A CHOLESKY LR ALGORITHM FOR THE POSITIVE DEFINITE SYMMETRIC DIAGONAL-PLUS-SEMISEPARABLE EIGENPROBLEM *

BOR PLESTENJAK , ELLEN VAN CAMP , AND MARC VAN BAREL

Abstract. We present a Cholesky LR algorithm with Laguerre's shift for computing the eigenvalues of a positive definite symmetric diagonal-plus-semiseparable matrix. By exploiting the semiseparable structure, each step of the method can be performed in linear time.

Key words. diagonal-plus-semiseparable matrix, LR algorithm, Laguerre's method, Cholesky decomposition

AMS subject classifications. 65F15

1. Introduction.

The symmetric eigenvalue problem is a well studied topic in numerical linear algebra. When the original matrix is an $n \times n$ symmetric matrix, very often an orthogonal transformation into a similar tridiagonal one is applied because the eigendecomposition of a tridiagonal matrix can be computed in $\mathcal{O}(n)$ and such an orthogonal similarity transformation always exists (see, for example, [5, 9]).

In [14], an orthogonal similarity reduction is presented that reduces any symmetric matrix into a diagonal-plus-semiseparable (from now on denoted by DPSS) one with free choice of the diagonal. This transformation has the same order of computational complexity as the reduction into tridiagonal form, only the second highest order term is a little bit larger. A good choice of the diagonal however, can compensate this small delay when computing the eigenvalues and eigenvectors afterwards.

Several algorithms are known for computing the eigendecomposition of symmetric DPSS matrices, for example, in [2] and [8] divide and conquer techniques are used. The authors of [1, 4, and references therein] focus on QR algorithms and in [10] an implicit QR algorithm is presented.

When the symmetric DPSS matrix is positive definite, also an LR algorithm, based on the Cholesky decomposition, can be applied in order to compute the eigenvalues. Such a Cholesky LR algorithm will be constructed in this paper.

Therefore, we show that the DPSS structure is preserved by the Cholesky decomposition and the LR algorithm. As a shift, Laguerre's shifts (also used for symmetric positive definite tridiagonal matrices in [6]) are used because one has to be sure that the shifted matrix is positive definite again. Exploiting the DPSS structure, one step of the Cholesky LR algorithm, including the computation of the shift, has a computational cost of order $\mathcal{O}(n)$. Because two steps of the LR algorithm are equivalent to one step of the QR algorithm (see, for example, [5]) there will be convergence towards the eigenvalues.

In contrast to the QR algorithm with shifts where the eigenvalues are not computed in any particular order, the eigenvalues in the LR algorithm are computed from the smallest to the largest one. This

*Version: March 8, 2005. The research of the first author was partially supported by the Ministry of Higher Education, Science and Technology of Slovenia, project P1-0294. The research of the second and third author was partially supported by the Research Council K.U.Leuven, project OT/00/16 (SLAP: Structured Linear Algebra Package), G.0176.02 (ANCILA: Asymptotic aNalysis of the Convergence behavior of Iterative methods in numerical Linear Algebra), G.0184.02 (CORFU: Constructive study of Orthogonal Functions) and G.0455.0 (RHPH: Riemann-Hilbert problems, random matrices and Padé-Hermite approximation), and by the Belgian Programme on Interuniversity Poles of Attraction, initiated by the Belgian State, Prime Minister's Office for Science, Technology and Culture, project IUAP V-22 (Dynamical Systems and Control: Computation, Identification and Modelling). The scientific responsibility rests with the authors.

makes it a very suitable algorithm for those applications where the smallest eigenvalues are needed.

The paper is organized as follows. In §2 the concepts used, are explained. The preservation of the DPSS structure under the Cholesky decomposition and the Cholesky LR algorithm is proven in §3. Also explicit fast algorithms for the Cholesky decomposition and the LR algorithm are constructed. In §4 a fast computation of Laguerre's shifts is studied. §5 focuses on the implementation, while numerical results are discussed in §6, followed by conclusions.

2. Preliminaries.

In this section we recall the definition of DPSS matrices and the Givens-vector representation that we will use. The idea of the LR algorithm based on the Cholesky decomposition is repeated as well as Laguerre's method.

DEFINITION 2.1. *An $n \times n$ matrix S is called a* lower- (upper-) semiseparable matrix *if every submatrix that can be taken out of the lower (upper) triangular part of the matrix S, has rank at most 1. If a matrix is lower- and upper-semiseparable, it is called* a semiseparable matrix.

The sum $D+S$ of a diagonal matrix D and a semiseparable matrix S is called a diagonal-plus-semiseparable matrix *or shortly a DPSS matrix.*

To represent a symmetric DPSS matrix, we use the Givens-vector representation based on a vector $f = [f_1, \ldots, f_n]^T$, $n-1$ Givens rotations

$$G_i = \begin{bmatrix} c_i & -s_i \\ s_i & c_i \end{bmatrix}, \quad i = 1, \ldots, n-1,$$

and a diagonal $d = [d_1, \ldots, d_n]^T$ (for more details, see, e.g., [13]).

$$D+S = \begin{bmatrix} c_1 f_1 + d_1 & c_2 s_1 f_1 & \cdots & c_{n-1} s_{n-2:1} f_1 & s_{n-1:1} f_1 \\ c_2 s_1 f_1 & c_2 f_2 + d_2 & \cdots & c_{n-1} s_{n-2:1} f_2 & s_{n-1:2} f_2 \\ \vdots & & \ddots & & \vdots \\ c_{n-1} s_{n-2:1} f_1 & c_{n-1} s_{n-2:2} f_2 & \cdots & c_{n-1} f_{n-1} + d_{n-1} & s_{n-1} f_{n-1} \\ s_{n-1:1} f_1 & s_{n-1:2} f_2 & \cdots & s_{n-1} f_{n-1} & f_n + d_n \end{bmatrix},$$

where $s_{a:b} = s_a s_{a-1} \cdots s_b$. We will denote $D+S = \mathrm{diag}(d) + \mathrm{Giv}(c, s, f)$.

The above representation of a DPSS matrix is not unique. One can see that the parameters d_1 and d_n can be chosen arbitrarily. If we change d_n into $\widetilde{d_n}$ then we can change f_n into $\widetilde{f_n} = f_n + d_n - \widetilde{d_n}$. If we change d_1 into $\widetilde{d_1}$, then one can check that by taking

$$\begin{aligned} \widetilde{f_1} &= \sqrt{(c_1 f_1 + d_1 - \widetilde{d_1})^2 + s_1^2 f_1^2}, \\ \widetilde{c_1} &= (c_1 f_1 + d_1 - \widetilde{d_1})/\widetilde{f_1}, \\ \widetilde{s_1} &= s_1 f_1 / \widetilde{f_1} \end{aligned}$$

we get the same matrix $D+S$. Most often, however, the diagonal d is known, so d_1 and d_n are fixed.

Next we recall another important concept, the Cholesky LR algorithm.

Let A be a symmetric positive definite (from now on denoted by s.p.d.) matrix. Starting from the matrix $A_0 = A$, a Cholesky LR algorithm generates a sequence of similar matrices

$$A_{k+1} = V_k^{-1} A_k V_k = V_k^T V_k, \quad k = 0, 1, \ldots,$$

where $V_k V_k^T = A_k$ is the Cholesky decomposition of A_k with V_k a lower-triangular matrix. The use of a shift at each step can speed up the convergence of the sequence A_k, $k = 0, 1, \ldots$, towards the Schur decomposition of A.

When applying the Cholesky LR algorithm to a s.p.d. DPSS matrix $D+S$, the shift can be included into the diagonal part and hence, when we are able to construct the Cholesky decomposition VV^T of an arbitrary s.p.d. DPSS matrix and the corresponding product $V^T V$, we can apply a step of the shifted Cholesky LR algorithm on a s.p.d. DPSS.

One important remark, however, is that the shift σ should be chosen such that $D+S-\sigma I$ is still positive definite or in other words, the shift σ should be smaller than the smallest eigenvalue of $D+S$. To fulfill this requirement, Laguerre's shifts are used.

Let A be a s.p.d. $n \times n$ matrix with eigenvalues $0 < \lambda_n \leq \lambda_{n-1} \leq \ldots \leq \lambda_1$. Let $f(\lambda) = \det(A - \lambda I)$ be the characteristic polynomial of A. If x is an approximation for an eigenvalue of A and we define

$$\begin{aligned} S_1(x) &= \sum_{i=1}^{n} \frac{1}{\lambda_i - x} &&= -\frac{f'(x)}{f(x)}, \\ S_2(x) &= \sum_{i=1}^{n} \frac{1}{(\lambda_i - x)^2} &&= \frac{f'^2(x) - f(x)f''(x)}{f^2(x)}, \end{aligned}$$

then the next approximation $\widetilde{x}$ by Laguerre's method is given by the equation

$$\widetilde{x} = x + \frac{n}{S_1(x) + \sqrt{(n-1)(nS_2(x) - S_1^2(x))}}. \tag{2.1}$$

Two important properties of Laguerre's method are that if λ_n is a simple eigenvalue and if $x < \lambda_n$ then $x < \widetilde{x} < \lambda_n$ and the convergence towards λ_n is cubic. For multiple eigenvalues the convergence is linear. More details on Laguerre's method and its properties can be found in, e.g., [15].

3. Cholesky decomposition.

In this section we show that the DPSS structure is preserved by both the Cholesky decomposition and the LR algorithm. Even better, if we use the Givens-vector representation, then we can show that some vectors from the representation are invariant to the Cholesky decomposition and the Cholesky LR algorithm. This enables us to produce a fast algorithm for the Cholesky LR step.

THEOREM 3.1. *Let A be a symmetric positive definite diagonal-plus-semiseparable matrix in the Givens-vector representation*

$$A = \mathrm{Giv}(c, s, f) + \mathrm{diag}(d).$$

1. *If V is a lower triangular matrix such that $A = VV^T$ is the Cholesky decomposition of A, then V can be represented in the Givens-vector representation as*

$$V = \mathrm{tril}(\mathrm{Giv}(c, s, \widetilde{f})) + \mathrm{diag}(\widetilde{d}).$$

2. *If $B = V^T V$, where V is the lower triangular Cholesky factor from 1. of A, then B is again a symmetric positive definite diagonal-plus-semiseparable matrix with the same diagonal part as the original matrix A:*

$$B = \mathrm{Giv}(\widehat{c}, \widehat{s}, \widehat{f}) + \mathrm{diag}(d).$$

Proof. 1. We use induction. As we generate A from the top to the bottom, the following relation holds between A_k and A_{k+1}, where $A_1 = [f_1 + d_1]$ and $A_n = A$. If we write

$$A_k = \begin{bmatrix} B_k & a_k \\ a_k^T & f_k + d_k \end{bmatrix},$$

where $B_k \in \mathbb{R}^{(k-1)\times(k-1)}$ and $a_k \in \mathbb{R}^{k-1}$, then

$$A_{k+1} = \begin{bmatrix} B_k & c_k a_k & s_k a_k \\ c_k a_k^T & c_k f_k + d_k & s_k f_k \\ s_k a_k^T & s_k f_k & f_{k+1} + d_{k+1} \end{bmatrix}.$$

If

$$V_k = \begin{bmatrix} W_k & 0 \\ v_k^T & * \end{bmatrix},$$

where $W_k \in \mathbb{R}^{(k-1)\times(k-1)}$ and $v_k \in \mathbb{R}^{k-1}$, is the Cholesky factor of A_k then one can see that the Cholesky factor of A_{k+1} has the form

$$V_{k+1} = \begin{bmatrix} W_k & 0 & 0 \\ c_k v_k^T & * & 0 \\ s_k v_k^T & * & * \end{bmatrix}.$$

It is easy to see that there exist $\widetilde{f_k}$ and $\widetilde{d_k}$ such that

$$V_k = \begin{bmatrix} W_k & 0 \\ v_k^T & \widetilde{f_k} + \widetilde{d_k} \end{bmatrix}$$

and

$$V_{k+1} = \begin{bmatrix} W_k & 0 & 0 \\ c_k v_k^T & c_k \widetilde{f_k} + \widetilde{d_k} & 0 \\ s_k v_k^T & s_k \widetilde{f_k} & * \end{bmatrix}.$$

In the last step, when $k = n-1$, one can also choose appropriate $\widetilde{f_n}$ and $\widetilde{d_n}$ for the right bottom element of V. Hence, the Givens transformations from A appear in the Givens-vector representation of the Cholesky factor V as well.

2. From 1. we know that $A = VV^T$ with V a nonsingular, lower-semiseparable and lower triangular matrix. A is also a s.p.d. DPSS matrix, so $A = D + S$. Hence,

$$D + S \;=\; VV^T.$$

This implies:

$$\begin{aligned} V^T V &= V^T (D+S) V^{-T} \\ &= V^T D V^{-T} + V^T S V^{-T} \\ &= D_1 + S_1. \end{aligned}$$

The matrix D_1 is an upper triangular matrix with the diagonal D as diagonal elements. All the submatrices of the lower triangular part of S_1 have rank at most 1. So, $D_1 + S_1$ can be rewritten as

$$D_1 + S_1 = D + \hat{S},$$

where all the submatrices of the lower triangular part of $\hat{S}$ have rank at most 1. Because of symmetry, also the submatrices of the upper triangular part of $\hat{S}$ have rank at most 1 and hence, $\hat{S}$ is a semiseparable matrix. This finishes the proof that $V^T V = D + \hat{S}$ is again a symmetric DPSS matrix with the same diagonal part as the original matrix A. □

The fact that the Givens transformations used in A and in the Cholesky factor V are the same, simplifies the computation of V. The same is true for the fact that the diagonal part of A is invariant under the LR algorithm. This will be exploited now.

The Cholesky factor of A has the form

$$V = \begin{bmatrix} c_1\widetilde{f}_1 + \widetilde{d}_1 & & & & \\ c_2 s_1 \widetilde{f}_1 & c_2\widetilde{f}_2 + \widetilde{d}_2 & & & \\ \vdots & & \ddots & & \\ c_{n-1}s_{n-2:1}\widetilde{f}_1 & c_{n-1}s_{n-2:2}\widetilde{f}_2 & \cdots & c_{n-1}\widetilde{f}_{n-1} + \widetilde{d}_{n-1} & \\ s_{n-1:1}\widetilde{f}_1 & s_{n-1:2}\widetilde{f}_2 & \cdots & s_{n-1}\widetilde{f}_{n-1} & \widetilde{f}_n + \widetilde{d}_n \end{bmatrix},$$

where $s_{a:b} = s_a s_{a-1} \cdots s_b$.

By comparing the elements of A and VV^T we get equations for the vectors $\widetilde{f}$ and $\widetilde{d}$. As we know all Givens rotations, it is enough to compare the elements on the diagonal and the main subdiagonal. Hence, we get the following equations

$$c_k f_k + d_k = \sum_{j=1}^{k-1} (c_k s_{k-1} \cdots s_j \widetilde{f}_j)^2 + (c_k \widetilde{f}_k + \widetilde{d}_k)^2, \qquad k = 1, \ldots, n, \tag{3.1}$$

$$\begin{aligned} c_{k+1} s_k f_k = {} & \sum_{j=1}^{k-1} c_k c_{k+1} s_k (s_{k-1} \cdots s_j \widetilde{f}_j)^2 \\ & + c_{k+1} s_k \widetilde{f}_k (c_k \widetilde{f}_k + \widetilde{d}_k), \qquad k = 1, \ldots, n-1, \end{aligned} \tag{3.2}$$

where we assume that $c_n = 1$. If we denote

$$q_k := \sum_{j=1}^{k-1} (s_{k-1} s_{k-2} \cdots s_j \widetilde{f}_j)^2,$$

then we can write (3.1) and (3.2) as

$$c_k f_k + d_k = c_k^2 q_k + (c_k \widetilde{f}_k + \widetilde{d}_k)^2, \qquad k = 1, \ldots, n, \tag{3.3}$$

$$c_{k+1} s_k f_k = c_k c_{k+1} s_k q_k + c_{k+1} s_k \widetilde{f}_k (c_k \widetilde{f}_k + \widetilde{d}_k), \qquad k = 1, \ldots, n-1. \tag{3.4}$$

The solution of (3.3) and (3.4) for $\widetilde{f}_k$ and $\widetilde{d}_k$ is

$$\widetilde{f}_k = \frac{f_k - c_k q_k}{\sqrt{d_k + c_k(f_k - c_k q_k)}}, \qquad k = 1, \ldots, n, \tag{3.5}$$

$$\widetilde{d}_k = \frac{d_k}{\sqrt{d_k + c_k(f_k - c_k q_k)}}, \qquad k = 1, \ldots, n, \tag{3.6}$$

where we assume that $c_n = 1$ and $q_1 = 0$.

For later use, let us define the common factors in the numerator and the denominator of (3.5) and (3.6) as follows:

$$\begin{aligned} z_k &= f_k - c_k q_k, \\ y_k &= \sqrt{d_k + c_k z_k}. \end{aligned}$$

One can see from (3.1) that y_k is in fact the diagonal element of V because

$$c_k \widetilde{f}_k + \widetilde{d}_k = \sqrt{d_k + c_k(f_k - c_k q_k)} = \sqrt{d_k + c_k z_k} = y_k. \tag{3.7}$$

As in the standard Cholesky algorithm, a negative or zero value under the square root appears if A is not positive definite, so this is a way to check whether A is positive definite or not.

Let us remark that $\widetilde{f}_n$ and $\widetilde{d}_n$ are not uniquely determined. We choose the values (3.5) and (3.6) because of consistency.

From the above equations we can obtain an algorithm that computes the Cholesky factorization of a s.p.d. DPSS matrix in $11n + \mathscr{O}(1)$ flops.

ALGORITHM 3.2. *An algorithm for the Cholesky decomposition $VV^T = A$ of a s.p.d. DPSS matrix $A = \mathrm{Giv}(c,s,f) + \mathrm{diag}(d)$. The result are vectors $\widetilde{f}$ and $\widetilde{d}$ such that $V = \mathrm{tril}(\mathrm{Giv}(c,s,\widetilde{f})) + \mathrm{diag}(\widetilde{d})$. In the algorithm we assume that $c_n = 1$.*

```
function [$\widetilde{f}$, $\widetilde{d}$] = Cholesky($c$, $s$, $f$, $d$)
$c_n = 1$
$q_1 = 0$
for $k = 1,\dots,n$ :
    $z_k = f_k - c_k \cdot q_k$
    $y_k = \sqrt{d_k + c_k \cdot z_k}$
    $\widetilde{f}_k = z_k / y_k$
    $\widetilde{d}_k = d_k / y_k$
    $q_{k+1} = s_k^2(q_k + \widetilde{f}_k^2)$
```

Next we study how to construct the product $V^T V$ in an efficient way. The product $B = V^T V$ is again a s.p.d. DPSS matrix. A short calculation shows that the diagonal and subdiagonal elements of B are equal to

$$b_{kk} = (c_k \widetilde{f}_k + \widetilde{d}_k)^2 + (s_k \widetilde{f}_k)^2, \tag{3.8}$$

$$b_{jk} = s_k s_{k+1} \cdots s_{j-1} \widetilde{f}_k (\widetilde{f}_j + c_j \widetilde{d}_j), \tag{3.9}$$

where $k = 1,\dots,n$, $j > k$, and we assume that $c_n = 1$ which implies that $s_n = 0$. Let us denote $B = \mathrm{Giv}(\widehat{c},\widehat{s},\widehat{f}) + \mathrm{diag}(d)$. From the equality

$$\widehat{s}_k^2 \widehat{f}_k^2 = \sum_{j=k+1}^{n} b_{jk}^2$$

and (3.9) it follows that

$$\widehat{s}_k^2 \widehat{f}_k^2 = \widetilde{f}_k^2 p_k, \qquad k = 1,\dots,n-1 \tag{3.10}$$

where

$$p_k = \sum_{j=k+1}^{n} (s_k s_{k+1} \cdots s_{j-1})^2 (\widetilde{f}_j + c_j \widetilde{d}_j)^2.$$

For p_k, $k = n-1, \dots, 1$, we can apply the recursion

$$p_k = s_k^2 \left(p_{k+1} + (\widetilde{f}_{k+1} + c_{k+1} \widetilde{d}_{k+1})^2 \right)$$

that starts with $p_n = 0$.

From (3.8) we obtain

$$\widehat{c}_k \widehat{f}_k = (c_k \widetilde{f}_k + \widetilde{d}_k)^2 + (s_k \widetilde{f}_k)^2 - d_k. \tag{3.11}$$

By applying the relation (3.7) we simplify (3.11) into

$$\widehat{c}_k \widehat{f}_k = c_k z_k + (s_k \widetilde{f}_k)^2 \tag{3.12}$$

and reduce the possibility of cancellation. From (3.10) and (3.12) we can compute the vectors $\widehat{c}, \widehat{s}$, and $\widehat{f}$.

ALGORITHM 3.3. *An algorithm for the product $B = V^T V$, where $V = \text{tril}(\text{Giv}(c,s,\widetilde{f})) + \text{diag}(\widetilde{d})$ is the lower triangular Cholesky factor of a s.p.d. DPSS matrix $A = \text{Giv}(c,s,f) + \text{diag}(d)$. The vector z was already computed in Algorithm 3.2. The result are vectors $\widehat{c}, \widehat{s}$, and $\widehat{f}$ such that $B = \text{Giv}(\widehat{c}, \widehat{s}, \widehat{f}) + \text{diag}(d)$.*

```
function [$\widehat{c}$, $\widehat{s}$, $\widehat{f}$] = VTV($c, s, \widetilde{f}, z$)
$c_n = 1$
$\widehat{f}_n = (\widetilde{f}_n + \widetilde{d}_n)^2 - d_n$
$p_n = 0$
for $k = n-1, \dots, 2, 1$
    $p_k = s_k^2 \left( p_{k+1} + (\widetilde{f}_{k+1} + c_{k+1}\widetilde{d}_{k+1})^2 \right)$
    [$\widehat{c}_k$, $\widehat{s}_k$, $\widehat{f}_k$] = Givens($c_k z_k + s_k^2 \widetilde{f}_k^2, \widetilde{f}_k \sqrt{p_k}$)
```

The function $[c,s,f] = \text{Givens}(x,y)$ in Algorithm 3.3 returns the Givens transformation such that

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix}.$$

A stable implementation that guards against overflow requires 7 flops (see, for example, [5]). Note that some quantities such as $\widetilde{f}_k^2$ and s_k^2 already appear in Algorithm 3.2, so we have to compute them only once. As a result an efficient implementation of Algorithm 3.3 requires $16n + \mathcal{O}(1)$ flops and one step of the Cholesky LR algorithm without shifts can be performed in $27n + \mathcal{O}(1)$ flops.

Let us remark that in Algorithm 3.3 we do not care about the sign of $\widehat{s}_k$ as the eigenvalues are invariant to the sign of $\widehat{s}_k$, $k = 1, \dots, n-1$.

4. Computation of Laguerre's shift. As indicated in (2.1), for Laguerre's shift we need to compute S_1 and S_2. It is easy to see that

$$S_1(\sigma) = \sum_{i=1}^{n} \frac{1}{\lambda_i - \sigma} = \mathrm{Tr}((A - \sigma I)^{-1})$$

and

$$S_2(\sigma) = \sum_{i=1}^{n} \frac{1}{(\lambda_i - \sigma)^2} = \mathrm{Tr}((A - \sigma I)^{-2}).$$

So, if $A - \sigma I = VV^T$ is the Cholesky decomposition of the s.p.d. DPSS matrix $A - \sigma I$ and $W = V^{-1}$, then

$$S_1(\sigma) = \mathrm{Tr}(W^T W) = \|W\|_F^2$$

and

$$S_2(\sigma) = \mathrm{Tr}(W^T W W^T W) = \mathrm{Tr}(W W^T W W^T) = \|W W^T\|_F^2.$$

The aim is to compute S_1 and S_2 in a stable and efficient way.

Let us assume that $W = \mathrm{tril}(\mathrm{Giv}(\bar{c}, \bar{s}, \bar{f})) + \mathrm{diag}(\bar{d})$. We will later show that the algorithm derived under the above assumption is correct also when W is not DPSS. One can check that W is not DPSS when $\widetilde{d_i} = 0$ for some $i = 2, \ldots, n-1$.

In the next lemmas and remark, we will show that S_1 and S_2 can be computed in an efficient way.

LEMMA 4.1. *If* $A = \mathrm{Giv}(c, s, f) + \mathrm{diag}(d)$ *is a symmetric* $n \times n$ *diagonal-plus-semiseparable matrix then*

$$\|A\|_F^2 = \sum_{k=1}^{n} (c_k f_k + d_k)^2 + 2 \sum_{k=1}^{n-1} s_k^2 f_k^2,$$

where we assume that $c_n = 1$.

Proof. As A is symmetric,

$$\|A\|_F^2 = \sum_{k=1}^{n} a_{kk}^2 + 2 \sum_{k=1}^{n-1} \sum_{j=k+1}^{n} a_{jk}^2.$$

If follows from the structure of A that $a_{kk} = c_k f_k + d_k$ and

$$\sum_{j=k+1}^{n} a_{jk}^2 = s_k^2 f_k^2.$$

□

Based on Lemma 4.1, we can derive the following expressions for S_1 and S_2:

LEMMA 4.2. *If* $W = \mathrm{tril}(\mathrm{Giv}(\bar{c}, \bar{s}, \bar{f})) + \mathrm{diag}(\bar{d})$ *is a lower nonsingular triangular matrix, such that* $\bar{c}_k \neq 0$ *for* $k = 2, \ldots, n-1$*, then*

$$\|W W^T\|_F^2 = \sum_{k=1}^{n} (W W^T)_{kk}^2 + 2 \sum_{k=1}^{n-1} \left(\frac{(W W^T)_{k+1,k}}{\bar{c}_{k+1}} \right)^2, \tag{4.1}$$

$$\|W\|_F^2 = \sum_{k=1}^{n} (WW^T)_{kk}, \tag{4.2}$$

where we assume that $\bar{c}_n = 1$.

Proof. WW^T is a s.p.d. DPSS matrix. As a consequence of point 1. of Theorem 3.1, the Givens transformations of the representation of W are preserved in the product WW^T. Hence, there exist two vectors $x, y \in \mathbb{R}^n$ such that $WW^T = \mathrm{Giv}(\bar{c}, \bar{s}, x) + \mathrm{diag}(y)$. Applying Lemma 4.1 and the relations

$$\begin{aligned} \bar{s}_k x_k &= \frac{(WW^T)_{k+1,k}}{\bar{c}_{k+1}} \quad &&\text{for} \quad k = 1, \dots, n-1, \\ \bar{c}_k x_k + y_k &= (WW^T)_{k,k} \quad &&\text{for} \quad k = 1, \dots, n \end{aligned}$$

finishes the proof. □

REMARK 4.3. *The formula (4.1) of Lemma 4.2 can be generalized such that the condition $\bar{c}_k \neq 0$ for $k = 2, \dots, n-1$ is no longer required. If we denote by $t(k)$ the smallest index j, $j > k$, such that $\bar{c}_j \neq 0$, then*

$$\|WW^T\|_F^2 = \sum_{k=1}^{n} (WW^T)_{kk}^2 + 2 \sum_{k=1}^{n-1} \left(\frac{(WW^T)_{t(k),k}}{\bar{c}_{t(k)}} \right)^2. \tag{4.3}$$

Since $\bar{c}_n = 1$, we always have $k < t(k) \leq n$ and (4.3) is well defined.

In addition to $\widetilde{d}_i \neq 0$ for $k = 1, \dots, n$, such that W is a DPSS, let us assume from now on also that $c_k \neq 0$ for $k = 2, \dots, n-1$ in the Cholesky factor V. Under this assumptions it follows from Lemma 4.2 that only the Givens transformations of W and the diagonal and subdiagonal elements of WW^T are required for computing S_1 and S_2.

One can check that

$$(WW^T)_{kk} = \bar{c}_k^2 \sum_{i=1}^{k-1} \left(\bar{s}_{k-1} \cdots \bar{s}_i \bar{f}_i \right)^2 + (\bar{c}_k \bar{f}_k + \bar{d}_k)^2$$

and

$$(WW^T)_{k+1,k} = \bar{c}_{k+1} \bar{c}_k \bar{s}_k \sum_{i=1}^{k-1} \left(\bar{s}_{k-1} \cdots \bar{s}_i \bar{f}_i \right)^2 + \bar{c}_{k+1} \bar{s}_k \bar{f}_k (\bar{c}_k \bar{f}_k + \bar{d}_k).$$

Because V is a lower triangular matrix and $W = V^{-1}$, the diagonal and subdiagonal elements of W are of the form:

$$w_{kk} = \bar{c}_k \bar{f}_k + \bar{d}_k = y_k^{-1}, \qquad k = 1, \dots, n, \tag{4.4}$$

$$w_{k+1,k} = \bar{c}_{k+1} \bar{s}_k \bar{f}_k = -\frac{c_{k+1} s_k \widetilde{f}_k}{y_k y_{k+1}}, \qquad k = 1, \dots, n-1, \tag{4.5}$$

where $y_k = c_k \widetilde{f}_k + \widetilde{d}_k$ is the diagonal element of V computed in Algorithm 3.2.

If we define $r_k = \sum_{i=1}^{k-1} \left(\bar{s}_{k-1} \cdots \bar{s}_i \bar{f}_i \right)^2$ then we can write

$$(WW^T)_{kk} = \bar{c}_k^2 r_k + y_k^{-2}$$

and

$$\frac{(WW^T)_{k+1,k}}{\bar{c}_{k+1}} = \bar{c}_k\bar{s}_k r_k + \frac{\bar{s}_k\bar{f}_k}{y_k}.$$

For r_k, $k = 1,\dots,n$, we use the recursion $r_{k+1} = \bar{s}_k^2 r_k + \bar{s}_k^2\bar{f}_k^2$ that starts with $r_1 = 0$.

From the relations (4.4) and (4.5) it follows that in order to compute the diagonal and the subdiagonal elements of WW^T, it is enough to know the Givens rotations and the diagonal and the subdiagonal elements of W.

The following lemma, which follows from the results in [3], helps us to compute the necessary elements of W.

LEMMA 4.4. *Let $V = \mathrm{tril}(\mathrm{Giv}(c,s,\widetilde{f})) + \mathrm{diag}(\widetilde{d})$ be a nonsingular lower triangular matrix such that $\tilde{d}_i \neq 0$ for $i = 1,\dots,n$. Then $W = V^{-1}$ can be represented in the Givens-vector representation as $W = \mathrm{tril}(\mathrm{Giv}(\bar{c},\bar{s},\bar{f})) + \mathrm{diag}(\bar{d})$, where $\bar{d}_i = \widetilde{d}_i^{-1}$ for $i = 1,\dots,n$.*

Hence, the diagonal elements of W can be written as

$$w_{kk} = \bar{c}_k\bar{f}_k + \widetilde{d}_k^{-1} = y_k^{-1}, \qquad k = 1,\dots,n. \tag{4.6}$$

If we rearrange the equations (4.5) and (4.6) into

$$\bar{c}_k\bar{f}_k = -\frac{c_k\widetilde{f}_k}{\widetilde{d}_k y_k}.$$

and

$$\bar{s}_k\bar{f}_k = -\frac{c_{k+1}s_k\widetilde{f}_k}{\bar{c}_{k+1}y_k y_{k+1}}, \tag{4.7}$$

then it follows that $\bar{c}_k$ and $\bar{s}_k$ form a Givens transformation such that

$$\begin{bmatrix} \bar{c}_k & \bar{s}_k \\ -\bar{s}_k & \bar{c}_k \end{bmatrix} \begin{bmatrix} c_k y_{k+1}\bar{c}_{k+1} \\ c_{k+1}s_k\widetilde{d}_k \end{bmatrix} = \begin{bmatrix} * \\ 0 \end{bmatrix}.$$

Again, for $k = n-1$ we assume that $c_n = \bar{c}_n = 1$. One can see by induction that $\bar{c}_k \neq 0$ for $k = n-1,\dots,2$ because we assumed that $c_k \neq 0$ for $k = 2,\dots,n-1$ and $y_{k+1} = 0$ would contradict the fact that A is s.p.d.

Now we can write an algorithm for the computation of $\|WW^T\|_F^2$ and $\|W\|_F^2$. In the algorihtm ξ_k denotes $(WW^T)_{k+1,k}/\bar{c}_{k+1}$ and ω_k denotes the diagonal element $(WW^T)_{kk}$. These are the values that appear in equations (4.1) and (4.2) for S_1 and S_2. We use β_k for the intermediate result (4.7). A careful implementation of the algorithm, where the values that appear in Algorithms 3.2 and 3.3 are computed only once, requires $31n + \mathcal{O}(1)$ flops.

ALGORITHM 4.5. *An algorithm that computes $S_1 = \|W\|_F^2$ and $S_2 = \|WW^T\|_F^2$, where $W = V^{-1}$ and $V = \mathrm{tril}(\mathrm{Giv}(c,s,\widetilde{f})) + \mathrm{diag}(\widetilde{d})$ is the Cholesky factor of a s.p.d. DPSS matrix $A = \mathrm{Giv}(c,s,f) + \mathrm{diag}(d)$, and $y = \mathrm{diag}(V)$. In the algorithm we assume $c_k \neq 0$ for $k = 2,\dots,n-1$ and $c_n = \bar{c}_n = 1$.*

```
function [$S_1$, $S_2$] = invtrace($c, s, \widetilde{f}, \widetilde{d}, y$)
$c_n = \bar{c}_n = 1$
for $k = n-1,\dots,2,1$ :
    [$\bar{c}_k$, $\bar{s}_k$] = Givens($c_k\bar{c}_{k+1}y_{k+1}, c_{k+1}s_k\widetilde{d}_k$)
```

```
$r_1 = 0$
for $k = 1,\dots,n-1$
    $\beta_k = -c_{k+1}s_k\widetilde{f}_k/(\bar{c}_{k+1}y_ky_{k+1})$
    $\omega_k = \bar{c}_k^{\,2}\, r_k + y_k^{-2}$
    $\xi_k = \bar{c}_k\bar{s}_kr_k + \beta_k/y_k$
    $r_{k+1} = \bar{s}_k^2 r_k + \beta_k^2$
$\omega_n = r_n + y_n^{-2}$
$S_1 = \sum_{k=1}^n \omega_k$
$S_2 = \sum_{k=1}^n \omega_k^2 + 2\sum_{k=1}^{n-1}\xi_k^2$
```

What remains to be considered is the case that W is not a DPSS matrix. If $\widetilde{d}_k = 0$ for some $k = 2,\dots,n-1$ then W has a zero block $W(k+1:n,1:k-1)$, see, e.g., [7, Lemma 2.5] and it is not a DPSS matrix anymore. However, Algorithm 4.5, that was derived under the assumption that W is a DPSS matrix, returns correct values for $\|W\|_F^2$ and $\|WW^T\|_F^2$ in such case as well. There are no divisions by $\widetilde{d}_k$ in the algorithm that could cause problems. We only use $\widetilde{d}_k$ to compute y_k. If we change $\widetilde{d}_k$ in V then one can see that as long as V is nonsingular, $\|W\|_F^2$ and $\|WW^T\|_F^2$ are continuous functions of $\widetilde{d}_k$. So, the algorithm is correct also in the limit when $\widetilde{d}_k = 0$.

Another restriction in Algorithm 4.5 is the assumption $c_k \neq 0$ for $k = 2,\dots,n-1$. When this assumption is not valid, we can still compute S_1 and S_2 if we apply formula (4.3) from Remark 4.3. One can see that in the kth column of W we need the elements w_{kk} and $w_{t(k),k}$. Because $w_{jk} = 0$ for $k < j < t(k)$, Laguerre's shift can still be computed in $\mathcal{O}(n)$ flops.

5. Implementation. In this section we discuss some details on the implementation of the algorithm presented in the previous sections. The software can be downloaded freely at: *http://www-lp.fmf.uni-lj.si/plestenjak/papers.htm*.

First we discuss how to deflate. If $|s_k|$ is small enough for some $k = 1,\dots,n-1$, then we decouple the problem into two smaller problems with matrices $A(1:k,1:k)$ and $A(k+1:n,k+1:n)$. In the special case when $|s_{n-1}|$ is small enough, we take $f_n + d_n$ as an approximation of an eigenvalue of A and continue with vectors $c(1:n-2)$, $s(1:n-2)$, $f(1:n-1)$, and $d(1:n-1)$. As initial shift for the smaller problem we take $f_n + d_n$.

Another important problem that can appear during the implementation is the shift. If a shift in the QR algorithm is by chance an exact eigenvalue then we can immediately extract this eigenvalue and continue with the smaller problem. This is not true in the Cholesky LR algorithm where shifts σ_k have to be strictly below the smallest eigenvalue λ_n, otherwise the Cholesky factorization does not exist. Without the Cholesky factorization we can not compute $A_{k+1} = V_k^{-1}A_kV_k$ and deflate. In numerical computations, even when $\sigma_k < \lambda_n$, the Cholesky factorization can fail if the difference is too small. This can cause a problem as usually Laguerre's shifts converge faster to the smallest eigenvalue than the elements $A_k(n,n)$. A good strategy is to insert a factor τ close, but smaller, to 1 into (2.1) and use

$$\sigma_{k+1} = \sigma_k + \tau\frac{n}{S_1(\sigma_k) + \sqrt{(n-1)(nS_2(\sigma_k) - S_1^2(\sigma_k))}}$$

as a shift in the new iteration. Based on our numerical experiments we suggest the value $\tau = 1 - 10^{-4}$. If it happens anyway that the shift is so large that the Cholesky factorization fails, we first reduce the shift by the factor $\tau = 1 - 10^{-4}$ and if the new shift is still too large, we start again with the shift 0.

The computation of Laguerre's shift requires more than half of the operations in one step of the Cholesky LR algorithm. We can save work by using the same shift once the shift improvement is small enough. Our numerical experiments show a speed up up to 15% if we stop improving the shift after $(\sigma_{k+1}-\sigma_k)/\sigma_{k+1} \leq 10^{-6}$.

The eigenvalues should be computed from the smallest to the largest one, however, it might happen that $|s_{n-1}|$ is so small that we deflate, and the extracted eigenvalue is not the smallest one. This causes a problem in the next phase as we use the extracted eigenvalue as initial shift and this shift is too large. The strategy from the previous paragraph overcomes this problem and the shift goes to zero after two unsuccessful Cholesky factorizations.

At the end of §4 we proposed a modification of Algorithm 4.5 that handles the case $c_k = 0$ for some $k = 2,\ldots,n-1$. Without this modification we get zero divided by zero in such a situation. In practice we can implement a simpler solution. If we perturb c_k into 10^{-20} whenever $c_k = 0$ then a small c_k results in a small $\bar{c}_k$. These two quantities avoid the zero divided by zero problem in Algorithm 4.5 and we end up with accurate results.

6. Numerical results. The following numerical results were obtained with Matlab 7.0 running on a Pentium4 2.6 GHz Windows XP operating system. We compared the Cholesky LR algorithm with a Matlab implementation of the implicit QR algorithm for DPSS matrices [10] and with the Matlab function `eig`. Exact eigenvalues were computed in Mathematica 5 using variable precision. For all numerical examples in this section the cutoff criterion for both Cholesky LR and implicit QR is 10^{-16}. With the maximum relative error we denote $\max_{1\leq i\leq n} \frac{|\lambda_i-\widetilde{\lambda}_i|}{|\lambda_i|}$ where λ_i, $i = 1,\ldots,n$, are the exact eigenvalues of the test matrix and $\widetilde{\lambda}_i$, $i = 1,\ldots,n$, the computed ones.

EXAMPLE 6.1. In our first example we use random s.p.d. DPSS matrices of the form

$$A = \mathrm{diag}(1,\ldots,n) + \mathrm{triu}(uv^T,1) + \mathrm{triu}(uv^T,1)^T + \alpha I,$$

where u and v are vectors of uniformly distributed random entries on $[0,1]$, obtained by the Matlab function `rand`, and the shift α is such that the smallest eigenvalue of A is 1. The condition numbers of these matrices are approximately n. The exact eigenvalues of A are computed in Mathematica using variable precision. Before using `eig` we compute all the elements of A accurately in double precision, so that the initial data for all three methods are of full precision. The comparison is not completely fair as in `eig` we first have to reduce the matrix to the tridiagonal form where additional numerical errors could occur.

The results in Table 6.1 show that the Cholesky LR method is competitive in accuracy with the other two methods. In most cases, especially for larger matrices, it is slightly more accurate than the implicit QR method. The comparison with `eig` shows that by exploiting the structure we can get more accurate results. In `eig` some accuracy is lost in the reduction to the tridiagonal form. One step of the Cholesky LR method has approximately the same complexity as one step of the implicit QR method, but although Cholesky LR requires roughly 3.5 times more steps than the implicit QR method, it runs much faster. This is due to a more efficient Matlab implementation. The same holds for `eig` which runs faster than Cholesky LR although it has $\mathscr{O}(n^3)$ complexity while the complexity of Cholesky LR is $\mathscr{O}(n^2)$. The difference in number of steps is also due to the fact that in implicit QR we may choose the shift more freely as in Cholesky LR, where the shifted matrix must remain positive definite.

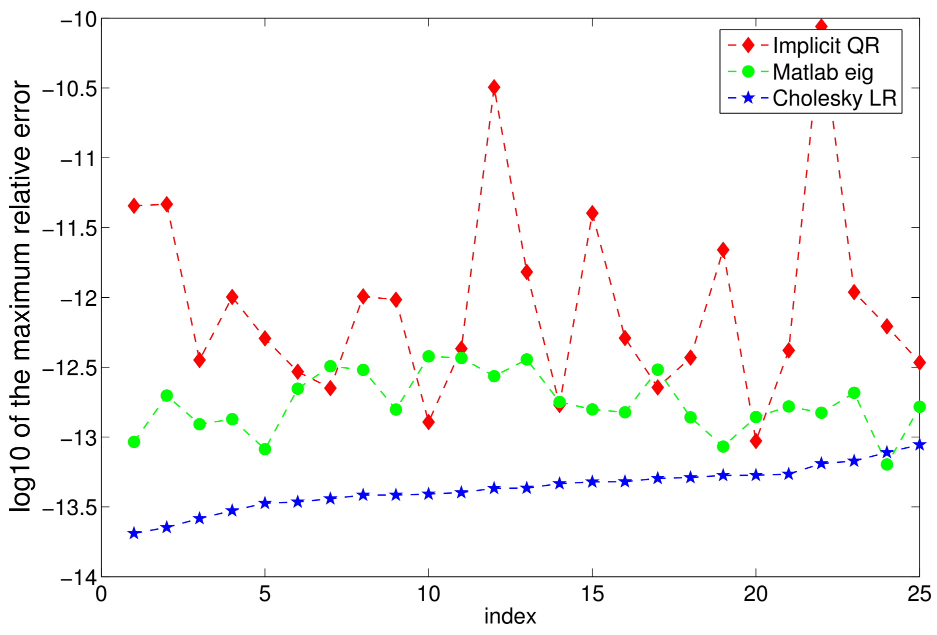
EXAMPLE 6.2. We use the same construction of the test matrices as in Example 6.1. For $n =$ 200 we generate 25 random matrices and compare the accuracy of the eigenvalues computed by the

TABLE 6.1
Comparison of the Cholesky LR method, implicit QR for DPSS matrices, and `eig` *from Matlab on random s.p.d. DPSS matrices of sizes $n = 50$ to $n = 500$ and small condition numbers. The columns are: t: running time in seconds; steps: number of LR (QR) steps; error: the maximum relative error of the computed eigenvalues.*

	Cholesky LR			Implicit QR			`eig`	
n	t	steps	error	t	steps	error	t	error
50	0.06	274	$9.2 \cdot 10^{-15}$	0.19	83	$4.3 \cdot 10^{-15}$	0.00	$1.8 \cdot 10^{-14}$
100	0.16	557	$1.0 \cdot 10^{-14}$	0.69	164	$4.8 \cdot 10^{-14}$	0.00	$1.0 \cdot 10^{-13}$
150	0.28	832	$1.8 \cdot 10^{-14}$	1.45	242	$2.9 \cdot 10^{-13}$	0.00	$1.4 \cdot 10^{-13}$
200	0.49	1104	$2.6 \cdot 10^{-14}$	2.59	311	$3.6 \cdot 10^{-13}$	0.02	$1.2 \cdot 10^{-13}$
250	0.72	1390	$6.4 \cdot 10^{-14}$	4.22	414	$6.7 \cdot 10^{-13}$	0.03	$7.6 \cdot 10^{-13}$
300	0.97	1660	$1.3 \cdot 10^{-13}$	6.18	486	$4.7 \cdot 10^{-13}$	0.05	$1.2 \cdot 10^{-13}$
350	1.25	1933	$4.8 \cdot 10^{-14}$	8.86	564	$1.5 \cdot 10^{-12}$	0.09	$5.6 \cdot 10^{-13}$
400	1.59	2194	$1.3 \cdot 10^{-13}$	11.95	684	$4.3 \cdot 10^{-12}$	0.14	$7.8 \cdot 10^{-13}$
450	1.94	2479	$9.8 \cdot 10^{-14}$	15.78	730	$2.2 \cdot 10^{-12}$	0.22	$6.8 \cdot 10^{-13}$
500	2.34	2741	$1.0 \cdot 10^{-13}$	19.72	821	$4.5 \cdot 10^{-12}$	0.28	$3.8 \cdot 10^{-13}$

Cholesky LR method, implicit QR for DPSS matrices, and `eig`. Again, the exact eigenvalues of A are computed in Mathematica using variable precision.

FIG. 6.1. *Comparison of the Cholesky LR method, implicit QR for s.p.d. DPSS matrices, and* `eig` *from Matlab on 25 random s.p.d. matrices of size $n = 200$.*



Results, ordered by the maximum relative error of the Cholesky LR method, are shown in Figure 6.1. We can see that the most accurate method for this particular class of matrices is the Cholesky

LR algorithm. The results from `eig` are comparable while the results of the implicit QR are slightly worse in general.

EXAMPLE 6.3. In this example we use s.p.d. matrices $A = Q\,\mathrm{diag}(1:n)Q^T$, where Q is a random orthogonal matrix, obtained in Matlab as `orth(randn(n))`. As in the previous examples we compare the Cholesky LR method, implicit QR for DPSS matrices, and `eig` from Matlab. The difference from the previous examples is that now we have to reduce the matrix into a similar DPSS matrix before we can apply Cholesky LR or implicit QR. We do this using the algorithm of [14], where we choose the diagonal elements as random numbers distributed uniformly on $[0,1]$. There is a connection between the Lanczos method and the reduction into a similar DPSS matrix [11] which causes that the largest eigenvalues of A are approximated by the lower right diagonal elements of the DPSS matrix. This is not good for the Cholesky LR method where the smallest eigenvalues are computed first. Therefore, we apply a method that reverses the direction of the columns and rows of the DPSS matrix in linear time [12, Chapter 2, § 8.1].

TABLE 6.2

Comparison of the Cholesky LR method, implicit QR for DPSS matrices, and `eig` *from Matlab on random s.p.d. matrices of sizes* $n = 500$ *to* $n = 2000$ *with the exact eigenvalues* $1,\dots,n$. *The columns are: t: running time in seconds (time for LR and QR does not include reduction into a DPSS matrix); steps: number of LR (QR) steps; error: the maximum relative error of the computed eigenvalues.*

	Cholesky LR			Implicit QR			`eig`	
n	t	steps	error	t	steps	error	t	error
500	1.7	2738	$3.5\cdot10^{-14}$	22.1	942	$2.9\cdot10^{-13}$	1.8	$2.4\cdot10^{-13}$
1000	9.4	5404	$2.1\cdot10^{-13}$	108.9	1804	$6.6\cdot10^{-13}$	13.4	$6.2\cdot10^{-13}$
1500	19.8	8048	$1.6\cdot10^{-13}$	292.7	2641	$4.6\cdot10^{-13}$	50.9	$4.5\cdot10^{-13}$
2000	33.8	10672	$4.6\cdot10^{-13}$	602.2	3448	$2.8\cdot10^{-12}$	123.0	$3.0\cdot10^{-12}$
2500	52.7	13281	$6.0\cdot10^{-13}$	1064.7	4263	$4.7\cdot10^{-12}$	279.6	$4.1\cdot10^{-12}$

The results in Table 6.2 show that the eigenvalues of a s.p.d. matrix can be computed accurately using a reduction into a DPSS matrix followed by the Cholesky LR method or the implicit QR method. For larger matrices, the Cholesky LR algorithm tends to be slightly more accurate than the implicit QR. Since both methods use the same reduced DPSS matrices, this implies that Cholesky LR is more accurate than implicit QR. The computational times are hard to compare because of different implementations and because the time for `eig` includes the reduction to the tridiagonal form while the reduction to DPSS matrices is excluded from the times of the Cholesky LR and the implicit QR method.

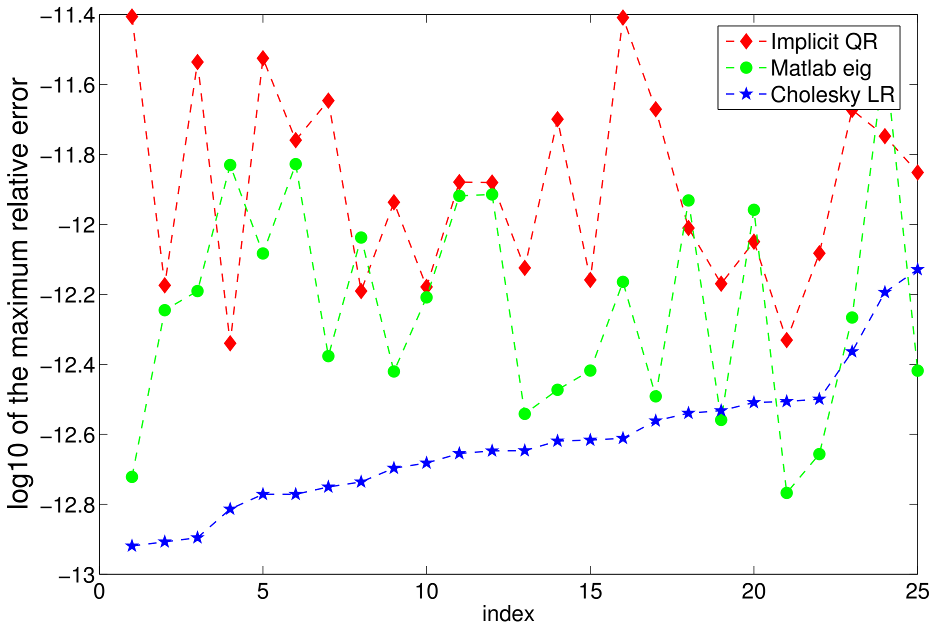
EXAMPLE 6.4. We use the same construction of the test matrices as in Example 6.3. For $n = 1000$ we generate 25 random matrices and compare the accuracy of the eigenvalues computed by the Cholesky LR method, implicit QR for DPSS matrices, and `eig`. For the reduction into a similar DPSS matrix we use the same approach as in Example 6.3.

Results are shown in Figure 6.2. Similar to the previous examples, the Cholesky LR method is comparable with `eig` and usually gives slightly better results than the implicit QR method.

Similar tests on matrices with multiple eigenvalues and with eigenvalues $\lambda_i = 2^i, i = 1,\dots,n$, were performed. The results obtained by the three algorithms also in these cases are comparable.

7. Conclusions. We have presented a version of the Cholesky LR algorithm that exploits the structure of positive definite DPSS matrices. We propose to combine the method with Laguerre's

FIG. 6.2. *Comparison of the Cholesky LR method, implicit QR for DPSS matrices, and* `eig` *from Matlab on 25 random s.p.d. matrices of size* $n = 1000$ *with the exact eigenvalues* $1, \ldots, 1000$.



shifts. It seems natural to compare the method to the implicit QR for DPSS matrices [10]. In Cholesky LR the eigenvalues are computed from the smallest to the largest eigenvalue, therefore the method is very appropriate for applications where one is interested in few of the smallest or the largest eigenvalues. If the complete spectrum is computed, Cholesky LR is more expensive than implicit QR, but, as it tends to be slightly more accurate, it presents an alternative.

The proposed method combined with the reduction to DPSS matrices [14] can also be applied to a general s.p.d. matrix.

REFERENCES

[1] Bini, D.A., Gemignani, L., Pan, V.: QR-like algorithms for generalized semiseparable matrices. Tech. Report 1470, Department of Mathematics, University of Pisa, 2003

[2] Chandrasekaran, S., Gu, M.: A divide and conquer algorithm for the eigendecomposition of symmetric block-diagonal plus semi-separable matrices. Numer. Math. **96**, 723–731 (2004)

[3] Delvaux, S., Van Barel, M.: Structures preserved by matrix inversion. Report TW 414, Department of Computer Science, K.U.Leuven, Leuven, Belgium, December 2004

[4] Fasino, D.: Rational Krylov matrices and QR-steps on Hermitian diagonal-plus-semiseparable matrices. To appear in Numer. Linear Algebra Appl.; also available from ftp://ftp.dimi.uniud.it/pub/fasino/bari.ps

[5] Golub, G.H., Van Loan, C.F.: Matrix Computations, 3rd Edition. The Johns Hopkins University Press, Baltimore, 1996

[6] Grad, J., Zakrajšek, E.: LR algorithm with Laguerre shifts for symmetric tridiagonal matrices. Comput. J. **15**, 268–270 (1972)

[7] Fiedler, M., Vavřín, Z.: Generalized Hessenberg matrices. Linear Algebra Appl. **380**, 95–105 (2004)

[8] Mastronardi, N., Van Camp, E., Van Barel, M.: Divide and conquer type algorithms for computing the eigendecomposition of diagonal plus semiseparable matrices. Technical Report 7 (5/2003), Istituto per le Applicazioni del

Calcolo ”M. Picone”, Consiglio Nazionale delle Ricerche, Rome, Italy, 2003. To appear in Numerical Algorithms.

[9] Parlett, B.N.: The symmetric eigenvalue problem. Classics in Applied Mathematics, Prentice-Hall, Englewood Cliffs, N.J., 1980

[10] Van Camp, E., Delvaux, S., Van Barel, M., Vandebril, R., Mastronardi, N.: An implicit QR-algorithm for symmetric diagonal-plus-semiseparable matrices, Report TW 419, Department of Computer Science, K.U.Leuven, Leuven, Belgium, March 2005.

[11] Van Camp, E., Van Barel, M., Vandebril, R., Mastronardi, N.: Orthogonal similarity transformation of a symmetric matrix into a diagonal-plus-semiseparable one with free choice of the diagonal. Structured Numerical Linear Algebra Problems: Algorithms and Applications, Cortona, Italy, September 19-24, 2004. http://www.dm.unipi.it/∼cortona04/program.htm

[12] Vandebril, R.: Semiseparable matrices and the symmetric eigenvalue problem. PhD, K.U.Leuven, Leuven, May 2004

[13] Vandebril, R., Van Barel, R., Mastronardi, N.: A note on the representation and definition of semiseparable matrices. Report TW 393, Department of Computer Science, K.U.Leuven, Leuven, Belgium, May 2004. To appear in Numer. Linear Algebra Appl.

[14] Vandebril, R., Van Camp, R., Van Barel, M., Mastronardi, N.: Orthogonal similarity transformation of a symmetric matrix into a diagonal-plus-semiseparable one with free choice of the diagonal. Report TW 398, Department of Computer Science, K.U.Leuven, Leuven, Belgium, August 2004

[15] Wilkinson, J.: Algebraic eigenvalue problem. Numerical Mathematics and Scientific Computation, Oxford University Press, Oxford, 1999