Electronic Transactions on Numerical Analysis.
Volume 59, pp. 60–88, 2023.
Copyright © 2023, Kent State University.
ISSN 1068–9613.
DOI: 10.1553/etna_vol59s60

ETNA
Kent State University and
Johann Radon Institute (RICAM)

THE BISECTION EIGENVALUE METHOD FOR UNITARY HESSENBERG MATRICES VIA THEIR QUASISEPARABLE STRUCTURE[*]

YULI EIDELMAN[†] AND IULIAN HAIMOVICI[†]

Dedicated to Lothar Reichel, from whose papers we have learnt a lot, on the occasion of his 70th birthday.

Abstract. If N_0 is a normal matrix, then the Hermitian matrices $\frac{1}{2}(N_0 + N_0^*)$ and $\frac{i}{2}(N_0^* - N_0)$ have the same eigenvectors as N_0. Their eigenvalues are the real part and the imaginary part of the eigenvalues of N_0, respectively. If N_0 is unitary, then only the real part of each of its eigenvalues and the sign of the imaginary part is needed to completely determine the eigenvalue, since the sum of the squares of these two parts is known to be equal to 1. Since a unitary upper Hessenberg matrix U has a quasiseparable structure of order one and we express the matrix $A = \frac{1}{2}(U + U^*)$ as quasiseparable matrix of order two , we can find the real part of the eigenvalues and, when needed, a corresponding eigenvector x, by using techniques that have been established in the paper by Eidelman and Haimovici [Oper. Theory Adv. Appl., 271 (2018), pp. 181–200].

We describe here a fast procedure, which takes only 1.7% of the bisection method time, to find the sign of the imaginary part. For instance, in the worst case only, we build one row of the quasiseparable matrix U and multiply it by a known eigenvector of A, as the main part of the procedure. This case occurs for our algorithm when among the 4 numbers $\pm\cos t \pm i \sin t$ there are exactly 2 eigenvalues and they are opposite, so that we have to distinguish between the case $\lambda, -\lambda$ and the case $\overline{\lambda}, -\overline{\lambda}$.

The performance of the developed algorithm is illustrated by a series of numerical tests. The algorithm is more accurate and many times faster (when executed in Matlab) than for general Hermitian matrices of quasiseparable order two, because the action of the quasiseparable generators, which are small matrices in the previous cited paper, can be replaced by scalars, most of them real numbers.

Key words. quasiseparable, eigenstructure, Sturm property, bisection, unitary Hessenberg

AMS subject classifications. 15A18, 15B10, 15B57, 65F15

1. Introduction.

Let U be an $N \times N$ matrix with scalar entries. The matrix U is called *upper Hessenberg* if its entries below the first subdiagonal are zeros, i.e., $U_{ij} = 0$ for $i > j+1$. To find the eigenvalues of a unitary matrix, it is common to first reduce it to a unitary upper Hessenberg similar matrix and then compute the eigenvalues of the latter matrix.

The adjoint matrix N_0^* of a normal matrix, i.e., a matrix N_0 which commutes with its adjoint matrix, has the same eigenvectors as the matrix itself and its (possibly complex) corresponding eigenvalues are the conjugates of the eigenvalues of the matrix N_0. Thus the Hermitian matrices $\frac{1}{2}(N_0 + N_0^*)$ and $\frac{i}{2}(N_0^* - N_0)$ have the same eigenvectors as N_0 and their eigenvalues are the real and the imaginary part of the eigenvalues of N_0, respectively.

A unitary matrix U of size $N \times N$ is normal and its eigenvalues are on the unit circle, i.e., they are of the form

$$\lambda_k = \cos\theta_k + i\sin\theta_k, \quad \theta_k \in [0, 2\pi), \quad k = 1, \dots, N.$$

Hence, the eigenvalues of the Hermitian matrices

$$A = \frac{1}{2}(U + U^*), \quad B = \frac{i}{2}(U^* - U) \tag{1.1}$$

are in fact the cosine part and the sine part, respectively. This means that, if we find the eigenvalues of A, we readily know the eigenvalues of U, up to the sign of the imaginary part

$$\lambda_k = \cos\theta_k \pm i\sqrt{1 - \cos^2\theta_k}, \quad \theta_k \in [0, 2\pi), \quad k = 1, \dots, N.$$

[*]Received March 10, 2022. Accepted April 4, 2023. Published online on June 5, 2023. Recommended by R. Vandebril.

[†]School of Mathematical Sciences, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel-Aviv University, Ramat-Aviv 69978, Israel, (eideyu@post.tau.ac.il, iulianh@zahav.net.il).

A unitary upper Hessenberg matrix U is quasiseparable of order one (see, for instance, [5, p. 155]) and the Hermitian matrices A and B from (1.1) are quasiseparable of order two; see their quasiseparable generators of order two in formulas (3.12)–(3.15) obtained further in the paper. For any Hermitian matrix which is given by its quasiseparable structure (of any order), an algorithm for finding the eigenvalues in a fast and precise manner, is devised in [9], where the bisection method is employed. The core of the bisection procedure is an algorithm which finds quite fast how many eigenvalues are below a certain real number λ. We bring this algorithm in the present paper (see Theorem 4.3), but then we adapt it for each of the matrices A and B from (1.1), for it to be faster.

For a known eigenvalue of multiplicity one, we can then find immediately a corresponding eigenvector; see Section 6. For the eigenvectors needed in this paper, both A and B have a suited eigenvalue of multiplicity one.

Our strategy consists in finding first the cosines, by using the bisection method on the matrix A, which takes less than $670N^2$ arithmetical operations between scalars, including comparisons to 0 and conjugation, but not including loop counters and assignments.

If we consider only upper Hessenberg unitary matrices which are unreduced, i.e., they have nonzero entries on the lower subdiagonal, then the unitary matrix U will only have eigenvalues of multiplicity one; see, for instance, Ammar, Gragg, and Reichel [2, p. 386]. They speak only about positive entries, but any unitary matrix which is described by the Schur parameters has already nonnegative entries on its lower subdiagonal even if the matrix is reduced. If one subdiagonal entry is 0, we can divide the matrix into two unreduced smaller unitary Hessenberg matrices and continue to work with each of them.

The matrix A from (1.1) however, that has as eigenvalues the cosine parts, can also have double eigenvalues, since a certain cosine can correspond to both of $\pm\sqrt{1-\cos^2\theta_k}$. It follows that, if a cosine is a double root, then we already know two eigenvalues of U.

On the other hand, if a certain eigenvalue of A is simple, we have to determine whether the sine part is positive or negative. To this end, we use again, this time for the matrix B from (1.1), the algorithm which gives how many eigenvalues, i.e., sines, are below a certain nonnegative real number λ. The algorithm is slightly modified in order for it to deal with the matrix B. More precisely, for a certain $\cos\theta_k$, we want to count, for the matrix B, how many positive sines it has, just below and just above $s_k = \sqrt{1-\cos^2\theta_k}$, say at the levels

$$s_k - \delta,\ s_k + \delta, \quad \text{with} \quad \delta = 2\epsilon \log_2 N, \tag{1.2}$$

where ϵ is the machine precision and δ is equal to the worst errors in the bisection algorithm that has already been performed on the matrix A. Then, we would look at their difference and we would know if the sine corresponding to that certain $\cos\theta_k$ is equal to s_k or to $-s_k$.

Of course, we have to find first the overall multiplicity of the eigenvalues of A which correspond to plus or minus the same cosine, if there are any as such, since they both generate sines with the same absolute value; see Definition 5.1. This overall multiplicity of a cosine can be a positive integer up to 4, since, as we explained before, $-\cos\theta_k$ and $\cos\theta_k$ can have multiplicity up to 2, if they are both among the eigenvalues of A.

In Fig. 1.1, the cosines are on the horizontal axis from -1 to 1, while the height of the point on the (blue) semicircle is equal to the absolute value of the sine, which would belong to a certain cosine on the horizontal axis just below that (blue) point. Specifically, in Fig. 1.1 we show a negative cosine $-|c|$ of multiplicity 2, for which $|c|$ is also an eigenvalue of A, of multiplicity 1. Therefore, the overall multiplicity of these cosines is 3. Fig. 1.1 shows that this group of 3 eigenvalues pairs with 2 positive sines, so that we already know that the 3 eigenvalues of U are $-|c| \pm i|s|$ and $|c| + i|s|$, where $|s| = \sqrt{1-|c|^2}$.

We then determine values of interest between 0 and 1, which are borders between absolute values of sines which belong to different (possibly multiple) eigenvalues of A and we find

ETNA
Kent State University and
Johann Radon Institute (RICAM)

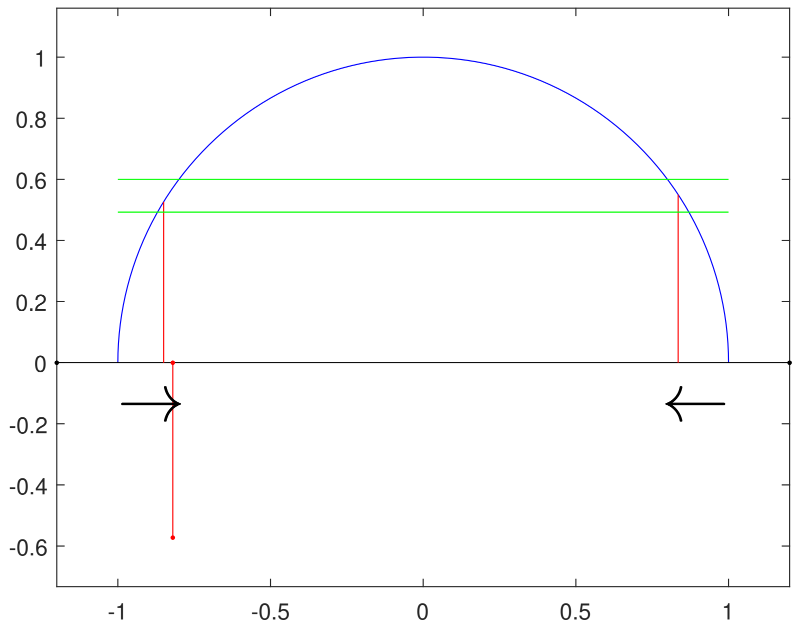

FIG. 1.1. *Starting from the left (where the cosine value is -1) and simultaneously from the right (where the cosine value is 1), we check how many cosines, from 1 up to 4, have the same absolute value and how many positive sines they pair (0, 1, or 2). The horizontal (black) line above the arrows is the axis where, between -1 at left and 1 at right, each point represents a possible cosine. The height of a point on the (blue) semicircle, is equal to the absolute value of the sine which corresponds to the cosine below that specific (blue) point. We detected 3 cosines, two of which are negative (almost equal) cosines, which of course correspond to a positive and to a negative sine. These 2 sines are depicted by two vertical (red) segments, one towards the positive sines on the semicircle and one downwards. The third cosine involved is positive and the vertical (red) segment which starts from it goes upwards. The two horizontal (green) lines cut the semicircle at two almost adjacent levels, just below and just above the two positive sines; see (1.2). When we count with our algorithm how many sines are below each of the two (green) nonnegative levels, the difference will show 2, since this is the number of the (positive) sines in between.*

for these levels the number of positive sines which are below that certain level. For example, in Fig. 1.1, the heights of the almost adjacent horizontal (green) lines which cut the (blue) semicircle are just below and just above of the positive sines which belong to those overall multiplicity 3 eigenvalues. At each of these (green) levels we check how many sines are below with a function adapted to the matrix B from (1.1). Then, when we subtract from this number the corresponding number for the previous (lower) level, we obtain the number of the positive sines that this eigenvalue of overall multiplicity 3 has. It might be only 1 or 2. If it were 0 or 3, we would have for the Hessenberg matrix U a double eigenvalue with a negative or positive sine, respectively. Up to now, we know how many (which means either 1 or 2) negative cosine values $-|c|$ we have, how many positive values $|c|$ (for the same c) and how many positive sines $|s|$ (and thus how many negative sines $-|s|$) correspond to them. This means that we know all what is needed for all but one case.

Only in certain rare circumstances we need to decide between the pair $-c + is$, $c - is$ and the pair $c + is$, $-c - is$. In this case we build in linear time the eigenvector of B which corresponds to its (multiplicity one) eigenvalue $|s|$ and we also build one row of the unitary matrix U itself, to check whether an eigenvalue of U, with a real part c and an imaginary part $|s|$, is present or not. In this way, we can settle this case, since all we have to find is whether, for $\lambda = c + is$, the eigenvalues of U are $\pm\lambda$ or $\pm\overline{\lambda}$.

Finding how many positive values of the sines are below a certain level between 0 and 1

accounts for most of the time spent, besides the bisection algorithm used to find the eigenvalues of A, which takes at least 98.3% of the total time. It follows that the time needed to find the sign of the sines is up to 1.7% of the total time and it is lower when there are more multiple eigenvalues.

Numerical experiments show that, even for matrices of size $2^{13} \times 2^{13}$, the error in finding eigenvalues does not increase, but it reaches and remains at about 20 times the machine precision, which means that on our computer the errors, computed with formula (7.1) with respect to the eigenvalues computed by Matlab, are on average $5E-15$. This small error is probably due partly to the process of building the whole matrix out of its quasiseparable generators, so that we can check our results against Matlab eigenvalues.

The paper consists of eight sections. The first section is the introduction. Section 2 recalls the basics about unitary upper Hessenberg matrices and Section 3 contains the definition of the quasiseparable representation of a matrix and the particular cases which are used in this paper. Section 4 is devoted to the bisection method, where we derive a simple algorithm to compute the real part of the eigenvalues of the unitary Hessenberg matrix U. In Section 5 we find the imaginary part of these eigenvalues. Even if one chooses to find the eigenvalues of the Hermitian matrix A by other means, this section is still needed in the present form. Section 6 is devoted to eigenvectors. We can find them only for eigenvalues of multiplicity one and only for Hermitian matrices, i.e., for the matrices from (1.1) or for Hessenberg matrices, i.e., for U itself. Section 7 contains the results of numerical tests. Section 8 contains concluding remarks.

Sections 2, 3.1, 3.2, 4.1, 4.2, 4.3, and the beginning of Section 4.4, which occupy in total a little more than 3 pages, do not contain new results, but are brought here to establish notation and in order for the paper to be self-contained.

For an $N \times N$ matrix A we denote by A_{ij} or by $A(i,j)$ its element in row $1 \le i \le N$ and column $1 \le j \le N$ and the Matlab notation $A(i:j, p:q)$ is used for submatrices.

We use the scientific notation for very small numbers, for instance $5E-15$ stands for $5 \cdot 10^{-15}$.

2. Unitary upper Hessenberg matrices.

We consider here upper Hessenberg matrices U which are also unitary, upon [5, p. 156]. Also, we suppose that U is unreduced, which means that the lower subdiagonal entries are different from 0.

If the subdiagonal entries of the matrix U are positive, the matrix U has the representation

(2.1)
$$U = \begin{bmatrix} -\rho_1\rho_0^* & -\rho_2\mu_1\rho_0^* & -\rho_3\mu_2\mu_1\rho_0^* & \cdots & -\rho_{N-1}\mu_{N-2}\cdots\mu_1\rho_0^* & -\rho_N\mu_{N-1}\cdots\mu_1\rho_0^* \\ \mu_1 & -\rho_2\rho_1^* & -\rho_3\mu_2\rho_1^* & \cdots & -\rho_{N-1}\mu_{N-2}\cdots\mu_2\rho_1^* & -\rho_N\mu_{N-1}\cdots\mu_2\rho_1^* \\ 0 & \mu_2 & -\rho_3\rho_2^* & \cdots & -\rho_{N-1}\mu_{N-2}\cdots\mu_3\rho_2^* & -\rho_N\mu_{N-1}\cdots\mu_3\rho_2^* \\ \vdots & \ddots & \mu_3 & & \vdots & \vdots \\ \vdots & & \ddots & \ddots & -\rho_{N-1}\rho_{N-2}^* & -\rho_N\mu_{N-1}\rho_{N-2}^* \\ 0 & \cdots & \cdots & 0 & \mu_{N-1} & -\rho_N\rho_{N-1}^* \end{bmatrix},$$

where

(2.2)
$$\mu_k > 0, \quad |\rho_k|^2 + \mu_k^2 = 1 \quad (k = 1, \dots, N-1), \quad \rho_0 = -1, \quad |\rho_N| = 1.$$

Note that in the following we mostly use the notation

(2.3)
$$\mu(k), \quad \rho(k) \quad (k = 1, \dots, N-1), \quad \rho(N)$$

instead of μ_k, ρ_k from (2.2).

Note that some authors, for instance Gragg [12], set $\rho_0 = 1$. In this case, to obtain the same matrix, all the signs of the Schur parameters ρ_k, $k = 0, 1, \dots, N$, should be changed.

Note also that an unreduced unitary Hessenberg matrix has only eigenvalues of multiplicity 1 (see, for instance, [2, p. 386]) and that if the matrix is reduced, then it has in fact two or more unreduced smaller unitary Hessenberg matrices as blocks and these can then be treated separately.

3. The quasiseparable representation. In this section we define the basic notions concerning quasiseparable representations of matrices and we discuss the particular cases that are used in this paper. This type of structured matrices have been introduced in [4] and eigenvalues for quasiseparable matrices of order one have been found starting with [7]. For other classes of structured matrices, see the monographs of Vandebril, Van Barel, and Mastronardi [17, 18].

3.1. The general quasiseparable structure. The following definitions and more can be found in the the book [5, Part I].

Let $\{a(k)\}$ be a family of matrices of sizes $r_k \times r_{k-1}$. For positive integers i, j, $i > j$, define the operation $a^{>}_{ij}$ as follows

$$a^{>}_{ij} = a(i-1)\cdots a(j+1), \qquad \text{for} \quad i > j+1,$$

and

$$a^{>}_{j+1,j} = I_{r_j}, \tag{3.1}$$

i.e., the identity matrix of size $r_j \times r_j$.

Let $\{b(k)\}$ be a family of matrices of sizes $r_{k-1} \times r_k$. For positive integers i, j, $j > i$, define the operation $b^{<}_{ij}$ as follows

$$b^{<}_{ij} = b(i+1)\cdots b(j-1), \qquad \text{for} \quad j > i+1,$$

and

$$b^{<}_{i,i+1} = I_{r_i},$$

i.e., the identity matrix of size $r_i \times r_i$.

Let $A = \{A_{ij}\}_{i,j=1}^{N}$ be a matrix with scalar entries A_{ij}. Assume its entries are represented in the form

$$A_{ij} = \begin{cases} p(i)a^{>}_{ij}q(j), & 1 \le j < i \le N, \\ d(i), & 1 \le i = j \le N, \\ g(i)b^{<}_{ij}h(j), & 1 \le i < j \le N. \end{cases} \tag{3.2}$$

Here

$$\{p(i)\}_{i=2}^{N}, \quad \{q(j)\}_{j=1}^{N-1}, \quad \text{and} \quad \{a(k)\}_{k=2}^{N-1} \tag{3.3}$$

are matrices of sizes $1 \times r^L_{i-1}$, $r^L_j \times 1$, and $r^L_k \times r^L_{k-1}$, respectively,

$$\{g(i)\}_{i=1}^{N-1}, \quad \{h(j)\}_{j=2}^{N}, \quad \{b(k)\}_{k=2}^{N-1} \tag{3.4}$$

are matrices of sizes $1 \times r^U_i$, $r^U_{j-1} \times 1$, $r^U_{k-1} \times r^U_k$, respectively, and $\{d(i)\}_{i=1}^{N}$ are (possibly complex) numbers. The representation of a matrix A in the form (3.2) is called a *quasiseparable representation*. The elements in (3.3), (3.4) and $\{d(i)\}_{i=1}^{N}$ are called *quasiseparable generators* of the matrix A. The numbers r^L_k, r^U_k $(k = 1, \dots, N-1)$ are called the *orders* of these generators. The elements in (3.3) and (3.4) are called also *lower quasiseparable generators* and, respectively, *upper quasiseparable generators* of the matrix A. In fact, the generators $p(i), g(i)$, and $q(j), h(j)$ are row and column vectors, respectively, of appropriate size.

3.2. Quasiseparable structure of Hermitian matrices. For a Hermitian matrix the diagonal entries $d(k), k = 1, \dots, N$, are real and the upper quasiseparable generators can be obtained from the lower ones by taking

$$g(k) = (q(k))^*, \quad h(k) = (p(k))^*, \quad b(k) = (a(k))^*, \quad k = 2, \dots, N-1,$$

$$g(1) = (q(1))^*, \quad h(N) = (p(N))^*.$$

See more on the definition and properties of the quasiseparable structure of (Hermitian) matrices in the book [5], starting with Section 4.2.

3.3. The quasiseparable generators used in this paper. For a unitary upper Hessenberg matrix U as in (2.1), we use the generators

$$p(k) = \mu(k-1),\ h(k) = -\rho(k), \qquad k = 2, \dots, N, \tag{3.5}$$

$$g(k) = \mu(k)\overline{\rho(k-1)},\ q(k) = 1, \qquad k = 1, \dots, N-1, \tag{3.6}$$

$$a(k) = 0,\ b(k) = \mu(k), \qquad k = 2, \dots, N-1, \tag{3.7}$$

$$d(1) = \rho(1),\ d(k) = -\rho(k)\overline{\rho(k-1)}, \qquad k = 2, \dots, N, \tag{3.8}$$

where $\rho(k), \mu(k),\ k = 1, \dots, N-1$, and $\rho(N)$ have been defined in (2.3).

We note that for U^* the following quasiseparable generators can be used

$$h(k) = \mu(k-1),\ p(k) = -\overline{\rho(k)}, \qquad k = 2, \dots, N, \tag{3.9}$$

$$q(k) = \mu(k)\rho(k-1),\ g(k) = 1, \qquad k = 1, \dots, N-1, \tag{3.10}$$

$$b(k) = 0,\ a(k) = \mu(k), \qquad k = 2, \dots, N-1, \tag{3.11}$$

$$d(1) = \overline{\rho(1)},\ d(k) = -\rho(k-1)\overline{\rho(k)}, \qquad k = 2, \dots, N.$$

Then it follows that for the matrix $A = \frac{1}{2}(U + U^*)$ from (1.1) we can use the lower quasiseparable generators

$$p(k) = \begin{bmatrix} \frac{1}{2} & -\frac{\overline{\rho(k)}}{2} \end{bmatrix}, \qquad k = 2, \dots, N, \tag{3.12}$$

$$q(k) = \begin{bmatrix} \mu(k) & \mu(k)\rho(k-1) \end{bmatrix}^T, \qquad k = 1, \dots, N-1, \tag{3.13}$$

$$a(k) = \begin{bmatrix} 0 & 0 \\ 0 & \mu(k) \end{bmatrix}, \qquad k = 2, \dots, N-1, \tag{3.14}$$

$$d(1) = \Re(\rho(1)),\ \ d(k) = -\Re(\rho(k-1)\overline{\rho(k)}), \qquad k = 2, \dots, N,$$

where $\Re$ denotes the real part. From (3.2) one can see that only on the first (lower) subdiagonal $a(k)$ does not intervene, since by (3.1) it is the 2×2 identity matrix and then the product $p(k+1)q(k)$, $k = 1, \dots, N-1$, of the quasiseparable generators from (3.12) and (3.13) gives indeed the first subdiagonal of half of $U + U^*$. This is the only lower subdiagonal to which both U and U^* contribute. On the other lower subdiagonals, due to the zeroes in $a(k)$ from (3.14), the first components of $p(k)$ from (3.12) and $q(k)$ from (3.13) are annihilated, while the product of second components gives exactly the (lower) quasiseparable generators of U^*, as they have been mentioned in (3.9)–(3.11).

For $B = \frac{i}{2}(U^* - U)$ we use $q(k), a(k)$ as in (3.13), (3.14) respectively, while

$$p(k) = \begin{bmatrix} -\frac{i}{2} & -\frac{\overline{\rho(k)}i}{2} \end{bmatrix}, \quad k = 2, \dots, N, \tag{3.15}$$

$$d(1) = \Im(\rho(1)),\ d(k) = -\Im(\overline{\rho(k-1)}\rho(k)), \quad k = 2, \dots, N,$$

where $\Im$ denotes the imaginary part. Note that both A and B share the same $a(k)$ and $q(k)$.

Hermitian matrices given in quasiseparable form have all the data concentrated in their lower quasiseparable generators which contain, together with the diagonal entries, for quasiseparable matrices of order two, $9N - 12$ (possibly complex) numbers. However, for the matrices A and B from (1.1), about a third of these numbers, namely $3N-6$, must be zeros and another $4N - 4$ for A and $3N - 2$ for B are real numbers. Only at most $2N - 2$ of the numbers in the quasiseparable structure of A can be not real. Moreover, the middle generators (3.14), with which in [9] we multiply the most, have only one nonzero entry which is real. This gives us the opportunity to devise a very fast algorithm to find how many eigenvalues the matrices in (1.1) have that are smaller than a given real number λ. In order to answer this question we need less than $17N$ operations. This is one of the explanations for why, among all the quasiseparable Hermitian matrices of order higher than one, the matrices we consider produce better results in terms of time and precision. The best results have been obtained only for band symmetric Toeplitz matrices in [10], since almost all the entries of their quasiseparable generators, with which we multiply, are 0 or 1. That is why in that case the eigenvectors are exact.

4. The bisection method.

Here we present the basic algorithm to compute the eigenvalues of a Hermitian matrix with a given quasiseparable representation.

4.1. The Sturm sequences property for a Hermitian matrix.

In order to obtain information on the location of the eigenvalues of a Hermitian matrix, we will apply the well-known Sturm property in the form presented, for instance, in the monograph [15, p. 296].

THEOREM 4.1. *Let A be an $N \times N$ Hermitian matrix with determinants of leading principal submatrices $\gamma_k = \det A(1:k, 1:k),\ k = 1, \dots, N$. Set $\gamma_0 = 1$ and assume that*

$$\gamma_k \neq 0, \quad k = 1, 2, \dots, N.$$

Then, the number of negative eigenvalues of A (counting each eigenvalue in accordance with its multiplicity) is equal to the number of alternations of sign in the sequence

$$\gamma_0, \gamma_1, \dots, \gamma_{N-1}, \gamma_N.$$

In fact, we will use ratios of consecutive Sturm polynomials $\gamma_k(\lambda)$ and we will check whether these ratios are negative, as a method of counting sign changes.

COROLLARY 4.2. *Let A be an $N \times N$ Hermitian matrix, λ_0 be a real number and*

$$\gamma_0(\lambda_0) \equiv 1, \quad \gamma_k(\lambda_0) = \det(A(1:k, 1:k) - \lambda_0 I),\ k = 1, 2, \dots, N.$$

Assume that $\gamma_k(\lambda_0) \neq 0, k = 1, 2, \dots, N$, and set

$$D_k(\lambda_0) = \frac{\gamma_k(\lambda_0)}{\gamma_{k-1}(\lambda_0)}, \quad k = 1, 2, \dots, N. \tag{4.1}$$

Then, the number of eigenvalues of A which are smaller than λ_0 is equal to the number of negative entries in the sequence (4.1).

4.2. The bisection procedure.

We perform the bisection method similarly to the one presented in [11, p. 467] for symmetric tridiagonal matrices. For a real λ we denote by $\nu(\lambda)$ the number of negative entries in the sequence (4.1). Let b_L and b_U be lower and upper bounds for the range of the eigenvalues, obtained previously, for instance using some matrix norm. In our case, for the matrix $A = (U + U^*)/2$ we take $b_L = -1, b_U = 1$. To compute $\lambda = \lambda_k(A)$, i.e., the kth largest eigenvalue of A for some prescribed k, one proceeds as follows.

We start with $z = b_U, y = b_L$ and we perform repeatedly the assignment $\lambda = \frac{z+y}{2}$ (until we attain machine precision), and each time we set either $z = \lambda$, if $\nu(\lambda) \geq k$, or $y = \lambda$, otherwise.

4.3. The basic eigenvalue algorithm. The algorithm presented in [5, Theorem 18.2] yields in particular the ratios of determinants of principal leading submatrices of a matrix A with a given quasiseparable representation. Applying this result to the matrix $A - \lambda I$ we obtain recursive relations for the numbers in the sequence (4.1).

4.4. Find the number $\nu(\lambda)$ of eigenvalues that are less than a given real λ. We reproduce the relevant result from [9], which will help us showing that, under the same conditions, the faster algorithms we devised for the matrices in (1.1) work.

THEOREM 4.3. **(findNu(λ,generators))** *Let A_0 be an $N \times N$ Hermitian matrix and let λ be a real number such that*

$$\det\left(A_0(1:k, 1:k) - \lambda I_k\right) \neq 0, \quad k = 1, 2, \ldots, N. \tag{4.2}$$

Assume that A_0 has lower quasiseparable generators as in (3.3) of orders r_k^L, $k = 1, \ldots N-1$, and also diagonal entries $d(k)$, for $k = 1, \ldots, N$.

Then, the number $\nu(\lambda)$ of the eigenvalues of A_0 which are smaller than λ is obtained via the following algorithm.

Step 1. Compute

$$D_1(\lambda) = d(1) - \lambda, \quad u_1(\lambda) = q(1)\frac{1}{D_1(\lambda)}, \quad f_1(\lambda) = u_1(\lambda)q^*(1). \tag{4.3}$$

If $D_1(\lambda) < 0$, set $\nu = 1$, otherwise, set $\nu = 0$.

Step 2. For $k = 2, \ldots, N-1$ compute

$$D_k(\lambda) = d(k) - \lambda - p(k)f_{k-1}(\lambda)p^*(k), \tag{4.4}$$

$$u_k(\lambda) = [q(k) - a(k)f_{k-1}(\lambda)p^*(k)]\frac{1}{D_k(\lambda)}, \tag{4.5}$$

$$f_k(\lambda) = a(k)f_{k-1}(\lambda)a^*(k) + u_k(\lambda)[q^*(k) - p(k)f_{k-1}(\lambda)a^*(k)]. \tag{4.6}$$

If $D_k(\lambda) < 0$, set $\nu := \nu + 1$.

Step 3. Compute

$$D_N(\lambda) = d(N) - \lambda - p(N)f_{N-1}(\lambda)p^*(N).$$

If $D_N(\lambda) < 0$, set $\nu := \nu + 1$.

Set $\nu(\lambda) = \nu$.

We want to mention that condition (4.2) above is not too restrictive. For instance, for $N \times N$ matrices it requires that only N such computed real numbers not to be zero. The matrices which satisfy this condition are called strictly regular.

Since for a matrix of size 1024×1024, for instance, the following Algorithm 4.4 is used by the bisection procedure about $43N$ times to find the cosines with maximal precision (the coefficient of N descending gradually from $51.5N$ for $N = 4$, down to $39.7N$ for $N = 8192$), it is very important to prepare for it as many operations as we can, in the calling function. This is Step 0 in the following algorithm.

ALGORITHM 4.4. **Find ν for the matrix A from (1.1).** The present algorithm receives the Schur parameters $\rho(k)$, $k = 1, \ldots, N$, of the unitary upper Hessenberg matrix U, its size N and a real number λ and computes the number ν of the eigenvalues of the matrix A which are smaller than λ. In order for the algorithm to be fast and in order not to recompute the same data again and again, it will also receive from the calling function the results of Step 0 below.

Step 0 Compute $r(k) = \rho(k-1),\ k = 2, \dots, N-1$, $c_\rho(k) = \overline{\rho(k)},\ k = 2, \dots, N$, the diagonal entries of A, namely

$$d(1) = \mathrm{real}(\rho(1)),\ d(k) = -\mathrm{real}(r(k)c_\rho(k)), \quad k = 2, \dots, N, \tag{4.7}$$

and $\mu_2(k) = \mu^2(k)$, $k = 1, \dots, N-1$, in the function which calls this algorithm.

Step 1 1.1 Set $D = d(1) - \lambda$ and $\nu = 1$, if $D < 0$, or $\nu = 0$, otherwise. Compute $\tilde{f} = \mu_2(1)/D$ and set $\tilde{g} = -\tilde{f}, \tilde{h} = \tilde{f}$.

1.2 For $s = 2, \dots, N-1$ compute

$$P = \tilde{g} - \tilde{h}\rho(s), \quad m = r(s) - P/2, \tag{4.8}$$

$$D = d(s) - \lambda + \mathrm{real}(c_\rho(s)(\tilde{g} + P) - \tilde{f})/4, \tag{4.9}$$

$$\tilde{f} = \mu_2(s)/D, \quad \tilde{g} = \tilde{f}m, \quad \tilde{h} = \tilde{h}\mu_2(s) + \tilde{g}\overline{m} \tag{4.10}$$

and set $\nu = \nu + 1$, if $D < 0$.

1.3 If $d(N) - \lambda + \mathrm{real}(c_\rho(N)(2\tilde{g} - \tilde{h}\rho(N)) - \tilde{f})/4 < 0$, set $\nu = \nu + 1$.

Proof. We have to prove that in fact Algorithm 4.4 is just a specialized form of Theorem 4.3, which, in turn, has been proved to be true in [9].

First note that $a(k) = a^*(k) = \begin{bmatrix} 0 & 0 \\ 0 & \mu(k) \end{bmatrix}$, $k = 2, \dots, N$, so that, when multiplied from the left, only the second line of the result is not zero while, when multiplied from the right, only the second column will be not zero.

We denote the entries of the matrix f from (4.3)–(4.6) as $\begin{bmatrix} \tilde{f} & * \\ \tilde{g} & \tilde{h} \end{bmatrix}$. The entry $f(1,2)$ is the conjugate of $\tilde{g}$, so that it does not bring new information and it is not needed.

It follows that the first part of formula (4.6) is equal to $\begin{bmatrix} 0 & 0 \\ 0 & \mu^2(k)\tilde{h} \end{bmatrix}$. The second part is, using the notation of (4.5), $u_k u_k^* D_k$. Due to a multiplication by $a(k)$, the first entry of u_k remains equal to the first entry of $q(k)$, which for our matrix A is $\mu(k)/D_k$. The second entry is the second entry of $q(k)$, namely $\mu(k)\rho(k-1)$, minus $\tilde{g}\overline{p_1} + \tilde{h}\overline{p_2}$, where p_1, p_2 are the entries of $p(k)$, namely 0.5 and $-0.5\overline{\rho(k)}$.

Then, formula (4.10) becomes

$$\begin{bmatrix} \tilde{f} & * \\ \tilde{g} & \tilde{h} \end{bmatrix} = \mu^2(k) \left[\begin{bmatrix} 0 & 0 \\ 0 & \tilde{h} \end{bmatrix} + tt^*/D_k \right], \tag{4.11}$$

where t is in fact $t = u_k D_k/\mu(k)$, which is equal to $t = \begin{bmatrix} 1 \\ \rho(k-1) - (\tilde{g} - \tilde{h}\rho(s))/2 \end{bmatrix}$. In formula (4.8) we denoted $P = \tilde{g} - \tilde{h}\rho(s), m = \rho(k-1) - P/2$, therefore $t(2) = m$.

It follows from (4.11) that $\tilde{h} = \tilde{h}\mu_2(s) + \overline{m}m/D_k, \tilde{g} = \mu_2(s)m/D_k, \tilde{f} = \mu_2(s)/D_k$, thus we proved (4.10) with the notations (4.8). ∎

We extract the real part of a real number, since in practice we noticed that it might have a small, but non-vanishing, imaginary part which leads to wrong results.

REMARK 4.5. Note that compute the eigenvalues of the matrix $2A = U + U^*$ is cheaper than computing those of A itself and at the end of the process to divide all the such obtained eigenvalues by 2. In order to do that, just multiply by 2 all the N diagonal entries which are computed in (4.7) and do not divide by 2 in (4.8) and in Step 1.3, and by 4 in (4.9). This will save about $80N^2$ divisions between scalars, with the cost of N multiplications.

If we count elementary scalar operations including comparisons to 0, conjugation, and taking the real part, then the total complexity of Step 1 if we use Remark 4.5 is $14.5N - 18.5$. We did not include assignments or the loop counter. Note that the complexity, as compared to a general quasiseparable Hermitian matrix of order two (as in [9]) has been reduced by almost two thirds and that only about $3N$ operations involve also two complex numbers.

Since the following Algorithm 4.6 is used about $0.75N$ times when finding the signs of the sines, it is very important to prepare for it as many operations as we can, in the calling function. This is Step 0 in the following algorithm.

ALGORITHM 4.6. **Find ν for the matrix B from (1.1).** The present algorithm receives the Schur parameters $\rho(k),\ k = 1, \dots, N$, of the unitary upper Hessenberg matrix U, its size N and a real number λ and it computes the number ν of the eigenvalues of the matrix B which are smaller than λ. In order for the algorithm to be fast, it will also receive from the calling function the same results as in Step 0 of Algorithm 4.4, where (4.7) is replaced by

$$d(1) = \text{imag}(\rho(1)),\ d(k) = \text{imag}(r(k)c_\rho(k)), \quad k = 2, \dots, N.$$

Step 1 1.1 Perform Step 1.1. of Algorithm 4.4.
1.2 For $s = 2, \dots, N-1$, compute

$$P = \tilde{g} + \tilde{h}\rho(s), \quad m = r(s) - iP/2, \tag{4.12}$$

$$D = d(s) - \lambda - \text{real}(c_\rho(s)(\tilde{g} + P) + \tilde{f})/4 \tag{4.13}$$

and (4.10) and if $D < 0$ set $\nu = \nu + 1$.
1.3 If $d(N) - \lambda - \text{real}(c_\rho(N)(2\tilde{g} + \tilde{h}\rho(N)) + \tilde{f})/4 < 0$, set $\nu = \nu + 1$.

Proof. The matrix B has the same generators as the matrix A, except for a small change in $p(k),\ k = 2, \dots, N$. From (3.12) and (3.15) we see that $p(k) = \begin{bmatrix} \frac{1}{2} & -\frac{\overline{\rho(k)}}{2} \end{bmatrix}$ for the matrix A, while for the matrix B, we have $p(k) = \begin{bmatrix} -\frac{i}{2} & -\frac{\overline{\rho(k)}i}{2} \end{bmatrix}$. □

Like in Remark 4.5, it is cheaper to find ν for the matrix $2B = i(U^* - U)$ instead of for B itself. To this end, just multiply by 2 the N diagonal entries which are computed before Step 1 of Algorithm 4.6 and do not divide by 2 in (4.12) and in Step 1.3, and by 4 in (4.13). "

4.5. Bisection for the real part of the eigenvalues of a unitary upper Hessenberg matrix. The algorithm for bisection is an adaptation of the one in [3] in which the authors use two arrays (the ones we called L_0, R_0) to store the data about the left and right bounds of the eigenvalues that are yet to be found. This approach yields that the number of iterations in the *while* loop climbs down from about 51.5 for the first found eigenvalues of A, in our case, to about 27.5 for the last found eigenvalues, when 8192 eigenvalues are to be found.

No scaling, as the one described in [8, 9, 10], is needed here, since the eigenvalues have already an absolute value which is at most 1.

The following algorithm receives the Schur parameters and returns the kth larger eigenvalue as the kth entry of the array R_0.

ALGORITHM 4.7.

Step 1 1.1 Perform Step 0 of Algorithm 4.4.
1.2 For $k = 1, \dots, N$, set $L_0(k) = -1, R_0(k) = 1$.

```
        1.3  Set $U = 1$.
**Step 2** For $k = N, \dots, 2, 1$, perform the following steps
        2.1  Set $L = \max_{1 \le j \le k} L_0(j)$ and $U = \min\{U, R_0(k)\}$.
        2.2  While $U - L$ is larger than the machine precision, perform the following.
            2.2.1  Set $\lambda = (L + U)/2$.
            2.2.2  Use Step 1 of Algorithm 4.4 to obtain the number $\nu$ of eigenvalues of the
                   matrix $A$ which are less than $\lambda$.
            2.2.3  If $\nu \ge k$, set $U = \lambda$; otherwise, set $L = \lambda$, $L_0(\nu+1) = L$, and $R_0(\nu) = L$,
                   if $\nu \ne 0$ and $R_0(\nu) > L$.
        2.3  Set $R_0(k) = (U + L)/2$ to contain the $k$th eigenvalue.
```

If one uses Remark 4.5, as it is recommended, one has to change in Steps 1.2 and 1.3 of the preceding algorithm the assignments of 1 and -1 to 2 and -2, respectively

The above algorithm proves to be slightly faster than the alternative algorithm that we propose next, for matrices up to the size 1024×1024. For sizes of 2048×2048 and larger, the following algorithm proves to be slightly faster. We also write in two arrays, which we have called again L_0, R_0, data about the left and right bounds of the eigenvalues that are yet to be found.

The following algorithm receives the arrays L_0, R_0, the number f of the smaller eigenvalues and the number n of the eigenvalues that are yet to be found. Since it is a recursive algorithm, it returns them and calls itself again.

ALGORITHM 4.8.

```
**Step 1** 1.1. Set $L_0(1) = -1, R_0(N) = 1$, $f = 0, n = N$, and $\epsilon$ equal to the machine
           precision.
           1.2. Set $\lambda = (L_0(f+1) + R_0(f+n))/2$ and use Algorithm 4.4 to find the number
           $\nu$ of eigenvalues of the matrix $A$ which are smaller than $\lambda$.
           1.3. Set $n1 = \nu - f, n2 = n - n1$.
**Step 2**  2.1  If $n1 > 0$, set $R_0(\nu) = \lambda$ and, if $n1 \ne 1$, perform either 2.2 or 2.3.
            2.2  If $n1 = 2$, set $R_0(f+1) = \lambda, L_0(\nu) = L_0(f+1)$, and, if $\lambda - L_0(f+1) > \epsilon$,
                 use this Algorithm 4.8 again, starting with 1.2, for $n1$ instead of $n$, to find the
                 arrays $L_0, R_0$.
            2.3  If $n1 > 2$, use this Algorithm 4.8 again, starting with 1.2., for $n1$ instead of $n$,
                 to find the arrays $L_0, R_0$.
**Step 3**  3.1  If $n2 > 0$, set $L_0(\nu+1) = \lambda$ and, if $n2 \ne 1$, perform either 3.2 or 3.3.
            3.2  If $n2 = 2$, set $L_0(\nu+2) = \lambda, R_0(\nu+1) = R_0(\nu+2)$, and, if $R_0(\nu+2) - \lambda > \epsilon$,
                 use this Algorithm 4.8 again, starting with 1.2, for $n2$ instead of $n$ and $\nu$ instead
                 of $f$, to find the arrays $L_0, R_0$.
            3.3  If $n2 > 2$, use this Algorithm 4.8 again, starting with 1.2, for $n1$ instead of $n$
                 and $\nu$ instead of $f$, to find the arrays $L_0, R_0$.
**Step 4** For $k = 1, \dots, N$, set $U = R_0(k), L = L_0(k)$, and perform 4.1. and 4.2.
            4.1  While $U - L$ is larger than the machine precision, perform 4.1.1 and 4.1.2.
                4.1.1  Set $\lambda = (L + U)/2$ and use Algorithm 4.4 to obtain the number $\nu$ of
                       eigenvalues of the matrix $A$ which are smaller than $\lambda$.
                4.1.2  If $\nu \ge k$, set $U = \lambda$, otherwise, set $L = \lambda$.
            4.2  Set $R_0(k) = (U + L)/2$ to contain the $k$th eigenvalue.
```

If one uses Remark 4.5, then in Step 1.1 of Algorithm 4.8 one must take $L_0(1) = -2, R_0(N) = 2$, instead of -1 and 1, respectively.

5. Finding the sign of the imaginary part for an $N \times N$ upper Hessenberg matrix U.

In this section we present how to find the sign of the imaginary part for an upper Hessenberg matrix.

5.1. The case of an orthogonal U. If all the Schur parameters $\rho(k),\ k = 1,\dots,N$, of the matrix U are real, then U is an orthogonal matrix.

Once we obtain the eigenvalues of the Hermitian matrix $A = (U + U^*)/2$, we know everything we need. The eigenvalues $\lambda_A(k)$, $k = 1,\dots,N$, of A are sorted in increasing order and at least $N-2$ of them have multiplicity two. For an even N, either all the N eigenvalues are doubled, or $\lambda_A(1) = -1$, $\lambda_A(N) = 1$, and the other eigenvalues come in $(N-2)/2$ pairs. One can establish in a stable way in which case A is, by checking whether

$$\sum_{k=1}^{N/2}(\lambda_A(2k) - \lambda_A(2k-1)) < \sum_{k=1}^{N/2-1}(\lambda_A(2k+1) - \lambda_A(2k)),$$

since in exact arithmetic one of these sums is 0. If $\lambda_A(1) \neq -1$, the left sum is smaller and also $\lambda_A(N) \neq 1$.

If $\lambda_A(N) = 1$, then set $\lambda_U(N) = 1, M = N-2$, and, if N is odd, then the double eigenvalues start with $\lambda_A(p)$, where $p = 1$, otherwise, i.e., if N is even, $\lambda_U(1) = -1, p = 2$.

Else (if $\lambda_A(N) \neq 1$), then $M = N-1$ and if N is odd, then $\lambda_U(1) = -1, p = 2$, else, if N is even, $p = 1$.

In order to find the other eigenvalues of U, for $k = p$ up to $k = M$, set

$$c = (\lambda_A(k) + \lambda_A(k+1))/2, \quad s = \sqrt{1-c^2},$$

$$\lambda_U(k) = c + is, \quad \lambda_U(k+1) = c - is, \quad k = k+2.$$

The number of arithmetical operations is, thus, less than $11N + 5$.

In Section 7.3 we mention that more than 99% of the double eigenvalues are recognized as such, even if we asked for a very small distance between them. The other eigenvalues are not lost, but are treated as eigenvalues of multiplicity 1.

5.2. Finding the sign of the imaginary part for a complex matrix U. After obtaining the real part of the eigenvalues $\cos\theta \pm i\sqrt{1-\cos^2\theta}$, we first compute the absolute value of the sine (of the imaginary part). Note that the cosines are returned by the bisection algorithm of the matrix $A = \frac{1}{2}(U + U^*)$ in increasing order between -1 and 1, so that they do not need to be sorted. Denote the array which contains the cosines by C and the array which will contain the absolute values of the sines with S_A. Then,

$$-1 \le C(1) \le C(2) \le \cdots \le C(N) \le 1, \quad S_A(k) = \sqrt{1 - C^2(k)},\ k = 1,\dots,N.$$

The multiplicity of the eigenvalues of a unitary Hessenberg unreduced matrix can only be 1, but the multiplicity of a certain cosine can also be 2, since it can belong to 2 complex conjugate eigenvalues. We define a very small interval, inside which two cosines will be treated as belonging to a double eigenvalue of A. In order to determine such multiplicities of the cosines, we only have to check each cosine with its next neighbour. The small interval that we have chosen has a radius approximately equal to the worst errors that the bisection on the matrix A can give, i.e., $\delta = 2(\log_2 N)\epsilon$ from (1.2). Denote with M the array containing the multiplicities of the cosines from C, with σ_m, σ_M the minimal and, respectively the maximal sines which correspond to the $M(k)$ eigenvalues and by P an array which contains the pair of an eigenvalue for which $M(k) = 2$.

Summarizing the steps

Step 1 Take $M(N) = 1$ and initialize $P(k) = 0,\ k = 1,\dots,N$, and $k = 1$.

Step 2 While $k < N$ perform Step 3.

Step 3 3.1 While $C(k+1)-C(k) \geq \delta$ and $k < N$, set

$$M(k) = 1, \quad \sigma_m(k) = S_A(k), \quad \sigma_M(k) = S_A(k),$$

and $k = k+1$.

3.2 If $k < N$ perform $M(k) = 2, M(k+1) = 2, P(k) = k+1, P(k+1) = k,$

$$\begin{aligned} \sigma_m(k) &= \min(S_A(k), S_A(k+1)), \\ \sigma_m(k+1) &= \min(S_A(k), S_A(k+1)), \\ \sigma_M(k) &= \max(S_A(k), S_A(k+1)), \\ \sigma_M(k+1) &= \max(S_A(k), S_A(k+1)), \end{aligned}$$

and $k = k+2$.

If the multiplicity of a certain cosine satisfies $M(k) = 2$, then we already know two (conjugated) eigenvalues of the unitary matrix U. If a cosine and its opposite are both present in the array C, then they should be treated together, since they share the same range of sines.

DEFINITION 5.1. *We call* overall multiplicity *the sum of the multiplicities of a certain cosine and of its opposite cosine.*

The overall multiplicity would be a number between 1 and 4, since both the cosines $\pm c$ can be present and each can have a multiplicity up to 2. We build an array G which contains the ordering number g of the group (up to 4 members) to which a certain cosine and the corresponding absolute value of the sine belong. At the same time, we build an array O_M which contains for each group of eigenvalues their (common) overall multiplicity, in the following way. We examine one by one the cosines from the beginning (from the cosine $C(1)$, which is closer to -1) towards the end and, at the same time, step by step, from the end, (i.e., from the cosine $C(N)$ which is closer to 1) towards the beginning, so that we reach the two values $\pm c$, which generate almost the same sines, at almost the same time. Namely, we take two counters $s = 1, 2, 3, \ldots$, and $t = N, N-1, N-2, \ldots$, and while $C(s) < 0$ and $C(t) > 0$ we check whether $|C(s) + C(t)| < \delta$. If this happens, then the overall multiplicity O_M of the eigenvalues which belong to that group is $M(s) + M(t)$. Note that to build the arrays G and O_M, we need only about $2N$ comparisons with $\pm\delta$.

We denote by g the ordering number of the last group, which is in fact the number of the groups. Then $g \geq N/4$ (which could be reached if all the groups had 4 members) and $g \leq N$. We build also an array S_m of length g which contains the minimal absolute value of the sines in each group, a similar array S_M, which contains their maximums, and an array P_C that contains for each group the multiplicity of the positive cosine in the group, which can be 0 when the group has only eigenvalues with negative cosine.

After setting $s = 1, t = N, g = 1$ and while $C(s) < 0, C(t) > 0$, perform Step 1 and Step 2.

Step 1 While $\sigma_m(s) \leq \sigma_m(t)$ and $C(s) < 0, C(t) > 0$.

1.1 If $|C(s) + C(t)| \geq \delta$,

$$O_M(g) = M(s), S_M(g) = \sigma_M(s), P_C(g) = 0, \tag{5.1}$$

otherwise

$$\begin{aligned} O_M(g) &= M(s) + M(t), \quad P_C(g) = M(t), \\ S_M(g) &= \max(\sigma_M(s), \quad \sigma_M(t)), G(t) = g, \end{aligned}$$

and, if $P(t) \neq 0$, $G(P(t)) = g$. Set $t = t - M(t)$.

```
        1.2 Set
            (5.2)   $S_m(g) = \sigma_m(s),\ G(s) = g,$ if $P(s) \neq 0,\ G(P(s)) = g,$
            (5.3)   $s = s + M(s), g = g + 1.$
Step 2  While $\sigma_m(t) < \sigma_m(s)$ and $C(s) < 0, C(t) > 0,$
        2.1 If $|C(s) + C(t)| \geq \delta,$
            (5.4)   $O_M(g) = M(t), \quad S_M(g) = \sigma_M(t),$
            otherwise
                $O_M(g) = M(s) + M(t),\ S_M(g) = \max(\sigma_M(s),\ \sigma_M(t)), G(s) = g,$
            and, if $P(s) \neq 0$, $G(P(s)) = g$. Set $s = s + M(s)$.
        2.2
            (5.5)
              $S_m(g) = \sigma_m(t),\ P_C(g) = M(t),\ G(t) = g,$ if $P(t) \neq 0,\ G(P(t)) = g,$
            (5.6)
              $t = t - M(t),\ g = g + 1.$
Step 3  3.1 While $C(s) < 0$, perform (5.1), (5.2), and (5.3).
        3.2 While $C(t) > 0$, perform (5.4), (5.5), and (5.6).
        3.3 Check whether there are cosines which are equal to 0.
            3.3.1 Set $a = 0$ and compute $a = a + O_M(k), k = 1, \ldots, g-1$. Set $z = N - a$.
            3.3.2 If $z \neq 0$, set $O_M(g) = z, S_m(g) = S_M(g) = 1, P_C(g) = 0, g = g + 1.$
```

We build an array L at levels where we would have to use Algorithm 4.6. We take

$$L(1) = \min\{C(1), C(N)\} - \delta,$$

(which can be a slightly negative level) and

$$L(j) = (S_M(j-1) + S_m(j))/2, \quad j = 2, \ldots, g.$$

Suppose that we applied Algorithm 4.6 to the matrix B at all these levels, so that we know the numbers $\nu(j),\ j = 1, \ldots, g$, of how many nonnegative sines are below each level, where g is the number of the groups of eigenvalues. We take $\nu(g+1) = N$. We then know how many positive sines are in between, i.e., in each group. We write the number of positive sines in an array P_S of length g.

At this point, we use the arrays to find the sign of the imaginary part of the eigenvalues. We take two counters $s = 1, 2, 3, \ldots,\ t = N, N-1, N-2, \ldots$. They start with $s = 1$ and $t = N$, so that we check first the cosines $C(1)$ and $C(N)$; if $G(1) \neq 1$ then $G(N) = 1$. We recall that $G(i)$ is the group to which the ith larger eigenvalue of A belongs and that we build the groups $1 \leq j \leq g$ such that the absolute value of the sines of the corresponding eigenvalues of U grows when j grows.

Suppose that in the process, we reach a cosine $C(k_0)$ which is the first in a new, not already checked, group. If its overall multiplicity $O_M(k_0) = 4$, then we found 4 eigenvalues $\pm C(k_0) \pm iS_A(k_0)$. In fact, one of them would be $C(k_0+1) - iS_A(k_0+1)$, in order to have minimal errors and also the cosines on the opposite side would be used. Suppose that the overall multiplicity is $O_M(G(k_0)) = 3$. Then, the multiplicity $M(k_0)$ is either 2 or 1. If it is two, then $C(k_0) \pm iS_A(k_0)$ are two of the eigenvalues of U and either the number of positive sines is $P_S(G(k_0)) = 2$, or $P_S(G(k_0)) = 1$, which means that the third cosine in this group $G(k_0)$ must be paired with a positive sine, or respectively, a negative sine. In Fig. 1.1, we have such a case with overall multiplicity 3 and 2 positive sines, i.e., $P_S(G(k_0)) = 2$. It is now obvious how we proceed if $M(k_0) = 1$, or if $O_M(G(k_0)) = 1$.

Table 5.1 shows all the 14 cases.

The only case that we cannot settle is when $O_M(G(k_0)) = 2$, $M(k_0) = 1$, and $P_S(G(k_0)) = 1$. Then, if we denote $\lambda = C(k_0) + iS_A(k_0)$, we cannot distinguish between the case when the eigenvalues of the matrix U are $\pm\lambda$ or $\pm\overline{\lambda}$. In this case, we build the eigenvector of A from (1.1), which corresponds to its eigenvalue $C(k_0)$ and, for the unitary Hessenberg matrix U itself, we build in linear time, since U is a quasiseparable matrix of order one, the row in U which corresponds to the maximal absolute value entry x_0 of the eigenvector, since most of the entries of such eigenvectors prove to be of too small absolute value. We then multiply the respective row of U with the eigenvector and we divide by x_0 to obtain λ or $\overline{\lambda}$. Note that this is more expensive than using once the Algorithm 4.6.

TABLE 5.1

In this table $\lambda = |cos\theta| + i|sin\theta|$, where at least one of the numbers $\pm|\cos\theta|$ is an eigenvalue of the matrix A from (1.1). The table shows all the cases that can appear for a (multiple) eigenvalue. All these eigenvalues form a group g with 1 up to 4 members (according to the overall multiplicity $O_M(g)$). We recall that $P_C(g)$ shows the multiplicity of the positive cosine, which can be 0 up to 2. In parenthesis are reported the data that do not need to be checked, since it follows from the other data. If this happens in the column of $P_S(g)$ (the number of positive sines) then this saves us the use of Algorithm 4.6 for the group g.

$O_M(g)$	$P_C(g)$	$P_S(g)$	Eigvals	$O_M(g)$	$P_C(g)$	$P_S(g)$	Eigvals
4	(2)	(2)	$\pm\lambda, \pm\overline{\lambda}$	2	2	(1)	$\lambda, \overline{\lambda}$
3	1	2	$\pm\lambda, -\overline{\lambda}$	2	1	2	$\lambda, -\overline{\lambda}$
3	2	1	$\pm\lambda, \overline{\lambda}$	2	1	0	$-\lambda, \overline{\lambda}$
3	1	1	$-\lambda, \pm\overline{\lambda}$	1	0	1	$-\overline{\lambda}$
3	2	2	$\lambda, \pm\overline{\lambda}$	1	0	0	$-\lambda$
2	1	1	$\pm\lambda$ or $\pm\overline{\lambda}$	1	1	1	λ
2	0	(1)	$-\lambda, -\overline{\lambda}$	1	1	0	$\overline{\lambda}$

In fact, for two consecutive levels, if both have overall multiplicity 1, which happens often when working with random matrices, we use Algorithm 4.6 only once. If the answer is 2, i.e., both sines have a plus sign or 0, i.e., both sines have a minus sign, we do not call it again for the intermediate level. Only if the answer is 1 (which happens statistically in about half of the cases), we call Algorithm 4.6 for the middle level also. In this way, if we try to skip one level, we call Algorithm 4.6 only $0.75N$ instead of N times. This is important, since Algorithm 4.6 is the one that makes the time of the procedure described in the present section, to amount to 1.7% of the overall time. When checked for U with almost all the eigenvalues of A (the cosines) of multiplicity two, this percentage is lower.

Note that if we skip two levels instead of one level, statistics assures us that we needed, in this case too, $0.75N$ runs of Algorithm 4.6. If we skip three levels instead, we need to call Algorithm 4.6 for them only if Algorithm 4.6 did not show at the fourth level that the number ν grew by 4, or it did not grow at all. In all the other obtained results, $(1, 2$ or $3)$ we would then call Algorithm 4.6 for the middle level first, then, if needed, we would call it once or twice more. This would save us $\frac{9}{32}$ instead of one fourth of the N calls to Algorithm 4.6, so that we would call it in total $0.71875N$ times, instead of $0.75N$ times. We decided to skip only one level, as described before.

In order to compute the complexity of the above algorithm, we will count as elementary arithmetical operations addition, subtraction, multiplication, division, conjugation, and comparisons performed on scalars. The overall complexity of the algorithm in this section is at most $c = 12.5N^2$, not including assignments and loop counters, mostly because of the use of Algorithm 4.6.

6. Finding the eigenvectors of U.

In this section describes how to compute the eigenvectors of the matrix U.

6.1. Computing the common eigenvectors by using the Hermitian matrices A or B.

After finding the cosine (real) part of the eigenvalues of U, one can compute by Algorithm 6.1 the corresponding common eigenvector of A, B, and U, if that cosine is not a double eigenvalue of A. Note that if we compute the eigenvector only for deciding whether for a complex number $\lambda = \cos\theta + i\sin\theta$ the eigenvalues are $\pm\lambda$ or else $\pm\overline{\lambda}$, then the corresponding eigenvalue of A (the cosine) and the corresponding eigenvalue of B (the sine) have always multiplicity one.

ALGORITHM 6.1. **Eigenvectors via the matrix A.** The present algorithm receives the Schur parameters $\rho(k),\ k = 1, \dots, N$, of the unitary upper Hessenberg matrix U, its size N, and an eigenvalue λ of the matrix A from (1.1), i.e., the real part of an eigenvalue of U. It computes the common eigenvector of the matrices U, A, and B which corresponds to λ. In order for the algorithm to be fast and in order not to recompute the same data again, it will also receive from the calling function the results of Step 0 from Algorithm 4.4.

Step 1 1.1 Set $D = d(1) - \lambda$, compute $u(1) = \mu(1)/D$, $\tilde{f} = \mu(1)u(1)$, and set

$$\tilde{g} = -\tilde{f}, \tilde{h} = \tilde{f}.$$

1.2 Compute (4.8)–(4.10), $s = 2, \dots, N-1$, and $u(s) = \mu(s)/D,\ v(s) = u(s)\overline{m}$.

Step 2 Find the eigenvector.

2.1 Set $x(N) = 1$, $b = 0.5$, and $c = -\rho(N)/2$.

2.2 For $s = N-1, \dots, 3, 2$, compute

$$x(s) = -u(s)b - v(s)c, \quad b = x(s)/2, \quad c = \mu(s)c - b\rho(s).$$

2.3 Compute $x(1) = u(1)(c - b)$.

Step 3 Find the norm of the eigenvector and normalize it if needed. Set $n = 0$. Compute $n = n + x(s)\overline{x(s)}$, $s = 2, \dots, N-1$, and after the end of the loop $n = \sqrt{n+1}$. Normalize the eigenvector dividing it by n.

If one uses Remark 4.5, then in Step 2.1 of Algorithm 6.1, one must take $b = 1$ and $c = -\rho(N)$, and one must not divide by 2 in Step 2.2.

If we count elementary scalar operations including conjugation and taking the real part, then the total complexity of Steps 1 and 2 using Remark 4.5 is $25N - 37$. We did not include assignments, or the loop counter. The complexity of normalizing the vector, if needed, is $4N - 1$.

After finding the cosine (real) part of the eigenvalues of U, if that cosine is a double eigenvalue, one can compute the corresponding eigenvector by Algorithm 6.2, if the sine is not a double eigenvalue.

ALGORITHM 6.2. **Eigenvectors via the matrix B.** The present algorithm receives the Schur parameters $\rho(k),\ k = 1, \dots, N$, of the unitary upper Hessenberg matrix U, its size N, and an eigenvalue λ of the matrix B from (1.1), i.e., the imaginary part of an eigenvalue of U. It computes the common eigenvector of the matrices U, A, and B which corresponds to λ. In order for the algorithm to be fast and in order not to recompute the same data again, it will also receive from the calling function the results of Step 0 as Algorithm 4.6.

```
Step 1  1.1  Perform Step 1.1 of Algorithm 6.1.
        1.2  Compute (4.12), (4.13), (4.10), and $u(s) = \mu(s)/D$, $v(s) = u(s)\overline{m}$, for
             $s = 2, \dots, N-1$.
Step 2  Find the eigenvector.
        2.1  Set $x(N) = 1$, $b = i/2$, and $c = b\rho(N)$.
        2.2  For $s = N-1, \dots, 3, 2$, compute
                 $x(s) = -u(s)b - v(s)c, \quad b = ix(s)/2, \quad c = \mu(s)c + b\rho(s).$
        2.3  Compute $x(1) = u(1)(c - b)$.
Step 3  Perform Step 3 of Algorithm 6.1.
```

In order to use Remark 4.5, one must not divide by 2 in Steps 2.1 and 2.2.

If all the four numbers $\pm c \pm is$ are eigenvalues, then we cannot properly find the corresponding eigenvectors by using A or B, since the algorithm works only on Hermitian matrices and only for simple eigenvalues.

We suggest to use Algorithm 6.1 when the absolute value of the real part of an eigenvalue of U is larger than the absolute value of its imaginary part, otherwise it is better to resort to Algorithm 6.2. In this way, the algorithm will always work with an eigenvalue λ which satisfies $\frac{\sqrt{2}}{2} \leq |\lambda| \leq 1$, so that it is far from zero.

Since Algorithm 6.1 (or, equivalently, Algorithm 6.2) needs to allocate three arrays of size N, it is cheaper to compute all the needed eigenvectors together, using the same arrays u, v for them all.

If all the eigenvectors are needed, we would perform the *while* loop from Step 2.2 of Algorithm 4.7 only when the error is smaller than twice the machine precision. Hence, we call Algorithm 4.4 one time less and then, after finishing the loop we would call one last time a slightly modified version of Algorithm 6.1, so that it also counts how many times D of (4.9) is negative. In this way we would not lose anything in terms of precision and we would exploit the common part of Algorithm 6.1 and Algorithm 4.4. By doing so, we would find all the eigenvectors with half the operations, more precisely $15N^2$ operations less.

If we compute the eigenvectors of U from A or B instead of U itself, one may find without many operations to which eigenvalues of U they correspond. Indeed, the eigenvalues of U, as they come from Section 5, are sorted in increasing order of the absolute value of their sine, or, which is the same, in decreasing order of the absolute value of their cosines, while the eigenvalues of A, i.e., the cosines, come sorted in increasing order.

6.2. Computing the common eigenvectors by using the Hessenberg matrix. The eigenvectors of orthogonal matrices have been computed in a stable way in [1]. We will also propose an algorithm for the computation of the eigenvectors of a unitary upper Hessenberg matrix U.

Note that the eigenvalues of a diagonalizable upper Hessenberg unreduced matrix H have multiplicity one. From Algorithm 6.6 it follows that, for each of the N eigenvalues of H, there is only one normalized eigenvector, up to its sign.

LEMMA 6.3. *Let H be an unreduced upper Hessenberg diagonalizable $N \times N$ matrix with quasiseparable generators of order one*

$$\begin{aligned} p(i) &= H(i, i-1), & i &= 2, \dots, N, \\ q(j) &= 1, & j &= 1, \dots, N-1, \\ a(k) &= 0, & k &= 2, \dots, N-1, \end{aligned} \tag{6.1}$$

$$\{g(i)\}_{i=1}^{N-1}, \quad \{h(j)\}_{j=2}^{N}, \quad \{b(k)\}_{k=2}^{N-1}, \tag{6.2}$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

and diagonal elements $\{d(i)\}_{i=1}^N$*, which are (possibly complex) numbers.*

Then, if

$$g(k+1) \neq 0, \quad k = 2, \dots, N-2, \tag{6.3}$$

and λ *is an eigenvalue of* H*, one can compute a corresponding eigenvector* x *in the following way. Set*

$$x_N = 1, \qquad x_{N-1} = p^{-1}(N)(\lambda - d(N)), \tag{6.4}$$

$$x_{N-2} = p^{-1}(N-1)p^{-1}(N)((\lambda - d(N))(\lambda - d(N-1)) - p(N)g(N-1)h(N)), \tag{6.5}$$

and for $k = N-2, N-3, \dots, 2$*, compute*

$$\begin{aligned} x_{k-1} = p^{-1}(k)(g(k)b(k+1)g^{-1}(k+1)(p(k+1)x_k \\ - (\lambda - d(k+1))x_{k+1}) - g(k)h(k+1)x_{k+1} + (\lambda - d(k))x_k). \end{aligned} \tag{6.6}$$

Proof. The matrix H is unreduced, which means that its lower subdiagonal entries $p(k)$ are different from 0, for $k = 2, \dots, N$.

The last entry of the eigenvector x is not equal to 0, since otherwise the multiplication of the last row of H with x would be $p(N)x_{N-1} + d(N) \cdot 0 = \lambda \cdot 0$, which implies that $x_{N-1} = 0$ too. with a similar argument, we would obtain $x = 0$, which cannot be an eigenvector.

Thus, we can take $x_N = 1$ and, if needed, x will be normalized after its computation. Then we multiply $H(N, 1:N)$ with x and it follows that $p(N)x_{N-1} + d(N) \cdot 1 = \lambda \cdot 1$. Since H is unreduced, the coefficient of x_{N-1} is not 0, thus one can divide in (6.4) by it and this completes its proof.

For the proof of (6.5), we multiply the row $N-1$ of H by x, obtaining

$$p(N-1)x_{N-2} + d(N-1)x_{N-1} + g(N-1)h(N) = \lambda x_{N-1},$$

and then we use the fact that x_{N-1} is known from (6.4) and that the coefficient of x_{N-2} is invertible for an unreduced matrix.

In order to prove (6.6) for $k \in \{N-2, N-3, \dots, 2\}$, we perform the multiplication $H(k+1, 1:N)x$ and then

$$\begin{aligned} 0_{1\times(k-1)}x(1:k-1) + p(k+1)x_k d(k+1)x_{k+1} + \\ + H(k+1, k+2:N)x(k+2:N) = \lambda x_{k+1}. \end{aligned} \tag{6.7}$$

Note that for a quasiseparable matrix we have, keeping into account (6.3),

$$H(k, k+2:N) = g(k)b(k+1)g^{-1}(k+1)H(k+1, k+2:N),$$

so that

$$\begin{aligned} 0_{1\times(k-2)}x(1:k-2) + p(k)x_{k-1} + d(k)x_k + g(k)h(k+1)x_{k+1} + \\ g(k)b(k+1)g^{-1}(k+1)H(k+1, k+2:N)x(k+2:N) = \lambda x_k. \end{aligned} \tag{6.8}$$

By computing $H(k+1, k+2:N)x(k+2:N)$ from (6.7) and replacing it in (6.8) one obtains

$$\begin{aligned} p(k)x_{k-1} + d(k)x_k + g(k)h(k+1)x_{k+1} + \\ g(k)b(k+1)g^{-1}(k+1)(\lambda x_{k+1} - p(k+1)x_k - d(k+1)x_{k+1}) = \lambda x_k, \end{aligned}$$

from which (6.6) follows. □

If we compute in the calling function the inverses $p^{-1}(k)$, for $k = 2, \dots, N$, and the products $g(k)b(k+1)g^{-1}(k+1)$ and $g(k)h(k+1)$, for $k = 2, \dots, N-2$, which are reused by all the eigenvectors, it follows that the computation of an entry of the vector x by using (6.6) costs 11 arithmetical operations.

Note that unitary matrices U are normal, so they are diagonalizable, since $D = V^*UV$, where D is a diagonal matrix containing the N eigenvalues of U on the diagonal and V is a unitary matrix, the columns of which are the corresponding normalized eigenvectors.

LEMMA 6.4. *Let H be an unreduced upper Hessenberg diagonalizable $N \times N$ matrix with quasiseparable generators* (6.1), (6.2) *of order one and diagonal elements $\{d(i)\}_{i=1}^N$ and suppose that* (6.3) *does not hold. Let λ be an eigenvalue of H. Then an eigenvector of H corresponding to λ can be obtained in the following way.*

a) If $j \in \{1, 2, \dots, N-2\}$ and for an index $k \in \{j, j+1, \dots, N-2\}$ one has

$$g(k+1) \neq 0, \quad g(k) = g(k-1) = \ldots = g(k-j+1) = 0, \tag{6.9}$$

then

$$x_{k-s} = p^{-1}(k-s+1)(\lambda - d(k-s+1))x_{k-s+1}, \quad s = 1, 2, \dots, j, \tag{6.10}$$

and, moreover, if $k > j+1$ and $g(k-j) \neq 0$, then x_{k-j-1} can be computed as follows. Set

$$y = g^{-1}(k+1)((\lambda - d(k+1))x_{k+1} - p(k+1)x_k) \tag{6.11}$$

and, for $s = 1, 2, \dots, j$, set

$$y = h(k+2-s)x_{k+2-s} + b(k+2-s)y, \tag{6.12}$$

then

$$\begin{aligned} x_{k-j-1} = p^{-1}(k-j)((\lambda - d(k-j))x_{k-j} \\ - g(k-j)(h(k-j+1)x_{k-j+1} + b(k-j+1)y)). \end{aligned} \tag{6.13}$$

b) If $j \in \{1, 2, \dots, N-2\}$ and

$$g(N-1) = g(N-2) = \ldots = g(N-j) = 0, \tag{6.14}$$

then we set $x_N = 1$ and

$$x_{N-s} = p^{-1}(N-s+1)(\lambda - d(N-s+1))x_{N-s+1}, \quad s = 1, 2, \dots, j+1, \tag{6.15}$$

and moreover, if $j \neq N-2$ and $g(N-j-1) \neq 0$, x_{N-j-2} can be computed as follows. Set $y = h(N)$ and

$$y = h(N-s)x_{N-s} + b(N-s)y, \quad s = 1, 2, \dots, j. \tag{6.16}$$

Then

$$x_{N-j-2} = p^{-1}(N-j-1)((\lambda - d(N-j-1))x_{N-j-1} - g(N-j-1)y). \tag{6.17}$$

Proof.

a) If $g(t) = 0$ for an index $t \in \{k, k-1, \dots, k-j+1\}$ as in (6.9), then it follows that $H(t, t+1 : N)x(t+1 : N) = 0$ in (6.8), thus this formula becomes

$$p(t)x_{t-1} + d(t)x_t = \lambda x_t,$$

hence (6.10) follows.
The number in the outer brackets from (6.11) is in fact

$$H(k+1, k+2 : N)x(k+2 : N),$$

when computed from the equality $H(k+1, 1 : N)x = \lambda x_{k+1}$ from (6.7), while the iterations which follow build step by step

$$H(k-j, k-j-1 : k+1)x(k-j-1 : k+1) + H(k-j, k+2 : N)x(k+2 : N),$$

to compute x_{k-j-1}.

b) The proof of b) is similar. ∎

If $g(t) = 0$ for $j \geq 1$ consecutive indices, one obtains, by Lemma 6.4, $j+1$ entries of the eigenvector x with the cost of $6j + 10$ arithmetical operations in total. Indeed, if we compute the inverses in the calling function, then (6.10) and (6.12) need $3j$ operations each, while (6.11) and (6.13) need 6 and 4 operations, respectively.

ALGORITHM 6.5. Let H be an unreduced upper Hessenberg diagonalizable $N \times N$ matrix with scalar entries, given in quasiseparable form via the scalar lower quasiseparable generators (6.1), the upper quasiseparable generators (6.2) (of any order) and diagonal elements $\{d(i)\}_{i=1}^{N}$.

Let λ be an eigenvalue of H. Then, an eigenvector x of H corresponding to λ can be obtained in the following way.

Step 1 Set $x_N = 1, x_{N-1} = p^{-1}(N)(\lambda - d(N)), f = h(N)$.
Step 2 For $k = 1, \dots, N-3$ compute

$$x_{N-k-1} = p^{-1}(N-k)((\lambda - d(N-k))x_{N-k} - g(N-k)f),$$
$$f = h(N-k)x_{N-k} + b(N-k)f.$$

Step 3 Compute $x_1 = p^{-1}(2)((\lambda - d(2))x_2 - g(2)f)$.

Proof. The proof is similar to the proofs of the previous lemmas. ∎

If the order of the upper quasiseparable generators is $r > 1$, then Algorithm 6.5 uses less than $(2r^2 + 2r + 5)N$ arithmetical operations between scalars, if the needed inverses are computed in the calling function. If $r = 1$, then $8N - 20$ operations are performed.

For unitary upper Hessenberg matrices, we will obtain the following algorithm with lower complexity.

ALGORITHM 6.6. Let U be an unreduced upper Hessenberg unitary $N \times N$ matrix (2.1) with lower quasiseparable generators (6.1), where $p(k) = U(k, k-1) = \mu(k-1)$, upper quasiseparable generators (6.2) and diagonal entries which, according to (3.5)–(3.8), are

$$g(k) = \mu(k)\rho^*(k-1), \;\; g(1) = -\mu(1), \;\; b(k) = \mu(k), \qquad k = 2, \dots, N-1,$$
$$d(1) = \rho(1), \;\; d(k) = -\rho(k)\rho^*(k-1), \;\; h(k) = -\rho(k), \quad k = 2, \dots, N.$$

Then, if λ is an eigenvalue of U, one can compute, up to its sign, (only) one corresponding normalized eigenvector x in the following way.

$$x_N = 1, \quad x_{N-1} = \frac{\lambda - d(N)}{\mu_{N-1}}, \tag{6.18}$$

$$x_{N-2} = \frac{\lambda^2 - (d(N) + d(N-1))\lambda + \rho^*(N-2)\rho(N)}{\mu_{N-2}\mu_{N-1}}, \tag{6.19}$$

and, for $k = N-2, N-3, \dots, 2$, if

$$\mu(k) \neq 1, \tag{6.20}$$

then

$$x_{k-1} = \frac{1}{\mu(k-1)} \left(\left(\lambda + \frac{\rho^*(k-1)}{\rho^*(k)} \right) x_k - \frac{\rho^*(k-1)}{\rho^*(k)} \mu(k) \lambda x_{k+1} \right). \tag{6.21}$$

Otherwise, if

$$\mu(N-2) = \mu(N-3) = \dots = \mu(N-j-1) = 1, \tag{6.22}$$

then

$$x_{N-s} = \lambda x_{N-s+1} = \lambda^{s-1} \frac{\lambda + \rho(N)\rho^*(N-1)}{\mu(N-1)}, \quad s = 1, 2, \dots, j+1, \tag{6.23}$$

and, moreover, if $\mu(N-j-2) \neq 1$, then

$$x_{N-j-2} = \frac{\lambda^{j+1}(\lambda + \rho(N)\rho^*(N-1)) + \rho(N)\rho^*(N-j-2)}{\mu(N-j-2)\mu(N-1)}. \tag{6.24}$$

Otherwise, it follows that there are two natural numbers k, j such that $1 \leq j < k \leq N-2$ for which

$$\mu(k) \neq 1, \quad \mu(k-1) = \mu(k-2) = \cdots = \mu(k-j) = 1. \tag{6.25}$$

In this case, it follows that

$$x_{k-s} = \lambda^s x_k, \quad s = 1, 2, \dots, j, \tag{6.26}$$

and if $k \neq j+1$ and $\mu(k-j-1) \neq 1$, then

$$x_{k-j-1} = \left(\frac{\lambda^{j+1}}{\mu(k-j-1)} + \frac{\rho^*(k-j-1)}{\rho^*(k)} \right) x_k - \frac{\lambda\mu(k)\rho^*(k-j-1)}{\rho^*(k)} x_{k+1}. \tag{6.27}$$

Perform Step 3 of Algorithm 6.1 to normalize the eigenvector.

Proof. The equalities from (6.18) follow from (6.4). In order to prove (6.19) by using (6.5), we have to show, with the above quasiseparable generators, that

$$d(N)d(N-1) - p(N)g(N-1)h(N) = \rho^*(N-2)\rho(N). \tag{6.28}$$

Indeed, the left hand side is equal to

$$\rho(N)\rho^*(N-1)\rho(N-1)\rho^*(N-2) + \mu^2(N-1)\rho(N)\rho^*(N-2),$$

and then (6.28) follows, since $|\rho(N-1)|^2 + \mu^2(N-1) = 1$.

The constraint (6.20) is the same with (6.3), when applied to the matrix U, which is also unitary and can therefore be written as (2.1). Indeed, $g(k+1) \neq 0$ is equivalent to $\rho^*(k) \neq 0$, since the other factor of $g(k+1)$, which is $\mu(k+1)$, cannot be 0 for an unreduced matrix. This also means that $\rho(k) \neq 0$, which is equivalent to (6.20).

Note that if $\mu(N-2) = 1$, then $\rho^*(N-2) = 0$, thus $d(N-1) = 0$ and (6.19) becomes $x_{N-2} = \lambda x_{N-1}$, which is the same as (6.15) would give. Thus, both (6.18) and (6.19) remain the same, no matter $\mu(N-2)$.

With the above quasiseparable generators, (6.6) becomes

$$\begin{aligned}x_{k-1} =& \frac{1}{\mu(k-1)}\left(\frac{\mu(k)\rho^*(k-1)\mu(k+1)}{\mu(k+1)\rho^*(k)}(\mu(k)x_k-(\lambda+\rho(k+1)\rho^*(k))x_{k+1}\right.\\ &\left.+\,\mu(k)\rho^*(k-1)\rho(k+1)x_{k+1}+(\lambda+\rho(k)\rho^*(k-1))x_k\right)\end{aligned} \tag{6.29}$$

The coefficient of x_k in the right-hand side of (6.29) is

$$\frac{1}{\mu(k-1)}\left(\frac{\mu(k)\rho^*(k-1)}{\rho^*(k)}\mu(k)+\lambda+\rho(k)\rho^*(k-1)\right),$$

and if we compute the common denominator and we use (2.2) and (2.3), it follows that the coefficient of x_k in (6.29) is

$$\frac{1}{\mu(k-1)}\left(\lambda+\frac{\rho^*(k-1)}{\rho^*(k)}\right). \tag{6.30}$$

The coefficient of x_{k+1} in the right-hand side of (6.29) is

$$\frac{1}{\mu(k-1)}\left(\frac{\mu(k)\rho^*(k-1)}{\rho^*(k)}(-\lambda-\rho(k+1)\rho^*(k))+\mu(k)\rho^*(k-1)\rho(k+1)\right)$$

and, after a simplification, it follows that the coefficient of x_{k+1} in (6.29) is

$$-\frac{\mu(k)}{\mu(k-1)}\cdot\frac{\rho^*(k-1)}{\rho^*(k)}\lambda. \tag{6.31}$$

From (6.30) and (6.31) it follows that (6.6), which became for unitary matrices (6.29), simplifies to (6.21).

Note that (6.14) means that $\rho(N-2)=\rho(N-3)=\cdots=\rho(N-j-1)$. thus (6.22).

In order to prove (6.23) note that, if $\mu(k)=1$, then all the other entries of row $k+1$ vanish, since the rows of unitary matrices are normalized vectors, thus

$$\lambda x_{k+1}=U(k+1,1:N)x=\mu(k)x_k=x_k.$$

To obtain (6.24), note that $y=h(N)=-\rho(N)$ is then changed by (6.16), for $s=1$ it becomes $y=-\rho(N-1)x_{N-1}-\mu(N-1)\rho(N)$ which, by (6.23) becomes

$$\begin{aligned}y&=\frac{\rho(N-1)\lambda-\rho(N-1)\rho^*(N-1)\rho(N)-\lambda\rho(N-1)-\mu^2(N-1)\rho(N)}{\mu(N-1)}\\ &=-\frac{\rho(N)}{\mu(N-1)},\end{aligned}$$

while for $s=2,3,\ldots,j$, it does not change, since $y=-\rho(N-s)+\rho(N-s)y=0+1\cdot y$.

Then, from (6.17), x_{N-j-2} is equal to

$$\frac{(\lambda+\rho(N-j-1)\rho^*(N-j-2))x_{N-j-1}+\mu(N-j-1)\rho^*(N-j-2)\frac{\rho(N)}{\mu(N-1)}}{\mu(N-j-2)},$$

which, using (6.23), becomes

$$\begin{aligned}x_{N-j-2}&=((\lambda+\rho(N-j-1)\rho^*(N-j-2))\lambda^j(\lambda+\rho(N)\rho^*(N-1))\\ &\quad+\mu(N-j-1)\rho^*(N-j-2)\rho(N))\frac{1}{\mu(N-j-2)\mu(N-1)}\\ &=\frac{\lambda^j(\lambda+0)(\lambda+\rho(N)\rho^*(N-1))+1\cdot\rho^*(N-j-2)\rho(N)}{\mu(N-j-2)\mu(N-1)},\end{aligned}$$

and thus (6.24) is proved.

In the last case, from (6.10) we obtain

$$x_{k-s} = \frac{(\lambda - \rho(k-s+1)\rho^*(k-s))x_{k-s+1}}{\mu(k-s)}, \quad s = 1, 2, \dots, j,$$

and thus (6.25) implies that $x_{k-s} = \frac{(\lambda-0)x_{k-s+1}}{1}$, $s = 1, 2, \dots, j$, which proves (6.26).

It remains to prove (6.27). For the matrix U, (6.11) becomes

$$\begin{aligned} y &= \frac{(\lambda + \rho(k+1)\rho^*(k))x_{k+1} - \mu(k)x_k}{\mu(k+1)\rho^*(k)} \\ &= \frac{1}{\mu(k+1)}\left(\frac{\lambda}{\rho^*(k)} + \rho(k+1))x_{k+1} - \frac{\mu(k)}{\rho^*(k)}x_k\right). \end{aligned}$$

If we use (6.12) for $s = 1$, then y becomes $y = -\rho(k+1)x_{k+1} + \mu(k+1)y$, thus

$$y = \frac{\lambda x_{k+1} - \mu(k)x_k}{\rho^*(k)}, \tag{6.32}$$

and if $j > 1$, after (6.12) for $s = 2$, it becomes $y = -\rho(k)x_k + \mu(k)y$, therefore

$$y = \frac{-\rho(k)\rho^*(k)x_k + \lambda\mu(k)x_{k+1} - \mu^2(k)x_k}{\rho^*(k)} = \frac{\lambda\mu(k)x_{k+1} - x_k}{\rho^*(k)}.$$

For $s \geq 3$, we have $\rho(k+2-s) = 0, \mu(k+2-s) = 1$, hence y does not change. Thus, from (6.13) and from the last two equalities of (6.25),

$$x_{k-j-1} = \frac{1}{\mu(k-j-1)}\left(\lambda x_{k-j} - \rho^*(k-j-1)\mu(k-j-1)\frac{\lambda\mu(k)x_{k+1} - x_k}{\rho^*(k)}\right),$$

and then, using (6.26), one proves (6.27).

If $j = 1$, then the last y is given by (6.32) and it also follows that

$$\mu(k) \neq 1, \mu(k-1) = 1, \mu(k-2) \neq 1, \quad \rho(k-1) = 0, \quad x_{k-1} = \lambda x_k,$$

and, therefore,

$$x_{k-2} = \frac{\lambda^2 x_k\rho^*(k) + \rho^*(k-2)(\lambda\mu(k)x_{k+1} - (\rho^*(k)\rho(k) + \mu^2(k))x_k)}{\mu(k-2)\rho^*(k)}.$$

Since $|\rho(k)|^2 + \mu^2(k) = 1$, (6.27) is true for $j = 1$ too. □

By using the fact that $\frac{\rho^*(k-1)}{\rho^*(k)} = \frac{-d(k)}{\sqrt{1-\mu^2(k)}}$,the division by complex numbers can be avoided,

The complexity for the computation of an entry of x by (6.21) is 6 arithmetical operations, if we compute in the calling function for $k = 2, \dots, N$ the numbers $\frac{1}{\mu(k-1)}$, $\frac{\rho^*(k-1)}{\rho^*(k)}$, and $\frac{\rho^*(k-1)}{\rho^*(k)}\mu(k)$, for $k = 2, \dots, N-1$, -which are common to all the eigenvectors. For the whole eigenvector one needs $6N - 10$ operations with scalars. If $\mu(k) = 1$ for at least an index k, then the cost is even lower. The complexity of normalizing the vector, if needed, is $4N - 1$.

7. Numerical experiments.

All the numerical experiments have been performed on a machine with an $i7-5820$ microprocessor, with 31.9 gigabytes installed memory (RAM) at $3.30GHz$ and $4GB$ in the video card GTX, which is exploited by Matlab as well. The operating system is Windows 10, 64bits, the least positive number which is used by the machine is $2.2251E-308$, given by the Matlab command `realmin`, and the machine precision is $2.2204E-16$, given by the Matlab command `eps`. The Matlab version is R2016B.

We build generators for random unitary Hessenberg matrices, which in practice rarely have multiple eigenvalues, since this is not even possible if the matrix is unreduced and the latter takes place whenever none of the Schur parameters $\rho_k,\ k=1,\dots,N-1$, is on the unit circle.

We build the (sub)unit complex numbers $\rho_k,\ k=0,\dots,N$, such that $\rho_0=-1$ and ρ_N is on the unit circle, by first computing 2π and then a point on the unit circle for an angle between 0 and 2π. To this end, we use two Matlab functions, $\mathtt{exp}(2\pi\cdot\mathtt{rand})$, where `rand` command gives a pseudo-random number between 0 and 1. For $0<k<N$ we then multiply the obtained number by a subunit absolute value, given again by the `rand` command and we compute the subdiagonal real numbers

$$\mu_k=\sqrt{1-\rho_k\overline{\rho_k}},\quad k=1,\dots,N-1.$$

In this way any possible unitary upper Hessenberg unreduced matrix can be obtained.

After we find the quasiseparable generators of the matrix U, we build the whole unitary upper Hessenberg matrix U out of its generators. Then we use the Matlab function $\mathtt{eig}(U)$ to find the Matlab eigenvalues, which we consider to be the true ones.

We checked matrices of sizes that are a power of 2, from 4 to 8192. For each of the sizes up to 2048 we checked 20 matrices.

The formula used for the errors is the average error over the N eigenvalues of each of the $n=20$ considered matrices of each size $N\times N$, N being a power of 2 starting with 2^2 and finishing with 2^{11}. For $N=4096$ and $N=8192$ we take $n=1$.

If we denote by $\lambda_T^{(j)}(s)$ the true known sth eigenvalue of one of the $j=1,\dots,20$ considered matrices and by $\lambda^{(j)}(k)$ the eigenvalue obtained by bisection, then the formula is

$$\frac{1}{20}\sum_{j=1}^{20}\left(\frac{1}{N}\sum_{k=1}^{N}\min_{s\in S_j}|\lambda^{(j)}(k)-\lambda_T^{(j)}(s)|\right). \tag{7.1}$$

Here, the set of indices S_j contains those $s\in\{1,\dots,N\}$ that have not been previously used by another $\lambda^{(j)}(k)$. To this end, we use a two dimensional array

$$F(j,k),\quad j=1,...,20,\ k=1,...,N,$$

of boolean flags, which is initialized with zero entries. Once a certain $\lambda_T^{(j)}(k)$ is assigned as explained before, we set $F(j,k)=1$.

7.1. Comparison with implicit QR for unitary Hessenberg matrices.

In [12], Gragg establishes an implicit QR algorithm for Hessenberg matrices. He writes that he does not claim that it is the most stable or it has the least number of operations. We used it with the unit shift introduced in [19] and we optimized it for one QR step at its maximum, obtaining a version with no array allocation and without indices, except those of the Schur parameters of the matrix, which are the input. The eigenvalues are the output.

ALGORITHM 7.1. **Optimized implicit QR.** Let ρ_k, $k=0,\dots,N$, with $\rho_0=1$ and $|\rho(N)|=1$, and $\mu(k)$, $k=1,\dots,N-1$, be the parameters of a unitary upper Hessenberg matrix.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Step 1 Perform Steps 2 to 5 as long as $N > 1$.

Step 2 While the last subdiagonal entry $\mu(N-1)$ is larger than the machine precision, perform the following two steps.

Step 3 Compute the QR shift λ.

If $N = 2$ or $\rho(N-2) = 0$, set $f = \overline{\rho(N)}$, otherwise $f = \frac{\overline{\rho(N-2)}}{\rho(N-2)}$.
Set

$$d = \overline{\rho(N-1)}\rho(N), \quad b = -(f\rho(N-1)+d)/2, \quad s = \sqrt{b^2 - f\rho(N)}.$$

If $|b-s+d| < |b+s+d|$, set $\lambda = b - s$, otherwise, $\lambda = b + s$.

Step 4 Obtain the new parameters of a unitary upper Hessenberg matrix, after one implicit QR step with shift λ.

4.1

$$m = \lambda + \rho(1), \quad g = \sqrt{\mu^2(1) + m\overline{m}}, \quad a = m/g,$$
$$t = (1 + \overline{\rho(1)}\lambda)/g, \quad c = \mu(1)/g, \quad \rho(1) = a\overline{t} - c^2\rho(2)\overline{\lambda}.$$

4.2 For $k = 2, \ldots, N-1$, compute

$$m = \lambda a + \rho(k)t, \quad g = \sqrt{\mu^2(k) + m\overline{m}}, \quad t = (t + \overline{\rho(k)}\lambda a)/g,$$
$$a = m/g, \quad \mu(k-1) = cg, \quad c = \mu(k)/g, \quad \rho(k) = a\overline{t} - c^2\rho(k+1)\overline{\lambda}.$$

4.3 Compute $\mu(N-1) = c|\lambda a + \rho(N)t|$.

Step 5 Take the last diagonal entry $-\overline{\rho(N-1)}\rho(N)$ as an eigenvalue and set $N = N-1$.

Step 6 Take $-\rho(1)$ as the last found eigenvalue.

The complexity of Step 4.2 in Algorithm 7.1 is $21N - 42$ arithmetical operations, assignment not included. The shift of Wang and Gragg (obtained in Step 3) makes the algorithm converge very quickly.

We checked for bisection and implicit QR the time needed to solve a matrix, the worst error among the first obtained $N-2$ eigenvalues of an $N \times N$ matrix and the average error over those $N-2$ eigenvalues. For implicit QR we did not take into consideration the errors of the two last found eigenvalues, since they are obtained in a slightly modified way than the others and since their cumulated errors might be higher.

Implicit QR works much faster than the bisection algorithm presented here and it would for sure be faster, even without any optimization. However, surprisingly, the bisection, which is known to be in general faster, but less accurate than implicit QR, is much more accurate; see Fig. 7.1. Among all the matrices of size up to 2048 that we checked, the largest error for the bisection method has been $4E-13$, while for implicit QR $5E-07$ and errors of $5E-08$ are very common. In general, for sizes 16 to 32 bisection is only 5 times more accurate, while for sizes from 64 up to 256 it gives 3 correct decimal digits more than QR for the average error. For larger sizes it is 10^4 times more correct and 10^5 times in some experiments with matrices up to size 2048. For one matrix of size 8192, QR gave an error of $3E-03$ as its worst error, while the worst error of bisection has been better by 9 decimal digits. Moreover, the error of QR does not increase smoothly in proportion with the size of the matrix, since there are spikes in its graph as it can be seen in Fig. 7.1.

7.2. Comparison with divide and conquer for unitary Hessenberg matrices. As we show on the left part of Fig. 7.1, the divide and conquer eigenvalue algorithm of Gragg and Reichel [13], (see also [6, Ch. 28] and [14]) is faster than our bisection method but is slower than QR. On the right part of Fig. 7.1, where we plot the average error and the worst error,

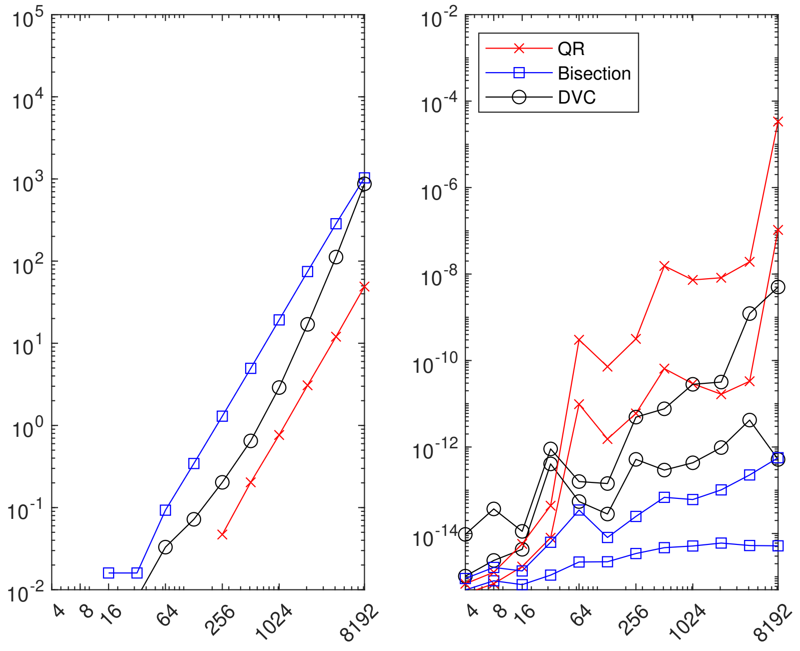

FIG. 7.1. *On the left, timing of QR and DVC versus the new algorithm. The data that we have not plotted is equal to 0. On the right, average and maximum error versus eigenvalues computed by Matlab. For each type of line (and colour), the upper values correspond to the worst eigenvalues and the lower values correspond to the average error over all the eigenvalues. The errors are computed by using formula* (7.1).

the divide and conquer errors are larger than our algorithm, but much smaller than QR errors. However, like for QR, the worst eigenvalue error for divide and conquer does not increase monotonically enough upon the size of the matrix and it was equal to almost 10^{-8} for one matrix of size 8192.

7.3. Double eigenvalues of the matrix A. If we take the Schur parameters to be real numbers, we obtain orthogonal matrices and the non real eigenvalues of U come in complex conjugate pairs. This means that most, or all, of the eigenvalues of A (i.e., the cosines) are double eigenvalues. The algorithm correctly finds all the eigenvalues of the orthogonal matrix U, as long as their cosines had been identified as double roots, since in this case the only possibility is that they are equal to $c \pm i\sqrt{1-c^2}$, where c is the cosine. The problem is that, when we restrain the criterion for the double eigenvalues to be at a very small distance in between, the bisection algorithm for A does not always recognize them as such.

We checked in detail this problem for matrices of size 2048×2048, on 20 matrices. We chose a small distance of

$$d = 5E - 15,$$

which is about 22 times our machine precision. First of all, we build for each matrix A the orthogonal matrix U out of its quasiseparable generators, we find its Matlab eigenvalues on the unit circle, we consider their real parts and we sort them in ascending order as well. We do not match each of our eigenvalues to the closest Matlab eigenvalue, instead we use the fact that our bisection finds the eigenvalues in ascending order and so we check if the absolute values of the differences between the kth eigenvalue in both sets of eigenvalues is smaller than d.

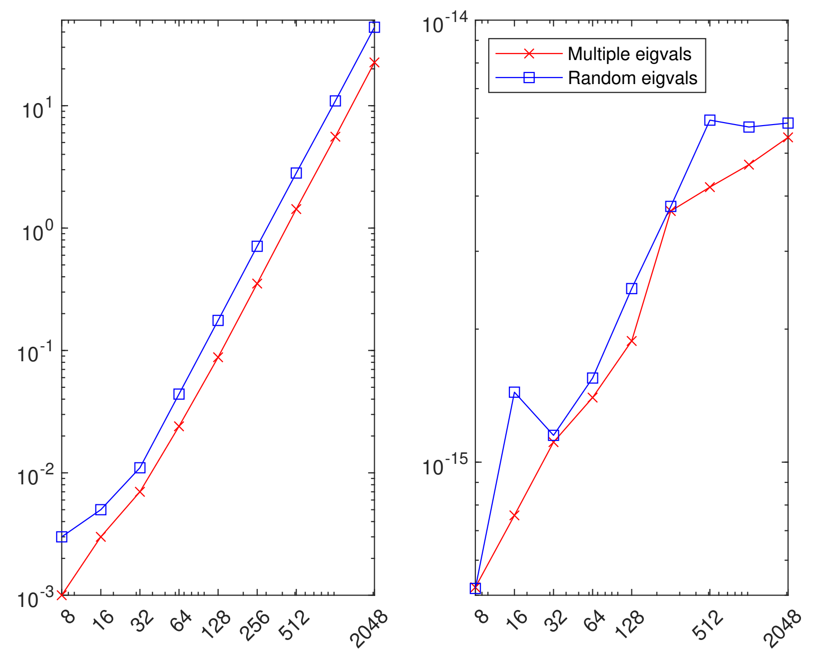

FIG. 7.2. *On the left, time for finding sign of sines for overall multiplicity* 3 *eigenvalues, vs. the larger time for random eigenvalues. On the right, the average error versus the eigenvalues computed by Matlab. The multiple eigenvalues which are known in advance (see Section 7.4) have overall multiplicity 3 (see Definition 5.1 in Section 5.2) and the Schur parameters have been obtained by using Ammar, Gragg, and Reichel [2].*

Although our way to measure error introduces small errors itself, 99.1% of our eigenvalues (about 2029 eigenvalues in average for each matrix) differ from the Matlab eigenvalues by less than d. However, there are a few bad eigenvalues: the worst out of 40960 eigenvalues has been $3E-09$ away from its corresponding Matlab eigenvalue. The best of the worst eigenvalues for the 20 matrices is at a distance of about $10d$.

The other 19 eigenvalues are not lost, but are treated further as if they were simple eigenvalues.

7.4. Eigenvalues known in advance. In [2] the Schur parameters of a unitary upper Hessenberg unreduced matrix are built from eigenvalues which are given a priori. We implemented this method and we checked our algorithm and also the Matlab `eig` function results, when compared to the true eigenvalues, which are known in advance. Since the Schur parameters are built in $O(N^2)$ operations and in order for comparing with Matlab results one must also build the whole matrix U, it follows that the bisection eigenvalues, as well as Matlab eigenvalues, contain larger errors than when comparing our results to those of Matlab. What is important is that there exists a strong positive correlation between our algorithm and Matlab results in all the eigenvalues; see Fig. 7.2.

The method of Ammar, Gragg, and Reichel from [2] has been of great help while checking the case when more than two among the 4 numbers $\pm c \pm si$ are eigenvalues. See also Reichel, Ammar, and Gragg [16, pp. 380–383].

7.5. Experiments with eigenvectors. For all the checked matrices U, we built also all the eigenvectors, which we found for the matrix A or B by using Algorithm 6.1 or, respectively, Algorithm 6.2. We choose between A and B, depending on whether the absolute value of the real part of the eigenvalue of U, or, respectively the absolute value of the imaginary part, was

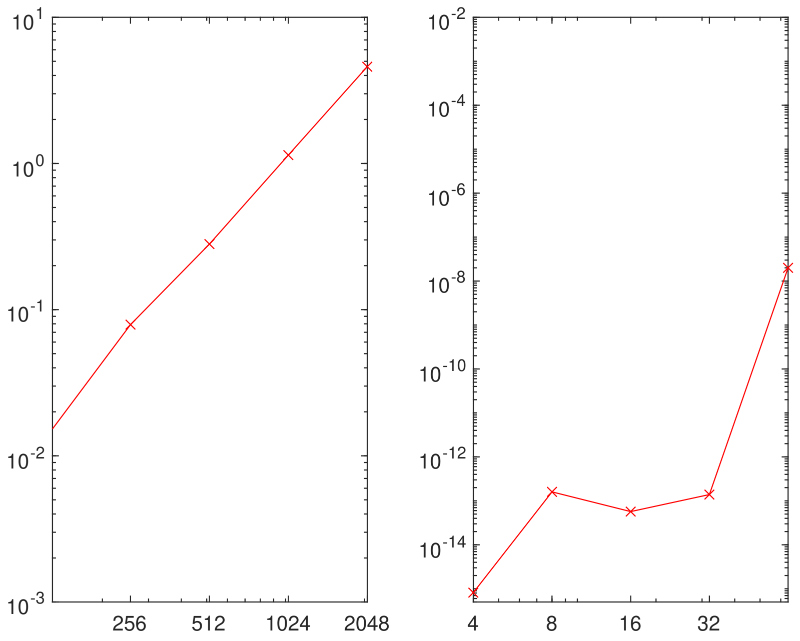

FIG. 7.3. *Time in seconds (left) has been plotted only starting with* $N = 128$*, since for up to* $N = 64$ *it is* 0 *and only up to* $N = 2048$*, since from* $N = 4096$ *it grows exponentially. The average entry of the* $N \times N$ *matrix* $X^*X - I_N$ *(right), where the columns of the matrix* X *are the eigenvectors, to measure orthogonality errors for eigenvectors.*

larger. In this way all the checked eigenvectors belonged to a real eigenvalue with absolute value between $\frac{\sqrt{2}}{2}$ and 1, thus far from zero.

We checked the overall time, the average error out of N normalized eigenvectors, and the mean entry of the matrix $E = X^*X - I_N$, where the columns of the $N \times N$ matrix X contain the N eigenvectors. Without errors, the matrix E would have only zero entries. By computing the matrix E, we checked only the orthogonality among the eigenvectors that we found, since they are for sure normal vectors.

The results are plotted in Fig. 7.3. The orthogonality is quite good up to $N = 32$, while for $N = 64$ we obtained errors starting at the eighth decimal digit. For $N = 128$ and more, there is no orthogonality at all. The time for finding all the eigenvectors has been not much more than twice the time for finding the signs of the sines with our algorithm, up to the size $N = 2048$. For $N = 4096$ and $N = 8192$ the time grew no more by a factor of 4 when N doubles, but by a factor of 10. That is why we plotted the time in Fig. 7.3 only up to $N = 2048$ and error only for $N \leq 64$.

The code for the algorithms of the bisection method can be received from the authors upon request.

8. Conclusions and future work.

Even when optimized, the present algorithm is much slower than an optimized implicit QR, but it is much more accurate. Divide and conquer is also faster than our algorithm, but its errors are larger than ours. The errors are small only about 20 times the machine precision and even that probably comes also from the way in which we measure it: by building the whole matrix to apply on it the Matlab function `eig`. We plan to improve the method for finding eigenvectors in future work, to achieve a much better orthogonality.

Acknowledgements. The authors would like to thank Raf Vandebril and the anonymous referees for many useful remarks.

REFERENCES

[1] G. S. AMMAR, W. B. GRAGG, AND L. REICHEL, *On the eigenproblem for orthogonal matrices*, Proceedings of the 25th IEEE Conference on Decision and Control, Athens, Greece, (1986), IEEE Conference Proceedings, Los Alamitos, 1986, pp. 1963–1966.

[2] ———, *Constructing a unitary Hessenberg matrix from spectral data*, Numerical Linear Algebra, Digital Signal Processing and Parallel Algorithms, G. H. Golub and P. Van Dooren, eds., NATO ASI Series, Springer, Berlin, 1991, pp. 385–395.

[3] W. BARTH, R. S. MARTIN, AND J. H. WILKINSON, *Calculations of the eigenvalues of a symmetric tridiagonal matrix by the method of bisection*, Numer. Math., 9 (1967), pp. 386–393.

[4] Y. EIDELMAN AND I. GOHBERG, *On a new class of structured matrices*, Integr. Equ. Oper. Theory, 34 (1999), pp. 293–324.

[5] Y. EIDELMAN, I. GOHBERG, AND I. HAIMOVICI, *Separable Type Representations of Matrices and Fast Algorithms. Vol. I. Basics. Completion problems. Multiplication and Inversion Algorithms*, Oper. Theory Adv. Appl., Springer Nature, Berlin, Germany, 2013.

[6] ———, *Separable Type Representations of Matrices and Fast Algorithms. Vol. II. Eigenvalue Method*, Oper. Theory Adv. Appl., Springer Nature, Berlin, Germany, 2013.

[7] Y. EIDELMAN, I. GOHBERG, AND V. OLSHEVSKY, *Eigenstructure of order-one-quasiseparable matrices. Three-term and two-term recurrence relations*, Linear Algebra Appl., 405 (2005), pp. 1–40.

[8] Y. EIDELMAN AND I. HAIMOVICI, *The fast bisection eigenvalue method for Hermitian order one quasiseparable matrices and computations of norms*, Electron. Trans. Numer. Anal., 44 (2015), pp. 342–366. https://etna.ricam.oeaw.ac.at/vol.44.2015/pp342-366.dir/pp342-366.pdf

[9] ———, *Bisection eigenvalue method for Hermitian matrices with quasiseparable representation and a related inverse problem*, in Operator Theory, Analysis and the State Space Approach, H. Bart, S. ter Horst, A. C. M. Ran, H. J. Woerdeman, eds., Oper. Theory Adv. Appl., 271, Springer Nature, Berlin, Germany, 2018, pp. 181–200.

[10] ———, *Improved bisection eigenvalue method for band symmetric Toeplitz matrices*, Electron. Trans. Numer. Anal., 58 (2023), pp. 316–347. https://etna.ricam.oeaw.ac.at/vol.58.2023/pp316-347.dir/pp316-347.pdf

[11] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, Johns Hopkins University Press, Baltimore, 2013.

[12] W. B.GRAGG, *The QR algorithm for unitary Hessenberg matrices*, J. Comput. Appl. Math., 16 (1986), pp. 1–8.

[13] W. B. GRAGG AND L. REICHEL, *A divide and conquer method for unitary and orthogonal eigenproblem*, Numer. Math., 57 (1990), pp. 695–718.

[14] M. GU, R. GUZZO, X. B. CHI, AND X. Q. CAO, *A stable divide and conquer algorithm for the unitary eigenproblem*, SIAM J. Matrix Anal. Appl., 25 (2003), pp. 385–404.

[15] P. LANCASTER AND M. TISMENETSKY, *The Theory of Matrices*, New York, Academic Press, Cambridge, 1984.

[16] L. REICHEL, G. S. AMMAR, AND W. B. GRAGG, *Discrete least squares approximation by trigonometric polynomials*, Math. Comp., 57 (1991), pp. 273–289.

[17] R. VANDEBRIL, M. VAN BAREL, AND N. MASTRONARDI, *Matrix Computations and Semiseparable Matrices, Vol. I. Linear Systems*, Johns Hopkins University Press, Baltimore, 2008.

[18] ———, *Matrix Computations and Semiseparable Matrices, Vol. II. Eigenvalue and Singular Value Methods*, Johns Hopkins University Press, Baltimore, 2008.

[19] T. L. WANG AND W. B. GRAGG, *Convergence of the unitary QR algorithm with a unitary Wilkinson shift*, Math. Comp., 72 (2002), pp. 375–385.