SIAM J. Matrix Anal. Appl.
Vol. 30, No. 4, pp. 1658–1676

FAST CONDITION ESTIMATION FOR A CLASS OF STRUCTURED EIGENVALUE PROBLEMS*

A. J. LAUB† AND J. XIA‡

Abstract. We present a fast condition estimation algorithm for the eigenvalues of a class of structured matrices. These matrices are low rank modifications to Hermitian, skew-Hermitian, and unitary matrices. Fast structured operations for these matrices are presented, including Schur decomposition, eigenvalue block swapping, matrix equation solving, compact structure reconstruction, etc. Compact semiseparable representations of matrices are used in these operations. We use these operations in a new, improved version of the statistical condition estimation method for eigenvalue problems. The estimation algorithm costs $O(n^2)$ flops for all eigenvalues, instead of $O(n^3)$ as in traditional algorithms, where n is the order of the matrix. The algorithm provides reliable condition estimates for both eigenvalues and eigenvalue clusters. The proposed structured matrix operations are also useful for additional eigenvalue problems and other applications. Numerical examples are used to illustrate the reliability and efficiency of the algorithm.

Key words. statistical condition estimation, low rank modification, sequentially semiseparable structure, structured Schur decomposition, diagonal block swapping, structured Sylvester equation

AMS subject classifications. 65F15, 65F30, 65F35

DOI. 10.1137/070707713

1. Introduction.

The condition number of an eigenvalue or eigenvalue cluster measures the sensitivity of the eigenvalue or eigenvalue cluster to small changes in the input matrix, and it may be used to bound the error in the computed approximation. For a general order-n matrix, it usually costs $O(n^3)$ flops or more to estimate the sensitivity of all its eigenvalues, for example, by considering separations of eigenvalues, considering angles between left eigenvectors and right eigenvectors, and other methods [1], [6], [19], [21], [28]. For structured eigenvalue problems, on the one hand, it is important to capture the structures [13]. We can take advantage of the structures to obtain fast condition estimation. The development of new fast structured algorithms for these matrices in recent years makes it possible to obtain fast condition estimation. In this paper, we consider condition estimation for the eigenvalues of a class of structured matrices, and we present a reliable estimation scheme which costs only $O(n^2)$ flops. This class of matrices has the following structures:

1. Low rank modifications to Hermitian and skew-Hermitian matrices. Examples include Frobenius matrices, diagonal plus rank one matrices, and arrowhead matrices which arise in applications such as bidiagonal SVD, divide-and-conquer algorithms for some eigenvalue problems [18], etc.
2. Low rank modifications to unitary matrices such as companion matrices, which are closely related to the problems of finding polynomial roots and solving certain differential equations.

Any such matrix $C \in \mathbb{R}^{n\times n}$ is a low rank perturbation to a *rank symmetric* matrix [3] (a matrix $\hat{C}$ is said to be rank symmetric if for any 2×2 block partition of

*Received by the editors November 9, 2007; accepted for publication (in revised form) by M. Van Barel September 9, 2008; published electronically January 23, 2009.
http://www.siam.org/journals/simax/30-4/70771.html

†Department of Electrical Engineering and Department of Mathematics, University of California, Los Angeles, CA 90095 (laub@ucla.edu).
‡Department of Mathematics, Purdue University, West Lafayette, IN 47907 (xiaj@math.purdue.edu).

$\hat{C} = \left[\begin{smallmatrix} \hat{C}_{11} & \hat{C}_{12} \\ \hat{C}_{21} & \hat{C}_{22} \end{smallmatrix}\right]$ with $\hat{C}_{11}$ and $\hat{C}_{22}$ square, the ranks of $\hat{C}_{12}$ and $\hat{C}_{21}$ are equal). That is,

$$C = \hat{C} + xy^T, \tag{1.1}$$

where $x, y \in \mathbb{C}^{n\times k}$ with $k \ll n$, and $\hat{C}$ is rank symmetric. Some fast methods for finding the eigenvalues of these matrices have been proposed in recent years (see, e.g., [3], [4], [5], [11], [16]). These methods exploit certain rank structure of the QR iterates when using QR iterations to find the eigenvalues. In this paper we show that the rank structure can also be used to accelerate the condition estimation of the eigenvalues.

As an example, the companion matrix

$$C = \begin{bmatrix} -a_{n-1} & -a_{n-2} & \cdots & -a_0 \\ 1 & 0 & \cdots & 0 \\ & \ddots & \ddots & \vdots \\ 0 & & 1 & 0 \end{bmatrix} \tag{1.2}$$

can be written as

$$C = \begin{bmatrix} 0 & \cdots & 0 & \pm 1 \\ 1 & 0 & \cdots & 0 \\ & \ddots & \ddots & \vdots \\ 0 & & 1 & 0 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} a_{n-1} & a_{n-2} & \cdots & a_0 \mp 1 \end{bmatrix}.$$

To quickly find the eigenvalues of (1.2), the fast structured QR iteration algorithm in [11] computes structured QR iterates which are unitarily similar to C. The iterations are done via the Q and R factors of the QR iterates. It can be shown that the QR iterates have small off-diagonal ranks [3], [5], [11]. Thus the Q and R factors can be efficiently represented by rank structures called *sequentially semiseparable* (SSS) matrix forms, proposed in [7], [8], [9]. An SSS matrix looks like

$$\begin{bmatrix} \mathcal{D}_1 & \mathcal{U}_1\mathcal{V}_2^T & \mathcal{U}_1\mathcal{W}_2\mathcal{V}_3^T & \mathcal{U}_1\mathcal{W}_2\mathcal{W}_3\mathcal{V}_4^T \\ \mathcal{P}_2\mathcal{Q}_1^T & \mathcal{D}_2 & \mathcal{U}_2\mathcal{V}_3^T & \mathcal{U}_2\mathcal{W}_3\mathcal{V}_4^T \\ \mathcal{P}_3\mathcal{R}_2\mathcal{Q}_1^T & \mathcal{P}_3\mathcal{Q}_2^T & \mathcal{D}_3 & \mathcal{U}_3\mathcal{V}_4^T \\ \mathcal{P}_4\mathcal{R}_3\mathcal{R}_2\mathcal{Q}_1^T & \mathcal{P}_4\mathcal{R}_3\mathcal{Q}_2^T & \mathcal{P}_4\mathcal{Q}_3^T & \mathcal{D}_4 \end{bmatrix}, \tag{1.3}$$

where the matrices $\{\mathcal{D}_i\}$, $\{\mathcal{U}_i\}$, $\{\mathcal{W}_i\}$, $\{\mathcal{V}_i\}$, $\{\mathcal{P}_i\}$, $\{\mathcal{R}_i\}$, $\{\mathcal{Q}_i\}$ are called (SSS) generators. SSS structures are useful for problems where the off-diagonal blocks have small ranks (see, e.g., [10], [11]). When the off-diagonal ranks of an SSS matrix are small and the sizes of $\{\mathcal{W}_i\}$ and $\{\mathcal{R}_i\}$ are close to the off-diagonal ranks, the matrix is said to be in *compact* SSS form. In such a situation, the matrix can be represented by only a linear amount of data, and operations on the compact SSS matrices are very efficient. For example, it costs only linear time to solve compact SSS linear systems and to multiply two compact SSS matrices with the same partition. More details can be found in [7], [8], [9], [10], [11]. The use of compact SSS matrices in the algorithm in [11] provides an $O(n^2)$ cost eigensolver for companion matrices (and polynomial root problems).

1.1. Main results. This paper shows that we can also exploit the rank structure of the above class of eigenvalue problems (1.1) to provide efficient condition estimation for the eigenvalues. We use the *statistical condition estimation* (SCE) method by Kenney and Laub [23]. In [19], a perturbation analysis for the *average eigenvalues* of a general matrix based on SCE has been given, and an SCE condition estimator is provided. The cost is $\mathrm{O}(n^3)$ flops for all eigenvalues. Here, we first improve the estimator in [19] from various points of view. Then we take into account the rank structure of the above class of matrices in SCE and extend the estimator in [19] to all the eigenvalues or eigenvalue clusters of these matrices. Given the facts that the related matrix computations become structured, and that SCE is good at respecting matrix structures, we can reduce the total condition estimation cost for all eigenvalues to $\mathrm{O}(n^2)$.

We present the main idea based on the companion matrix (1.2). For convenience, we consider only real matrices and maintain real arithmetic in this paper. Similar techniques can be easily extended to other matrices in the above class. This is discussed in section 4. In this paper, the following structured matrix algorithms are used or developed:

1. Use the existing algorithm in [11] to compute a compact SSS Schur decomposition of C in $\mathrm{O}(n^2)$ flops. Improve the algorithm so that the maximum generator size is smaller than the original one in [11] (see the next item).
2. Preserve *quasi-triangular* (a block upper triangular matrix with 1×1 and/or 2×2 diagonal blocks) Schur form when bringing any desired eigenvalue or eigenvalue cluster to any other position. This is done by swapping the contiguous 1×1 or 2×2 diagonal blocks of the Schur form in structured form. Quickly update the initial Schur form to recover a new compact Schur form. A recovery procedure in [11] is improved (with even less cost) so that it also works for quasi-triangular matrices, and the computational rank result is consistent with the theoretical prediction as well. The total cost for all eigenvalues is $\mathrm{O}(n^2)$ flops.
3. Represent certain Sylvester equations in structured forms. Quickly solve a structured Sylvester equation for each eigenvalue so that the total cost for all eigenvalues is $\mathrm{O}(n^2)$ instead of $\mathrm{O}(n^3)$.
4. Use structured perturbation in SCE and evaluate all the condition estimates in $\mathrm{O}(n^2)$ flops by taking advantage of matrix structures. Efficiently reconstruct certain invariant subspaces.

Our estimator works for both simple or multiple eigenvalues or eigenvalue clusters. The paper [26] presents some similar work. However, [26] requires that the eigenvalues of C all be distinct. The new operations here are more general, more efficient, and even simpler to implement than those in [26]. For example, after the diagonal block swapping, [26] uses SSS matrix multiplications to get the SSS representation of the new Schur form. But when the matrix has multiple eigenvalues, many SSS multiplications may be needed and the SSS Schur form may not be compact anymore. Instead, here we use a recovery strategy which always guarantees that the SSS Schur form is compact. As another example, here we simplify the reconstruction of the invariant subspace corresponding to an eigenvalue or eigencluster after the diagonal block swapping.

Similar techniques can also speed up the condition estimation for the eigenvectors of (1.1). We emphasize that the structured matrix operations in this paper are also useful in many other problems, in addition to condition estimation. Condition estimation for the eigenvalues of a companion matrix C can be used to assess the accuracy of polynomial roots including multiple or clustered roots.

1.2. Overview and notation. The rest of this paper is organized as follows. Section 2 reviews SCE for general eigenvalue problems and gives some new improvements. The fast structured condition estimation scheme is presented in section 3 in detail. Related matrix algorithms are derived. We also briefly discuss the extension of the techniques to general matrices (1.1) in the above class. Section 4 provides the algorithm, together with detailed flop counts, and shows some numerical examples. We draw some concluding remarks in section 6.

The following notation is used in this paper:

- The ith row (or block row) and the jth column (or block column) of $A \equiv (A_{ij})_{n\times n}$ are denoted by $A_{i,:}$ and $A_{:,j}$, respectively. Similarly, $A_{1:i,1:j}$ denotes the submatrix of A at (block) rows 1 through i and (block) columns 1 through j.
- $\mathrm{vec}(A)$ denotes the column vector formed by stacking the columns of A from left to right.
- If A is an SSS matrix, $\mathcal{D}_i(A)$, $\mathcal{U}_i(A)$, etc. represent its SSS generators as in (1.3).
- δA means the product of a small scalar δ with A.
- If a vector u is selected uniformly and randomly from the unit n-sphere S_{n-1}, we write $u \in \mathrm{U}(S_{n-1})$.

2. Condition estimation for general eigenvalue problems.

2.1. General SCE scheme for average (mean) eigenvalues. For a general $n \times n$ real matrix C, Gudmundsson, Kenney, and Laub derived an SCE condition estimator for its average or mean eigenvalues in the following way [19]. Assume we have a block Schur decomposition of C,

$$C = UTU^T, \quad U = [U_1, U_c], \quad T = \begin{bmatrix} T_1 & H \\ 0 & T_c \end{bmatrix}, \tag{2.1}$$

where U is orthogonal and T_1 and T_c have orders n_1 and $n - n_1$, respectively. The average eigenvalue of T_1 is defined to be [1], [19]

$$\mu(T_1) = \frac{\mathrm{trace}(T_1)}{n_1}.$$

If the spectra of T_1 and T_c are well separated [22], [31], then the sensitivity of $\mu(T_1)$ is well defined. A condition number κ for $\mu(T_1)$ is given in [19].

In SCE, a condition estimate for $\mu(T_1)$ can be obtained by perturbing C to $C + \delta E$ with a relative perturbation matrix δE, where δ is a small number, and $E = (C_{ij}Z_{ij})_{n\times n}$ with $Z = (Z_{ij})_{n\times n}$ satisfying $\mathrm{vec}(Z) \in \mathrm{U}(S_{n^2-1})$. Accordingly, $\mu(T_1)$ is perturbed to [19]

$$\mu(\tilde{T_1}) \approx \mu(T_1 + \delta B) = \mu(T_1) + \delta\mu(B), \tag{2.2}$$

where

$$B = U_1^T E U_1 + Y U_c^T E U_1, \tag{2.3}$$

and Y is an $n_1 \times (n - n_1)$ matrix satisfying a Sylvester equation

$$T_1 Y - Y T_c = H. \tag{2.4}$$

Based on (2.2), SCE leads to a relative condition estimate to $\mu(T_1)$ in the following form:

$$\nu = \frac{1}{\omega_p |\mu(T_1)|} |\mu(B)|, \tag{2.5}$$

where p is the number of parameters that define C (for a general $n \times n$ matrix, $p = n^2$; for the companion matrix (1.2), $p = n$), and ω_p is the Wallis factor which can be approximated by [23]

$$\omega_p \approx \sqrt{\frac{2}{\pi(p - \frac{1}{2})}}.$$

The expected value of the estimate $\mathrm{E}(\nu)$ is equal to the exact condition number κ [19].

Multiple samples of Z can be used to increase the accuracy of the estimation. For example, assume we use m samples of Z, denoted $Z^{(i)}$, $i = 1, 2, \dots, m$, which are properly orthonormalized [23], and accordingly, T_1 is perturbed to $T_1 + \delta B^{(i)}$, $i = 1, 2, \dots, m$. Then the m-sample condition estimator is defined as

$$\nu^{(m)} = \frac{1}{\omega_p |\mu(T_1)|} \sqrt{[\mu(B^{(1)})]^2 + \cdots + [\mu(B^{(m)})]^2}.$$

The accuracy of this estimator is given by [19]

$$\Pr\left(\frac{\kappa}{\gamma} \le \nu^{(m)} \le \gamma\kappa\right) \ge 1 - \frac{1}{m!}\left(\frac{2m}{\gamma\pi}\right)^m + \mathrm{O}\left(\frac{1}{\gamma^{m+1}}\right), \qquad \gamma > 1.$$

For example, with $m = 2$, the probability of $\nu^{(m)}$ being within a factor of $\gamma = 10$ of the exact condition number κ is greater than 0.9919. Even with only one sample, this probability is greater than 0.9363.

2.2. Improvements. We make several improvements over Gudmundsson, Kenney, and Laub's general SCE estimator for average eigenvalues. First, for simple real eigenvalues and complex eigenpairs, more specific forms based on (2.3) and (2.5) can be derived. When T_1 is a 1×1 block (eigenvalue), B in (2.3) is reduced to a scalar which can be calculated by using Y (a vector) and the first row and the first column of U. When T_1 is a 2×2 block, T_1 has a conjugate pair of complex eigenvalues. The condition number of this eigenpair is generally different from the condition number of their average. Thus, (2.5) may not precisely reflect the sensitivity of this eigenpair. In fact, by assuming

$$T_1 \equiv \begin{bmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{bmatrix}, \qquad B \equiv \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix},$$

we can derive a more accurate estimator for the actual condition of the eigenpair [26]

$$\nu = \frac{1}{\omega_p \sqrt{\det(T_1)}} \sqrt{\frac{\alpha^2 \det(T_1) - \alpha\beta \mathrm{trace}(T_1) + \beta^2}{[\mathrm{trace}(T_1)]^2 - 4\det(T_1)}},$$

where $\alpha = b_{11} + b_{22}$, $\beta = t_{11}b_{22} + t_{22}b_{11} - t_{12}b_{21} - t_{21}b_{12}$.

The second improvement is that, for average eigenvalues corresponding to a diagonal block T_i other than T_1, we can employ diagonal block swapping techniques as in

[1], [12], [27] to obtain a new Schur decomposition such that T_i (in its similar form) appears in the leading (upper left) position of the new Schur form

$$\tilde{T} = GTG^T, \tag{2.6}$$

where G is an orthogonal transformation matrix.

Another improvement is to rewrite (2.3) as

$$\begin{aligned} B &= \begin{bmatrix} I_{n_1} & Y \end{bmatrix} U^T E U_1 \\ &= \begin{bmatrix} I_{n_1} & Y \end{bmatrix} (U^T E U) \begin{bmatrix} I_{n_1} \\ 0 \end{bmatrix}, \end{aligned} \tag{2.7}$$

where E is isolated from Y. The new representation (2.7) indicates that the operations involving E can be independent of different eigenvalues. In order to compute B for different eigenvalues, we can precompute the matrix $U^T EU$ just once. Then for different eigenvalues we need only solve for Y and then compute the trace of the left $n_1 \times n_1$ submatrix of

$$\hat{B} = \begin{bmatrix} I_{n_1} & Y \end{bmatrix} (U^T E U), \tag{2.8}$$

where appropriate transformations may be applied to $U^T EU$ (see section 4). On the other hand, if multiple samples of E are used, then we can reuse the matrices Y and U. This will be further discussed in section 4.

Finally, for structured matrices C, the perturbation matrix E may also be structured, and it is also possible to compute Y and B quickly by taking advantage of the structure of C. The total cost of the condition estimation can then be less than $O(n^3)$. This is the actual situation for the class of matrices (1.1) where only $O(n^2)$ flops are needed. We elaborate on this in the rest of this paper.

Remark 2.1. The algorithm in this paper can be used to estimate the condition of polynomial roots. For the companion matrix (1.2), it can be shown that the exact condition number κ for $\mu(T_1)$ is actually the condition number defined in [14], [15] for the roots of the polynomial $\sum_{i=0}^{n} a_i x^i$ (with $a_n \equiv 1$) when all roots are distinct [26].

3. Fast structured condition estimation.

For a structured matrix C in (1.1), the perturbation matrix E generally corresponds to the low rank modifications (see, e.g., [18] for an error analysis based on perturbing the low rank part of a diagonal plus rank one matrix). In such a situation, the relative perturbation matrix E has the form

$$E = xE_2^T + E_1 y^T = \begin{bmatrix} x & E_1 \end{bmatrix} \begin{bmatrix} E_2^T \\ y^T \end{bmatrix} \equiv \hat{x}\hat{y}^T, \tag{3.1}$$

where $E_1 = (x_{ij} Z_{ij}^x)_{n \times k}$, $E_2 = (y_{ij} Z_{ij}^y)_{n \times k}$ with Z^x and Z^y random matrices satisfying $\mathrm{vec}([Z^x, Z^y]) \in \mathrm{U}(S_{2nk-1})$. For the example of the companion matrix (1.2), E can be further simplified to

$$E = \begin{array}{c} \begin{bmatrix} e^T \\ 0 \end{bmatrix} \end{array} \begin{array}{c} 1 \\ n-1 \end{array}, \tag{3.2}$$

where $e^T = [-a_{n-1} z_{n-1}, -a_{n-2} z_{n-2}, \ldots, -(a_0 \mp 1) z_0]$ and $[z_{n-1}, \ldots, z_0]^T \in \mathrm{U}(S_{n-1})$. This is because, usually, $a_{n-1}, a_{n-2}, \ldots, a_0$ are the parameters of interest.

The special structure of E saves the cost of computing B. Moreover, based on the rank structure of C and its similarity transformations, all the major steps in the SCE scheme can be quickly done by structured matrix computations. They include

1. structured Schur decomposition in (2.1),
2. structured diagonal block swapping in (2.6),
3. structured Sylvester equation solution for (2.4), and
4. evaluation of $\mu(B)$ in (2.5) using the low rank structure.

We discuss the details in the following subsections.

3.1. Structured Schur decomposition. We can find a structured Schur decomposition of C by using the fast structured eigensolver in [11]. The traditional Hessenberg QR iterations for the eigenvalues of C are

$$\begin{aligned}&C^{(0)} = C,\\ &C^{(k)} = Q^{(k)}R^{(k)},\ C^{(k+1)} = R^{(k)}Q^{(k)}, \qquad k = 0,1,2,\ldots.\end{aligned}$$

Clearly, any $C^{(k)}$ is a rank one update to an orthogonal matrix since C is. The QR algorithm in [11] is based on the result that each $C^{(k)}$ actually has small off-diagonal ranks.

THEOREM 3.1 (see [3], [4], [11]). *The ranks of all off-diagonal blocks $C^{(k)}_{1:j,j+1:n}$, $j = 1,\ldots,n-1$, of $C^{(k)}$ are no larger than* 3.

The algorithm in [11] uses a sequence of Givens matrices for $Q^{(k)}$ and an SSS form $R^{(k)}$. When the algorithm converges, it yields a quasi-triangular Schur form T. Appropriate Givens matrices form an orthogonal matrix U such that $C = UTU^T$, which is a structured form of (2.1). That is, U is a product of $\mathrm{O}(n^2)$ Givens matrices in general, and T is an SSS matrix.

Clearly, T also has off-diagonal ranks bounded by 3. However, we can compute the QR factorization $T = QR$, where Q is a block diagonal matrix with 1×1 or 2×2 diagonal blocks and R is also a rank one update to an orthogonal matrix. The matrix R has maximum off-diagonal rank bounded by 2 with a proof similar to that of Theorem 3.1 [3], [11]. The application of Q to R does not increase this rank. Thus, we have the following result.

THEOREM 3.2. *For any quasi-triangular matrix orthogonally similar to C, its maximum off-diagonal rank is no larger than* 2.

The algorithm in [11] can be used to obtain a compact SSS form of T with maximum generator size 3. This algorithm can be improved by using the techniques in subsection 3.3.2 so that the maximum generator size of T is 2.

3.2. Structured Sylvester equation solver. The Sylvester equation (2.4) can be solved in different ways. It can be converted into a linear system using Kronecker products. Alternatively, since T_1 and T_c are quasi triangular, the Bartels–Stewart algorithm [2] can be applied conveniently. However, both methods cost $\mathrm{O}(O(n_1(n-n_1)^2+n_1^2(n-n_1)))$ if T_1 and T_c are general quasi-triangular matrices. This makes the total condition estimation cost as large as $\mathrm{O}(n^3)$. Notice that T_1 and T_c are also SSS matrices. We can reduce the Sylvester equation solution cost to $\mathrm{O}(n_1(n-n_1))$ by a structured form of the Bartels–Stewart algorithm. This consists of three parts: making one coefficient matrix lower quasi triangular, quickly formulating certain (simpler) linear or Sylvester systems in the Bartels–Stewart algorithm, and quickly solving those systems.

3.2.1. Transforming the Sylvester equation. We first make one of the two coefficient matrices lower quasi triangular. Since T_1 is block upper triangular, we employ a permutation matrix P such that

$$L \equiv PT_1P^T$$

is lower quasi triangular. The matrix P is simply the anti-identity matrix

$$P = \begin{bmatrix} 0 & & 1 \\ & \iddots & \\ 1 & & 0 \end{bmatrix}.$$

Then (2.4) can be written as

$$L(PY) - (PY)T_c = PH. \tag{3.3}$$

It is clear that the SSS generators of L can be obtained directly from those of T_1 as follows:

$$\mathcal{D}_i(L) = \mathcal{D}_i(T_1)^T, \quad \mathcal{P}_i(L) = \mathcal{V}_i(T_1), \quad \mathcal{Q}_i(L) = \mathcal{U}_i(T_1), \quad \mathcal{R}_i(L) = \mathcal{W}_i(T_1)^T.$$

For notational convenience, we write (3.3) as the SSS Sylvester equation

$$LX - XR = K, \tag{3.4}$$

where L is a block lower triangular compact SSS matrix with generators $\{\mathcal{D}_i(L) \equiv L_{ii}\}_1^{l_1}$, $\{\mathcal{P}_i\}_2^{l_1}$, $\{\mathcal{Q}_i\}_1^{l_1-1}$, and $\{\mathcal{R}_i\}_2^{l_1-1}$, and R is a block upper triangular compact SSS matrix with generators $\{\mathcal{D}_i(R) \equiv R_{ii}\}_1^{l_2}$, $\{\mathcal{U}_i\}_1^{l_2-1}$, $\{\mathcal{V}_i\}_2^{l_2}$, and $\{\mathcal{W}_i\}_2^{l_2-1}$. Here, $l_1 + l_2 = l$ is the total number of diagonal blocks (eigenvalue clusters) in T.

3.2.2. Formulating (simpler) systems in the Bartels–Stewart algorithm. Next, we apply the Bartels–Stewart algorithm to (3.4). Since H is an off-diagonal block of T, we assume $K = PH$ has the form

$$\begin{aligned} K &= \begin{bmatrix} K_{:,1} & K_{:,2} & \cdots & K_{:,l_2} \end{bmatrix} \\ &= \begin{bmatrix} u_1 w_2 \cdots w_{l_1} v_{l_1+1}^T & u_1 w_2 \cdots w_{l_1+1} v_{l_1+2}^T & \cdots & u_1 w_2 \cdots w_{l-1} v_l^T \\ \vdots & \vdots & & \vdots \\ u_{l_1-1} w_{l_1} v_{l_1+1}^T & u_{l_1-1} w_{l_1} w_{l_1+1} v_{l_1+2}^T & \cdots & u_{l_1-1} w_{l_1} \cdots w_{l-1} v_l^T \\ u_{l_1} v_{l_1+1}^T & u_{l_1} w_{l_1+1} v_{l_1+2}^T & \cdots & u_{l_1} w_{l_1+1} \cdots w_{l-1} v_l^T \end{bmatrix}. \end{aligned} \tag{3.5}$$

We also assume that all the matrices in (3.4) have conformal partitions. When we get the solution X of (3.4), the solution of (2.4) can be simply obtained by $Y = P^T X$.

The Bartels–Stewart algorithm solves (3.4) by successively solving

$$L_{ii} X_{ij} - X_{ij} R_{jj} = K_{ij} - \sum_{k=1}^{i-1} L_{ik} X_{kj} + \sum_{k=1}^{j-1} X_{ik} R_{kj},$$
$$i = 1, 2, \ldots, l_1, \qquad j = 1, 2, \ldots, l_2,$$

where L_{ii} and R_{jj} are 1×1 or 2×2 blocks, and X_{ij} denotes the (i, j) block of X, which is partitioned conformally according to the blocks of L and R. These equations can be rewritten as a set of linear equations or (simpler) Sylvester equations

$$LX_{:,j} - X_{:,j} R_{jj} = \hat{K}_j, \qquad j = 1, 2, \ldots, l_2, \tag{3.6}$$

where $\hat{K}_j = K_{:,j} + X_{:,1:j-1} R_{1:j-1,j}$, and $X_{:,j}$ has one or two columns. Since L is an $n_1 \times n_1$ SSS matrix and R_{jj} is a 1×1 or 2×2 block, we can solve (3.6) for each j in

$O(n_1)$ flops, provided that the right-hand side $\hat{K}_j$ can be evaluated in $O(n_1)$ flops. In fact, the evaluation of both $K_{:,j}$ and $X_{:,1:j-1}R_{1:j-1,j}$ for all $j = 1, \ldots, l_2$ can be done successively as follows (when $j = 1$, let $X_{:,1:j-1}R_{1:j-1,j} \equiv 0$).

For $K_{:,j}$, $j = 1, \ldots, l_2$ in (3.5), introduce auxiliary matrices Ω_k defined by

$$\Omega_{l_1} = I,\ \Omega_k = w_{k+1}\Omega_{k+1}, \qquad k = l_1 - 1,\ l_1 - 2, \ldots, 2, 1. \tag{3.7}$$

Then compute each block K_{kj} by

$$K_{kj} = u_k \Omega_k v_j^T, \qquad k = 1, 2, \ldots, l_1. \tag{3.8}$$

After the calculation of each $K_{:,j}$, replace all Ω_k by

$$\hat{\Omega}_k = \Omega_k w_{l_1+j}. \tag{3.9}$$

For $X_{:,1:j-1}R_{1:j-1,j}$, $j = 1, \ldots, l_2$, notice that the block column $R_{1:j-1,j}$ has the following form:

$$\begin{bmatrix} \mathcal{U}_1 \mathcal{W}_2 \cdots \mathcal{W}_{j-1} \mathcal{V}_j^T \\ \vdots \\ \mathcal{U}_{j-2} \mathcal{W}_{j-1} \mathcal{V}_j^T \\ \mathcal{U}_{j-1} \mathcal{V}_j^T \end{bmatrix}.$$

Introduce auxiliary matrices Φ_j defined by

$$\Phi_0 = 0,\ \Phi_j = \Phi_{j-1}\mathcal{W}_j + X_{:,j}\mathcal{U}_j, \qquad j = 1, 2, \ldots, l_2 - 1. \tag{3.10}$$

Then clearly,

$$X_{:,1:j-1}R_{1:j-1,j} = \Phi_{j-1}\mathcal{V}_j^T, \qquad j = 1, 2, \ldots, l_2. \tag{3.11}$$

It can be shown that the cost of evaluating $\hat{K}_j$ in (3.6) for each j by (3.7)–(3.11) is $O(n_1)$.

3.2.3. Solving (3.6). Finally, we consider the solution of (3.6). For each j, when R_{jj} is a scalar, (3.6) is an order-n_1 lower triangular SSS system

$$(L - R_{jj})X_{:,j} = \hat{K}_j^T.$$

The coefficient matrix $L - R_{jj}$ has the same generators as L except that the $\mathcal{D}_i$ generators are replaced by $L_{ii} - R_{jj}$ or $L_{ii} - R_{jj}I_2$, depending on the size of L_{ii}. This system can be solved in linear time by the fast SSS system solver in [9], and the details are shown in [26].

When R_{jj} is 2×2, (3.6) is a simple Sylvester equation, which can be converted into an order-$2n$ lower triangular SSS system. Note that for this situation, the Bartels–Stewart algorithm does not apply to (3.6) anymore since we want to maintain real arithmetic and R_{jj} does not have a real Schur form. However, we can rewrite (3.6) as a Sylvester equation in terms of $X_{:,j}^T$ as follows:

$$-R_{jj}^T X_{:,j}^T + X_{:,j}^T L^T = \hat{K}_j^T.$$

This equation can be converted into a lower triangular SSS system

$$(L \otimes I_2 - I_{n_1-2} \otimes R_{jj}^T)\mathrm{vec}(X_{:,j}^T) = \mathrm{vec}(\hat{K}_j^T).$$

The SSS generators of the coefficient matrix are given by those of $L \otimes I_2$, except the diagonal generators are $\mathcal{D}_i(L \otimes I_2) - R_{jj}^T$ or $\mathcal{D}_i(L \otimes I_2) - I_2 \otimes R_{jj}^T$, depending on whether the order of $\mathcal{D}_i(L)$ is 1 or 2. The generators of $L \otimes I_2$ are listed in Table 3.1.

For both cases, the solution of (3.6) costs $O(n_1)$ for each j.

TABLE 3.1
SSS generators of $L \otimes I_2$ in terms of the generators of L, where $\mathcal{P}_i(L) \equiv \left[\begin{smallmatrix} \mathcal{P}_{i,1}(L) \\ \mathcal{P}_{i,2}(L) \end{smallmatrix}\right]$ and $\mathcal{Q}_i(L) \equiv \left[\begin{smallmatrix} \mathcal{Q}_{i,1}(L) \\ \mathcal{Q}_{i,2}(L) \end{smallmatrix}\right]$.

Order of $\mathcal{D}_i(L)$	$\mathcal{D}_i(L \otimes I_2)$	$\mathcal{P}_i(L \otimes I_2)$	$\mathcal{Q}_i(L \otimes I_2)$	$\mathcal{R}_i(L \otimes I_2)$
1	$\mathcal{D}_i(L) \otimes I_2$	$I_2 \otimes \mathcal{P}_i(L)$	$I_2 \otimes \mathcal{Q}_i(L)$	$I_2 \otimes \mathcal{R}_i(L)$
2	$\mathcal{D}_i(L) \otimes I_2$	$\left[\begin{smallmatrix} I_2 \otimes \mathcal{P}_{i,1}(L) \\ I_2 \otimes \mathcal{P}_{i,2}(L) \end{smallmatrix}\right]$	$\left[\begin{smallmatrix} I_2 \otimes \mathcal{Q}_{i,1}(L) \\ I_2 \otimes \mathcal{Q}_{i,2}(L) \end{smallmatrix}\right]$	$I_2 \otimes \mathcal{R}_i(L)$

3.3. Swapping the diagonal blocks of the Schur form T. In order to use (2.5) to evaluate the condition of any eigenvalue cluster corresponding to diagonal blocks other than T_1, we can use a swapping procedure to bring those blocks to the leading upper left position of T. Assume that the eigenvalue cluster of interest corresponds to the diagonal blocks $\{T_{i_1}, T_{i_2}, \ldots, T_{i_k}\}$ of T. The matrix T will be transformed into $\tilde{T}$ in (2.6), for which we will derive a new compact SSS form.

3.3.1. Swapping procedure for contiguous blocks. We make use of a fundamental swapping procedure in [1], [12], [27] for two diagonal blocks of a matrix

$$\begin{bmatrix} T_i & H_i \\ 0 & T_j \end{bmatrix},$$

where T_i and T_j have orders n_i and n_j, respectively, and T_i and T_j have no eigenvalue in common. The swapping procedure in [1], [12], [27] finds an orthogonal matrix G_i, which is the product of some Givens matrices such that

$$G_i \begin{bmatrix} T_i & H_i \\ 0 & T_j \end{bmatrix} G_i^T = \begin{bmatrix} M_j T_j M_j^{-1} & \bar{H}_i \\ 0 & M_i T_i M_i^{-1} \end{bmatrix},$$

where M_i and M_j are approximate invertible matrices. Thus T_i and T_j have been swapped.

In order to preserve the quasi-triangular form of T, we apply this swapping procedure to contiguous 1×1 or 2×2 diagonal blocks of T, even if T may have multiple eigenvalues. The details are as follows. To bring the 1×1 or 2×2 blocks $\{T_{i_1}, T_{i_2}, \ldots, T_{i_k}\}$ of an eigencluster to the leading position, we partition T as

$$T = \begin{bmatrix} \hat{T}_1 & \hat{H}_1 & \cdots \\ 0 & \hat{T}_2 & \cdots \\ 0 & 0 & \ddots \end{bmatrix},$$

where $\hat{T}_2$ has diagonal blocks $\{T_{i_1}, T_{i_2}, \ldots, T_{i_k}\}$. If we directly apply the above swapping procedure to $\left[\begin{smallmatrix} \hat{T}_1 & \hat{H}_1 \\ 0 & \hat{T}_2 \end{smallmatrix}\right]$ to bring $\hat{T}_2$ to the leading position, then the quasi-triangular form of T may be destroyed, and also the structures of the related matrices and matrix equations are hard to explore.

Thus, instead, we apply the above procedure to contiguous 1×1 or 2×2 diagonal blocks of T and bring each T_i in $\{T_{i_1}, T_{i_2}, \ldots, T_{i_k}\}$ to the leading position in one round of swapping. After each round of swapping, T is transformed into a new quasi-triangular matrix $\tilde{T} = GTG^T$ as in (2.6), where G is a product of Givens matrices. (The current structure of this $\tilde{T}$ needs to be compressed. See the next subsection.) The number of Givens matrices depends on the size of T_i. If T_i is 1×1, then $k_i - 1$

Givens matrices are needed, where k_i is the row or column index of T_i in T. The matrix G has the form

$$G = \prod_{j=1}^{k_i-1} \operatorname{diag}\left[I_{j-1}, \begin{bmatrix} c_j & s_j \\ -s_j & c_j \end{bmatrix}, I_{n-j-1}\right], \tag{3.12}$$

which is upper Hessenberg. If T_i is 2×2, then $2(k_i - 1)$ Givens matrices are needed, where k_i is the row or column index of the leading entry of T_i in T. Some of these Givens matrices commute, and after reordering the matrices, we have $G = G_1 G_2$, where each of G_1 and G_2 has the form (3.12). The details of the generation of G are similar to those in [26]. Clearly, G in (3.12) is an SSS matrix, with its generators given in Table 3.2 in terms of its Schur parameters $\{c_i, s_i\}$ [11].

Table 3.2
SSS generators of G in (3.12).

$\mathcal{D}_j(G)$	$\mathcal{U}_j(G)$	$\mathcal{V}_j(G)$	$\mathcal{W}_j(G)$	$\mathcal{P}_j(G)$	$\mathcal{Q}_j(G)$	$\mathcal{R}_j(G)$
$c_{j-1}c_j$	$c_{j-1}s_j$	c_j	s_j	1	$-s_j$	0

The matrix $\tilde{T}$ is then still quasi triangular. We can get its SSS form by multiplying three SSS matrices in (2.6) using the formulas for SSS matrix multiplications in [8]. This is the method used in [26] for the situation of distinct eigenvalues. Notice that the off-diagonal generator sizes increase accumulatively with the multiplications in (2.6). If the $\mathcal{W}_i(T)$ generators have size 2, then the $\mathcal{W}_i(\tilde{T})$ generators have size up to 6, since the $\mathcal{W}_i$ generators of G and G^T have size 1 or 2.

3.3.2. Recovery of compact SSS representation of $\tilde{T}$. Since here we are considering general eigenclusters instead of simple distinct eigenvalues, the matrix multiplication technique in [26] is inefficient. For example, if the swapping process is applied to an eigencluster which has k multiple eigenvalues or eigenpairs, then $\tilde{T}$ needs to be multiplied by up to $\mathrm{O}(kn)$ Givens matrices, and the off-diagonal generator sizes of $\tilde{T}$ increase significantly. Therefore, $\tilde{T}$ is generally not compact anymore. On the other hand, the actual off-diagonal ranks of $\tilde{T}$ do not increase, according to Theorem 3.2. Thus, we use a recovery strategy similar to the one in [11] to reconstruct a compact SSS form for $\tilde{T}$. The recovery strategy in [11] is designed for certain strictly triangular matrices with maximum off-diagonal rank 2 such that their generator sizes are bounded by 3. Here we improve the strategy such that it works for the quasi-triangular matrix $\tilde{T}$, and furthermore, the generators sizes are bounded by 2, which is consistent with the maximum off-diagonal rank. This also saves one SSS matrix multiplication.

The matrix T is orthogonally similar to C and is obviously orthogonal plus rank one, which we assume to be $T \equiv P + uv^T$. Also, assume that $\tilde{T}$ has a QR factorization $\tilde{T} = \tilde{Q}\tilde{R}$. The matrix $\tilde{Q}$ is a block diagonal matrix with 1×1 or 2×2 diagonal blocks, since $\tilde{T}$ is quasi triangular. We have

$$\begin{aligned} \tilde{R} &= \tilde{Q}^T \tilde{T} = \tilde{Q}^T G(P + uv^T)G^T \\ &= \tilde{Q}^T G P G^T + (\tilde{Q}^T G u)(Gv)^T \equiv \tilde{P} + \tilde{u}\tilde{v}^T. \end{aligned}$$

There exists an orthogonal upper Hessenberg matrix P_1 which is a product of Givens matrices such that

$$P_1 \tilde{u} = ||\tilde{u}||_2 e_1, \tag{3.13}$$

TABLE 3.3
Upper SSS generators of $P_1^TP_2$ in terms of the Schur parameters of P_1 and P_2, where $\rho_{i-1} \equiv \prod_{k=1}^{i-1} s_k\tilde{s}_k + \sum_{j=1}^{i-1}\left(c_j\tilde{c}_j\prod_{k=j+1}^{i-1} s_k\tilde{s}_k\right)$. Other generators can be obtained by symmetry in the structure.

$D_i(P_1^TP_2)$	$U_i(P_1^TP_2)$	$V_i(P_1^TP_2)$	$W_i(P_1^TP_2)$
$c_i\tilde{c}_i\rho_{i-1} + s_i\tilde{s}_i$	$c_i\tilde{s}_i\rho_{i-1} - s_i\tilde{c}_i$	$\tilde{c}_i$	$\tilde{s}_i$

TABLE 3.4
SSS generators of $\tilde{u}\tilde{v}^T$.

$\mathcal{D}_i(\tilde{u}\tilde{v}^T)$	$\mathcal{U}_i(\tilde{u}\tilde{v}^T)$	$\mathcal{V}_i(\tilde{u}\tilde{v}^T)$	$\mathcal{W}_i(\tilde{u}\tilde{v}^T)$	$\mathcal{P}_i(\tilde{u}\tilde{v}^T)$	$\mathcal{Q}_i(\tilde{u}\tilde{v}^T)$	$\mathcal{R}_i(\tilde{u}\tilde{v}^T)$
$\tilde{u}_i\tilde{v}_i^T$	$\tilde{u}_i$	$\tilde{v}_i$	1	$\tilde{u}_i$	$\tilde{v}_i$	1

where e_1 is the first unit vector. Then we have that $P_1\tilde{P} = P_1\tilde{R} - ||\tilde{u}||_2e_1\tilde{v}^T$ is orthogonal and also upper Hessenberg. Thus, there exists another orthogonal upper Hessenberg matrix P_2 such that

$$P_2 = P_1\tilde{P} = P_1\tilde{R} - ||\tilde{u}||_2e_1\tilde{v}^T, \tag{3.14}$$

$$\tilde{R} = P_1^TP_2 + ||\tilde{u}||_2P_1^Te_1\tilde{v}^T = P_1^TP_2 + \tilde{u}\tilde{v}^T, \tag{3.15}$$

$$\tilde{T} = \tilde{Q}\tilde{R} = \tilde{Q}(P_1^TP_2 + \tilde{u}\tilde{v}^T). \tag{3.16}$$

Both P_1 and P_2 are orthogonal upper Hessenberg and have maximum off-diagonal rank one, and $||\tilde{u}||_2P_1^Te_1\tilde{v}^T$ is a rank one matrix. Therefore, if SSS multiplication and addition formulas are used as in [11], the SSS form of $\tilde{R}$ has maximum generator size 3. However, a more compact form of $\tilde{R}$ is available.

THEOREM 3.3. *For any orthogonal upper Hessenberg matrices P_1 and P_2, the matrix $P_1^TP_2$ has maximum off-diagonal rank one.*

Proof. Denote a submatrix $P_1(1:i, 1:i+1)$ by $P_{1;(1:i,1:i+1)}$. An upper off-diagonal block of $P = P_1^TP_2$ is given by

$$P_{1:i,i+1:n} = P_{1;(1:i+1,1:i)}^T P_{2;(1:i+1,i+1:n)}.$$

The submatrix $P_{2;(1:i+1,i+1:n)}$ has rank one, since all its rows are multiples of

$$[\tilde{c}_i,\ \tilde{s}_i\tilde{c}_{i+1},\ \ldots,\ \tilde{s}_i\cdots\tilde{s}_{n-1}\tilde{c}_n],$$

where $\{\tilde{c}_i, \tilde{s}_i\}$ are the Schur parameters of P_2. (See Table 3.2.) Thus, $P_{1:i,i+1:n}$ has rank one also. □

The SSS form of $P_1^TP_2$ in (3.16) is given in Table 3.3 in terms of the Schur parameters $\{c_i, s_i\}$ and $\{\tilde{c}_i, \tilde{s}_i\}$ of P_1 and P_2, respectively. The generators can be obtained in O(n) complexity by computing $\rho_{i-1} = \prod_{k=1}^{i-1} s_k\tilde{s}_k + \sum_{j=1}^{i-1}\left(c_j\tilde{c}_j\prod_{k=j+1}^{i-1} s_k\tilde{s}_k\right) \equiv \alpha_{i-1} + \beta_{i-1}$ recursively as follows:

$$\alpha_0 = 1,\ \alpha_i = (s_i\tilde{s}_i)\alpha_{i-1}, \qquad i = 1, 2, \ldots, n-1,$$
$$\beta_0 = 0,\ \beta_i = (s_i\tilde{s}_i)\beta_{i-1} + c_i\tilde{c}_i, \qquad i = 1, 2, \ldots, n-1.$$

The SSS form of $\tilde{u}\tilde{v}^T$ is given in Table 3.4. Then one SSS addition gives an SSS form for $\tilde{R}$ whose maximum generator size is 2.

The left multiplication of $\tilde{R}$ by $\tilde{Q}$ does not increase the off-diagonal block ranks because $\tilde{Q}$ is a block diagonal matrix with 1×1 or 2×2 diagonal blocks. Thus, $\tilde{T}$ has maximum generator size 2. This construction process provides an alternative way of proving Theorem 3.2.

Therefore, we can use (3.14)–(3.16) to recover a compact SSS form for $\tilde{T}$. After each round of swapping to bring a single eigenvalue or eigenpair to a desired position, we apply the recovery procedure to $\tilde{T}$. First, we form the redundant SSS form of $\tilde{T}$ in (2.6) by SSS matrix multiplications, and then we compute the QR factorization $\tilde{Q}\tilde{R}$ for $\tilde{T}$. The matrix $\tilde{Q}$ can be obtained by computing the Givens QR factorization

$$\tilde{T}_i = \tilde{Q}_i\tilde{R}_i \tag{3.17}$$

for any 2×2 diagonal block $\tilde{T}_i$ of T. The block diagonal matrix $\tilde{Q}$ has each diagonal block being either 1 or $\tilde{Q}_i$. The matrix $\tilde{R}$ is still an SSS matrix with the same generators as $\tilde{T}$ except that for any i corresponding to a 2×2 diagonal block,

$$\mathcal{D}_i(\tilde{R}) = \tilde{R}_i,\ \mathcal{U}_i(\tilde{R}) = \tilde{Q}_i^T\mathcal{U}_i(\tilde{T}). \tag{3.18}$$

Note that the Schur parameters of P_2 are obtained by updating only certain SSS generators of $P_1\tilde{R}$. With $P_1^TP_2$ and $\tilde{u}\tilde{v}^T$ available in SSS forms, a new compact SSS form of $\tilde{T}$ is straightforward according to (3.17) and (3.18). This improved recovery procedure requires about $14i_1$ operations, as compared with the $40i_1$ cost in [11]. In addition, the SSS generators have maximum size 2 instead of 3, which reduces the cost for later operations also.

3.4. Computing the condition estimate (2.5). The major work in evaluating ν in (2.5) is to compute $\mu(B)$, where B is given by (2.3). In general, for companion matrices we can compute (2.5) using the method which will be presented in section 4. But since E has a special form (3.2) with only one nonzero row, an alternative method is to use (2.3) directly.

We first find $U_{1,:}$. Note that the fast eigensolver in [11] provides U in the form of a sequence of $\mathrm{O}(n^2)$ Givens rotation matrices. Thus, the application of these matrices on the right to e_1^T, the first unit vector of length n, yields the initial $U_{1,:}$. This costs $\mathrm{O}(n^2)$ operations. Later, for each cluster of diagonal blocks with size n_i, the row $U_{1,:}$ needs to be updated when T is updated by the swapping process. According to the previous subsection, U is updated to $\tilde{U} = UG^T$, where G is also represented by Givens rotation matrices. Thus, the updated vector is

$$\tilde{U}_{1,:} = U_{1,:}G^T. \tag{3.19}$$

The computation of EU_1 in (2.3) can be done by considering EU. (If the diagonal blocks of T are swapped, then we compute $E\tilde{U}$ similarly.) Since U is represented by $\mathrm{O}(n^2)$ Givens matrices, the cost for computing EU is $\mathrm{O}(n^2)$. The matrix EU_1 has only one nonzero row, which we assume to be u_1^T. Also, let $U_{1,:} = [\begin{smallmatrix} U_{1:n_1,:} \\ U_{n_1+1:n,:} \end{smallmatrix}]$. Then we have

$$B = U_{1,1:n_1}^T u_1^T + (YU_{1,n_1+1:n}^T)u_1^T,$$

which is the sum of two rank one matrices. We first evaluate $YU_{1,n_1+1:n}^T$ and then compute the diagonal entries of B.

4. The case of general C in (1.1).

For a general C in (1.1), which is a low rank modification to a symmetric, skew-symmetric, or orthogonal matrix, the main operations in previous sections are similar. For example, we can quickly get an SSS form Schur decomposition. A major difference is that the computation of the condition estimate (2.5) can be done in a more general way.

For C in (1.1), the perturbation matrix E has the form (3.1). We can precompute

$$U^T E U = (U^T \hat{x})(U^T \hat{y})^T \equiv \tilde{x}\tilde{y}^T.$$

The computations of $\tilde{x}$ and $\tilde{y}$ cost $\mathrm{O}(n^2)$ flops, since U is a product of $\mathrm{O}(n^2)$ Givens rotation matrices and both $\tilde{x}$ and $\tilde{y}$ have a finite number of columns.

The direct computation of the trace of $\hat{B}$ in (2.8) is thus straightforward. Since $\hat{B} = \left(\begin{bmatrix} I_{n_1} & Y \end{bmatrix}\tilde{x}\right)\tilde{y}^T$, we first form $\begin{bmatrix} I_{n_1} & Y \end{bmatrix}\tilde{x}$ and then compute the trace of the left $n_1 \times n_1$ submatrix of $\hat{B}$. For different eigenvalues, permutations are applied to U, and $\hat{B}$ now has the form

$$\hat{B} = \begin{bmatrix} I_{n_1} & Y \end{bmatrix}(G\tilde{x})(G\tilde{y})^T, \tag{4.1}$$

where G is a product of Givens rotation matrices. (Here, Y should also be different, but the same notation is used for convenience.) We form $G\tilde{x}$ and $G\tilde{y}$ first, and the rest of the computations are similar.

5. Algorithm, flop counts, and numerical experiments.

We outline the major steps in the following algorithm in terms of a companion matrix C.

Algorithm 1 (condition estimation for the eigenvalues/eigenclusters of C).

```
1. Compute an initial structured Schur decomposition $C = UTU^T$.
2. Choose a perturbation matrix $E$ as in (3.2) or (3.1). Precompute $U^T E U$ as
   in section 4.
3. Repeat for each eigenvalue cluster $i$ corresponding to $\{T_{i_1}, T_{i_2}, \ldots\}$.
   (a) If $i > 1$, use the swapping technique in subsection 3.3 to bring cluster $i$
       to the leading position, one block $T_{i_j}$ per round.
   (b) Solve the Sylvester equation (2.4) as in subsection 3.2.
   (c) Compute the condition estimate (2.5) via the diagonal entries of $\hat{B}$ in
       (4.1).
4. If additional samples of $E$ are used, repeat steps 2 and 3(c).
```

5.1. Flop counts.

To obtain detailed flop counts for a companion matrix C, we make the following assumptions:

- The number of iterations required for the Hessenberg QR iteration to converge is cn^2, where c is a constant (c is usually small).
- Each compact SSS matrix A has maximum off-diagonal rank p which is 2 for T and 1 for an orthogonal upper Hessenberg matrix. All $\mathcal{W}_i$ and $\mathcal{R}_i$ generators of A have dimension p.
- A simplified problem is considered, where all diagonal blocks T_i of T are 1×1.
- The matrix T has m eigenvalue clusters and the ith cluster has n_i eigenvalues $\{T_{i_1}, T_{i_2}, \ldots\}$.

Step 1 costs about the same as the structured eigensolver in [11], and we do not discuss the details here. Step 2 costs about $6cn^2$ flops. The cost of computing the condition estimate of each cluster i in step 3 is as follows.

1. In step 3(a), the operations and the required flops are given by the following:
   (a) Swapping the diagonals of T to bring T_{i_j} to the leading position and computing a redundant SSS form for $\tilde{T}$ cost
   $$(80p^3 + 122p^2 + 130p + 81)(i_j - j),$$
   where we have used the result that it costs at most $40p^3(i_j - j)$ flops to multiply two order-$(i_j - j)$ SSS matrices whose maximum off-diagonal ranks are p [8].
   (b) Recovery of a compact SSS form for $\tilde{T}$ approximately costs
   $$(47p^3 + 242p^2 + 464p + 374)(i_j - j),$$
   where, for simplicity, the cost for a general rank p recovery process is mainly counted based on SSS multiplications, although it is possible to extend the idea in subsection 3.3.2 to further reduce the cost.

   The total cost for the entire cluster i is thus
   $$(127p^3 + 364p^2 + 594p + 455)\sum_{j=1}^{n_i}(i_j - j).$$
2. The cost for step 3(b) using the Bartels–Stewart algorithm is
   $$(2p^3 + 8p^2 + 11p + 2)n_i(n - n_i).$$
3. Step 3(c) costs
   $$2n_i(n - n_i) + 2n_i + 12\sum_{j=1}^{n_i}(i_j - j).$$

Therefore, the cost for all the eigenvalue clusters is

$$(127p^3 + 364p^2 + 594p + 467)\sum_{i=2}^{m}\sum_{j=1}^{n_i}(i_j - j) + (2p^3 + 8p^2 + 11p + 4)\sum_{i=1}^{m} n_i(n - n_i) + 2\sum_{i=1}^{m} n_i.$$

Since $\sum_{i=1}^{m} n_i = n$, we have

$$\sum_{i=2}^{m}\sum_{j=1}^{n_i}(i_j - j) = \sum_{i=2}^{m} n_i(i_1 - 1) \le n\sum_{i=2}^{m} n_i \le n^2,$$
$$\sum_{i=1}^{m} n_i(n - n_i) \le n\sum_{i=1}^{m} n_i = n^2.$$

The total cost is thus approximately bounded by

$$(129p^3 + 372p^2 + 605p + 471)n^2. \quad (5.1)$$

This bound can highly overestimate the cost. For example, when there are only two eigenvalue clusters with equal size $n/2$, the cost is bounded by

$$\left(33p^3 + 95p^2 + 154p + 119\right)n^2.$$

If multiple samples are used, we need only repeat steps 2 and 3(c), and the results from other steps can be reused. Since the total cost for steps 2 and 3(c) is bounded by $26n^2$, which is much smaller than (5.1), the amount of work required for each additional sample of SCE is insignificant.

5.2. Numerical examples. A MATLAB implementation of Algorithm 1 is available at http://www.math.purdue.edu/˜xiaj/work/sceeig.

We apply it to some companion matrices and demonstrate the efficiency and accuracy. Note that [26] also includes some results with a different algorithm which requires all the eigenvalues to be distinct.

Example 5.1. Consider a companion matrix C whose eigenvalues are $\lambda_i = i,\ i = 1, 2, \ldots, n$. These eigenvalues are the roots of the Wilkinson polynomial. According to [26], the SCE estimator (2.5) is an estimate of the following exact condition number for λ_i:

$$\kappa_i = ||(k_{i,1}, k_{i,2}, \ldots, k_{i,n})^T||_2,\ \kappa_{i,j} = \left| \frac{a_j \lambda_i^{j-1}}{\prod_{k\neq i} (\lambda_i - \lambda_k)} \right|, \tag{5.2}$$

where a_j is the coefficient of the λ^j term of the polynomial (see (1.2)).

For $n = 15$, we calculate the exact condition numbers κ_i, their 1-sample SCE estimates, and the estimates by the MATLAB routine `condeig`, which computes the reciprocals of the cosines of the angles between the left and right eigenvectors of C. According to Figure 5.1, SCE provides favorable estimates, while `condeig` gives large estimates for nearly all eigenvalues except the first one.

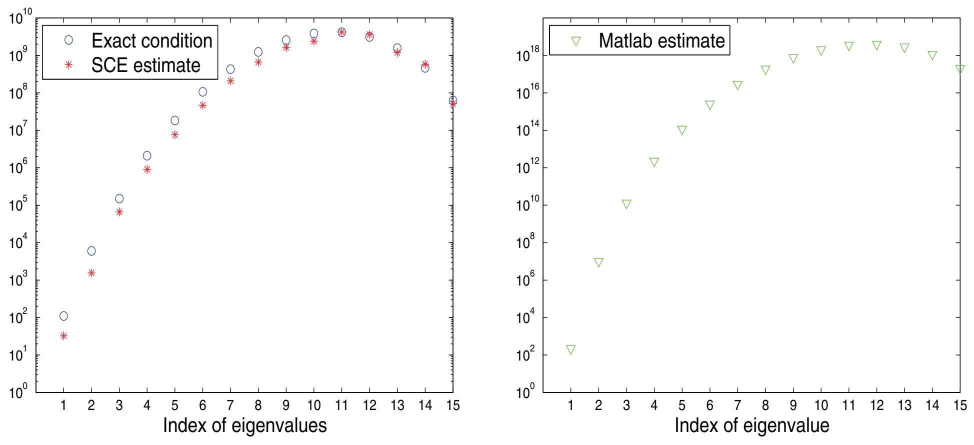

FIG. 5.1. *Condition numbers and their estimates for Example* 5.1.

Example 5.2. We show the quadratic complexity of the estimator with an example where C in (1.2) has $a_0 = 1$ and $a_i = 0,\ i = 1, \ldots, n-1$. We report only the flop counts of our preliminary MATLAB implementation of the algorithm.

This companion matrix has eigenvalues at the roots of unity. The eigenvalues are all well conditioned, with κ_i in (5.2) given by $1/n$ [15]. Our SCE estimator also reflects this fact. We run our algorithm for n ranging from 32 to 1024 and count the flops, denoted flops_n. Then we compute the flop ratios $\frac{\text{flops}_n}{\text{flops}_{n/2}}$. The numerical results

TABLE 5.1
SCE for Example 5.2 *with different* n.

n	32	64	128	256	512	1024
κ_{exact}	0.0313	0.0156	0.0078	0.0039	0.0020	0.0010
κ_{SCE}	0.0432	0.0268	0.0124	0.0083	0.0009	0.0007
$\frac{\text{flops}_n}{\text{flops}_{n/2}}$	4.9	4.4	4.2	4.1	4.0	4.0

TABLE 5.2
SCE for Example 5.4 *with multiplicity* $m = 2$.

n	32	64	128	256	512	1024
$\frac{\text{flops}_n}{\text{flops}_{n/2}}$	4.8	4.4	4.2	4.1	4.0	4.0

show that the ratios are close to 4, which is consistent with the $O(n^2)$ complexity. See Table 5.1.

Example 5.3. We consider a companion matrix C with multiple eigenvalues $\{2^{-i}, 2^{-i}\}$, $i = 1, 2, \ldots, n/2$.

For $n = 10$, SCE gives five estimates for the five eigenvalue clusters $\{2^{-i}, 2^{-i}\}$: $2.5E2$, $9.9E2$, $6.4E1$, $3.4E2$, $4.7E1$. We see that the eigenvalue clusters are still well conditioned. This somehow is consistent with the result that multiple roots of polynomials may be well conditioned if the multiplicities are preserved on a proper pejorative manifold [22], [32].

However, for large problems with multiple eigenvalues, eigensolvers such as the one in [11] give inaccurate results. Thus, it is not clear if SCE with those eigensolvers accurately reflects the condition. With multiplicity preserving eigensolvers, it is possible to further explore the potential of SCE. This remains an open problem. Therefore, in the following example, we report only the complexity results.

Example 5.4. Consider a companion matrix C whose eigenvalues are the roots of unity with different multiplicities m. (We chose this example because its nonzero entries are relatively easy to compute accurately.) Tables 5.2 shows the flop ratios $\frac{\text{flops}_n}{\text{flops}_{n/2}}$ for $m = 2$. Very similar results can be observed for m to be a fraction of n such as $\frac{n}{4}$. Thus we omit them.

6. Conclusions.

We develop a condition estimation scheme for the eigenvalues of a class of matrices which are low rank perturbations to rank symmetric matrices. Rank structures of these matrices are exploited and fast structured matrix operations are presented, such as Schur decomposition, matrix equation solution, Schur form update, compact semiseparable form reconstruction, etc. These operations may be used in the condition estimation of other structured matrices and more general problems such as invariant subspace computations.

Similar techniques can also be used to estimate the condition of the eigenvectors. The information in the condition estimation for the eigenvalues can be reused. It is also possible to derive a condition estimate for the average eigenvalue of the block T_c in (2.1). In this way, we can save about 3/4 of the diagonal swapping work, on average, for all the eigenvalues. We also notice that the cost for the structured Sylvester solver can be possibly reduced further, since K in (3.4) is a low rank matrix.

REFERENCES

[1] Z. Bai, J. W. Demmel, and A. McKenney, *On computing condition numbers for the nonsymmetric eigenproblem*, ACM Trans. Math. Software, 19 (1993), pp. 202–223.

[2] R. H. Bartels and G. W. Stewart, *Solution of the matrix equation $AX + XB = C$*, Comm. ACM, 15 (1972), pp. 820–826.

[3] D. Bindel, S. Chandresekaran, J. Demmel, D. Garmire, and M. Gu, *A Fast and Stable Nonsymmetric Eigensolver for Certain Structured Matrices*, Technical report, University of California, Berkeley, CA, 2005.

[4] D. A. Bini, F. Daddi, and L. Gemignani, *On the shifted QR iteration applied to companion matrices*, Electron. Trans. Numer. Anal., 18 (2004), pp. 137–152.

[5] D. A. Bini, Y. Eidelman, L. Gemignani, and I. Gohberg, *Fast QR Eigenvalue Algorithms for Hessenberg Matrices which Are Rank-One Perturbations of Unitary Matrices*, Technical Report 1587, Department of Mathematics, University of Pisa, Pisa, Italy, 2005.

[6] S. P. Chan, R. Feldman, and B. N. Parlett, *A program for computing the condition numbers of matrix eigenvalues without computing eigenvectors*, ACM Trans. Math. Software, 3 (1977), pp. 186–203.

[7] S. Chandrasekaran, P. Dewilde, M. Gu, T. Pals, X. Sun, A.-J. van der Veen, and D. White, *Some fast algorithms for sequentially semiseparable representations*, SIAM J. Matrix Anal. Appl., 27 (2005), pp. 341–364.

[8] S. Chandrasekaran, P. Dewilde, M. Gu, T. Pals, X. Sun, A.-J. van der Veen, and D. White, *Fast Stable Solvers for Sequentially Semi-separable Linear Systems of Equations and Least Squares Problems*, Technical report, University of California, Berkeley, CA, 2003.

[9] S. Chandrasekaran, P. Dewilde, M. Gu, T. Pals, and A.-J. van der Veen, *Fast stable solver for sequentially semi-separable linear systems of equations*, in High Performance Computing (HiPC 2002): 9th International Conference, Lecture Notes in Comput. Sci. 2552, Springer-Verlag, Heidelberg, 2002, pp. 545–554.

[10] S. Chandrasekaran, M. Gu, X. Sun, J. Xia, and J. Zhu, *A superfast algorithm for Toeplitz systems of linear equations*, SIAM J. Matrix Anal. Appl., 29 (2007), pp. 1247–1266.

[11] S. Chandrasekaran, M. Gu, J. Xia, and J. Zhu, *A fast QR algorithm for companion matrices*, in Recent Advances in Matrix and Operator Theory, Oper. Theory Adv. Appl. 179, Birkhäuser, Basel, 2008, pp. 111–143.

[12] J. J. Dongarra, S. Hammarling, and J. H. Wilkinson, *Numerical considerations in computing invariant subspaces*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 145–161.

[13] H. Fassbender and D. Kressner, *Structured eigenvalue problems*, GAMM-Mitt., 29 (2006), pp. 297–318.

[14] W. Gautschi, *On the condition of algebraic equations*, Numer. Math., 21 (1973), pp. 405–424.

[15] W. Gautschi, *Questions of numerical condition related to polynomials*, in Studies in Numerical Analysis, MAA Studies in Math. 24, G. H. Golub, ed., Math. Assoc. Amer., Washington, DC, 1984, pp. 140–177.

[16] L. Gemignani, *A unitary Hessenberg QR-based algorithm via semiseparable matrices*, J. Comput. Appl. Math., 184 (2005), pp. 505–517.

[17] G. Golub and C. V. Loan, *Matrix Computation*, 3rd ed., The Johns Hopkins University Press, Baltimore, MD, 1996.

[18] M. Gu and S. C. Eisenstat, *A stable and efficient algorithm for the rank-one modification of the symmetric eigenproblem*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 1266–1276.

[19] T. Gudmundsson, C. Kenney, and A. J. Laub, *Small-sample statistical estimates for the sensitivity of eigenvalue problems*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 868–886.

[20] T. Gudmundsson, C. S. Kenney, A. J. Laub, and M. S. Reese, *Applications of small-sample statistical condition estimation in control*, in Proceedings of the 1996 IEEE International Symposium on Computer-Aided Control System Design, IEEE Press, Piscataway, NJ, 1996, pp. 164–169.

[21] B. Kågstrom and P. Poromaa, *Computing eigenspaces with specified eigenvalues of a regular matrix pair (A, B) and condition estimation: Theory, algorithms and software*, Numer. Algorithms, 12 (1996), pp. 369–407.

[22] W. Kahan, *Conserving Confluence Curbs Ill-Condition*, Technical Report 6, Department of Computer Science, University of California, Berkeley, CA, 1972.

[23] C. S. Kenney and A. J. Laub, *Small-sample statistical condition estimates for general matrix functions*, SIAM J. Sci. Comput., 15 (1994), pp. 36–61.

[24] C. S. Kenney, A. J. Laub, and M. S. Reese, *Statistical condition estimation for linear systems*, SIAM J. Sci. Comput., 19 (1998), pp. 566–583.

[25] C. S. Kenney, A. J. Laub, and M. S. Reese, *Statistical condition estimation for linear least squares*, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 906–923.

[26] A. J. Laub and J. Xia, *Statistical condition estimation for the roots of polynomials*, SIAM J. Sci. Comput., 31 (2008), pp. 624–643.

[27] G. W. Stewart, *Algorithm* 506*: HQR*3 *and EXCHNG: Fortran subroutines for calculating and ordering eigenvalues of a real upper Hessenberg matrix*, ACM Trans. Math. Software, 2 (1976), pp. 275–280.

[28] C. Van Loan, *On estimating the condition of eigenvalues and eigenvectors*, Linear Algebra Appl., 88/89 (1987), pp. 715–732.

[29] J. H. Wilkinson, *Rounding Errors in Algebraic Processes*, Prentice-Hall, Englewood Cliffs, NJ, 1963.

[30] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon, Oxford, UK, 1965.

[31] J. H. Wilkinson, *Sensitivity of eigenvalues*, Utilitas Math., 25 (1984), pp. 5–76.

[32] Z. Zeng, *A method computing multiple roots of inexact polynomials*, in Proceedings of ISSAC 2003, ACM Press, New York, 2003, pp. 266–272.