq)$. Two elements $a, b \in S$ are *comparable* if and only if $a \succcurlyeq b$ or $b \succcurlyeq a$. Any totally ordered subset of a poset is called a *chain*. Every pair of elements in a chain is comparable.

We may equip any poset with the ordering topology, where each basic open set is given by $\{a \neq b : b \succcurlyeq a\}$ for some point b. This is the minimal topology in which the ordering is continuous [14].

Observe that vectors are comparable by majorization only when their entries have the same sum. Let $\mathscr{C}_\alpha$ denote the set of N-dimensional vectors with trace α. Then each $\mathscr{C}_\alpha$ is an isolated component of the poset $(\mathbb{R}^N, \succcurlyeq)$. Moreover, every $\mathscr{C}_\alpha$ has a unique majorization-maximal element: the constant vector with entries α/N. On the other hand, there are no minimal vectors under the majorization relation; every $\boldsymbol{z}$ succeeds an infinite number of other vectors.

Definition 5. We say that two Hermitian matrices A and B are Schur-Horn equivalent if and only if they have identical spectra and identical diagonals (up to permutation). We write $A \equiv B$, and we use $[A]$ to denote the equivalence classes induced by this relation.

We indicate the collection of Schur-Horn equivalence classes by $\mathscr{H}$. Notice that the members of $\mathscr{H}$ vary significantly. For example, the Schur-Horn equivalence class of a diagonal matrix is the set of diagonal matrices with the same entries in permuted order. Meanwhile, the equivalence class of a matrix with unit diagonal and non-negative eigenvalues $\boldsymbol{\lambda}$ is the set of "correlation matrices" that have spectrum $\boldsymbol{\lambda}$ [7]. Even though similarity transformations preserve the eigenvalues of a Hermitian matrix, very few simultaneously preserve the diagonal. Therefore, Schur-Horn equivalence classes are not stable under most transformations. Exceptions include symmetric permutations and diagonal similarity transforms.

Definition 6. For any two elements of $\mathscr{H}$, $[A]$ and $[Z]$, we say that $[Z] \succcurlyeq [A]$ if and only if the two matrices have the same spectrum and $\operatorname{diag} Z \succcurlyeq \operatorname{diag} A$.

It is not hard to check that this construction yields a well-defined partial ordering on $\mathscr{H}$. Clearly, two Schur-Horn equivalence classes are comparable only if their members have the same spectrum. Suppose that the entries of $\boldsymbol{\lambda} \in \mathbb{R}^N$ already occur in non-decreasing order, viz. $\lambda_k = \lambda_{(k)}$ for each k. Then we may write $\mathscr{H}_{\boldsymbol{\lambda}}$ to denote the elements of $\mathscr{H}$ with spectrum $\boldsymbol{\lambda}$. Each $\mathscr{H}_{\boldsymbol{\lambda}}$ forms an isolated component of the poset $(\mathscr{H}, \succcurlyeq)$, and it has a unique maximal element: the equivalence class of matrices with eigenvalues $\boldsymbol{\lambda}$ and with a constant diagonal. A significant difference between majorization and the matrix ordering is that every chain under the matrix ordering has a minimal element: $[\operatorname{diag} \boldsymbol{\lambda}]$, where $\boldsymbol{\lambda}$ lists the (common) eigenvalues of the members of the chain.

2.3. **Algorithms.** Now we discuss two algorithms which have been proposed for constructing Hermitian matrices with diagonal and spectral properties. In the sequel, we use $\mathbb{M}_N$ to denote the set of complex $N \times N$ matrices and $\mathbb{M}_{d,N}$ to denote the set of complex $d \times N$ matrices.

The Bendel-Mickey algorithm produces random (Hermitian) correlation matrices with given spectrum [3]. Suppose that $A \in \mathbb{M}_N$ is a Hermitian matrix with $\operatorname{Tr} A = N$. If A does not have a unit diagonal, we can locate two diagonal elements so that $a_{jj} < 1 < a_{kk}$; otherwise, the trace condition would be violated. It is then possible to construct a real rotation Q in the jk-plane for which $(Q^* A Q)_{jj} = 1$. The transformation $A \mapsto Q^* A Q$ preserves the conjugate symmetry and the spectrum of A, but it reduces the number of non-unit diagonal entries by at least one. Therefore, at most $(N-1)$ rotations are required before the resulting matrix has a unit diagonal. If the output matrix is Z, it follows that $[Z] \succcurlyeq [A]$. Indeed, $[Z]$ is the unique $\succcurlyeq$-maximal element in every chain that contains $[A]$.

The Chan-Li algorithm, on the other hand, was developed as a constructive proof of the Schur-Horn Theorem [4]. Suppose that $\boldsymbol{a} \succcurlyeq \boldsymbol{\lambda}$. The Chan-Li algorithm begins with the diagonal matrix $\Lambda \stackrel{\text{def}}{=} \operatorname{diag} \boldsymbol{\lambda}$. Then it applies a sequence of $(N-1)$ cleverly chosen (real) plane rotations to generate a real, symmetric matrix A with the same eigenvalues as Λ but with diagonal entries listed by $\boldsymbol{a}$. Once again, the output and input satisfy the relationship $[A] \succcurlyeq [\Lambda]$. Where the Bendel-Mickey algorithm starts from any element of a chain and moves to the top, the Chan-Li algorithm starts at the bottom of a chain and moves upward.

The Bendel-Mickey algorithm is a surjective map from the set of Hermitian matrices with spectrum $\boldsymbol{\lambda}$ onto the set of correlation matrices with spectrum $\boldsymbol{\lambda}$. If the initial matrix is chosen uniformly at random (which may be accomplished with standard techniques [19]), the result may be construed as a random correlation matrix. The distribution of the output, however, is unknown [11]. On the other hand, due to the special form of the initial matrix and the rigid choice of rotations, the Chan-Li algorithm cannot construct very many distinct matrices with a specified diagonal. It would be interesting to develop a procedure which can calculate every member of a given equivalence class.

A brief discussion of how to use plane rotations to equalize the diagonal entries of a Hermitian matrix appears on page 77 of Horn and Johnson [12]. Problems 8.4.1 and 8.4.2 of Golub and van Loan outline the Bendel-Mickey algorithm [10]. Davies and Higham present a numerically stable version of the Bendel-Mickey algorithm in their article [7]. Other references on this topic include [13, 25].

3. Generalized Algorithms

We propose methods that generalize the Bendel-Mickey and Chan-Li algorithms. Like them, our techniques use a sequence of $(N-1)$ or fewer plane rotations to move upward between two points in a chain. The crux of the matter is the strategy for selecting the planes of rotation. The two methods we present can be viewed respectively as direct generalizations of the Chan-Li strategy and the Bendel-Mickey strategy. Unlike the earlier algorithms, these new techniques do not require ending at the top of a chain like Bendel-Mickey nor starting at the bottom like Chan-Li. Therefore, our techniques allow the construction of a much larger set of matrices than the Chan-Li algorithm, while retaining its ability to select the final diagonal entries.

3.1. **Generalized Chan-Li.** Let $\boldsymbol{z}$ and $\boldsymbol{a}$ be N-dimensional vectors for which $\boldsymbol{z} \succcurlyeq \boldsymbol{a}$. Using induction on the dimension, we show how to transform a Hermitian matrix with diagonal $\boldsymbol{a}$ and spectrum $\boldsymbol{\lambda}$ into a Hermitian matrix with diagonal $\boldsymbol{z}$ and spectrum $\boldsymbol{\lambda}$ using a sequence of plane rotations. It is enough to prove the result when the components of $\boldsymbol{a}$ and $\boldsymbol{z}$ are sorted in ascending order, so we place that restriction in the sequel.

Suppose first that $N = 2$ and that A has diagonal $\boldsymbol{a}$. Since $\boldsymbol{z} \succcurlyeq \boldsymbol{a}$, we have $a_1 \le z_1 \le z_2 \le a_2$. We can explicitly construct a real plane rotation Q so that the diagonal of Q^*AQ equals $\boldsymbol{z}$. Recall that a two-dimensional plane rotation is an orthogonal matrix of the form

$$Q = \begin{bmatrix} c & s \\ -s & c \end{bmatrix},$$

where $c^2 + s^2 = 1$ [10]. The desired plane rotation yields the matrix equation

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix}^* \begin{bmatrix} a_1 & a_{21}^* \\ a_{21} & a_2 \end{bmatrix} \begin{bmatrix} c & s \\ -s & c \end{bmatrix} = \begin{bmatrix} z_1 & z_{21}^* \\ z_{21} & \tilde{z}_2 \end{bmatrix}.$$

The equality of the upper-left entries can be stated as

$$c^2 a_1 - 2sc \operatorname{Re} a_{21} + s^2 a_2 = z_1.$$

This equation is quadratic in $t \stackrel{\text{def}}{=} s/c$:

$$(a_2 - z_1)\, t^2 - 2t \operatorname{Re} a_{21} + (a_1 - z_1) = 0, \tag{1}$$

whence

$$t = \frac{\operatorname{Re} a_{21} \pm \sqrt{(\operatorname{Re} a_{21})^2 - (a_1 - z_1)(a_2 - z_1)}}{a_2 - z_1}. \tag{2}$$

Notice that the discriminant is non-negative due to the majorization condition. The $\pm$ sign in (2) is taken to avoid sign cancellations with $\operatorname{Re} a_{21}$. If necessary, we can extract the other root of (1) using the fact that the product of its roots equals $(a_1 - z_1)/(a_2 - z_1)$. Finally, determine the parameters of the rotation using

$$c = \frac{1}{\sqrt{1+t^2}} \qquad \text{and} \qquad s = ct. \tag{3}$$

Floating-point arithmetic is inexact, so the rotation may not yield $(Q^*AQ)_{11} = z_1$. A better implementation sets this entry to z_1 explicitly. Davies and Higham have shown that this method of computing rotations is numerically stable [7]. Since Q is orthogonal, $Q^* A Q$ preserves the spectrum of A but replaces its diagonal with $\boldsymbol{z}$.

Grant us for a moment that we can perform the advertised feat on Hermitian matrices of size $(N-1)$. Now we consider N-dimensional vectors for which $\boldsymbol{z} \succcurlyeq \boldsymbol{a}$, and suppose that $\operatorname{diag} A = \boldsymbol{a}$. The majorization condition implies that $a_1 \le z_1 \le z_N \le a_N$, so it is always possible to select a least integer $j > 1$ so that $a_{j-1} \le z_1 \le a_j$. Let P_1 be a permutation matrix for which

$$\operatorname{diag}(P_1^* A P_1) = (a_1, a_j, a_2, \ldots, a_{j-1}, a_{j+1}, \ldots, a_N).$$

Observe that $a_1 \le z_1 \le a_j$ and $a_1 \le a_1 + a_j - z_1 \le a_j$. Thus we modify equations (2) and (3) to construct a two-dimensional plane rotation Q_2 that sets the upper left entry of

$$Q_2^* \begin{bmatrix} a_1 & a_{j1}^* \\ a_{j1} & a_j \end{bmatrix} Q_2$$

to z_1. If we define the rotation

$$P_2 \stackrel{\text{def}}{=} \begin{bmatrix} Q_2 & \mathsf{0}^* \\ \mathsf{0} & \mathsf{I}_{N-2} \end{bmatrix},$$

then

$$P_2^* P_1^* A P_1 P_2 = \begin{bmatrix} z_1 & \boldsymbol{v}^* \\ \boldsymbol{v} & A_{N-1} \end{bmatrix},$$

where $\boldsymbol{v}$ is an appropriate vector and A_{N-1} is an appropriate sub-matrix with

$$\operatorname{diag}(A_{N-1}) = (a_1 + a_j - z_1, a_2, \ldots, a_{j-1}, a_{j+1}, \ldots, a_N).$$

In order to apply the induction hypothesis, it remains to check that the vector $(z_2, z_3, \ldots, z_N)$ majorizes the diagonal of A_{N-1}. We accomplish this in three steps. First, recall that $a_k \le z_1$ for $k = 2, \ldots, j-1$. Therefore,

$$\sum_{k=2}^{m} z_k \ge (m-1)\, z_1 \ge \sum_{k=2}^{m} a_k$$

for each $m = 2, \ldots, j-1$. The sum on the right-hand side obviously exceeds the sum of the smallest $(m-1)$ entries of the vector $\operatorname{diag} A_{N-1}$, so the first $(j-2)$ majorization inequalities are in force. Second, we use the fact that $\boldsymbol{z} \succcurlyeq \boldsymbol{a}$ to calculate that, for $m = j, \ldots, N$,

$$\begin{aligned} \sum_{k=2}^{m} z_k &= \sum_{k=1}^{m} z_k - z_1 \\ &\ge \sum_{k=1}^{m} a_k - z_1 = (a_1 + a_j - z_1) + \sum_{k=2}^{j-1} a_k + \sum_{k=j+1}^{m} a_k. \end{aligned}$$

Once again, observe that the sum on the right-hand side exceeds the sum of the smallest $(m-1)$ entries of the vector $\operatorname{diag} A_{N-1}$, so the remaining majorization inequalities are in force. Finally, rearranging the relation $\sum_{k=1}^{N} z_k = \sum_{k=1}^{N} a_k$ yields $\sum_{k=2}^{N} z_k = \operatorname{Tr} A_{N-1}$.

In consequence, the induction furnishes a rotation Q_{N-1} that sets the diagonal of A_{N-1} equal to the vector $(z_2, \ldots, z_N)$. Defining

$$P_3 \stackrel{\text{def}}{=} \begin{bmatrix} 1 & \mathbf{0}^* \\ \mathbf{0} & Q_{N-1} \end{bmatrix},$$

we see that conjugating A by the orthogonal matrix $P = P_1 P_2 P_3$ transforms the diagonal entries of A to $\boldsymbol{z}$ while retaining the spectrum $\boldsymbol{\lambda}$.

This proof leads to the following algorithm.

Algorithm 1 (Generalized Chan-Li). *Let A be an $N \times N$ Hermitian matrix with diagonal $\boldsymbol{a}$, and let $\boldsymbol{z}$ be a vector such that $\boldsymbol{z} \succcurlyeq \boldsymbol{a}$, where both $\boldsymbol{a}$ and $\boldsymbol{z}$ are arranged in ascending order. The following procedure computes a Hermitian matrix with diagonal entries $\boldsymbol{z}$ and eigenvalues equal to that of A.*

(1) Set $n = 1$.

(2) Find the least $j > n$ so that $a_{j-1,j-1} \le z_n \le a_{jj}$.
(3) Use a symmetric permutation to set $a_{n+1,n+1}$ equal to a_{jj} while shifting diagonal entries $n+1, \dots, j-1$ one place down the diagonal.
(4) Construct a plane rotation Q in the $(n, n+1)$-plane using equations (2) and (3) with appropriate modifications.
(5) Replace A by Q^*AQ.
(6) Use a symmetric permutation to re-sort the diagonal entries of A in ascending order.
(7) Increment n, and repeat Steps 2–7 while $n < N$.

This algorithm requires about $12N^2$ floating-point operations if conjugate symmetry is exploited. It requires the storage of about $N(N+1)/2$ floating-point numbers, including the vector $\boldsymbol{z}$. It is conceptually simpler to perform the permutations described in the algorithm, but it can be implemented without them. The bookkeeping just becomes more laborious. The MATLAB code in the Appendix demonstrates a simple implementation where such bookkeeping is used instead of permutations.

3.2. **Generalized Bendel-Mickey.** Distinct algorithms arise by changing the strategy for selecting the planes of rotation. Let $\boldsymbol{z}$ and $\boldsymbol{a}$ be N-dimensional vectors for which $\boldsymbol{z} \succcurlyeq \boldsymbol{a}$. As before, we assume that they are sorted in ascending order, and suppose that A is a Hermitian matrix with diagonal $\boldsymbol{a}$. We now exhibit a different method for transforming the diagonal of A to $\boldsymbol{z}$ while preserving its eigenvalues. It can be viewed as a generalization of the Bendel-Mickey algorithm [3].

Suppose that $\operatorname{diag} A \ne \boldsymbol{z}$. On account of the majorization relationship, it is possible to select indices $i < j$ that satisfy two properties: $a_{ii} < z_i \le z_j < a_{jj}$ and $a_{kk} = z_k$ for all k strictly between i and j. If $z_i - a_{ii} \le a_{jj} - z_j$, then we construct a plane rotation Q in the (i,j)-plane such that $(Q^*AQ)_{ii} = z_i$. Otherwise, we find Q such that $(Q^*AQ)_{jj} = z_j$. Either rotation can be calculated using appropriate versions of equations (2) and (3). To see that this strategy can be repeated, we just need to check that $\boldsymbol{z}$ majorizes the diagonal of Q^*AQ. In the first case, the plane rotation transforms a_{ii} to z_i and a_{jj} to $a_{ii} + a_{jj} - z_i$, while the remaining diagonal entries do not change. Since $a_{ii} < z_i \le z_j \le a_{ii} + a_{jj} - z_i < a_{jj}$ the diagonal entries of Q^*AQ remain in ascending order. The first $(i-1)$ majorization conditions are clearly unaffected. Notice that

$$\sum_{\ell=1}^{i-1} a_{\ell\ell} + z_i \le \sum_{\ell=1}^{i-1} z_\ell + z_i,$$

which proves the i-th majorization condition. The next $(j-i-1)$ majorization inequalities follow in consequence of a_{kk} being equal to z_k whenever $i < k < j$. The rest of the majorization conditions hold since

$$\sum_{\ell=1}^{i-1} a_{\ell\ell} + z_i + \sum_{k=i+1}^{j-1} a_{kk} + (a_{ii} + a_{jj} - z_i) = \sum_{\ell=1}^{j} a_{\ell\ell} \le \sum_{\ell=1}^{j} z_\ell.$$

The argument in the case when $z_i - a_{ii} > a_{jj} - z_j$ is similar. It follows that our rotation strategy may be applied until $\operatorname{diag} A = \boldsymbol{z}$. This proof leads to the following algorithm.

Algorithm 2 (Generalized Bendel-Mickey). *Let A be an $N \times N$ Hermitian matrix with diagonal $\boldsymbol{a}$ and furthermore let $\boldsymbol{z}$ be a vector such that $\boldsymbol{z} \succcurlyeq \boldsymbol{a}$, where both $\boldsymbol{a}$ and $\boldsymbol{z}$ are arranged in ascending order. The following procedure computes a Hermitian matrix with diagonal entries $\boldsymbol{z}$ and eigenvalues equal to that of A.*

(1) Find $i < j$ for which $a_{ii} < z_i$ and $z_j < a_{jj}$ and $a_{kk} = z_k$ for $i < k < j$ (in our implementation we pick the smallest such i). If no such pair exists, we are either done ($\boldsymbol{z} = \boldsymbol{a}$) or the majorization condition is violated.
(2) Construct a plane rotation Q in the (i,j)-plane using equations (2) and (3) with appropriate modifications to transform a_{ii} to z_i in the case $z_i - a_{ii} \le a_{jj} - z_j$ or transform a_{jj} to z_j otherwise.

(3) Replace A by Q^*AQ.
(4) Repeat Steps 1–3 until the diagonal is transformed to $\boldsymbol{z}$.

This algorithm has the same functionality and complexity as Algorithm 1 but it is different in the plane rotations used. For further comparison we provide a simple MATLAB implementation in the Appendix.

3.3. **One-Sided Algorithms.** It is well known that any positive semi-definite matrix $A \in \mathbb{M}_N$ can be expressed as the product X^*X where $X \in \mathbb{M}_{d,N}$ and $d \geq \operatorname{rank} A$. With this factorization, the two-sided transformation $A \mapsto Q^*AQ$ is equivalent to a one-sided transformation $X \mapsto XQ$. In consequence, the machinery of Algorithm 1 requires little adjustment to produce these factors.

Algorithm 3 (One-sided generalized Chan-Li)**.** *Suppose that $\boldsymbol{z}$ and $\boldsymbol{a}$ are non-negative vectors of length N with ascending entries. Assume, moreover, that $\boldsymbol{z} \succcurlyeq \boldsymbol{a}$. The following algorithm takes as input a $d \times N$ matrix X whose squared column norms are listed by $\boldsymbol{a}$ and transforms it into a matrix with the same singular spectrum and with squared column norms listed by $\boldsymbol{z}$.*

(1) Let $n = 1$.
(2) Find the least $j > n$ so that $\|\boldsymbol{x}_{j-1}\|_2^2 \leq z_n \leq \|\boldsymbol{x}_j\|_2^2$.
(3) Move the j-th column of X to the $(n+1)$-st column, shifting the displaced columns to the right.
(4) Form the quantities

$$a_{nn} = \|\boldsymbol{x}_n\|_2^2, \qquad a_{n,n+1} = \langle \boldsymbol{x}_{n+1}, \boldsymbol{x}_n \rangle \qquad \text{and} \qquad a_{n+1,n+1} = \|\boldsymbol{x}_{n+1}\|_2^2 .$$

(5) Compute a plane rotation Q in the $(n, n+1)$-plane using modified versions of equations (2) and (3).
(6) Replace X by XQ.
(7) Sort columns $(n+1), \ldots, N$ in order of increasing norm.
(8) Increment n, and repeat Steps 2–7 while $n < N$.

The algorithm requires about $12dN$ real floating-point operations and storage of $N(d+2)$ real floating-point numbers including the desired column norms and the current column norms. As before, the procedure can be implemented without any permutations.

A similar modification of our generalized Bendel-Mickey algorithm also leads to a one-sided version. The latter generalizes the one-sided version proposed by Davies and Higham in [7].

4. Illustrative Numerical Examples

We illustrate the generalized Chan-Li algorithm by comparing it with the classical algorithm on two examples. The generalized algorithms can produce a richer set of matrices with prescribed diagonal entries and eigenvalues, making it possible to find solutions that satisfy additional properties or better suit the application.

Suppose we want to produce a Hermitian matrix with eigenvalues $(1, 4, 5, 7, 9)$ and diagonal entries $(2, 5, 6, 6, 7)$. This example was presented in the Chan-Li paper [4]; our generalized algorithm (essentially) yields the same result:

$$A^{(1)}_{\mathrm{ChanLi}} = \begin{bmatrix} 2.0000 & 0 & 0.7071 & -0.9487 & 0.7746 \\ 0 & 5.0000 & 0 & 0 & 0 \\ 0.7071 & 0 & 6.0000 & 1.3416 & -1.0954 \\ -0.9487 & 0 & 1.3416 & 6.0000 & 2.4495 \\ 0.7746 & 0 & -1.0954 & 2.4495 & 7.0000 \end{bmatrix}.$$

Notice the sparsity structure in the above matrix. In applications, such as designing matrices for testing eigenvalue solvers [9], it would be better to produce a more random Hermitian matrix that satisfies the diagonal and eigenvalue constraints. The generalized algorithms can be used for this purpose. We first generate a sequence of six vectors satisfying:

$$(1,4,5,7,9) = z_0 \preccurlyeq z_1 \preccurlyeq z_2 \preccurlyeq z_3 \preccurlyeq z_4 \preccurlyeq z_5 = (2,5,6,6,7).$$

The z_1, z_2, z_3, z_4 vectors are randomly generated as convex combinations of the prescribed z_0 and z_5 vectors. This construction makes sure that the majorization conditions hold. Five steps of the generalized Chan-Li algorithm are successively used to transform the diagonal matrix diag (z_0) to have diagonals $z_1, z_2, \ldots, z_5$. We arrive at the final matrix

$$A^{(1)}_{\mathrm{GenChanLi}} = \begin{bmatrix} 2.0000 & 1.0400 & 1.4517 & -0.6294 & -0.3720 \\ 1.0400 & 5.0000 & 0.3620 & -0.2157 & 1.4731 \\ 1.4517 & 0.3620 & 6.0000 & 1.6901 & -0.6544 \\ -0.6294 & -0.2157 & 1.6901 & 6.0000 & -1.2822 \\ -0.3720 & 1.4731 & -0.6544 & -1.2822 & 7.0000 \end{bmatrix}.$$

For the wireless application mentioned in the introduction, the matrices in question must have all non-zero eigenvalues equal to one. (See the Appendix for more details.) The following example calls for the generation of matrices with eigenvalues $(0,0,1,1,1)$ and diagonal $(0.4, 0.6, 0.6, 0.6, 0.8)$. The Chan-Li algorithm produces

$$A^{(2)}_{\mathrm{ChanLi}} = \begin{bmatrix} 0.4000 & -0.4899 & 0 & 0 & 0 \\ -0.4899 & 0.6000 & 0 & 0 & 0 \\ 0 & 0 & 0.6000 & 0.4000 & -0.2828 \\ 0 & 0 & 0.4000 & 0.6000 & 0.2828 \\ 0 & 0 & -0.2828 & 0.2828 & 0.8000 \end{bmatrix}.$$

In the wireless application, it is often desirable to have lower variance in the magnitudes of the off-diagonal entries, which are also known as "cross-correlations". The generalized Chan-Li algorithm applied in five steps as described above produces a more desirable matrix:

$$A^{(2)}_{\mathrm{GenChanLi}} = \begin{bmatrix} 0.4000 & -0.2312 & -0.3503 & 0.1636 & 0.1926 \\ -0.2312 & 0.6000 & 0.1116 & 0.3955 & -0.1331 \\ -0.3503 & 0.1116 & 0.6000 & -0.1681 & 0.2768 \\ 0.1636 & 0.3955 & -0.1681 & 0.6000 & 0.1691 \\ 0.1926 & -0.1331 & 0.2768 & 0.1691 & 0.8000 \end{bmatrix}.$$

Here is an alternate strategy to construct a richer set of Hermitian matrices with prescribed eigenvalues and diagonal entries. Generate random matrices with the given spectrum and select those which have a diagonal majorized by the target diagonal. Then apply either one of the generalized algorithms.

Conclusions

We have shown that a sequence of $(N-1)$ rotations is sufficient to replace the original diagonal of $N \times N$ Hermitian matrix with any set of diagonal entries that majorizes the original set, all the while preserving the spectrum of the matrix. The algorithms we have presented can move up a chain in the poset of Schur-Horn equivalence classes as given in Definition 5.

An obvious question is whether it is possible to obtain an algorithm that moves down a chain instead. In other words, is it possible to construct a finite sequence of rotations to replace the diagonal with a set of entries that majorizes the eigenvalues but not necessarily the original diagonal? Since the diagonal matrix of eigenvalues lies at the bottom of the chain, it might seem at first glance that we are attempting to calculate the eigenvalues in finite time. We avoid this paradox since we assume that the target diagonal is already known, In fact, to get to the bottom of the chain, $O(n^2)$ Givens rotations can be used to reduce the initial matrix to tridiagonal form and then to the desired diagonal matrix of eigenvalues (these include the application of perfect shifts to the tridiagonal matrix [8]). Nevertheless, it seems much harder to construct a transformation of a vector into one of its predecessors than into one of its successors. Entropy may provide a reasonable explanation: it is easier to average things together than to un-average them.

Other interesting questions arise. What is the structure of a general Schur-Horn equivalence class of Hermitian matrices? Is there a procedure to construct every member of a given equivalence class? Is it possible to define a uniform probability measure on each class and to construct members from a class uniformly at random? In this paper, we have restricted our attention to finite step algorithms. Iterative algorithms are an alternative, especially for the case of producing Hermitian matrices that satisfy additional constraints. It would be useful to understand these problems better, and we hope that other researchers will take interest.

Acknowledgments

Part of this research was supported by NSF CAREER Award No. ACI-0093404.

References

[1] P. Anigstein and V. Anantharam, *Ensuring convergence of the MMSE iteration for interference avoidance to the global optimum*, IEEE Trans. Inform. Th., 49 (2003), pp. 873–885.
[2] B. C. Arnold, *Majorization and the Lorenz Order: A Brief Introduction*, no. 43 in Lecture Notes in Statistics, Springer-Verlag, Berlin, 1987.
[3] R. B. Bendel and M. R. Mickey, *Population correlation matrices for sampling experiments*, Commun. Statist. Simul. Comp., B7 (1978), pp. 163–182.
[4] N. N. Chan and K.-H. Li, *Diagonal elements and eigenvalues of a real symmetric matrix*, J. Math. Anal. Appl., 91 (1983), pp. 562–566.
[5] O. Christensen, *An introduction to frames and Riesz bases*, Birkhäuser, Boston, 2003.
[6] M. T. Chu, *Constructing a Hermitian matrix from its diagonal entries and eigenvalues*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 207–217.
[7] P. I. Davies and N. J. Higham, *Numerically stable generation of correlation matrices and their factors*, BIT, 40 (2000), pp. 640–651.
[8] I. S. Dhillon and A. N. Malyshev, *Inner deflation for symmetric tridiagonal matrices*, Lin. Alg. Appl., 358 (2002), pp. 139–144.
[9] Z. Drmač, *Accurate computation of the product-induced singular value decomposition with applications*, SIAM J. Num. Anal., 35 (1998), pp. 1969–1994.
[10] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, 3rd ed., 1996.
[11] R. B. Holmes, *On random correlation matrices*, SIAM Journal on Matrix Analysis and Applications, 12 (1991), pp. 239–272.
[12] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, 1985.
[13] K. D. Ikramov, *A remark on "A note on constructing a symmetric matrix with specified diagonal entries and eigenvalues"*, BIT, 38 (1998), p. 807.
[14] J. L. Kelley, *General Topology*, Van Nostrand, 1975.
[15] R. S. Leite, R. W. Richa, and C. Tomei, *Geometric proofs of some theorems of Schur-Horn type*, Linear Algebra and Appl., 286 (1999), pp. 149–173.
[16] A. W. Marshall and I. Olkin, *Inequalities: Theory of Majorization and its Applications*, Academic Press, 1979.
[17] J. L. Massey and T. Mittelholzer, *Welch's Bound and sequence sets for code division multiple-access systems*, in Sequences II: Methods in communcation, security and computer science, Springer-Verlag, 1994, pp. 63–78.
[18] C. Rose, *CDMA codeword optimization: Interference avoidance and convergence via class warfare*, IEEE Trans. Inform. Th., 47 (2001), pp. 2368–2382.
[19] G. W. Stewart, *The efficient generation of random orthogonal matrices with an application to condition estimation*, SIAM J. Numer. Anal., 17 (1980), pp. 403–409.
[20] J. A. Tropp, I. S. Dhillon, and R. W. Heath, *Finite-step algorithms for constructing optimal CMDA signature sequences*, Wireless Networking and Communications Group Technical Report WNCG-TR-2003-05-08, The University of Texas at Austin, 2003. Available from `http://www.ece.utexas.edu/~rheath/papers/2003/finitestep/paper.pdf`.
[21] S. Ulukus and R. D. Yates, *Iterative construction of optimum signature sequence sets in synchronous CDMA systems*, IEEE Trans. Inform. Th., 47 (2001), pp. 1989–1998.
[22] P. Viswanath and A. Anantharam, *Optimal sequences for CDMA under colored noise: A Schur-saddle function property*, IEEE Trans. Inform. Th., 48 (2002), pp. 1295–1318.
[23] P. Viswanath and V. Anantharam, *Optimal sequences and sum capacity of synchronous CDMA systems*, IEEE Trans. Inform. Th., 45 (1999), pp. 1984–1991.

[24] L. R. WELCH, *Lower bounds on the maximum cross-correlation of signals*, IEEE Trans. Inform. Th., 20 (1974), pp. 397–399.

[25] H. ZHA AND Z. ZHANG, *A note on constructing a symmetric matrix with specified diagonal entries and eigenvalues*, BIT, 35 (1995), pp. 448–452.

Appendix A. Appendix

A.1. **Total Squared Correlation.** The purpose of this appendix is to provide a short explanation of the eigenvalue property that motivated us to pursue this work. Let X be a $d \times N$ matrix with fixed column norms. Its *total squared correlation* is defined by the formula

$$\mathrm{TSC}(X) \stackrel{\text{def}}{=} \|X^* X\|_{\mathrm{F}}^2 = \sum_{m,n=1}^{N} |\langle \boldsymbol{x}_m, \boldsymbol{x}_n \rangle|^2,$$

where the vectors $\boldsymbol{x}_1, \dots, \boldsymbol{x}_N$ indicate the columns of X. Roughly speaking, the total squared correlation measures how similar these vectors are to each other. Matrices with minimal TSC are important for code-division multiple access (CDMA) systems. This connection is developed at length in [1, 20], and it serves as the basis for several iterative algorithms [18, 21]. Here, we shall only demonstrate that matrices with minimal TSC have nearly constant singular values and discuss some implications.

Suppose that the numbers $w_1, \dots, w_N$ represent the squared column norms of X, which are *fixed*, and write $W = \sum_{n=1}^{N} w_n$. We shall use the symbols $\sigma_1, \dots, \sigma_k$ to denote the d largest singular values of X.

Proposition 1. *Minimizing the total squared correlation of X with the column norms fixed is the same as minimizing*

$$\sum_{k=1}^{d} (\sigma_k^2 - W/d)^2$$

In words, the TSC reaches its minimum when the squared singular values of X are as constant as possible. A lower bound on the total squared correlation is W^2/d. This bound is attained if and only if the non-zero singular values of X are identically equal to $\sqrt{W/d}$.

If the column norms are identically one, the inequality reduces to a famous result called the Welch Bound [24]. The necessary and sufficient condition for attainment is due to Massey-Mittelholzer [17].

Proof. The squared Frobenius norm of X can be written in two fundamentally different ways:

$$\sum_{n=1}^{N} w_n = \|X\|_{\mathrm{F}}^2 = \sum_{k=1}^{d} \sigma_k^2.$$

Meanwhile, we can write the TSC as

$$\mathrm{TSC}(X) = \|X^* X\|_{\mathrm{F}}^2 = \sum_{k=1}^{d} \sigma_k^4.$$

To complete the first part of the proof, perform the expansion

$$\sum_{k=1}^{d} (\sigma_k^2 - W/d)^2 = \sum_{k=1}^{d} \sigma_k^4 - \frac{2W}{d} \sum_{k=1}^{d} \sigma_k^2 + \frac{W^2}{d}.$$

The last two terms of the right-hand side are constant. It follows that minimizing the left-hand member of the equation is the same as minimizing $\sum \sigma_k^4$, the total squared correlation.

To produce a lower bound on the TSC, we minimize $\sum c_k^4$ over arbitrary positive numbers $c_1, \dots, c_d$ that satisfy the constraint $\sum c_k^2 = W$. Using the theory of Lagrange multipliers, it is straightforward to check that the minimum occurs if and only if the numbers $c_1^2, \dots, c_d^2$ are all equal. According to the sum constraint, they must all equal W/d. It follows that the sum of their squares equals W^2/d. $\square$

The diagonal entries of $X^* X$ are equal to the squared column norms of X, and the eigenvalues of $X^* X$ are equal to the squared singular values of X. As discussed in Section 2.1, the numbers must always satisfy a majorization condition. In the special case that the non-zero singular values of X are constant, the majorization relation reduces to a condition on the column norms:

$$0 \leq w_n \leq W/d \qquad \text{for } n = 1, \dots, N.$$

If this condition fails, the singular values of X cannot be equal, and the lower bound of Proposition 1 cannot be met.

A $d \times N$ matrix, $d < N$, whose d singular values are equal is called a *tight frame*. Tight frames provide a natural generalization of unitary matrices, and, as we have seen, they also arise from minimizing the TSC. If X is a tight frame, then X^*X is a Hermitian matrix whose non-zero eigenvalues are identical. A Hermitian matrix whose non-zero eigenvalues all equal one is called an *orthogonal projector*, so minimizing the TSC is also equivalent to constructing a (scaled) orthogonal projector with a specified diagonal. For an introduction to tight frames, see [5].

A.2. **Matlab Code.** MATLAB code for the generalized Chan-Li algorithm (Algorithm 1):

```
function [A, Q] = genchanli(A, z);
%GENCHANLI Implements the generalized Chan-Li algorithm.
%        [A, Q] = GENCHANLI(A, z) calculates Q and replaces A by
%        Q'AQ such that diag(Q'AQ) = z provided that A is a Hermitian
%        matrix and z majorizes diag(A).

% The algorithm operates on the upper triangular part of A, the lower triangular
% part is calculated at the end of the procedure for convenience.

[N, m] = size(A);
z = sort(z);
[d, Perm] = sort(diag(A));
% Now A(Perm(i), Perm(i)) i = 1, 2, ..., N is in ascending order.
if (A(Perm(1), Perm(1)) > z(1))
    error('Majorization condition violated');
end
if (nargout == 2)
    Q = eye(N);
end

for n = 1:N-2
    % Find the smallest j > n with:
    % A(Perm(j-1), Perm(j-1)) <= z(n) <= A(Perm(j), Perm(j)).
    j = n + 1;
    while (j <= N & z(n) > A(Perm(j), Perm(j)))
        j = j + 1;
    end
    if (j == N + 1 | A(Perm(j-1), Perm(j-1)) > z(n))
        error('Majorization condition violated');
    end
    % Transform the diagonal entry in row Perm(n) to z(n):
    [c, s] = drotug(A, Perm(n), Perm(j), z(n));
    A = drotcu(A, Perm(n), Perm(j), c, s, z(n));
    if (nargout == 2)
        Q = drot(Q, Perm(n), Perm(j), c, s);
    end
    % A has been changed so we need to update Perm accordingly.
    % First find the new place for the j-th element A(Perm(j), Perm(j)):
    i = n + 1;
    while (i < j & A(Perm(i), Perm(i)) < A(Perm(j), Perm(j)))
        i = i + 1;
    end
    % A(Perm(j), Perm(j)) needs to move to the ith position:
    temp = Perm(j);
    Perm(i+1:j) = Perm(i:j-1);
    Perm(i) = temp;
    % Note that Perm records the order of the diagonal A except the first n
    % entries (which have already been processed).  In the subsequent iterations
    % only Perm(n+1:N) is used.
end
```

```
% The last rotation:
[c, s] = drotug(A, Perm(N-1), Perm(N), z(N-1));
A = drotcu(A, Perm(N-1), Perm(N), c, s, z(N-1));
if (nargout == 2)
    Q = drot(Q, Perm(N-1), Perm(N), c, s);
end

% Copy the elements above the diagonal to the other half of the matrix for
% convenience:
for i = 1:N-1
    A(i+1:N, i) = A(i, i+1:N)';
end
```

Matlab code for the generalized Bendel-Mickey algorithm (Algorithm 2):

```
function [A, Q] = genbendelmickey(A, z, tol)
%GENBENDELMICKEY Implements the generalized Bendel-Mickey algorithm.
%        [A, Q] = GENBENDELMICKEY(A, z, tol) calculates Q and replaces A by
%        Q'AQ such that diag(Q'AQ) = z provided that A is a Hermitian matrix
%        and z majorizes diag(A) and z and diag(A) are in ascending order.

% The algorithm operates on the upper triangular part of A, the lower
% triangular part is calculated at the end of the procedure for convenience.

[m, N] = size(A);
if (nargout == 2)
    Q = eye(N);
end

% The following tolerance is necessary to ensure that no false reports of
% violations of the majorization condition is generated due to loss of
% precision.
if (nargin == 2)
    tol = 2*N*max(abs(z(N)), abs(z(1)))*eps;
end

% The algorithm attempts to find i < j indices such that the following two
% conditions hold: A(i, i) < z(i) <= z(j) < A(j, j) and A(k, k) = z(k) for
% all i < k < j.  A plane rotation in the (i, j) plane transforms A such that
% either A(i, i) or A(j, j) becomes equal to z(i) or z(j) respectively.  If
% the majorization condition is satisfied, this step can be repeated until
% diag(A) = z.  The first (i, j) pair is separately calculated:
i = 0;
j = 1;
while (j <= N & z(j) >= A(j, j))
    if (z(j) > A(j, j))
        i = j;
    end
    j = j + 1;
end
while (1 <= i & j <= N)
    % Decide which diagonal element can be made equal to the corresponding
```

```
    % z vector element and use a plane rotation for the transformation:
    if (z(i) - A(i, i) <= A(j, j) - z(j))
        [c, s] = drotug(A, i, j, z(i));
        A = drotcu(A, i, j, c, s, z(i));
        if (nargout == 2)
            Q = drot(Q, i, j, c, s);
        end
    else
        [c, s] = drotug(A, j, i, z(j));
        A = drotcu(A, j, i, c, s, z(j));
        if (nargout == 2)
            Q = drot(Q, j, i, c, s);
        end
    end
    % Find the new pair:
    while (j <= N & z(j) > A(j, j) - tol)
        if (z(j) > A(j, j))
            i = j;
        end
        j = j + 1;
    end
    while (i >= 1 & z(i) < A(i, i) + tol)
        i = i - 1;
    end
end
if (i >= 1 | j <= N)
    error('The majorization condition is violated.');
end
% Copy the elements above the diagonal to the other half of the matrix for
% convenience.
for i = 1:N-1
    A(i+1:N, i) = A(i, i+1:N)';
end
```

Auxiliary functions to calculate and apply plane rotations:

```
function [c, s] = drotug(A, i, j, z)
%DROTUG  Generate a plane rotation for a Hermitian matrix.
%        [c, s] = DROTUG(A, i, j, z) calculates the parameters of a plane
%        rotation Q such that conjugating with Q transforms A(i, i) to z.
%        The A matrix is assumed to be Hermitian and only its upper triangular
%        part is used in the calculation.

a1 = A(i, i);
a2 = A(j, j);
if (i < j)
    b = real(A(i, j));
else
    b = real(A(j, i));
end

D = b^2 - (a1 - z)*(a2 - z);
if (b > 0)                          % To avoid cancellations.
    t = b + sqrt(D);
else
    t = b - sqrt(D);
end

if (a2 == z)
    c = 0;
    s = 1;
else
    t = t/(a2 - z);
    c = 1/sqrt(1 + t^2);
    s = c*t;
end;
return;

function A = drot(A, i, j, c, s)
%DROT    Apply a plane rotation.
%        A = DROT(A, i, j, c, s) replaces A by AQ where the Q plane rotation
%        has parameters [c, s] and acts in the (i, j) plane.

A(:, [i, j]) = A(:, [i, j])*[c, s; -s, c];
return;

function A = drotcu(A, i, j, c, s, z)
%DROTCU  Conjugates with a plane rotation.
%        A = DROTCU(A, i, j, c, s, z) replaces A by Q'AQ where the Q plane
%        rotation acts in the (i, j) plane and has parameters c and s.  The A
%        matrix is assumed to be Hermitian and only its upper triangular part
%        is used in the calculation.

[m, N] = size(A);

if (i < j)
```

```
    A(1:i-1, [i, j]) = A(1:i-1, [i, j])*[c, s; -s, c];
    tmp(i+1:j-1)  = c*A(i, i+1:j-1) - s*A(i+1:j-1, j)';
    A(i+1:j-1, j) = s*A(i, i+1:j-1)' + c*A(i+1:j-1, j);
    A(i, i+1:j-1) = tmp(i+1:j-1);
    A([i, j], j+1:N) = [c, -s; s, c]*A([i, j], j+1:N);
    A(i, j) = s*c*(A(i, i) - A(j, j)) + c^2*A(i, j) - s^2*A(i, j)';
else
    A(1:j-1, [i, j]) = A(1:j-1, [i, j])*[c, s; -s, c];
    tmp(j+1:i-1)  = c*A(j+1:i-1, i) - s*A(j, j+1:i-1)';
    A(j, j+1:i-1) = s*A(j+1:i-1, i)' + c*A(j, j+1:i-1);
    A(j+1:i-1, i) = tmp(j+1:i-1)';
    A([i, j], i+1:N) = [c, -s; s, c]*A([i, j], i+1:N);
    A(j, i) = s*c*(A(i, i) - A(j, j)) + c^2*A(j, i) - s^2*A(j, i)';
end
A(j, j) = A(j, j) + A(i, i) - z;
A(i, i) = z;
return;
```