Electronic Transactions on Numerical Analysis.
Volume 44, pp. 342–366, 2015.
Copyright © 2015, Kent State University.
ISSN 1068–9613.

ETNA
Kent State University
http://etna.math.kent.edu

THE FAST BISECTION EIGENVALUE METHOD FOR HERMITIAN ORDER ONE QUASISEPARABLE MATRICES AND COMPUTATIONS OF NORMS[*]

YULI EIDELMAN[†] AND IULIAN HAIMOVICI[†]

Abstract. Since we can evaluate the characteristic polynomial of an $N \times N$ order one quasiseparable Hermitian matrix A in less than $10N$ arithmetical operations by sharpening results and techniques from Eidelman, Gohberg, and Olshevsky [Linear Algebra Appl., 405 (2005), pp. 1–40], we use the Sturm property with bisection to compute all or selected eigenvalues of A. Moreover, linear complexity algorithms are established for computing norms, in particular, the Frobenius norm $\|A\|_F$ and $\|A\|_\infty, \|A\|_1$, and other bounds for the initial interval to be bisected. Upper and lower bounds for eigenvalues are given by the Gershgorin Circle Theorem, and we describe an algorithm with linear complexity to compute them for quasiseparable matrices.

AMS subject classifications. 15A18, 15A15, 65F35, 15A57, 65F15

Key words. quasiseparable, Hermitian, Sturm property, matrix norm, eigenvalues, bisection

1. Introduction.

The bisection method is one of the most customary tools to compute all or selected eigenvalues of a matrix. The application of this method to Hermitian matrices is essentially based on the Sturm sequence property, which means that for any given real number λ, the number of sign changes in the sequence of the characteristic polynomials of the principal leading submatrices of an $N \times N$ Hermitian matrix A equals the number of eigenvalues which are less than that λ. We will denote the polynomials in such a sequence by

$$\gamma_0(\lambda) \equiv 1, \gamma_1(\lambda), \gamma_2(\lambda), \ldots, \gamma_N(\lambda). \tag{1.1}$$

For rank-structured matrices there exist fast algorithms to evaluate the polynomials in (1.1) in $O(N)$ arithmetic operations and subsequently to compute the roots of γ_N, i.e., the eigenvalues of A, in a fast way. Realizations of this method for symmetric tridiagonal matrix can be found, for instance, in the monograph [9, Section 8.4.1] or in the paper [1].

Other implementations for tridiagonal symmetric matrices use the LDL^T approach with gains in both stability and speed compared to approaches based on the computation of polynomial values; see [4, p. 231]. However, the recursion formula which is obtained in this way is identical to the recursion already obtained by other means in [1]. In the present paper we extend the approach suggested in [1] to Hermitian quasiseparable matrices.

Related results for semiseparable matrices are contained in [12, Section 9.7.2], where an algorithm using the Sturm property and bisection is devised based on the computation of polynomial values as done in [7].

In this paper we present a complete algorithm using this method for the much broader class of (order one) quasiseparable matrices; see [7]. Because Hermitian matrices have only real eigenvalues and since (also by using techniques from [7]) for a given such matrix A and a given real λ, we succeed in evaluating all the polynomials in the sequence (1.1)—in particular the characteristic polynomial γ_N of A itself—in less than $6N$ additions (subtractions) and less than $4N$ multiplications (divisions), it is entirely feasible to find the roots of this polynomial (the eigenvalues of A) by Sturm plus bisection.

For comparison, one can count in [9, Formula (8.4.2)] that for symmetric tridiagonal matrices, $5N$ arithmetical operations are needed as well in order to compute the sequence (1.1).

[*]Received April 18, 2014. Accepted March 28, 2015. Published online on June 17, 2015. Recommended by Froilan Dopico.

[†]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`).

ETNA
Kent State University
http://etna.math.kent.edu

All the other costs, such as the few arithmetic or comparison operations for managing and coordinating the bisection mechanism and especially the second half of each and every bisection step, i.e., finding the number of sign alternations (involving $O(N)$ operations), are exactly the same for the simpler case of symmetric tridiagonal matrices as well as for the broader case presented here. Thus, the $5N$ extra operations multiplied by the number of bisections that we need are a dwarfish supplement, yielding an inexpensive $O(N^2)$ algorithm in the present case.

The Sturm property with bisection is also most suited when we want only a few eigenvalues, especially if we want particular ones like, for instance, the last (largest) $k < N$ eigenvalues. However, the low complexity compared to other existing methods for general Hermitian matrices makes the present method appropriate for such matrices even when the complete set of eigenvalues and not only selected ones are to be computed. In Section 3.4 of the current paper, we show that finding only the largest eigenvalue can be done in linear time.

Similarly as it was done in the paper [1] for tridiagonal matrices, we use here the ratios $D_k(\lambda) = \frac{\gamma_k(\lambda)}{\gamma_{k-1}(\lambda)}$ instead of the polynomials $\gamma_k(\lambda)$ in order to avoid overflow or underflow in the computing process. The recurrences relations for $D_k(\lambda)$ are computed for Hermitian quasiseparable matrices in Section 3.

Also for the same reason, for matrix sizes larger than 150, we have to scale the matrix (i.e., to divide some of its quasiseparable generators by the Frobenius norm $\|A\|_F$ of the matrix) to temporarily obtain smaller eigenvalues with absolute values in the unit interval and thus smaller polynomial ratios. This latter technique has not been used in [1] and permits to reduce overflow for matrices of any size. Without it, even for matrices of the size of tens, the bisection method does not work. See step 5 in Section 3.3 for further details. The scaling and subsequent descaling of the eigenvalues are operations with linear complexity in time. In particular, in Section 4 we describe how to obtain the Frobenius norm in linear time. If we use instead of the Sturm polynomials, ratios of consecutive polynomials and additionally employ scaling, then the algorithm works well for practically any matrix size.

To start with the bisection method, one should determine at first bounds for the eigenvalues of a matrix. Thus, in a natural way, we are led to the problem of developing fast algorithms for computing the norms and other bounds for eigenvalues of order one quasiseparable matrices. Lower and upper bounds for the initial, larger interval to be bisected can be found by the Gershgorin Circle Theorem (see [9]), which we translate in Section 4.2 into an $O(N)$-algorithm for finding eigenvalue bounds for general quasiseparable matrices. Also, such bounds could be found by the more general Ostrowski Circle Theorem, and several matrix norms such as the Frobenius norm $\|A\|_F$ (as we decided to do), the absolute values norm, the 1-norm $\|A\|_1$, or the ∞-norm $\|A\|_\infty$ can be computed for quasiseparable matrices in low, linear complexity, as we show indeed in Sections 4 and 5.

Numerical experiments in Section 6 compare the bisection method with two other methods for Hermitian order one quasiseparable matrices: implicit QR after a reduction to tridiagonal form and divide and conquer. They show that the bisection is the method of choice since it is the fastest (even faster than divide and conquer) and much more accurate. Implicit QR is just at least as accurate, but it is even slower than Matlab.

2. Notation, definitions, and preliminaries.

For a scalar $N \times N$ matrix A, we denote by A_{ij} or by $A(i,j)$ its element in row $1 \leq i \leq N$ and in column $1 \leq j \leq N$ and by $A(i : j, p : q)$ the submatrix containing the entries in the rows with index from i to j, $1 \leq i \leq j \leq N$, and in the columns with index from p to q, $1 \leq p \leq q \leq N$. If $i = j$, then we succinctly write $A(i, p : q)$, and similar for $i \leq j, p = q$, we write $A(i : j, p)$.

2.1. Semiseparable structure (of order one). Matrices with a semiseparable structure include, among others, tridiagonal matrices or companion matrices and are an important example of quasiseparable matrices. According definitions and more can be found, for instance, in Part I of the book [5].

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

$$A_{ij} = \begin{cases} p(i)q(j), & 1 \le j < i \le N, \\ d(i), & 1 \le i = j \le N, \\ g(i)h(j), & 1 \le i < j \le N. \end{cases}$$

Here $p(i)$, $g(i)$, $i = 2, \dots, N$, $q(j)$, $h(j)$, $j = 1, \dots, N-1$, and $d(i)$, $i = 1, \dots, N$, are (possibly complex) numbers.

2.2. Quasiseparable structure (of order one). The class of matrices with quasiseparable structure contains, among others, unitary upper Hessenberg matrices and inverses of certain semiseparable matrices. The following definitions can be found, for instance, in Part I of the book [5].

Following [7, p. 7], let $A = \{A_{ij}\}_{i,j=1}^{N}$ be a matrix with scalar entries A_{ij}. Assume that these are represented in the form

$$A_{ij} = \begin{cases} p(i)a^{>}_{ij}q(j), & 1 \le j < i \le N, \\ d(i), & 1 \le i = j \le N, \\ g(i)b^{<}_{ij}h(j), & 1 \le i < j \le N. \end{cases} \tag{2.1}$$

Here

$$\begin{array}{lll} p(i),\ i = 2, \dots, N, & q(j),\ j = 1, \dots, N-1, & a(k),\ k = 2, \dots, N-1, \\ g(i),\ i = 1, \dots, N-1, & h(j),\ j = 2, \dots, N, & b(k),\ k = 2, \dots, N-1, \\ d(i),\ i = 1, \dots, N, & & \end{array} \tag{2.2}$$

are (possibly complex) numbers. Also, the operations $a^{>}_{ij}$ and $b^{<}_{ji}$ are defined for positive integers i, j, $i > j$ as $a^{>}_{ij} = a(i-1) \cdot \dots \cdot a(j+1)$, for $i > j+1$, and $a^{>}_{j+1,j} = 1$ and $b^{<}_{ji} = b(j+1) \cdot \dots \cdot b(i-1)$, for $i > j+1$, and $b^{<}_{j,j+1} = 1$. It is easy to see that

$$a^{>}_{ij} = a^{>}_{ik} a^{>}_{k+1,j}, \quad i > k \ge j, \tag{2.3}$$

and

$$b^{<}_{ij} = b^{<}_{i,k+1} b^{<}_{k,j}, \quad i \le k < j. \tag{2.4}$$

The matrix A is called *order one quasiseparable*, or simply *quasiseparable*. The representation of a matrix A in the form (2.1) is called a *quasiseparable representation*. The elements in (2.2) are called *quasiseparable generators* of the matrix A. The elements $p(i)$, $q(j)$, $a(k)$ and $g(i)$, $h(j)$, $b(k)$ are called *lower quasiseparable generators* and *upper quasiseparable generators,* respectively, of the matrix A.

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

$$\begin{array}{llll} g(k) = \overline{q(k)}, & h(k) = \overline{p(k)}, & b(k) = \overline{a(k)}, & k = 2, \dots, N-1, \\ g(1) = \overline{q(1)}, & h(N) = \overline{p(N)}. & & \end{array} \tag{2.5}$$

2.3. The Sturm sequences property for a quasiseparable Hermitian matrix. This property yields information on the location of the eigenvalues of matrices being order one quasiseparable. More details can be found, e.g., in [6, Part I]. Following [7, p. 15] we have an analog of the well-known Sturm property for the characteristic polynomials of the principal leading submatrices.

THEOREM 2.1. *Let A be a Hermitian matrix with scalar entries with the quasiseparable generators as in (2.2) with orders equal to one. Assume that $p(k)h(k) \neq 0, \;\; k = 2, \dots, N-2$. Let $\gamma_0(\lambda) \equiv 1$ and $\gamma_k(\lambda) = \det(A(1:k, 1:k) - \lambda I)$, $k = 1, \dots, N$, be the characteristic polynomials of the principal leading submatrices of the matrix A. Let $\nu(\lambda)$ be the number of sign changes in the sequence*

$$\gamma_N(\lambda), \gamma_{N-1}(\lambda), \dots, \gamma_1(\lambda), \gamma_0(\lambda), \tag{2.6}$$

and let (α, β) be an interval on the real axis such that $\gamma_N(\alpha) \neq 0$, $\gamma_N(\beta) \neq 0$. Then

1. *$\nu(\beta) \geq \nu(\alpha)$, and*
2. *the difference $\nu(\beta) - \nu(\alpha)$ equals the number of eigenvalues of the matrix A in the interval (α, β) counting each eigenvalue in accordance with its multiplicity.*

2.4. Recursive relations for the characteristic polynomials. These are useful for computing the sequence of $N+1$ Sturm polynomials of Section 2.3 in linear time. The following results are taken from [7, pp. 10–12]. Consider the characteristic polynomials

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

of the principal submatrices of a matrix A being order one quasiseparable.

THEOREM 2.2. *Let A be an $N \times N$ matrix with scalar entries with the given quasiseparable generators as in (2.2) with orders equal to one. With these generators, determine the following expressions linear in λ:*

$$c_k(\lambda) = (d(k) - \lambda)\, a(k)b(k) - q(k)p(k)b(k) - h(k)g(k)a(k), \quad k = 2, \dots, N-1. \tag{2.7}$$

Let $\gamma_k(\lambda) = \det(A(1:k, 1:k) - \lambda I)$ be the characteristic polynomials of the principal submatrices of the matrix A. Then the following recursive relations hold:

$$\begin{aligned}
\gamma_1(\lambda) &= d(1) - \lambda, \qquad f_1(\lambda) = q(1)g(1), \\
\gamma_k(\lambda) &= (d(k) - \lambda)\gamma_{k-1}(\lambda) - p(k)h(k)f_{k-1}(\lambda), && k = 2, \dots, N-1, \\
f_k(\lambda) &= c_k(\lambda)f_{k-1}(\lambda) + q(k)g(k)\gamma_{k-1}(\lambda), && k = 2, \dots, N-1, \\
\gamma_N(\lambda) &= (d(N) - \lambda)\gamma_{N-1}(\lambda) - p(N)h(N)f_{N-1}(\lambda).
\end{aligned}$$

Moreover, if $p(k) \neq 0$, $h(k) \neq 0$, $k = 2, \dots, N-1$, then

$$\gamma_k(\lambda) = \Psi_k(\lambda)\gamma_{k-1}(\lambda) - \Phi_k(\lambda)\gamma_{k-2}(\lambda), \qquad k = 3, \dots, N, \tag{2.8}$$

where

$$\Psi_k(\lambda) = d(k) - \lambda + \frac{p(k)h(k)}{p(k-1)h(k-1)} c_{k-1}(\lambda), \tag{2.9}$$

$$\Phi_k(\lambda) = \frac{p(k)h(k)}{p(k-1)h(k-1)} l_{k-1}(\lambda)\delta_{k-1}(\lambda), \tag{2.10}$$

$$l_{k-1}(\lambda) = (d(k) - \lambda)a(k) - q(k)p(k), \tag{2.11}$$

$$\delta_{k-1}(\lambda) = (d(k) - \lambda)b(k) - h(k)g(k), \tag{2.12}$$

and

$$(d(k)-\lambda)c_k(\lambda) = l_k(\lambda)\delta_k(\lambda) - q(k)p(k)h(k)g(k). \tag{2.13}$$

2.5. The bisection method. Following [9, Section 8.4.1], for symmetric tridiagonal matrices, if we consider the integer $\nu(\lambda)$ from Section 2.3 and want to compute $\lambda = \lambda_k(A)$, the kth largest eigenvalue of A for some prescribed k, and if b_L, b_U are a lower and an upper bound for the range of the eigenvalues, then we can use the bisection method. Namely, if we start with $z = b_U$, $y = b_L$, and perform repeatedly (until we attain machine precision) the assignment $\lambda = \frac{z+y}{2}$ and then $z = \lambda$ if $\nu(\lambda) \geq k$ and $y = \lambda$ otherwise, then this will yield the desired eigenvalue.

2.6. Matrix norms. In order to apply the bisection method just described, we a priori need a lower and an upper bound for the eigenvalues. The following matrix norms equipped with appropriate $\pm$ signs could be used for that purpose.

Upon [10, Chapter 5], consider the Frobenius norm, namely

$$\|A\|_F = \sqrt{\sum_{i=1}^{N}\sum_{j=1}^{N}|A_{ij}|^2}, \tag{2.14}$$

which is like the other mentioned norms a bound for the eigenvalues. This norm is used, for instance, in [9, Subsection 11.2.2 and Section 12.3].

The maximum value over all columnwise sums of matrix entries is

$$\|A\|_1 = \max_{1\leq j\leq N}\sum_{i=1}^{N}|A_{ij}|, \tag{2.15}$$

which is also called the absolute values norm of a matrix or the 1-norm..

The ∞-norm is the maximum of all rowwise sum of the entries, i.e.,

$$\|A\|_\infty = \max_{1\leq i\leq N}\sum_{j=1}^{N}|A_{ij}|. \tag{2.16}$$

The *condition number* (see also [3]), which indicates the number of significant digits of a solution x of the system $Ax = b$, is computed in this norm as it is given by $\kappa_\infty = \|A\|_\infty\|A^{-1}\|_\infty$.

2.7. The Gershgorin Circle Theorem. The same linear complexity algorithms that we will present for computing matrix norms can be used to compute quantities needed for some other theorems. Here is a particular case, namely that of the Gershgorin Circle Theorem; see [9, p. 341, Theorem 7.2.1, Section 7.2.1].

THEOREM 2.3. *If $A = D + F$ where $D = \operatorname{diag}(d(1), \ldots, d(N))$ and F has zero diagonal entries, then the eigenvalues of A lie in the union of the circles D_i, $i = 1, \ldots, N$, where*

$$D_i = \{z \in \mathbf{C} : |z - d(i)| \leq \sum_{j=1}^{N}|F_{ij}|\}.$$

This theorem has been generalized by Ostrowski as follows:

THEOREM 2.4. *If $A = D + F$ where $D = \operatorname{diag}(d(1), \dots, d(N))$ and F has zero diagonal entries and $\alpha \in [0,1]$, then the eigenvalues of A lie in the union of the circles D_i, $i = 1, \dots, N$, where*

$$D_i = \left\{ z \in \mathbf{C} : |z - d(i)| \le \left(\sum_{j=1}^{N} |F_{ij}| \right)^{\alpha} \left(\sum_{i=1}^{N} |F_{ij}| \right)^{1-\alpha} \right\}. \tag{2.17}$$

2.8. Strictly diagonal dominant matrices. By the same means we can decide in linear time when a quasiseparable matrix satisfies the following definition.

DEFINITION 2.1 ([9, p. 119]). *A matrix A is called strictly diagonal dominant if*

$$|d(i)| > \sum_{i \ne j=1}^{N} |A_{ij}|, \qquad i = 1, \dots, N. \tag{2.18}$$

3. Sturm property with bisection for eigenvalues of Hermitian quasiseparable matrices.

We aim at finding the eigenvalues of an $N \times N$ scalar Hermitian (order one) quasiseparable matrix A with given lower quasiseparable generators and diagonal entries. To this end, we use the Sturm sequence property of the polynomials (2.6) and the bisection method described in Section 2.5.

The eigenvalues are found one by one in the form

$$\lambda_1 \le \lambda_2 \le \ldots \le \lambda_N$$

starting with the largest (from $k = N$ to $k = 1$). First we need to find a lower bound $b_L \le \lambda_1$ and an upper bound $b_U \ge \lambda_N$, but we address this problem in detail in Sections 4 and 5. We suppose for the present section that this task has been completed.

The important rudiments for this method, Sturm property with bisection, appeared first in 1962 in [13]. However, in our method, when finding an eigenvalue, we keep relevant data which are obtained during the search for λ_k for further use when finding neighboring smaller eigenvalues $\lambda_p,\ 1 \le p < k$. To this end, like in [1], we write the continuously improving left and right bounds for a certain eigenvalue into two arrays with N entries each, called left and right. Finally when a stopping criterion is met, the procedure outputs the values $\mathrm{right}(k)$, $k = 1, 2, \dots, N$, starting from $k = N$, which are approximations to the respective eigenvalues λ_k. This is reasonable since both left and right margins are in the end almost equal to each other and to the eigenvalue which is in between them.

We now describe the algorithm starting with the inner operations. The remaining part of this section is divided into three subsections. In the first one, Section 3.1, the quasiseparable generators of a matrix A are used for finding the number $\nu(\lambda)$ of eigenvalues of A smaller than a given value λ. Then in Section 3.2, the eigenvalues themselves are found one by one in a *for* loop, and in Section 3.3, we describe how to prepare data for the loop and put everything together.

3.1. Finding the ratios of consecutive Sturm polynomials and the number $\nu(\lambda)$ of eigenvalues of A smaller than a given λ.

The task is to find $\nu(\lambda)$ for a given λ which is not an eigenvalue itself. We will succeed to compute the sequence of characteristic polynomials (2.6) in less than $10N$ arithmetic operations (additions and multiplications only) as opposed to about half this number which is needed for the simpler case of a symmetric tridiagonal matrix (one can count them in [9, Formula (8.4.2)]). Moreover, only these $5N$ operations are added after all, since for both mentioned cases (the previously known case of symmetric tridiagonal matrices and the present case of general quasiseparable Hermitian matrices), once the sequence (2.6) is computed, all the other operations required to find $\nu(\lambda)$ are the same and so is the small number of remaining operations needed to manage the bisection step itself. These facts permit to generalize the bisection method from the particular case of symmetric tridiagonal matrices to the much broader case of Hermitian quasiseparable matrices in an inexpensive way. Thus, for general Hermitian quasiseparable matrices, the number of arithmetic operations needed increases only by $5N$ for each bisection step.

The modified recursions which we use result from the fact that, like in [1], when trying to compute (2.6), we encountered heavy overflow (i.e., certain numbers are too large for the computer to handle) in the vicinity of the upper and lower bounds of the possible eigenvalue range and, on the other hand, underflow in regions with many eigenvalues, where the values in (2.6) are too low. For that reason we divide the relation (2.8) by $\gamma_{k-1}(\lambda)$ and make other obvious changes to obtain new recursive relations but now for the ratio $D_k(\lambda) = \frac{\gamma_k(\lambda)}{\gamma_{k-1}(\lambda)}$. It is easy to see that the number of sign changes in the sequence (2.6) equals the number of negative values in the sequence $D_k(\lambda)$.

PROPOSITION 3.1. *Let $A = \{A_{ij}\}_{i,j=1}^N$ be a scalar Hermitian matrix with diagonal entries $d(k)$ and order one lower quasiseparable generators $p(i) \neq 0$, $q(i)$, $a(i)$ as in* (2.2). *The number of eigenvalues of A situated at the left of a given real value λ is equal to the number $\nu(\lambda)$ of negative elements in the sequence*

$$D_1(\lambda) = d(1) - \lambda, \quad D_k(\lambda) = \Psi_k(\lambda) - \frac{\Phi_k(\lambda)}{D_{k-1}(\lambda)}, \qquad k = 2, \dots, N, \tag{3.1}$$

where $\Psi_k(\lambda), \Phi_k(\lambda)$ are given by

$$\begin{aligned} &\Psi_2(\lambda) = d(2) - \lambda, \\ &\Psi_k(\lambda) = d(k) - \lambda + \frac{|p(k)|^2}{|p(k-1)|^2} c_{k-1}(\lambda), \qquad k = 3, \dots, N, \end{aligned} \tag{3.2}$$

$$\begin{aligned} &\Phi_2(\lambda) = |p(2)|^2 |q(1)|^2, \\ &\Phi_k(\lambda) = \frac{|p(k)|^2}{|p(k-1)|^2} \left((d(k-1) - \lambda) c_{k-1}(\lambda) + |q(k-1)|^2 |p(k-1)|^2 \right), \\ &\qquad\qquad\qquad\qquad\qquad\qquad k = 3, \dots, N, \end{aligned} \tag{3.3}$$

where

$$c_k(\lambda) = (d(k) - \lambda)|a(k)|^2 - 2\mathrm{Re}\left(q(k) p(k) \overline{a(k)} \right), \quad k = 2, \dots, N-1. \tag{3.4}$$

Proof. If we divide formula (2.8) by $\gamma_{k-1}(\lambda)$, then we obtain (3.1), where $\Psi_k(\lambda), \Phi_k(\lambda)$ are given by (2.9), (2.10). By (2.5), the relations (2.9), (2.10) become formulas (3.2),

$$\Phi_k(\lambda) = \frac{|p(k)|^2}{|p(k-1)|^2} |l_{k-1}(\lambda)|^2 , \tag{3.5}$$

and (3.3), where we also use that the left-hand side in (2.11) and (2.12) are conjugate (complex) numbers. The same argument shows that (2.13) becomes

$$(d(k) - \lambda) c_k(\lambda) = |l_k(\lambda)|^2 - |q(k)|^2 |p(k)|^2 .$$

Now we use this formula with $k-1$ instead of k in order to replace $|l_k(\lambda)|^2$ in (3.5) since we anyhow compute $c_{k-1}(\lambda)$ in (3.2). We thus obtain from (3.5) the formula (3.3). Changes in formula (2.7) in order to reach (3.4) are similar to the ones mentioned above. ☐

The formulas (3.2), (3.3) contain a number of operations which do not involve the value λ, so that these operations can be executed in advance, such that in practice less operations are required for each bisection.

Since it is needed for each of the many bisection steps for each eigenvalue, we have to repeatedly evaluate the sequence in (3.1) for numerous values of λ. Several coefficients in the recursion have to be computed only once for the given matrix A, for instance, the squares of the absolute values of all the lower quasiseparable generators. By doing so, we succeed to compute the recursive sequence $D_k(\lambda)$, $k = 1, \dots, N$, in less than $10N$ arithmetical operations. In Lemma 3.2 we detail the computations which have to be performed only once to prepare suitable data for the fast computation of $\nu(\lambda)$ in Theorem 3.1.

LEMMA 3.2. *Let $A = \{A_{ij}\}_{i,j=1}^N$ be a scalar Hermitian matrix as in Proposition 3.1. Compute the numbers*

$$\begin{aligned} &\alpha(k), \pi(k), \eta(k), \beta(k), \qquad k = 2, \dots, N-1, \\ &\pi(N), \eta(1), \mu(2), \\ &\mu(k), \omega(k) \qquad k = 3, \dots, N, \end{aligned} \tag{3.6}$$

via the following algorithm:

1. *Compute $\eta(1) = q(1)\overline{q(1)}$.*
2. *For $k = 2, \dots, N-1$, compute*

$$\begin{aligned} &\alpha(k) = a(k)\overline{a(k)}, \qquad \pi(k) = p(k)\overline{p(k)}, \qquad \eta(k) = q(k)\overline{q(k)}, \\ &\delta(k) = q(k)p(k)\overline{a(k)}, \quad \delta_R(k) = \delta(k) + \overline{\delta(k)}, \end{aligned} \tag{3.7}$$

$$\begin{aligned} &\beta(k) = \alpha(k)d(k) - \delta_R(k), \\ &\mu(k) = \eta(k-1)\pi(k), \\ &\omega(k) = \pi(k)/\pi(k-1). \end{aligned} \tag{3.8}$$

3. *Compute*

$$\begin{aligned} &\mu(N) = \eta(N-1)\pi(N), \qquad \pi(N) = p(N)\overline{p(N)}, \qquad \text{and} \\ &\omega(2) = 0, \qquad \omega(N) = \pi(N)/\pi(N-1). \end{aligned}$$

Then the values $\Phi_k(\lambda), \Psi_k(\lambda)$ satisfy the relations

$$\begin{aligned} &\Psi_k(\lambda) = d(k) - \lambda + \chi_k(\lambda), \qquad k = 2, \dots, N, \\ &\Phi_k(\lambda) = (d(k-1) - \lambda)\, \chi_k(\lambda) + \mu(k), \end{aligned}$$

where

$$\chi_2(\lambda) = 0, \qquad \chi_k(\lambda) = (\beta(k-1) - \alpha(k-1)\lambda)\omega(k), \quad k = 3, \dots, N-1.$$

Proof. Using (3.2), (3.3), we get

$$\begin{aligned} \Psi_2(\lambda) &= d(2) - \lambda, \\ \Psi_k(\lambda) &= d(k) - \lambda + w(k)c_{k-1}(\lambda), \qquad k = 3, \dots, N-1, \\ \Phi_2(\lambda) &= \mu(2), \\ \Phi_k(\lambda) &= w(k)\,(d(k-1) - \lambda)c_{k-1}(\lambda) + \pi(k-1)\eta(k-1)) \\ &= (d(k-1) - \lambda)\, w(k)c_{k-1}(\lambda) + \mu(k), \qquad k = 3, \dots, N-1. \end{aligned}$$

Hence it only remains to prove that

$$c_{k-1}(\lambda) = \beta(k-1) - \alpha(k-1)\lambda. \tag{3.9}$$

Indeed using (3.4), we get

$$c_{k-1}(\lambda) = (d(k-1) - \lambda)\alpha_{k-1} - \delta_R(k),$$

which implies (3.9). □

The complexity of step 2 of the algorithm in Lemma 3.2 is $16(N-2)$ arithmetic operations (additions, multiplications, and conjugations). The overall complexity is $c_0 = 16N - 27$. Therefore we concentrate much of the complexity into the steps of Lemma 3.2, which are performed only once for a matrix A. Later we work with the numbers (3.6) instead of the lower quasiseparable generators of the matrix A. These numbers could be called *generator-derived Sturm parameters*.

THEOREM 3.1 (**findNu(λ, generators)**). *Let $A = \{A_{ij}\}_{i,j=1}^N$ be a Hermitian (order one) quasiseparable matrix with scalar entries which has diagonal entries $d(k)$, $k = 1, \dots, N$, and the generator-derived Sturm parameters*

$$\begin{aligned}
&\pi(i),\ i = 2,\dots,N, \qquad \eta(j),\ j = 1,\dots,N-1, \qquad \alpha(k), \beta(k),\ k = 2,\dots,N-1,\\
&\omega(k),\ k = 3,\dots,N, \qquad \mu(k),\ k = 2,\dots,N,
\end{aligned}$$

given by formulas (3.7), (3.8) using the lower quasiseparable generators of the matrix A. Let λ be a real number which is not an eigenvalue of A. Then the integer $\nu(\lambda)$ of the eigenvalues of A which are less than λ is obtained via the following algorithm:

1. *Compute the the ratios of consecutive pairs of Sturm sequence of polynomials (2.6).*
   1.1. *Compute*

$$D_1(\lambda) = d(1) - \lambda, \quad \chi_2(\lambda) = 0. \tag{3.10}$$

   1.2. *For $k = 2, \dots, N-1$, compute $t = k - 1$ and*

$$\begin{aligned}
\chi_k(\lambda) &= (\beta(t) - \alpha(t)\lambda)\omega(k),\\
\Psi_k(\lambda) &= d(k) - \lambda + \chi_k(\lambda),\\
\Phi_k(\lambda) &= (d(t) - \lambda)\chi_k(\lambda) + \mu(k),
\end{aligned}$$

   and if $D_t(\lambda) > \epsilon$ (the machine precision), then

$$D_k(\lambda) = \Psi_k(\lambda) - \frac{\Phi_k(\lambda)}{D_t(\lambda)}.$$

2. *Find the number of sign changes in the Sturm sequence of polynomials (2.6) which is in fact equal to the number of negative $D_k(\lambda)$, $k = 1, \dots, N$, that have been just obtained since these are ratios of pairs of consecutive Sturm polynomials. To this end perform the following steps:*
   2.1. *Set $\nu = 0$.*
   2.2. *For $k = 1, \dots, N$, do: if $D_k(\lambda) < 0$, then set $\nu = \nu + 1$.*
   2.3. *Set $\nu(\lambda) = \nu$.*

The complexity of step 1 of the algorithm in Theorem 3.1 is dominated by the operations in step 1.2, namely 3 multiplications, one division, and 6 additions or subtractions, which are performed each of them $N - 1$ times. Therefore the overall complexity of step 1 is $c_1 = 10N - 9$, half of them additions and (almost) half of them multiplications.

Step 2 of the algorithm requires only cheap additions by 1 in half of the cases and cheap comparisons with 0 when the sign (the first bit) is checked. In total, the complexity of one bisection step in the present theorem is

$$c_B \leq 11.5N - 8. \tag{3.11}$$

In the semiseparable (of order one) case in Lemma 3.2, one does not need to compute $\alpha(k) = a(k)\overline{a(k)}$, $k = 2, \ldots, N-1$, in formula (3.7) since all these numbers are equal to one. In formula (3.8) one does not have to multiply with these numbers twice. Therefore the complexity of the algorithm is $4N - 8$ arithmetical operations less.

In Theorem 3.1, which is used in the following to find the number $\nu(\lambda)$, formula (3.10) requires one multiplication less for the semiseparable case, and this is why the bisection algorithm works 10% faster than for quasiseparable matrices.

3.2. Finding the k^{th} largest eigenvalues one by one. We will find each eigenvalue λ_k, $k = N, N-1, \ldots, 1$, by repeated bisection: after we have determined a lower bound L and an upper bound U, we choose λ in the middle, and if our eigenvalue is in the left interval, then we set $U = \lambda$, otherwise $L = \lambda$, so that in either case in the next step the bisection interval in which the eigenvalue lies is of half size. We continue the bisection until we reach the previously determined machine precision, denoted ϵ. The corresponding *while* loop can be described as follows:

- While $\text{abs}(U - L) > \epsilon$, set $\lambda = (U + L)/2$, and we find ν for this λ by the algorithm in Section 3.1, in particular by Theorem 3.1.
- If $\nu = k$, then $U = \lambda$ since the new eigenvalue λ_k is leftwards of λ, otherwise, i.e., if $\nu < k$, set $L = \lambda$.
  In this latter case we also keep this λ as a left bound for the eigenvalue $\lambda_{\nu+1}$, and (if $\nu \neq 0$) we also keep it as a right bound for λ_ν. To this end, like in [1], we have previously defined two arrays with N entries each called left and right.

The complete loop of the bisection step is illustrated in the following. The function $\nu = \text{findNu}(\lambda, \text{ generators})$ in the algorithm below gathers the operations in Theorem 3.1 for finding the number ν of sign changes in the Sturm sequence. This is a simplified—but still well working—version of the algorithm in [1].

The first two lines, before the call to this function, perform 2 arithmetical operations each, and the last 3 lines perform all 3, yielding in total an average of 3.5 arithmetical operations. So one bisection step costs (mostly because of (3.11)), $12N - 3$ arithmetical operations.

Bisection.

```
1.   while $|U - L| > \epsilon$
2.       $\lambda = (U + L)/2$
3.       $\nu = \text{findNu}(\lambda, \text{ generators})$
4.       if $\nu = k$
5.           set $U = \lambda$
6.       end if
7.       else
8.           $L = \lambda, \text{ left}(\nu + 1) = \lambda$
9.           if $\nu \neq 0$
10.              set $\text{right}(\nu) = \lambda$.
11.          end if
12.      end else
13.  end while
```

Now we describe how to find the eigenvalues of A themselves in a *for* loop which includes the just introduced *while* loop. When a new eigenvalue λ_k is to be found, this means that, if there exist $\lambda_{k+1}, \lambda_{k+2}, \ldots, \lambda_N$ rightwards, then they have been found already. We must therefore start again with the left margin L of the bisection interval from the general lower bound of all the eigenvalues, namely b_L. Concerning the right margin U, we might improve it if in $\mathrm{right}(k)$ a better (smaller) value has been written. Now is also the moment to make use of the data written in the array left, but here, since we wrote in there $\mathrm{left}(\nu+1)$ for $\nu < k$, which might be very low, we have to improve L with a larger margin by seeking all of $\mathrm{left}(1), \mathrm{left}(2), \ldots, \mathrm{left}(N)$. This is all we have to do before the *while* loop.

After the *while* loop we only have to write down the recently found λ_k into $\mathrm{right}(k)$. Hence, the complete *for* loop reads as follows.

Bisection for all eigenvalues.

```
1.   for $k = N, N-1, \ldots, 1$,
2.       set $L = b_L$
3.       for $j = 1, \ldots, k$
4.           if $L < \mathrm{left}(j)$
5.               set $L = \mathrm{left}(j)$
6.           end if
7.       end for
8.       if $U > \mathrm{right}(k)$
9.           set $U = \mathrm{right}(k)$
10.      end if
11.      while $(\ldots \quad , \quad \ldots)$
12.      set $\mathrm{right}(k) = \frac{L+U}{2}$ $\lambda_k = \mathrm{right}(k)$
13.  end for
```

In the above *for* loop, at most $2N$ comparisons and assignments are required. If we do not count, as usual, assignments, and if we take in average $k = \frac{N}{2}$, then only $\frac{N}{2}$ operations count. Only 5 other operations are performed outside the *while* loop for each k. Since $k = N, N-1, \ldots, 1$ we have a total complexity of $\frac{N^2}{2} + 5N$ arithmetical operations outside the inner *while* loop.

3.3. Preparations for the loop and putting everything together. In order to prepare the loops in Section 3.2, we first need to perform the following:

1. Find a lower bound b_L and an upper bound b_U for the set of eigenvalues on the real line; this is described in Section 4.
2. Initialize with b_L the array $\mathrm{left}(1:N)$, which contains the larger last known individual left margins of the eigenvalues λ_k, $k = 1, \ldots, N$, and with b_U the array $\mathrm{right}(1:N)$ containing the smaller last known right margins of the same eigenvalues.
3. Initialize once and for all the upper bound U of the current interval to be bisected by $U = b_U$.
4. Receive from another program or compute the machine precision epsilon ϵ.
5. Scale the problem as explained below.
6. Use the algorithm contained in Lemma 3.2 in order to obtain and prepare better coefficients, which are then used to compute the values in

$$\gamma_0(\lambda) = 1, \gamma_2(\lambda), \ldots, \gamma_{N-1}(\lambda), \gamma_N(\lambda)$$

   for many values of the real number λ as fast as possible.

We yet need to explain step 5. Since computing the values (1.1) gives underflow, especially in the vicinity of eigenvalues of A, and overflow for λ large enough (say close to b_L or b_U whichever is farther from the diagonal entries of A) and for large values of N, we scale the matrix A and thus the eigenvalues. Both the underflow and the overflow problems have been pointed out in [1] for symmetric tridiagonal matrices as well.

The problem with overflow is harder, so we scale the lower quasiseparable generators $p(i), i = 2, \dots, N$, and $d(j), j = 1, \dots, N$, by dividing them by the Frobenius norm $\|A\|_F$ of the matrix A. Hence, the Frobenius norm of the resulting matrix will be 1 and the eigenvalues satisfy $|\lambda_k| \leq 1$. Thus, we can take the upper bound $b_U = 1$ and the lower bound $b_L = -1$ for the eigenvalues. This is because the Frobenius norm is the square root of the sum of the square of the singular values, while the eigenvalues are bounded already by the largest singular value alone. The scaling amounts in fact (and is equivalent to) diminishing the entries of A if we had worked with the N^2 matrix entries and not with the few quasiseparable generators. Of course, after solving the problem we will descale back the eigenvalues that we found, this time by multiplying each of them by the Frobenius norm $\|A\|_F$. The scaling is intended to assure that the algorithm works for larger N as addressed in Section 6, where we also indicate the typical size of N.

3.4. Bisection for selected eigenvalues and finding the spectral norm $\|A\|_2$ for a general $N \times N$ matrix A whenever A^*A is quasiseparable. The spectral norm (the 2-norm $\|B\|_2$) of a general matrix B is the largest singular value of that matrix, which in turn is the square root of the largest eigenvalue of the Hermitian matrix $A = B^*B$. Since by the bisection method which we use for Hermitian matrices, it is easy to find only one desired eigenvalue of our choice, we can find $\|B\|_2$ in linear time whenever $A = B^*B$ happens to be quasiseparable and we have its lower quasiseparable generators and its diagonal entries available.

To this end we choose the lower bound L of the interval to be bisected as $\frac{\|Bx\|_2}{\|x\|_2}$ for an arbitrary vector $x \neq 0$ and the upper bound L of that interval as $\|A\|_F$, and we compute the generator-derived Sturm parameters (3.6) by Lemma 3.2. Then we only perform the following loop instead of the large *for* loop above.

Bisection for one eigenvalue.
```
1.   while $|U - L| > \epsilon$
2.       $\lambda = (U + L)/2$
3.       $\nu = \text{findNu}(\lambda, \text{ generators})$
4.       if $\nu = N$
5.           set $U = \lambda$
6.       else
7.           $L = \lambda$
8.       end if else
9.   end while
10.  set $\|A\|_2 = \sqrt{\frac{U+L}{2}}$
```

In the case when B itself is a Hermitian quasiseparable matrix, $\|B\|_2$ equals the maximal absolute value of its eigenvalues. This means that

$$\|B\|_2 = \max\{|\lambda_1|, |\lambda_N|\},$$

and if moreover B is positive definite, then $\|B\|_2 = \lambda_N$ and we can apply the above algorithm using B itself instead of $A = B^*B$.

4. The Frobenius $\|A\|_F$ norm of an order one quasiseparable matrix. In the algorithm presented above, we use as initial bounds for the eigenvalues

$$b_L = -\|A\|_F, \quad b_U = \|A\|_F.$$

In this section we derive fast algorithms to compute the Frobenius norm for an order one quasiseparable matrix.

Computation in linear time of the Frobenius norm for a proper represented (in the G-d, Givens vector representation) semiseparable symmetric matrix can be found in [11]. In the following, we extend this to the larger class of quasiseparable matrices in two ways.

Using (2.14) we have

$$\|A\|_F^2 = \sum_{1\le j<i\le N} |A_{ij}|^2 + \sum_{i=1}^{N} |A_{ii}|^2 + \sum_{1\le i<j\le N} |A_{ij}|^2. \tag{4.1}$$

We derive a backward and a forward recursion algorithms to compute this value in $O(N)$ operations.

4.1. The backward recursion algorithm. In this subsection we rewrite (4.1) in the form

$$\|A\|_F^2 = \sum_{j=1}^{N-1} \sum_{i=j+1}^{N} |A_{ij}|^2 + \sum_{i=1}^{N} |A_{ii}|^2 + \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} |A_{ij}|^2. \tag{4.2}$$

THEOREM 4.1. *Let $A = \{A_{ij}\}_{i,j=1}^N$ be a scalar matrix with diagonal entries $d(k)$ and lower and upper quasiseparable generators $p(i)$, $q(j)$, $a(k)$, $g(i)$, $h(j)$, $b(k)$ as in (2.2) all of order one. Then the Frobenius norm n of the matrix A is obtained via the following algorithm:*

1. *For $k = N$ initialize*

$$s_U(N-1) = |h(N)|^2, \quad s_L(N-1) = |p(N)|^2 \qquad \textit{and} \tag{4.3}$$

$$n = |d(N)|^2. \tag{4.4}$$

2. *For $k = N-1, \ldots, 2$, perform the following steps: compute*

$$n = n + |q(k)|^2 s_L(k) + |d(k)|^2 + |g(k)|^2 s_U(k), \tag{4.5}$$

$$s_U(k-1) = |h(k)|^2 + |b(k)|^2 s_U(k), \tag{4.6}$$

$$s_L(k-1) = |p(k)|^2 + |a(k)|^2 s_L(k). \tag{4.7}$$

3. *Compute*

$$n = n + |q(1)|^2 s_L(1) + |d(1)|^2 + |g(1)|^2 s_U(1), \tag{4.8}$$

$$n = \sqrt{n}. \tag{4.9}$$

Proof. Inserting (2.1) in (4.2) we get

$$\begin{aligned}\|A\|_F^2 = &\sum_{j=1}^{N-1} \left(\sum_{i=j+1}^{N} |p(i)|^2 |a_{ij}^{>}|^2 \right) |q(j)|^2 \\ &+ \sum_{i=1}^{N} |d(i)|^2 + \sum_{i=1}^{N-1} |g(i)|^2 \left(\sum_{j=i+1}^{N} |b_{ij}^{<}|^2 |h(j)|^2 \right),\end{aligned} \tag{4.10}$$

i.e.,

$$\|A\|_F^2 = \sum_{j=1}^{N-1} \left(s_L(j)|q(j)|^2 + |d(j)|^2 + |g(j)|^2 s_U(j) \right) + |d(N)|^2, \tag{4.11}$$

with

$$s_L(j) = \sum_{i=j+1}^{N} |p(i)|^2 |a^{>}_{ij}|^2, \quad s_U(j) = \sum_{i=j+1}^{N} |b^{<}_{ji}|^2 |h(i)|^2.$$

One has $s_L(N-1) = |p(N)|^2$, and from (2.3) and $a^{>}_{j+1,j} = 1$, it follows that $s_L(j)$ satisfies the recursive relations

$$\begin{aligned} s_L(j-1) &= \sum_{i=j}^{N} |p(i)|^2 |a^{>}_{i,j-1}|^2 = |p(j)|^2 + \sum_{i=j+1}^{N} |p(i)|^2 |a^{>}_{ij}|^2 |a(j)|^2 \\ &= |p(j)|^2 + s_L(j)|a(j)|^2, \qquad j = N-1, \dots, 2. \end{aligned}$$

Similarly, one has $s_U(N-1) = |h(N)|^2$, and from (2.4) and $b^{>}_{j,j+1} = 1$, the recursion for $s_U(j)$ follows:

$$\begin{aligned} s_U(j-1) &= \sum_{i=j}^{N} |b^{<}_{j-1,i}|^2 |h(i)|^2 = |h(j)|^2 + \sum_{i=j+1}^{N} |b(j)|^2 |b^{<}_{ji}|^2 |h(i)|^2 \\ &= |h(j)|^2 + |b(j)|^2 |s_U(j), \qquad j = N-1, \dots, 2. \end{aligned}$$

Hence the recursive relations (4.3), (4.6), and (4.7) hold. Next, the relations (4.4), (4.5), (4.8) follow directly from (4.11). The equality (4.9) is obvious. ☐

The complexity of step 1 of the algorithm in Theorem 4.1 involves 6 arithmetic operations, where additions, multiplications, and conjugation (since $|z|^2 = z \cdot \overline{z}$) are counted. The complexity of step 2 in formula (4.5) is $11(N-2)$ operations; the complexity of formulas (4.6) and (4.7) is $6(N-2)$ operations each. Together with 12 operation in step 3, the overall complexity of this algorithm is $23N - 28$ arithmetical operations. For the particular case of Hermitian quasiseparable matrices, we obtain roughly half the complexity.

COROLLARY 4.1. *Let $A = \{A_{ij}\}_{i,j=1}^N$ be a scalar Hermitian matrix with diagonal entries $d(k)$ and order one lower quasiseparable generators $p(i)$, $q(j)$, $a(k)$ as in* (2.2). *Then the Frobenius norm n of the matrix A is obtained via the following algorithm:*

1. *For $k = N$ initialize*

$$s_L(N-1) = |p(N)|^2, \qquad n = d(N)^2.$$

2. *For $k = N-1, \dots, 2$, perform the following steps: compute*

$$\begin{aligned} n &= n + 2|q(k)|^2 s_L(k) + d(k)^2, \\ s_L(k-1) &= |p(k)|^2 + |a(k)|^2 s_L(k). \end{aligned} \tag{4.12}$$

3. *Compute*

$$\begin{aligned} n &= n + 2|q(1)|^2 s_L(1) + d(1)^2, \\ n &= \sqrt{n}. \end{aligned}$$

The complexity of step 1 of the algorithm in Corollary 4.1 involves 3 arithmetic operations where additions, multiplications, modulus, and initializations are counted. The complexity of step 2, formula (4.12), is $7(N-2)$ operations, the complexity of formula (4.7) is $6(N-2)$ operations. Together with 8 operation in step 3, the overall complexity of this algorithm is $13N - 15$ arithmetical operations.

4.2. The forward recursion algorithm. We can also compute the Frobenius norm using the representation

$$\|A\|_F^2 = \sum_{i=2}^{N}\sum_{j=1}^{i-1}|A_{ij}|^2 + \sum_{i=1}^{N}|A_{ii}|^2 + \sum_{j=2}^{N}\sum_{i=1}^{j-1}|A_{ij}|^2$$

instead of (4.2) and, respectively, the formula

$$\|A\|_F^2 = \sum_{i=2}^{N}|p(i)|^2\left(\sum_{j=1}^{i-1}|a^{>}_{ij}|^2|q(j)|^2\right) + \sum_{i=1}^{N}|d(i)|^2 + \sum_{j=2}^{N}\left(\sum_{i=1}^{j-1}|g(i)|^2|b^{<}_{ij}|^2\right)|h(j)|^2$$

instead of (4.10). As a result we obtain the following algorithm.

THEOREM 4.2. *Let $A = \{A_{ij}\}_{i,j=1}^N$ be a scalar matrix with diagonal entries $d(k)$ and lower and upper quasiseparable generators $p(i)$, $q(j)$, $a(k)$, $g(i)$, $h(j)$, $b(k)$ as in (2.2) all of order one. Then the Frobenius norm n of the matrix A is obtained via the following algorithm:*

1. *For $k = N$ initialize*

$$s_U(2) = |g(1)|^2, \quad s_L(2) = |q(1)|^2 \qquad \text{and}$$
$$n = |d(1)|^2.$$

2. *For $k = 2, \dots, N-1$ perform the following: compute*

$$n = n + |p(k)|^2 s_L(k) + |d(k)|^2 + |h(k)|^2 s_U(k),$$
$$s_U(k+1) = |g(k)|^2 + |b(k)|^2 s_U(k),$$
$$s_L(k+1) = |q(k)|^2 + |a(k)|^2 s_L(k).$$

3. *Compute*

$$n = n + |p(N)|^2 s_L(N) + |d(N)|^2 + |h(N)|^2 s_U(N),$$
$$n = \sqrt{n}.$$

The overall complexity of this algorithm is $23N - 28$ arithmetical operations.

COROLLARY 4.2. *Let $A = \{A_{ij}\}_{i,j=1}^N$ be a scalar Hermitian matrix with diagonal entries $d(k)$ and order one lower quasiseparable generators $p(i)$, $q(j)$, $a(k)$ as in (2.2). Then the Frobenius norm n of the matrix A is obtained via the following algorithm:*

1. *For $k = 1$ initialize*

$$s_L(2) = |q(1)|^2, \quad n = d(1)^2.$$

2. *For $k = 2, \dots, N-1$ perform the following: compute*

$$n = n + 2|p(k)|^2 s_L(k) + d(k)^2,$$
$$s_L(k+1) = |q(k)|^2 + |a(k)|^2 s_L(k).$$

3. *Compute*

$$n = n + 2|p(N)|^2 s_L(N) + d(N)^2,$$
$$n = \sqrt{n}.$$

The overall complexity of this algorithm is $13N - 15$ arithmetical operations.

For the semiseparable case, the norm-algorithms work about 10% faster than for quasiseparable matrices since, for instance, in Theorem 4.1 for the Frobenius norm, the number of operations is reduced by three in both formulas (4.6) and (4.7).

Similar algorithms, but without the equations that involve the diagonal entries $d(k)$ also permit the calculation of the Frobenius norm of the matrix $\|A - D\|_F$, where $D = \mathrm{diag}(d(1), \ldots, d(N))$. The central idea behind Jacobi's method in [9, Section 8.5] is to systematically reduce this quantity (called $\mathrm{off}(A)$ there), the "norm" of the off-diagonal entries of A.

5. Other norms and applications.

Here we present $O(N)$-algorithms to compute the 1-norm in (2.15) and the ∞-norm in (2.16) for an order one quasiseparable matrix. The proofs are similar to the ones for the Frobenius norm in the preceding section, and we skip them.

5.1. The $\|A\|_1$ norm.

THEOREM 5.1. *Let $A = \{A_{ij}\}_{i,j=1}^N$ be a scalar matrix with diagonal entries $d(k)$ and lower and upper quasiseparable generators $p(i)$, $q(j)$, $a(k)$, $g(i)$, $h(j)$, $b(k)$ as in (2.2) all of order one. Then the norm $n_1 = \|A\|_1$ of the matrix A is obtained via the following algorithm:*

1. *Compute the sums of the absolute values of the upper portion of the columns.*
   1.1. *Set*
$$s_U = |g(1)|, \quad u(1) = 0.$$
   1.2. *For $k = 2, \ldots, N-1$, perform the following: compute*
$$u(k) = s_U|h(k)|, \quad s_U = |g(k)| + s_U|b(k)|.$$
   1.3. *Set*
$$u(N) = s_U|h(N)|.$$
2. *Compute the sums of the absolute values of the lower portion of the columns.*
   2.1. *Set*
$$s_L = |p(N)|, \quad w(N) = 0.$$
   2.2. *For $k = N-1, \ldots, 2$, perform the following: compute*
$$w(k) = s_L|q(k)|, \quad s_L = |p(k)| + s_L|a(k)|.$$
   2.3. *Set*
$$w(1) = s_L|q(1)|.$$
3. *Find the maximal column.*
   3.1. *Set*
$$n_1 = 0.$$
   3.2. *For $k = 1, \ldots, N$, perform the following: compute*
$$c = u(k) + w(k) + |d(k)|,$$
   and set $n_1 = c$ whenever $c > n_1$.

Step 1 of this algorithm comprises the steps 1.1, 1.2, and 1.3, which involve $1 + 6(N-2) + 2$ arithmetic operations each, where additions, multiplications, or modulus are counted as one. Step 2 comprises steps 2.1, 2.2, and 2.3 which have $1 + 6(N-2) + 2$ arithmetic operations each. The complexity of step 3.2 involves $5N$ arithmetic operations respectively, where additions, multiplications, comparisons, or modulus are counted once. In total, the complexity of the algorithm in Theorem 5.1 is $17N - 18$ arithmetic operations.

5.2. The $\|A\|_\infty$ norm.

THEOREM 5.2. *Let $A = \{A_{ij}\}_{i,j=1}^{N}$ be a scalar matrix with diagonal entries $d(k)$ and lower and upper quasiseparable generators $p(i)$, $q(j)$, $a(k)$, $g(i)$, $h(j)$, $b(k)$ as in (2.2) all of order one. Then the norm $n_i = \|A\|_\infty$ of the matrix A is obtained via the following algorithm:*

1. *Compute the sums of the absolute values of the upper triangular portion of the rows.*
   1.1. *Set*
$$s_U = |h(N)|, \quad u(N) = 0.$$
   1.2. *For $k = N-1, \ldots, 2$, perform the following: compute*
$$u(k) = s_U|g(k)|, \quad s_U = |h(k)| + s_U|b(k)|.$$
   1.3. *Set*
$$u(1) = s_U|g(1)|.$$
2. *Compute the sums of the absolute values of the lower triangular portion of the rows.*
   2.1. *Set*
$$s_L = |q(1)|, \quad w(1) = 0.$$
   2.2. *For $k = 2, \ldots, N-1$, perform the following: compute*
$$w(k) = s_L|p(k)|, \quad s_L = |q(k)| + s_L|a(k)|.$$
   2.3. *Set*
$$w(1) = s_L|p(N)|.$$
3. *Find the maximal row.*
   3.1. *Set*
$$n_i = 0.$$
   3.2. *For $k = 1, \ldots, N$, perform the following: compute*
$$c = u(k) + w(k) + |d(k)|$$
   and set $n_i = c$ whenever $c > n_i$.

In total, the complexity of the algorithm in Theorem 5.2 is $17N - 18$ arithmetic operations.

5.3. Lower and upper bounds for the eigenvalues. By an application of Gershgorin's and Ostrowski's theorem and using the steps of Theorems 5.1 and 5.2 for the computation of (2.17), we obtain the following two theorems.

THEOREM 5.3. *Let $A = \{A_{ij}\}_{i,j=1}^{N}$ be a scalar matrix with diagonal entries $d(k)$ and lower and upper quasiseparable generators $p(i)$, $q(j)$, $a(k)$, $g(i)$, $h(j)$, $b(k)$ as in (2.2) all of order one. Then a lower bound b_L and an upper bound b_U for the absolute values of the eigenvalues of the matrix A is obtained via the following algorithm:*

ETNA
Kent State University
http://etna.math.kent.edu

1. *Perform step 1 from Theorem 5.1.*
2. *Perform step 2 from Theorem 5.1.*
3. *Find the bounds.*
   3.1. *Set*
$$b_U = d(1) + u(1) + w(1), \quad b_L = d(1) - u(1) - w(1).$$
   3.2. *For $k = 2, \ldots, N$ perform the following: compute*
$$s = w(k) + u(k),$$
$$c(k) = d(k) + s, \quad e(k) = d(k) - s,$$
and set $b_U = c(k)$ whenever $c(k) > b_U$ and $b_L = e(k)$ whenever $e(k) < b_L$.

In total the complexity of the algorithm in Theorem 5.3 is $19N - 21$ arithmetic operations.

THEOREM 5.4. *Let $A = \{A_{ij}\}_{i,j=1}^N$ be a scalar matrix with diagonal entries $d(k)$ and lower and upper quasiseparable generators $p(i)$, $q(j)$, $a(k)$, $g(i)$, $h(j)$, $b(k)$ as in (2.2) all of order one. Then a lower bound $\tilde{b}_L$ and an upper bound $\tilde{b}_U$ for the eigenvalues of the matrix A is obtained via the following algorithm:*

1. *Perform step 1 from Theorem 5.2.*
2. *Perform step 2 from Theorem 5.2.*
3. *Find the lower and the upper bounds.*
   3.1. *Set*
$$\tilde{b}_U = d(1) + u(1) + w(1). \quad \tilde{b}_L = d(1) - u(1) - w(1).$$
   3.2. *For $k = 2, \ldots, N$ perform the following: compute*
$$s = w(k) + u(k),$$
$$c(k) = d(k) + s, \quad e(k) = d(k) - s,$$
and set $\tilde{b}_U = c(k)$ whenever $c(k) > \tilde{b}_U$ and $\tilde{b}_L = e(k)$ whenever $e(k) < \tilde{b}_L$.

In total, the complexity of the algorithm in Theorem 5.4 is $19N - 21$ arithmetic operations. Note that we can apply Ostrowski's theorem as in (2.17) to quasiseparable matrices in a similar manner.

Summing up, a lower bound for the eigenvalues of a matrix is
$$\max\{-\|A\|_F, -\|A\|_1, -\|A\|_\infty, b_L, \tilde{b}_L\},$$
and an upper bound for the eigenvalues of a matrix is
$$\min\{\|A\|_F, \|A\|_1, \|A\|_\infty, b_U, \tilde{b}_U\}.$$

5.4. Strictly diagonal dominant order one quasiseparable matrices. We now determine when an order one quasiseparable matrix is strictly diagonal dominant as in (2.18); see, [9, p. 119]. Note that this property can ensure a nice, non-pivoting LU factorization, as
$$\|D^{-1}(L + U)\|_\infty = \max_{1 \le i \le N} \sum_{i \ne j = 1}^{N} \frac{|A_{ij}|}{|d(i)|} < 1,$$
where $D = \operatorname{diag}(d(1), \ldots, d(N))$.

THEOREM 5.5. *Let $A = \{A_{ij}\}_{i,j=1}^N$ be a matrix with diagonal entries $d(k)$ and lower and upper quasiseparable generators $p(i)$, $q(j)$, $a(k)$, $g(i)$, $h(j)$, $b(k)$ as in (2.2) all of order one. Then the fact that the matrix A is strictly diagonal dominant is obtained as the value of y in the following algorithm:*

ETNA
Kent State University
http://etna.math.kent.edu

```
1. Perform step 1 from Theorem 5.2.
2. Perform step 2 from Theorem 5.2.
3. Check the condition (2.18).
  3.1. Set $y = 1$.
  3.2. For $k = 1, \dots, N$, perform the following: set $y = 0$, and break the loop
       whenever $u(k) + w(k) \geq |d(k)|$.
```

In total, the complexity of the algorithm in Theorem 5.5 is $15N - 18$ arithmetic operations.

6. Numerical experiments.

All the numerical experiments have been performed on a laptop with an Intel i5-3470 processor, 3 gigabyte of memory, and the machine precision epsilon 1.8e-16.

A structure containing random quasiseparable generators is built. For each matrix size N, 20 random quasiseparable matrices and 20 random semiseparable matrices are considered. For a Hermitian matrix, complex lower quasiseparable generators $a(k)$, $p(k)$, $q(k)$ are built by using twice the Matlab function *rand* (once for the real part and once for the complex part). Diagonal entries $d(k)$ are created as real numbers. The mean and the deviation of the numbers generated by *rand* are 0.5. For semiseparable matrices, $p(k)$ and $q(k)$ are build in a similar way.

For each of the 20 random quasiseparable matrices of each size, 21 parameters are computed and written into an Excel file for statistical postprocessing. For semiseparable matrices, we stored only 10 parameters since we do not keep the results of the computations of norms, except the Frobenius norm. The reason being that the other norms for semiseparable matrices give results similar to the quasiseparable case.

Among the parameters are those which show the average or the maximum of the absolute values of the differences between our N eigenvalues compared to the eigenvalues found by Matlab. The computation of these parameters is simplified by the fact that both sets of eigenvalues are already sorted in ascending order. The Matlab results are considered the exact ones.

We describe how we measure the errors in the computation of the eigenvalues for the numerical experiments. If we denote by $\lambda(k)$ and $\lambda_M(k)$, $k = 1, \dots, N$, the eigenvalues found by the bisection method and, respectively, the Matlab eigenvalues, then the mean relative error for 20 matrices is

$$\frac{1}{20} \sum_{i=1}^{20} \frac{1}{N} \frac{\sum_{k=1}^{N} |\lambda(k) - \lambda_M(k)|}{|\lambda_M(k)|}, \tag{6.1}$$

while the worst relative error is computed by

$$\frac{1}{20} \sum_{i=1}^{20} \max_{1 \leq k \leq N} \frac{|\lambda(k) - \lambda_M(k)|}{|\lambda_M(k)|}. \tag{6.2}$$

Absolute errors are stored as well.

We note that when we use Matlab to compute the eigenvalues, we explicitly form the matrix from the generators, which is then passed on to Matlab for the eigenvalue computation. It can be shown that there can occur an instability in the step of converting generators to the quasiseparable matrix. This is a cause of concern since different algorithms for forming the quasiseparable matrix may give slightly different entries.

Figure 6.3 displays the average relative errors of the eigenvalues over 20 matrices of fixed size, where the size N is a power of 2 between 32 and 2048, for the algorithms Sturm bisection, divide and conquer, and reduction to tridiagonal plus implicit QR. The computed

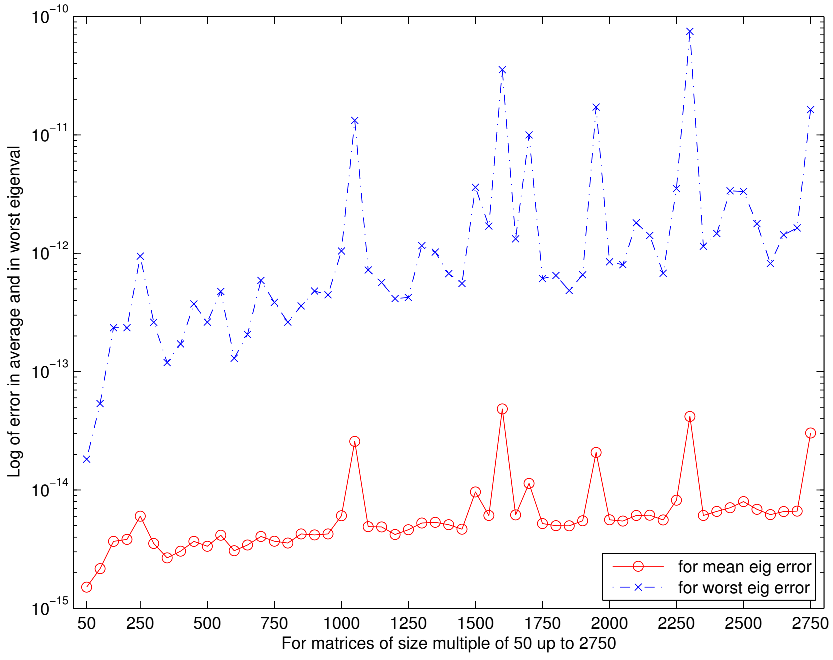

FIG. 6.1. *Average and worst absolute error in finding eigenvalues by Sturm bisection as compared to Matlab eigenvalues, which are considered the exact ones.*

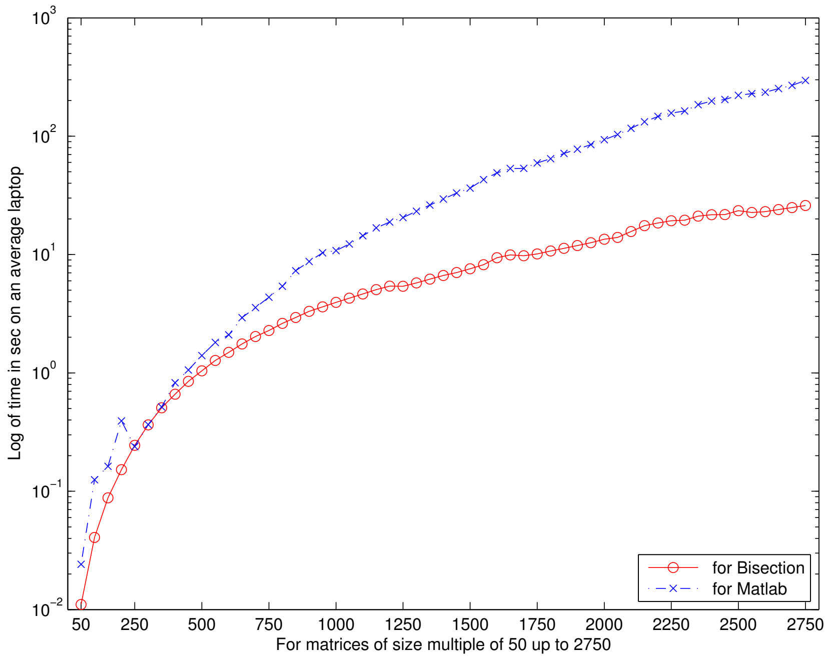

FIG. 6.2. *Time in seconds for Sturm bisection compared to Matlab for* 1100 *order one quasiseparable Hermitian matrices with* 20 *matrices for each size* $50, 100, \ldots, 2750$*. From* 750 *on, Sturm bisection is already twice more powerful. For matrices of size* 2750*, it works* 12 *times faster.*

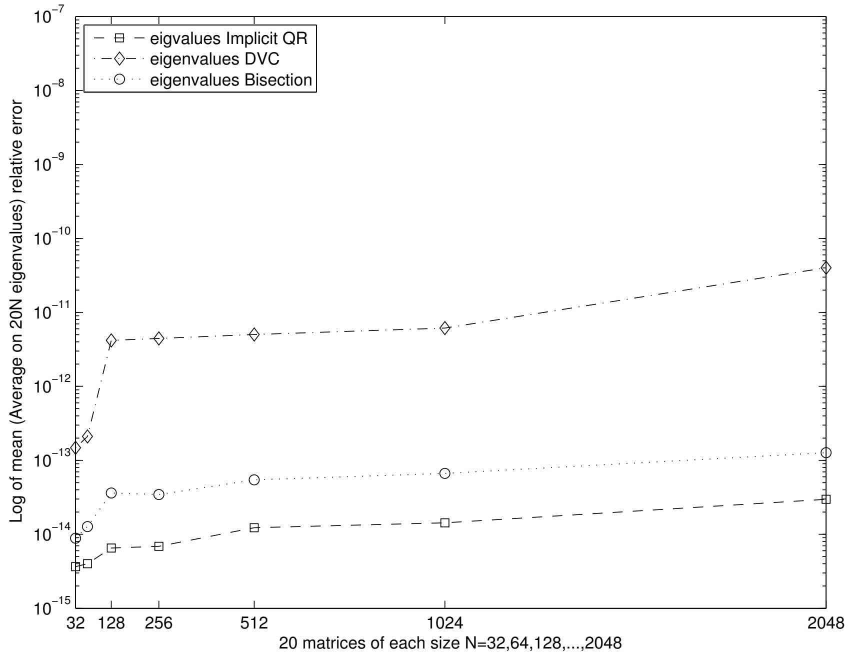

FIG. 6.3. *Average relative errors of* $20N$ *eigenvalues of* 20 *matrices of size* N, *where* N *is a power of* 2 *between* 32 *and* 2048, *for Sturm bisection, divide and conquer, and reduction to tridiagonal plus implicit* QR *as compared to Matlab eigenvalues.*

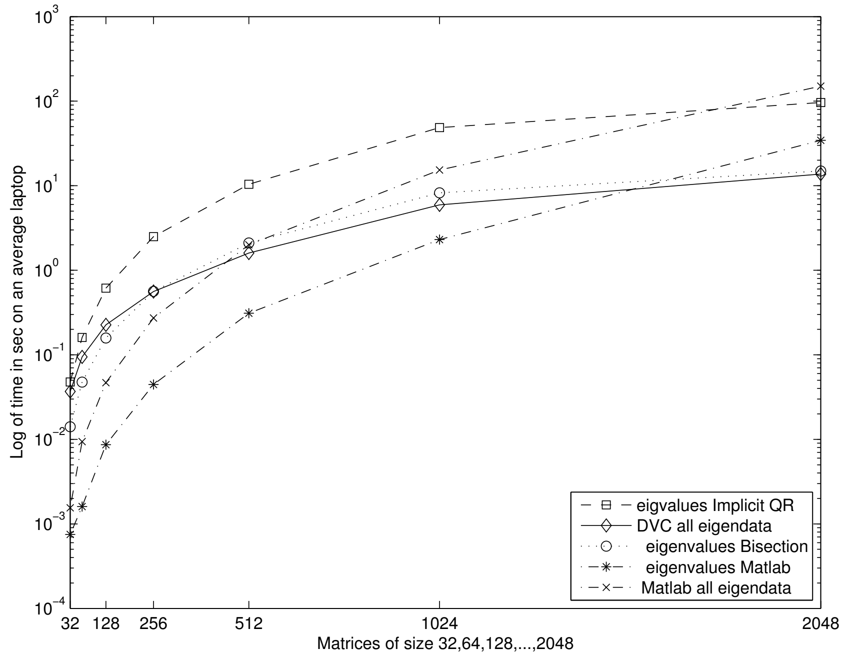

FIG. 6.4. *The average time for finding all the eigenvalues of* 20 *matrices of size* N, *where* N *is a power of* 2 *between* 32 *and* 2048, *for Sturm bisection, divide and conquer (including eigenvectors), and reduction to tridiagonal plus implicit* QR *as compared to Matlab (either only eigenvalues or all eigendata).*

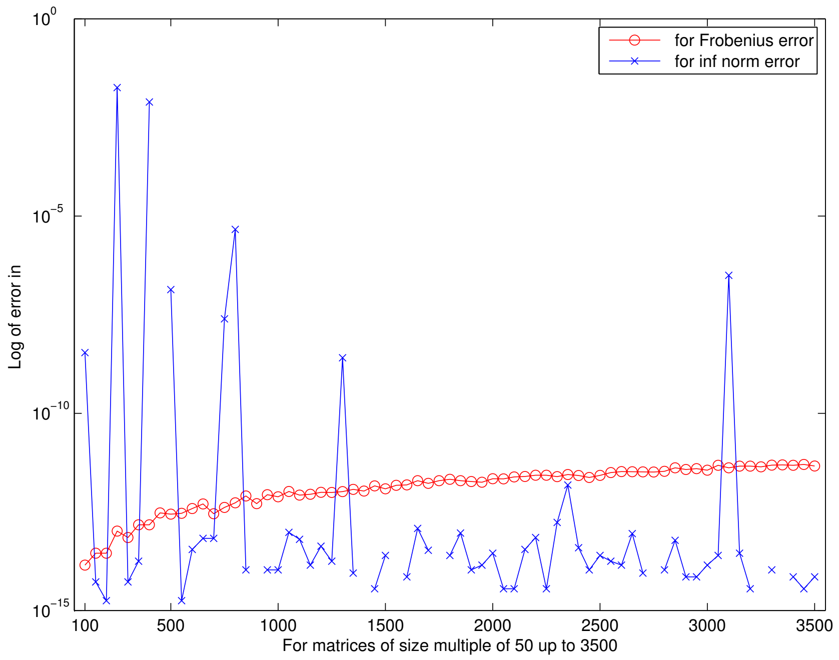

FIG. 6.5. *Error in computing the Frobenius norm* $\|A\|_F$ *and* $\|A\|_\infty$.

eigenvalues are compared to Matlab eigenvalues $\lambda_M(k)$, $k = 1, \ldots, N$, which are considered to be the exact ones. For each of the $20N$ eigenvalues, we compute the absolute value of the difference between it and the corresponding Matlab eigenvalue divided by its absolute value. The average over $20N$ eigenvalues of the maximal error of Matlab as announced by its developers is also plotted, and it is worse compared to any of the other 3 methods. The worst error over all of the $81,000$ eigenvalues obtained in our experiment was 1.15868e-09.

Also, Figure 6.1 displays the average and worst absolute error in finding eigenvalues for Sturm bisection as compared to the Matlab eigenvalues, which are considered the exact ones. The size N of the matrices in our numerical experiments is a multiple of 50 ranging from 50 to 2750. For each of the N eigenvalues for an $N \times N$ matrix, we compute the absolute value of the difference between it and the corresponding Matlab eigenvalue. Then the average error for a fixed matrix size N is computed as the mean over 20 random matrices of the average of the errors of its N eigenvalues as in (6.1) and (6.2). The worst result over all 1.54 million eigenvalues obtained in our experiment was 1.45e-09. We computed this number as the following maximum over 1100 matrices $\max_{1\leq j\leq 1100} \max_{1\leq k\leq N} \ |\lambda(k) - \lambda_M(k)|$; this number is not plotted. The worst condition number out of the N eigenvalues is computed for each matrix, and the time used by our algorithm, by Matlab, and by our checking program is stored for each matrix.

For large matrices, the computation time of our eigenvalues algorithm is much less than the corresponding time of Matlab. Both of them are given in Figure 6.2. Also, Figure 6.4 displays the time for divide and conquer, for reduction to tridiagonal plus implicit QR, bisection, and Matlab, while explicit QR is not plotted since it is the slowest and the less accurate.

For each matrix we also compute 11 norms and the Gershgorin and Ostrowski bounds for the eigenvalues which are all of them obtained from the quasiseparable structure in order to verify the algorithms, and we also give the total time of these computations. Some of them are equal to each other, for instance, for any matrix, the Frobenius norm computed by rows

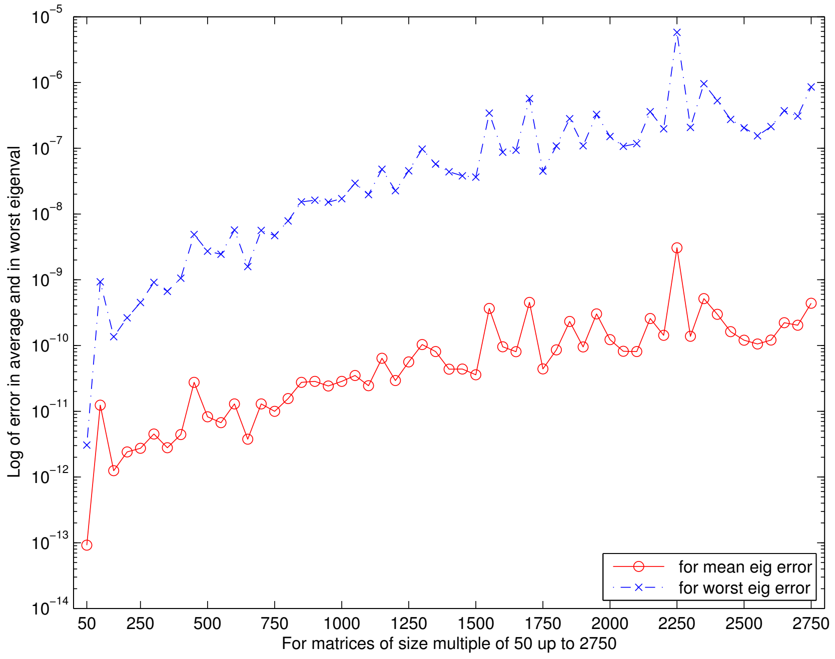

FIG. 6.6. *Absolute error of eigenvalues in semiseparable matrices.*

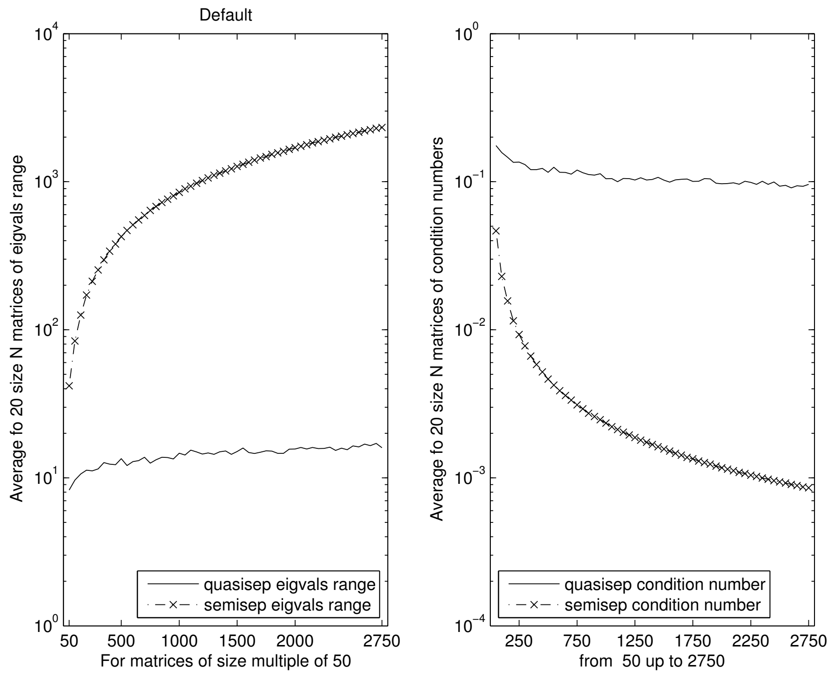

FIG. 6.7. *The range of the eigenvalues (at left) and (at right) the worst condition numbers for the N eigenvalues of semiseparable versus quasiseparable matrices.*

coincides with the one computed by columns, while for Hermitian matrices this value also equals the norm computed by the shorter algorithms in Corollaries 4.1, 4.2. For the 1100 semiseparable matrices, only the Frobenius norm is stored. Figure 6.5 displays the error in computing the Frobenius $\|A\|_F$ and the $\|A\|_\infty$ norm. Computing these norms using the quasiseparable generators involves much less operations than building the entire matrix and then computing again the corresponding norm with Matlab, so that in this case, the error comes from building the large matrix or from Matlab and not from our algorithm. The Frobenius norm is much more stable. Since we took only one matrix for each size, a multiple of 50 up to $3,500$, the results for $\|A\|_\infty$, which sometimes is strongly influenced by a single larger matrix entry, do not vary in a nice way proportionately to the size of the matrix.

Moreover, we computed many other parameters, such as the minimum, the maximum, the mean, and the standard deviation for each of the 40 different matrices for a certain instance out of the 55 sizes. This gives us the possibility to display the average, the worst case, and the best case out of the 20 matrices of a certain size and type.

We also computed more parameters, like, for instance, the numbers $\|A\|_\infty$ and $\|A\|_F$ divided by $\mathrm{norm2}$ (which we obtained by using the Matlab function $\mathrm{norm}(A)$) in order to find out how well the ∞-norm or Frobenius norm approximate the range were the eigenvalues are (i.e., the interval $[-\mathrm{norm2}, \mathrm{norm2}]$ since we work with Hermitian matrices only).

Calculating the results for all the 2200 matrices, which involves our fast Sturm bisection algorithm, and building the matrices from their generators in order to run the Matlab function *condeig* and making many other computations, took more more than five days on an average laptop.

The whole $N \times N$ matrices are computed out of their quasiseparable generators in order to use the Matlab function *condeig*. However, for large matrices, say of size 2000, there are differences of up to 0.00008 in the same entry of the matrix when the large matrix is computed from its quasiseparable generators twice, once starting with the diagonal, as we finally decided to do, and another time starting from the first row. This comes in favor of the Matlab *condeig* function since we then compare its result with ours and Matlab always uses the $N \times N$ matrix.

Finally, we want to underline the fact that for general quasiseparable matrices with random generators between 0 and 1, the range of the eigenvalues is very small as compared to semiseparable matrices, so that the absolute errors in finding eigenvalues are much higher. Figure 6.6 displays absolute (and not relative) errors of the eigenvalues in the semiseparable case, i.e., the average and worst errors in finding eigenvalues for Sturm bisection as compared to the Matlab eigenvalues. Again we consider the Matlab eigenvalues the exact ones. For each of the N eigenvalues of an $N \times N$ matrix, we compute the absolute value of the difference between it and the corresponding Matlab eigenvalue. Then the average is computed for each matrix size N as the mean over 20 random semiseparable matrices of the average of the errors of its N eigenvalues as in (6.1) and (6.2). The worst error of an eigenvalue for semiseparable matrices over all $1,54$ million eigenvalues obtained in our experiment was 1.08e-04 away from the corresponding Matlab eigenvalue. We computed this number as the following maximum over 1100 matrices $\max_{1\le j\le 1100} \max_{1\le k\le N} \ |\lambda(k) - \lambda_M(k)|$. Also, Figure 6.7 displays the range of the eigenvalues (at left), computed as $\frac{1}{20}\sum_{i=1}^{20}(\lambda_{\max,i} - \lambda_{\min,i})$ for quasiseparable matrices compared to semiseparable matrices. This figure shows that semiseparable matrices are harder to deal with. At the right-hand side, the worst condition numbers $\frac{1}{20}\sum_{i=1}^{20} 1/(\max_{1\le j\le N}(s_j) * \mathrm{norm}(A_i))$ for the N eigenvalues of quasiseparable matrices and semiseparable matrices are given. This figure shows that semiseparable matrices have a poorer condition number. In the above formula, s_j is the condition number for the eigenvalue λ_j, $j = 1, \ldots, N$, and $\mathrm{norm}(A_i)$ is the absolute value of the largest singular value of a certain matrix.

REFERENCES

[1] 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.

[2] F. L. Bauer and C. T. Fike, *Norms and exclusion theorems*, Numer. Math., 2 (1960), pp. 137–141.

[3] A. K. Cline, C. B. Moler, G. W. Stewart, and J. H. Wilkinson, *An estimate for the condition number of a matrix*, SIAM J. Numer. Anal., 16 (1979), pp. 368–375.

[4] J. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, 1997.

[5] Y. Eidelman, I. Gohberg, and I. Haimovici, *Separable Type Representations of Matrices and Fast Algorithms. Vol I.*, Birkhäuser, Basel, 2013.

[6] ———, *Separable Type Representations of Matrices and Fast Algorithms. Vol II.*, Birkhäuser, Basel, 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, *Divide and conquer method for eigenstructure of quasiseparable matrices using zeroes of rational matrix functions*, in A Panorama of Modern Operator Theory and Related Topics, H. Dym, M. A. Kaashoek, P. Lancaster, H. Langer, and L. Lerer, eds., Operator Theory: Advances and Applications, 218, Birkhäuser, Basel, 2012, pp. 299–328.

[9] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd. ed., The Johns Hopkins University Press, Baltimore, 1996.

[10] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, 1990.

[11] N. Mastronardi, M. Van Barel, and R. Vandebril, *Computing the rank revealing factorization of symmetric matrices by the semiseparable reduction*, Tech. Report TW 418, Dept. of Computer Science, Katholieke Universiteit Leuven, Leuven, 2005.

[12] R. Vandebril, M. Van Barel, and N. Mastronardi, *Matrix Computations and Semiseparable Matrices. Vol. II.*, The Johns Hopkins University Press, Baltimore, 2008.

[13] J. H. Wilkinson, *Calculation of the eigenvalue of a symmetric tridiagonal matrix by the method of bisection*, Numer. Math., 4 (1962), pp. 362–367.