University of Maryland College Park

Institute for Advanced Computer Studies UMIACS-TR-94-9.1
Department of Computer Science CS-TR-3211.1

THE ORTHOGONAL QD-ALGORITHM*

Urs von Matt†

January, 1994
revised September, 1994

Abstract. The orthogonal qd-algorithm is presented to compute the singular value decomposition of a bidiagonal matrix. This algorithm represents a modification of Rutishauser's qd-algorithm, and it is capable of determining all the singular values to high relative precision. A generalization of the Givens transformation is also introduced, which has applications besides the orthogonal qd-algorithm.

The shift strategy of the orthogonal qd-algorithm is based on Laguerre's method, which is used to compute a lower bound for the smallest singular value of the bidiagonal matrix. Special attention is devoted to the numerically stable evaluation of this shift.

Key words. Generalized Givens transformation, implicit Cholesky decomposition, Laguerre's method, orthogonal qd-algorithm, singular value decomposition.

AMS subject classifications. 65F20.

* This report is available by anonymous ftp from `cs.umd.edu` in the directory `/pub/papers/TRs`.

† Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742; e-mail: `na.vonmatt@na-net.ornl.gov`.

THE ORTHOGONAL QD-ALGORITHM

Urs von Matt*

Abstract. The orthogonal qd-algorithm is presented to compute the singular value decomposition of a bidiagonal matrix. This algorithm represents a modification of Rutishauser's qd-algorithm, and it is capable of determining all the singular values to high relative precision. A generalization of the Givens transformation is also introduced, which has applications besides the orthogonal qd-algorithm.

The shift strategy of the orthogonal qd-algorithm is based on Laguerre's method, which is used to compute a lower bound for the smallest singular value of the bidiagonal matrix. Special attention is devoted to the numerically stable evaluation of this shift.

Key words. Generalized Givens transformation, implicit Cholesky decomposition, Laguerre's method, orthogonal qd-algorithm, singular value decomposition.

AMS subject classifications. 65F20.

1. Introduction. In 1954 H. Rutishauser [20] introduced the qd-algorithm to compute the eigenvalues of a symmetric tridiagonal matrix. In this paper, we present a related algorithm to compute the singular values of a square bidiagonal matrix. Since all the transformations consist of Givens rotations we call it the orthogonal qd-algorithm.

We impose no restriction in only considering the singular value decomposition of a square bidiagonal matrix. For a general rectangular matrix it is common practice to first reduce it to the bidiagonal form (cf. [6]). In many applications one is also given a bidiagonal matrix right from the beginning. Furthermore, it is also known [2] that all the singular values of a bidiagonal matrix are determined to high relative precision by the entries of the matrix. We will give evidence that our algorithm can actually achieve this high relative accuracy.

An outline of the paper is as follows. In Section 2 we review Rutishauser's progressive qd-algorithm to compute the eigenvalues of a symmetric tridiagonal positive definite matrix. The calculation of the singular value decomposition has attracted a lot of attention recently, and we mention some of this work in Section 3. In Section 4 we show how Rutishauser's progressive qd-step can be expressed by means of orthogonal transformations. For this we have to introduce the generalized Givens transformation in Section 4.1. We discuss how to use Givens transformations to perform a progressive qd-step for triangular matrices (Section 4.2) and for bidiagonal matrices (Section 4.3). In Section 5 we derive the differential qd-algorithm to compute the singular values of a bidiagonal matrix. The basic orthogonal qd-steps are introduced in Sections 6 and 8. In order to compute these qd-steps we need the differential form of the generalized Givens transformation, which is introduced in Section 7. After that we are ready to present the orthogonal qd-algorithm in Section 9. In Sections 10 and 11 we consider deflation and the calculation of the singular vectors. The calculation of Newton's and Laguerre's shift is presented in Section 12. Finally, Section 13 gives some numerical results.

* Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742; e-mail: na.vonmatt@na-net.ornl.gov.

Some important algorithms are presented in pseudo-code, and we choose a notation corresponding to a modern procedural language of the ALGOL-family. This allows us to concentrate on the mathematical properties of the algorithms, while an actual implementation in any language can still be readily derived from our code.

2. Rutishauser's Quotient-Difference Algorithm. Rutishauser used his qd-algorithm in order to compute the eigenvalues of a symmetric tridiagonal positive definite matrix A. He started with the Cholesky decomposition

$$A = R^{\mathrm{T}} R,$$

where

$$R = \begin{bmatrix} \gamma_1 & \delta_1 & & \\ & \ddots & \ddots & \\ & & \ddots & \delta_{n-1} \\ & & & \gamma_n \end{bmatrix}$$

is an n-by-n upper bidiagonal matrix. Afterwards, a sequence of so-called progressive qd-steps is applied to the matrix R.

DEFINITION 2.1. *A progressive qd-step with shift s transforms an n-by-n upper bidiagonal matrix R into another n-by-n upper bidiagonal matrix R'. It is defined by the Cholesky decomposition*

$$RR^{\mathrm{T}} - sI = R'^{\mathrm{T}} R'. \tag{1}$$

This transformation is applicable if and only if $s \le \lambda_{\min}(R^{\mathrm{T}} R)$.

By a progressive qd-step the eigenvalues are shifted by the amount of s, i.e.

$$\lambda_i(R'^{\mathrm{T}} R') = \lambda_i(R^{\mathrm{T}} R) - s.$$

All the eigenvalues of A can be computed by a proper shift strategy. Algorithm 1 presents Rutishauser's progressive qd-algorithm in matrix terms (see also [5, p. 100]).

The performance of the qd-algorithm depends critically on the choice of the shifts s. In [23] Rutishauser shows that, for $s \equiv 0$, the matrices R converge to a diagonal matrix with the square roots of the eigenvalues of A. The convergence is linear and becomes worse when the eigenvalues are clustered. Algorithm 1 is not suited for this shift strategy because, in general, its inner loop would not terminate.

We can obtain a much better rate of convergence by choosing the shifts

$$s = \frac{1}{\operatorname{trace}(R^{\mathrm{T}} R)^{-1}}.$$

In [24, pp. 484–486] it is shown how this shift can be computed easily from the matrix R. Reinsch and Bauer observe in [19] that this shift can be seen as a Newton step for the solution of the characteristic polynomial

$$p(s) = \det(R^{\mathrm{T}} R - sI).$$

ALGORITHM 1. *Progressive qd-Algorithm.*

$t := 0$
Compute the Cholesky decomposition $A = R^{\mathrm{T}}R$.
for $k := n$ **to** 1 **by** -1 **do**
  while $\gamma_k \neq 0$ **do**
    Choose a shift s with $0 \leq s \leq \lambda_{\min}(R^{\mathrm{T}}R)$.
    Execute the progressive qd-step $RR^{\mathrm{T}} - sI = R'^{\mathrm{T}}R'$.
    $t := t + s$
    $R := R'$
  end
  $\lambda_k := t$
  Compute the Cholesky decomposition $RR^{\mathrm{T}} = R'^{\mathrm{T}}R'$.
  $R := (k-1)$-by-$(k-1)$ leading principal submatrix of R'
end

This observation also explains the quadratic convergence of this shift strategy.

In [22] Rutishauser describes a shift strategy that even leads to an asymptotically cubic convergence. He uses trial qd-steps with shifts $s > \lambda_{\min}(R^{\mathrm{T}}R)$. Although these steps must fail one can usually extract enough information to obtain an improved lower bound on $\lambda_{\min}(R^{\mathrm{T}}R)$.

3. Review of Related Work. In 1990 J. Demmel and W. Kahan presented a modified QR-algorithm which computes the smallest singular values to maximal relative accuracy and the others to maximal absolute accuracy [2]. As in the standard QR-algorithm the singular vectors are also available. Their work represents a major improvement over the original SVD-subroutine `svdc` as it has been implemented in the LINPACK linear algebra library [3].

On the other hand K. V. Fernando and B. N. Parlett discovered in 1992 a variant of the qd-algorithm for obtaining maximal relative accuracy for all the singular values [4]. Their approach is based on the so-called differential form of the progressive qd-algorithm. In contrast to the work of Demmel and Kahan their algorithm cannot compute the left and right singular vectors simultaneously with the singular values.

4. Progressive Quotient-Difference Step. The Cholesky decomposition (1) represents the heart of Rutishauser's qd-algorithm. As it turns out this transformation can also be expressed by means of an orthogonal matrix. In order to see this let us first look at a slightly more general problem.

Let L be an n-by-n lower triangular matrix, and let σ denote a nonnegative lower bound for the singular values of L. We consider the problem of computing an upper triangular matrix U such that

$$L^{\mathrm{T}}L - \sigma^2 I = U^{\mathrm{T}}U. \tag{2}$$

The matrix U may be obtained from the Cholesky decomposition of $L^{\mathrm{T}}L - \sigma^2 I$. However, it is also possible to compute U directly from L and σ by means of an orthogonal transformation. If we could find an orthogonal matrix Q such that

$$Q\begin{bmatrix} L \\ 0 \end{bmatrix} = \begin{bmatrix} U \\ \sigma I \end{bmatrix}, \tag{3}$$

$$
\begin{bmatrix}
l_{11} & & & \\
l_{21} & l_{22} & & \\
\vdots & \vdots & \ddots & \\
\vdots & \vdots & \cdots & \ddots \\
0 & & & \\
 & \ddots & & \\
 & & \ddots & \\
 & & & 0
\end{bmatrix}
\longmapsto
\begin{bmatrix}
\times & & & \\
l_{21} & l_{22} & & \\
\vdots & \vdots & \ddots & \\
\vdots & \vdots & \cdots & \ddots \\
\sigma & & & \\
 & 0 & & \\
 & & \ddots & \\
 & & & 0
\end{bmatrix}
\longmapsto
\begin{bmatrix}
\times & \times & & \\
0 & \times & & \\
\vdots & \vdots & \ddots & \\
\vdots & \vdots & \cdots & \ddots \\
\sigma & & & \\
 & 0 & & \\
 & & \ddots & \\
 & & & 0
\end{bmatrix}
\longmapsto
$$

$$
\cdots \longmapsto
\begin{bmatrix}
\times & \times & \cdots & \times \\
0 & \times & & \\
\vdots & \vdots & \ddots & \\
0 & \times & \cdots & \times \\
\sigma & & & \\
 & 0 & & \\
 & & \ddots & \\
 & & & 0
\end{bmatrix}
=
\begin{bmatrix}
u & & \mathbf{v}^{\mathrm{T}} & \\
 & & & \\
 & & L' & \\
 & & & \\
\sigma & & & \\
 & 0 & & \\
 & & \ddots & \\
 & & & 0
\end{bmatrix}
$$

FIG. 1. *Stage in the Implicit Cholesky Decomposition.*

we would have solved our problem, since (3) implies $L^{\mathrm{T}}L = U^{\mathrm{T}}U + \sigma^2 I$, which is equivalent to (2). We call this approach the *implicit Cholesky decomposition.*

We can obtain the decomposition (3) by a properly chosen sequence of Givens transformations. In Figure 1 one stage of this decomposition is depicted which reduces the dimension of the problem by one. By applying this step n times we can obtain the desired decomposition (3).

The nontrivial part of this stage consists in the first Givens transformation. Instead of zeroing the entry l_{11}, it introduces the value σ. Obviously, we cannot use an ordinary Givens transformation for this purpose. Rather, we have to introduce a generalization of the Givens transformation.

4.1. Generalized Givens Transformation. Usually the Givens transformation

$$G := \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \tag{4}$$

with $c^2 + s^2 = 1$ is determined such that

$$G \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}.$$

The matrix G is therefore used to selectively annihilate elements in a vector or a matrix. But it is also possible to introduce another value σ different from zero:

$$G \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} r \\ \sigma \end{bmatrix}. \tag{5}$$

ALGORITHM 2. *Generalized Givens Transformation* (`rotg2`).

scale := $\max(|x_1|, |x_2|)$
if scale = 0 **then**
    $c := 1$
    $s := 0$
else
    $x_1 := x_1/\text{scale}$
    $x_2 := x_2/\text{scale}$
    sig := σ/scale
    norm2 := $x_1^2 + x_2^2$
    $r := \sqrt{\text{norm2} - \text{sig}^2}$
    $c := (x_1 \cdot r + x_2 \cdot \text{sig})/\text{norm2}$
    $s := (x_2 \cdot r - x_1 \cdot \text{sig})/\text{norm2}$
    $x_1 := \text{scale} \cdot r$
    $x_2 := \sigma$
end

The value of r is given by

$$r := \pm\sqrt{x_1^2 + x_2^2 - \sigma^2}.$$

The choice of the sign is of no concern. One may opt for a positive r, or it may be more useful to let r have the same sign as x_1. Obviously, the transformation (5) is only possible if $|\sigma| \leq \sqrt{x_1^2 + x_2^2}$.

We now demonstrate how to compute the quantities c and s in G. It is easily verified that

$$\begin{bmatrix} c \\ s \end{bmatrix} = \frac{1}{x_1^2 + x_2^2} \begin{bmatrix} x_1 & x_2 \\ x_2 & -x_1 \end{bmatrix} \begin{bmatrix} r \\ \sigma \end{bmatrix}.$$

An implementation which avoids overflow is presented as Algorithm 2.

We intend to present a detailed error analysis in a future paper. Let us just mention that the computed matrix G is orthogonal up to a small multiple of the machine precision, and the equation (5) is also satisfied to high accuracy for the computed quantities.

4.2. Implicit Cholesky Decomposition. So far, we have only described one stage of the implicit Cholesky decomposition in Figure 1. By applying this step n times, we can compute the decomposition (3). This procedure is presented in pseudo-code as Algorithm 3. We describe the construction and application of ordinary Givens rotations by calls of the BLAS routines `rotg` and `rot`. Their precise definition is given in [3, 11]. We also postulate the procedure `rotg2` which calculates a generalized Givens transformation according to Algorithm 2. More specifically, the call `rotg2` (x_1, x_2, σ, c, s) determines the values of c and s such that equation (5) holds. The parameters x_1 and x_2 are overwritten by their transformed values. In the case of $\sigma^2 > x_1^2 + x_2^2$ an error condition is raised.

The following Theorem ensures that the generalized Givens transformation necessary in each stage will never fail.

ALGORITHM 3. *Implicit Cholesky Decomposition of an n-by-n Lower Triangular Matrix L.*

```
U := 0
for j := 1 to n do
    u_jj := l_jj
    tmp := 0
    rotg2 (u_jj, tmp, σ, c, s)
    for i := j + 1 to n do
        rotg (u_jj, l_ij, c, s)
        for k := j + 1 to i do
            rot (u_jk, l_ik, c, s)
        end
    end
end
```

THEOREM 4.1. *Let L be an n-by-n lower triangular matrix, and let σ denote a nonnegative lower bound for all the singular values of L. Under these assumptions, Algorithm 3 will compute the decomposition (3) which is equivalent to (2). In particular the generalized Givens transformation necessary in each stage will never fail.*

Proof. The proof of this Theorem is organized as follows. We consider a single stage of Algorithm 3 as depicted in Figure 1 and show:

1. This step can be carried out if the matrix $L^{\mathrm{T}}L - \sigma^2 I$ is positive semidefinite.
2. The matrix $L'^{\mathrm{T}}L' - \sigma^2 I$ will be positive semidefinite as well.

These two conditions establish Theorem 4.1.

First, we can see that $|l_{11}| \geq \sigma$. Let $L = U\Sigma V^{\mathrm{T}}$ be the singular value decomposition of L. This implies

$$l_{11}^2 = \|L^{\mathrm{T}}\mathbf{e}_1\|_2^2 = \|V\Sigma^{\mathrm{T}}U^{\mathrm{T}}\mathbf{e}_1\|_2^2 = \sum_{i=1}^{n}(\sigma_i u_{1i})^2 \geq \sum_{i=1}^{n}\sigma^2 u_{1i}^2 = \sigma^2,$$

which ensures that the generalized Givens transformation will succeed.

Now we show that the matrix $L'^{\mathrm{T}}L' - \sigma^2 I$ is positive semidefinite as well. If we take the notation from Figure 1, we get

$$L^{\mathrm{T}}L - \sigma^2 I = \begin{bmatrix} u^2 & u\mathbf{v}^{\mathrm{T}} \\ u\mathbf{v} & L'^{\mathrm{T}}L' + \mathbf{v}\mathbf{v}^{\mathrm{T}} - \sigma^2 I \end{bmatrix}. \tag{6}$$

Additionally, let us define the vector

$$\mathbf{x} := \begin{bmatrix} y \\ \mathbf{z} \end{bmatrix},$$

whose partition is commensurable with (6). Then

$$\begin{aligned} \mathbf{x}^{\mathrm{T}}(L^{\mathrm{T}}L - \sigma^2 I)\mathbf{x} &= \begin{bmatrix} y & \mathbf{z}^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} u^2 & u\mathbf{v}^{\mathrm{T}} \\ u\mathbf{v} & L'^{\mathrm{T}}L' + \mathbf{v}\mathbf{v}^{\mathrm{T}} - \sigma^2 I \end{bmatrix} \begin{bmatrix} y \\ \mathbf{z} \end{bmatrix} \\ &= u^2y^2 + 2uy\mathbf{v}^{\mathrm{T}}\mathbf{z} + \mathbf{z}^{\mathrm{T}}(L'^{\mathrm{T}}L' + \mathbf{v}\mathbf{v}^{\mathrm{T}} - \sigma^2 I)\mathbf{z} \\ &= (uy + \mathbf{v}^{\mathrm{T}}\mathbf{z})^2 + \mathbf{z}^{\mathrm{T}}(L'^{\mathrm{T}}L' - \sigma^2 I)\mathbf{z} \geq 0. \end{aligned}$$

First we assume $u \neq 0$. In this case we can determine a y corresponding to each $\mathbf{z}$ such that $uy + \mathbf{v}^{\mathrm{T}}\mathbf{z} = 0$. This implies

$$\mathbf{z}^{\mathrm{T}}(L'^{\mathrm{T}}L' - \sigma^2 I)\mathbf{z} \geq 0 \tag{7}$$

for all $\mathbf{z}$.

Finally, assume that $u = 0$. As the orthogonal Givens transformations leave the norm of the first column intact, we have

$$\sigma^2 = \sum_{i=1}^{n} l_{i1}^2 \geq l_{11}^2 \geq \sigma^2.$$

But this implies $|l_{11}| = \sigma$ and $l_{i1} = 0$ for $i = 2, \ldots, n$. Consequently, the ordinary Givens transformations in this stage need not be carried out. The important point here is, that we have $\mathbf{v} = \mathbf{0}$ in this special case. This implies $uy + \mathbf{v}^{\mathrm{T}}\mathbf{z} = 0$ too, and the inequality (7) can be established for all $\mathbf{z}$ as well. But this observation proves the claim that $L'^{\mathrm{T}}L' - \sigma^2 I$ is a positive semidefinite matrix. □

Note that if the decomposition (3) exists, then from (2) it follows immediately that $L^{\mathrm{T}}L - \sigma^2 I$ is a positive semidefinite matrix. Therefore, the implicit Cholesky decomposition of a square lower triangular matrix L exists if and only if $L^{\mathrm{T}}L - \sigma^2 I$ is a positive semidefinite matrix.

There are a number of problems where the implicit Cholesky decomposition exhibits its advantages over its explicit counterpart. The explicit Cholesky decomposition is based on the factorization of the matrix $A := L^{\mathrm{T}}L - \sigma^2 I$. The condition number of A is at least the square of the condition number of L. This means that even for a modestly ill-conditioned L the matrix A can become numerically singular. In this case the explicit Cholesky decomposition breaks down. Also, if A is close to a singular matrix the explicit Cholesky decomposition may loose accuracy. The reader may find a more detailed comparison in [26, pp. 47–53].

4.3. Bidiagonal Matrices. In the special case of a progressive qd-step the matrix L has the shape of the n-by-n lower bidiagonal matrix

$$L := \begin{bmatrix} \alpha_1 & & & \\ \beta_1 & \ddots & & \\ & \ddots & \ddots & \\ & & \beta_{n-1} & \alpha_n \end{bmatrix}.$$

Due to the special structure of the matrix L we can use a simplified version of the general Algorithm 3. This is shown graphically in Figure 2 and as Algorithm 4. Note that Algorithm 4 takes care of not overwriting the vectors α and β.

5. Differential Quotient-Difference Algorithm. The evaluation of the Cholesky decomposition (1) represents the heart of Rutishauser's qd-algorithm. As we have already seen in Section 4.2 this factorization can also be obtained with the help of generalized Givens transformations. Algorithm 4 gives us the decomposition

$$Q \begin{bmatrix} R^{\mathrm{T}} \\ 0 \end{bmatrix} = \begin{bmatrix} R' \\ \sqrt{s} I \end{bmatrix},$$

$$
\begin{bmatrix} \alpha_1 & & & \\ \beta_1 & \alpha_2 & & \\ & \beta_2 & \alpha_3 & \\ & & \ddots & \ddots \\ 0 & & & \\ & 0 & & \\ & & 0 & \\ & & & \ddots \end{bmatrix}
\longmapsto
\begin{bmatrix} \gamma_1' & & & \\ \beta_1 & \alpha_2 & & \\ & \beta_2 & \alpha_3 & \\ & & \ddots & \ddots \\ \sigma & & & \\ & 0 & & \\ & & 0 & \\ & & & \ddots \end{bmatrix}
\longmapsto
\begin{bmatrix} \gamma_1 & \delta_1 & & \\ 0 & \tilde{\gamma}_2 & & \\ & \beta_2 & \alpha_3 & \\ & & \ddots & \ddots \\ \sigma & & & \\ & 0 & & \\ & & 0 & \\ & & & \ddots \end{bmatrix}
\longmapsto
$$

$$
\begin{bmatrix} \gamma_1 & \delta_1 & & \\ 0 & \gamma_2' & & \\ & \beta_2 & \alpha_3 & \\ & & \ddots & \ddots \\ \sigma & & & \\ & \sigma & & \\ & & 0 & \\ & & & \ddots \end{bmatrix}
\longmapsto
\begin{bmatrix} \gamma_1 & \delta_1 & & \\ 0 & \gamma_2 & \delta_2 & \\ & 0 & \tilde{\gamma}_3 & \\ & & \ddots & \ddots \\ \sigma & & & \\ & \sigma & & \\ & & 0 & \\ & & & \ddots \end{bmatrix}
\longmapsto \cdots
$$

FIG. 2. *Implicit Cholesky Decomposition of a Lower Bidiagonal Matrix L.*

ALGORITHM 4. *Implicit Cholesky Decomposition of an n-by-n Lower Bidiagonal Matrix L.*

$\gamma_1 := \alpha_1$
for $k := 1$ **to** $n-1$ **do**
  tmp := 0
  `rotg2` (γ_k, tmp, σ, cos, sin)
  tmp := β_k
  `rotg` (γ_k, tmp, cos, sin)
  $\delta_k := 0$
  $\gamma_{k+1} := \alpha_{k+1}$
  `rot` (δ_k, γ_{k+1}, cos, sin)
end
tmp := 0
`rotg2` (γ_n, tmp, σ, cos, sin)

which is equivalent to (1).

It is now possible to modify the qd-Algorithm 1 to directly compute the singular values of an upper bidiagonal matrix R. We get Algorithm 5, which corresponds to the differential qd-algorithm by Fernando and Parlett (cf. [4, Section 5]).

This algorithm enables us to compute all the singular values of R to high relative accuracy (cf. [4, Section 7]). On the other hand, it is not suited to also compute the corresponding singular vectors simultaneously with the singular values. The primary reason for this is that the matrix R is transposed in each step. Consequently, it is not possible to interpret Algorithm 5 as a sequence of orthogonal transformations applied from the left and the right to the matrix R, as it is the case with the ordinary QR-algorithm (cf. [7, Section 8.3]).

In order to remedy this deficiency we will propose a new orthogonal qd-algorithm

ALGORITHM 5. *Differential qd-Algorithm.*

$\sigma := 0$
for $k := n$ **to** 1 **by** -1 **do**
  while $\gamma_k \neq 0$ **do**
    Choose a shift s with $0 \leq s \leq \sigma_{\min}(R)$.
    Compute the implicit Cholesky decomposition

$$Q \begin{bmatrix} R^{\mathrm{T}} \\ 0 \end{bmatrix} = \begin{bmatrix} R' \\ sI \end{bmatrix}.$$

    $\sigma := \sqrt{\sigma^2 + s^2}$
    $R := R'$
  end
  $\sigma_k := \sigma$
  Compute the QR-decomposition $R^{\mathrm{T}} = QR'$.
  $R := (k-1)$-by-$(k-1)$ leading principal submatrix of R'
end

which transforms the $(2n)$-by-n matrix

$$A := \begin{bmatrix} R \\ 0 \end{bmatrix}$$

to a diagonal matrix with the help of orthogonal transformations applied from the left and the right. So-called orthogonal qd-steps are the tool for doing this, and they will be presented in the next sections.

6. Orthogonal Quotient-Difference Steps I. Let

$$L := \begin{bmatrix} \alpha_1 & & & \\ \beta_1 & \ddots & & \\ & \ddots & \ddots & \\ & & \beta_{n-1} & \alpha_n \end{bmatrix}$$

denote an n-by-n lower bidiagonal matrix, and let

$$U := \begin{bmatrix} \gamma_1 & \delta_1 & & \\ & \ddots & \ddots & \\ & & \ddots & \delta_{n-1} \\ & & & \gamma_n \end{bmatrix}$$

be an n-by-n upper bidiagonal matrix.

DEFINITION 6.1. *Let Q be an orthogonal $(2n)$-by-$(2n)$ matrix. We call the transformation*

$$Q \begin{bmatrix} L \\ \sigma I \end{bmatrix} = \begin{bmatrix} U \\ \sqrt{\sigma^2 + s^2} I \end{bmatrix} \tag{8}$$

$$
\begin{bmatrix} \alpha_1 & & & & \\ \beta_1 & \alpha_2 & & & \\ & \beta_2 & \alpha_3 & & \\ & & \ddots & \ddots & \\ \sigma & & & & \\ & \sigma & & & \\ & & \sigma & & \\ & & & \ddots & \end{bmatrix}
\longmapsto
\begin{bmatrix} \gamma_1' & & & & \\ \beta_1 & \alpha_2 & & & \\ & \beta_2 & \alpha_3 & & \\ & & \ddots & \ddots & \\ \rho & & & & \\ & \sigma & & & \\ & & \sigma & & \\ & & & \ddots & \end{bmatrix}
\longmapsto
\begin{bmatrix} \gamma_1 & \delta_1 & & & \\ 0 & \tilde{\gamma}_2 & & & \\ & \beta_2 & \alpha_3 & & \\ & & \ddots & \ddots & \\ \rho & & & & \\ & \sigma & & & \\ & & \sigma & & \\ & & & \ddots & \end{bmatrix}
\longmapsto
$$

$$
\begin{bmatrix} \gamma_1 & \delta_1 & & & \\ 0 & \gamma_2' & & & \\ & \beta_2 & \alpha_3 & & \\ & & \ddots & \ddots & \\ \rho & & & & \\ & \rho & & & \\ & & \sigma & & \\ & & & \ddots & \end{bmatrix}
\longmapsto
\begin{bmatrix} \gamma_1 & \delta_1 & & & \\ 0 & \gamma_2 & \delta_2 & & \\ & 0 & \tilde{\gamma}_3 & & \\ & & \ddots & \ddots & \\ \rho & & & & \\ & \rho & & & \\ & & \sigma & & \\ & & & \ddots & \end{bmatrix}
\longmapsto \cdots
$$

FIG. 3. *Orthogonal Left lu-Step with Shift s.*

an orthogonal left lu-step with shift s.

It follows from Theorem 4.1 that this transformation exists if and only if the condition $|s| \leq \sigma_{\min}(L)$ holds. The singular values of L are diminished by the amount of the shift s, i.e.

$$\sigma_i^2(U) = \sigma_i^2(L) - s^2.$$

The matrix Q can be constructed by the sequence of Givens rotations depicted in Figure 3. The quantity ρ is an abbreviation for $\sqrt{\sigma^2 + s^2}$. The use of generalized Givens transformations may become problematic for small values of s, however. Assume, for instance, that α_1 and s are tiny, and σ is so large that $\sigma^2 + \alpha_1^2 = \sigma^2$ numerically. In such a case, Algorithm 2 would transform α_1 into $\gamma_1' = 0$ even if $\alpha_1^2 - s^2 > 0$. In order to avoid this pitfall we introduce the differential form of the generalized Givens transformation in the next section.

7. Differential Form of the Generalized Givens Transformation. The generalized Givens transformation, as it has been introduced in Section 4.1, is designed to introduce a given value into an element of a vector. Now, consider the related problem of determining a Givens transformation (4) such that

$$G \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}, \tag{9}$$

where

$$r_1 := \operatorname{sign}(x_1)\sqrt{x_1^2 - \sigma^2}, \tag{10}$$

$$r_2 := \operatorname{sign}(x_2)\sqrt{x_2^2 + \sigma^2}. \tag{11}$$

Algorithm 6. *Differential Form of the Generalized Givens Transformation* (`rotg3`).

if $\sigma = 0$ **then**
  $c := 1$
  $s := 0$
elsif $|x_2| \geq |x_1|$ **then**
  $r_1 := \sqrt{1 - (\sigma/x_1)^2}$
  $r_2 := \sqrt{1 + (\sigma/x_2)^2}$
  $\rho := x_1/x_2$
  $\text{denom} := 1 + \rho^2$
  $c := (\rho^2 \cdot r_1 + r_2)/\text{denom}$
  $s := \big(\rho \cdot (r_1 - r_2)\big)/\text{denom}$
  $x_1 := x_1 \cdot r_1$
  $x_2 := x_2 \cdot r_2$
elsif $|x_2| \geq |\sigma|$ **then**
  $r_1 := \sqrt{1 - (\sigma/x_1)^2}$
  $r_2 := \sqrt{1 + (\sigma/x_2)^2}$
  $\rho := x_2/x_1$
  $\text{denom} := 1 + \rho^2$
  $c := (r_1 + \rho^2 \cdot r_2)/\text{denom}$
  $s := \big(\rho \cdot (r_1 - r_2)\big)/\text{denom}$
  $x_1 := x_1 \cdot r_1$
  $x_2 := x_2 \cdot r_2$
else
  $r_1 := \sqrt{1 - (\sigma/x_1)^2}$
  $r_2 := \sqrt{1 + (x_2/\sigma)^2}$
  $\rho := x_2/x_1$
  $\text{denom} := 1 + \rho^2$
  $c := \big(r_1 + |\rho \cdot (\sigma/x_1)| \cdot r_2\big)/\text{denom}$
  $s := \big(\rho \cdot r_1 - (\text{sign}(x_2) \cdot |\sigma|/x_1) \cdot r_2\big)/\text{denom}$
  $x_1 := x_1 \cdot r_1$
  $x_2 := \text{sign}(x_2) \cdot |\sigma| \cdot r_2$
end

Of course, this is only possible if $|\sigma| \leq |x_1|$. In this context the sign-function is defined by

$$\text{sign}(x) := \begin{cases} 1, & \text{if } x \geq 0, \\ -1, & \text{if } x < 0. \end{cases}$$

Because we only prescribe the difference σ in x_1 and x_2 we call this matrix G the differential form of the generalized Givens transformation.

It is easily verified that the values of c and s are given by

$$\begin{bmatrix} c \\ s \end{bmatrix} = \frac{1}{x_1^2 + x_2^2} \begin{bmatrix} x_1 & x_2 \\ x_2 & -x_1 \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}. \tag{12}$$

A stable numerical implementation must be immune to overflow and underflow as long as the results r_1 and r_2 can be represented by floating-point numbers. This objective can be achieved by an appropriate scaling of the expressions in (10,11,12). A numerical implementation is presented as Algorithm 6. In the following we will

ALGORITHM 7. *Orthogonal Left lu-Step with Shift s.*

$\gamma_1 := \alpha_1$
for $k := 1$ **to** $n-1$ **do**
  tmp $:= \sigma$
  `rotg3` (γ_k, tmp, s, cos, sin)
  tmp $:= \beta_k$
  `rotg` (γ_k, tmp, cos, sin)
  $\delta_k := 0$
  $\gamma_{k+1} := \alpha_{k+1}$
  `rot` (δ_k, γ_{k+1}, cos, sin)
end
tmp $:= \sigma$
`rotg3` (γ_n, tmp, s, cos, sin)

abbreviate this computation by the subroutine call `rotg3` (x_1, x_2, σ, c, s) which overwrites the parameters x_1 and x_2 by their transformed values. The name `rotg3` stems from the BLAS subroutine `rotg` for the calculation of an ordinary Givens transformation [3, 11]. It is erroneous to call `rotg3` with $|\sigma| > |x_1|$.

It is planned to give an error analysis in a future paper. We will show that the matrix G is orthogonal up to a small multiple of the machine precision. Furthermore, the key equation (9) always holds to high accuracy for the computed quantities.

8. Orthogonal Quotient-Difference Steps II. An implementation of the orthogonal left lu-step of Figure 3 is given by Algorithm 7.

DEFINITION 8.1. *Let Q denote an orthogonal $(2n)$-by-$(2n)$ matrix. We call the transformation*

$$Q \begin{bmatrix} U \\ \sigma I \end{bmatrix} = \begin{bmatrix} L \\ \sqrt{\sigma^2 + s^2} I \end{bmatrix} \tag{13}$$

an orthogonal left ul-step with shift s.

This transformation represents the dual version of the orthogonal left lu-step. It can be carried out if and only if $|s| \leq \sigma_{\min}(U)$. The singular values of U are reduced by the amount of the shift s, i.e.

$$\sigma_i^2(L) = \sigma_i^2(U) - s^2.$$

An orthogonal left ul-step, too, can be executed by a sequence of Givens rotations. The mechanism is the same as in Figure 3, with the only exception that the transformations are applied from bottom to top.

DEFINITION 8.2. *Let Q denote an orthogonal n-by-n matrix. We call the transformation*

$$\begin{bmatrix} I & \\ & Q^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} U \\ \sigma I \end{bmatrix} Q = \begin{bmatrix} L \\ \sigma I \end{bmatrix} \tag{14}$$

an orthogonal right ul-step.

$$\begin{bmatrix} \gamma_1 & \delta_1 & & \\ & \gamma_2 & \delta_2 & \\ & & \gamma_3 & \ddots \\ & & & \ddots \end{bmatrix} \longmapsto \begin{bmatrix} \alpha_1 & 0 & & \\ \beta_1 & \tilde{\alpha}_2 & \delta_2 & \\ & & \gamma_3 & \ddots \\ & & & \ddots \end{bmatrix} \longmapsto \begin{bmatrix} \alpha_1 & 0 & & \\ \beta_1 & \alpha_2 & 0 & \\ & \beta_2 & \tilde{\alpha}_3 & \ddots \\ & & & \ddots \end{bmatrix} \longmapsto \cdots$$

FIG. 4. *Orthogonal Right ul-Step.*

ALGORITHM 8. *Orthogonal Right ul-Step.*

$\alpha_1 := \gamma_1$
for $k := 1$ **to** $n-1$ **do**
  tmp $:= \delta_k$
  `rotg` (α_k, tmp, cos, sin)
  $\beta_k := 0$
  $\alpha_{k+1} := \gamma_{k+1}$
  `rot` (β_k, α_{k+1}, cos, sin)
end

This transformation can always be executed, and it leaves the singular values of U unchanged. If $\gamma_n = 0$ we have $\alpha_n = \beta_{n-1} = 0$ after the transformation. This property will be useful to deflate a matrix U with $\gamma_n = 0$.

The sequence of Givens rotations necessary for an orthogonal right ul-step is depicted as Figure 4. The corresponding implementation is presented as Algorithm 8.

DEFINITION 8.3. *Let Q denote an orthogonal n-by-n matrix. We call the transformation*

$$\begin{bmatrix} I & \\ & Q^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} L \\ \sigma I \end{bmatrix} Q = \begin{bmatrix} U \\ \sigma I \end{bmatrix} \tag{15}$$

an orthogonal right lu-step.

This transformation represents the dual version of the orthogonal right ul-step. It can also be executed unconditionally, and it preserves the singular values of the matrix L. If the first row of L is zero, i.e. if $\alpha_1 = 0$, we get a matrix U with $\gamma_1 = \delta_1 = 0$. We will therefore use this transformation to deflate a matrix L with $\alpha_1 = 0$.

An orthogonal right lu-step can also be carried out by a sequence of Givens rotations. The same technique is used as shown in Figure 4, except that the transformations are applied from bottom to top.

We will refer to the four transformations, that have been introduced in this section, by the generic term of *orthogonal qd-steps.*

Fernando and Parlett also present a root-free version of the orthogonal left lu-step in [4]. However, such a root-free algorithm has the disadvantage that it operates on the squares of the singular values. Consequently, the largest singular value must not exceed the square root of the largest machine-representable number. Similarly, singular values smaller than the square root of the smallest machine number underflow to zero. We refrain from this version in order not to restrict the domain of calculation more than necessary.

TABLE 1
Choice of Orthogonal qd-Steps.

Matrix	Grading	qd-Step	Result
lower bidiagonal	$\lvert\alpha_1\rvert \geq \lvert\alpha_n\rvert$	left lu-step	$\lvert\gamma_1\rvert \geq \lvert\gamma_n\rvert$
	$\lvert\alpha_1\rvert < \lvert\alpha_n\rvert$	right lu-step	$\lvert\gamma_1\rvert < \lvert\gamma_n\rvert$
upper bidiagonal	$\lvert\gamma_1\rvert \geq \lvert\gamma_n\rvert$	right ul-step	$\lvert\alpha_1\rvert \geq \lvert\alpha_n\rvert$
	$\lvert\gamma_1\rvert < \lvert\gamma_n\rvert$	left ul-step	$\lvert\alpha_1\rvert < \lvert\alpha_n\rvert$

The orthogonal qd-steps are closely related to the QR-algorithm for calculating the singular value decomposition (cf. [7, Section 8.3]). If we execute an orthogonal left lu-step (8) with shift zero and an orthogonal right ul-step (14) in succession we get the same result as by applying one unshifted QR-step. This has also been observed in [14, 15]. The same transformations are also useful for refining a URV-decomposition (see [14, 25] for more details).

9. Orthogonal Quotient-Difference Algorithm. Now, we have available the full set of orthogonal qd-steps to introduce the orthogonal qd-algorithm. We start from a given n-by-n lower bidiagonal matrix B whose singular value decomposition is desired. The orthogonal qd-algorithm can be expressed as a sequence of left and right qd-steps applied to the $(2n)$-by-n matrix

$$A := \begin{bmatrix} B \\ 0 \end{bmatrix}.$$

The matrix A is transformed into the $(2n)$-by-n matrix

$$\begin{bmatrix} 0 \\ \Sigma \end{bmatrix},$$

where Σ denotes an n-by-n diagonal matrix with the singular values of B. In matrix terms this process can be described by the equation

$$\begin{bmatrix} B \\ 0 \end{bmatrix} = P \begin{bmatrix} 0 \\ \Sigma \end{bmatrix} Q^{\mathrm{T}}, \tag{16}$$

where P denotes an orthogonal $(2n)$-by-$(2n)$ matrix, and Q denotes an orthogonal n-by-n matrix.

In order to obtain rapid convergence the orthogonal qd-steps must preserve the grading of the bidiagonal matrix (cf. [2, p. 891]). For instance, if we have a lower bidiagonal matrix L with large entries in the top left corner and small entries in the lower right corner we will execute an orthogonal left lu-step. If the same matrix L were graded the opposite way we would choose an orthogonal right lu-step. In our implementation we only look at the first and last diagonal entry to determine the grading of the matrix. We present all the four possible cases as Table 1. If, for instance, we have a lower bidiagonal matrix L with $|\alpha_1| \geq |\alpha_n|$, we execute a left lu-step and get an upper bidiagonal matrix U with $|\gamma_1| \geq |\gamma_n|$.

10. Deflation. We would like to execute a deflation step as soon as an element in the bidiagonal matrix L or U has become so small that it can be neglected. This means that we can set this matrix entry to zero without changing the *numerical* singular values of the matrix B in (16).

Let us first analyse the situation when we set to zero a diagonal element α_k in the matrix L. For this we compare the original matrix

$$A := \begin{bmatrix} L \\ \sigma I \end{bmatrix} \tag{17}$$

with the modified matrix

$$\tilde{A} := \begin{bmatrix} \tilde{L} \\ \sigma I \end{bmatrix}, \tag{18}$$

where we obtain the matrix

$$\tilde{L} := L - \alpha_k \mathbf{e}_k \mathbf{e}_k^{\mathrm{T}} \tag{19}$$

from the matrix L by setting α_k to zero. As an immediate consequence of the definition (17) we get the equation

$$\lambda_i(A^{\mathrm{T}}A) = \lambda_i(L^{\mathrm{T}}L) + \sigma^2 = \lambda_i(LL^{\mathrm{T}}) + \sigma^2. \tag{20}$$

By taking into account the definition (19) we can express the matrices $L^{\mathrm{T}}L$ and LL^{T} by

$$L^{\mathrm{T}}L = \tilde{L}^{\mathrm{T}}\tilde{L} + E_1, \tag{21}$$

$$LL^{\mathrm{T}} = \tilde{L}\tilde{L}^{\mathrm{T}} + E_2, \tag{22}$$

where

$$E_1 := \alpha_k^2 \mathbf{e}_k \mathbf{e}_k^{\mathrm{T}} + \alpha_k \beta_{k-1}(\mathbf{e}_{k-1}\mathbf{e}_k^{\mathrm{T}} + \mathbf{e}_k \mathbf{e}_{k-1}^{\mathrm{T}}),$$
$$E_2 := \alpha_k^2 \mathbf{e}_k \mathbf{e}_k^{\mathrm{T}} + \alpha_k \beta_k(\mathbf{e}_k \mathbf{e}_{k+1}^{\mathrm{T}} + \mathbf{e}_{k+1}\mathbf{e}_k^{\mathrm{T}}).$$

The Euclidean norms of the two perturbation matrices E_1 and E_2 are given by

$$\|E_1\|_2 = \frac{1}{2}|\alpha_k|\Big(|\alpha_k| + \sqrt{\alpha_k^2 + 4\beta_{k-1}^2}\Big) \le |\alpha_k|\Big(|\alpha_k| + |\beta_{k-1}|\Big),$$
$$\|E_2\|_2 = \frac{1}{2}|\alpha_k|\Big(|\alpha_k| + \sqrt{\alpha_k^2 + 4\beta_k^2}\Big) \le |\alpha_k|\Big(|\alpha_k| + |\beta_k|\Big).$$

As a consequence of Weyl's monotonicity theorem (cf. [18, pp. 191–194] and [27, pp. 101–103]) the norms of the perturbations E_1 and E_2 represent an upper bound for the change of the eigenvalues in (21,22). More precisely, we can express the eigenvalues of $L^{\mathrm{T}}L$ and LL^{T} by

$$\lambda_i(L^{\mathrm{T}}L) = \lambda_i(\tilde{L}^{\mathrm{T}}\tilde{L}) + u_i\|E_1\|_2, \tag{23}$$

$$\lambda_i(LL^{\mathrm{T}}) = \lambda_i(\tilde{L}\tilde{L}^{\mathrm{T}}) + v_i\|E_2\|_2, \tag{24}$$

with $|u_i| \le 1$ and $|v_i| \le 1$. If we substitute these results into equation (20), we get

$$\sigma_i^2(A) = \sigma_i^2(\tilde{L}) + \sigma^2 + u_i\|E_1\|_2 = \sigma_i^2(\tilde{L}) + \sigma^2 + v_i\|E_2\|_2. \tag{25}$$

We conclude that the singular values of A and $\tilde{A}$ are equal to working precision, if the equation

$$\sigma^2 + \min(\|E_1\|_2, \|E_2\|_2) = \sigma^2$$

applies numerically. Therefore, the condition

$$\sigma^2 + |\alpha_k|\Big(|\alpha_k| + \min(|\beta_{k-1}|, |\beta_k|)\Big) = \sigma^2 \tag{26}$$

represents our numerical deflation criterion for neglecting a diagonal element α_k. It should be noted that $\beta_0 = \beta_n = 0$. And, of course, the equivalent deflation criterion for the upper bidiagonal matrix U is given by

$$\sigma^2 + |\gamma_k|\Big(|\gamma_k| + \min(|\delta_{k-1}|, |\delta_k|)\Big) = \sigma^2, \tag{27}$$

where $\delta_0 = \delta_n = 0$.

Secondly, let us analyse the situation when an off-diagonal element β_k in L becomes small. We also compare the two matrices A and $\tilde{A}$ from (17,18) except that the matrix $\tilde{L}$ is now given by

$$\tilde{L} := L - \beta_k \mathbf{e}_{k+1}\mathbf{e}_k^{\mathrm{T}}.$$

Equations (20,21,22) also apply for this case, and the perturbations E_1 and E_2 are given by

$$\begin{aligned} E_1 &:= \beta_k^2 \mathbf{e}_k \mathbf{e}_k^{\mathrm{T}} + \alpha_{k+1}\beta_k(\mathbf{e}_{k+1}\mathbf{e}_k^{\mathrm{T}} + \mathbf{e}_k\mathbf{e}_{k+1}^{\mathrm{T}}), \\ E_2 &:= \beta_k^2 \mathbf{e}_{k+1} \mathbf{e}_{k+1}^{\mathrm{T}} + \alpha_k\beta_k(\mathbf{e}_k\mathbf{e}_{k+1}^{\mathrm{T}} + \mathbf{e}_{k+1}\mathbf{e}_k^{\mathrm{T}}), \end{aligned}$$

with the norms

$$\begin{aligned} \|E_1\|_2 &= \frac{1}{2}|\beta_k|\Big(|\beta_k| + \sqrt{\beta_k^2 + 4\alpha_{k+1}^2}\Big) \le |\beta_k|\Big(|\beta_k| + |\alpha_{k+1}|\Big), \\ \|E_2\|_2 &= \frac{1}{2}|\beta_k|\Big(|\beta_k| + \sqrt{\beta_k^2 + 4\alpha_k^2}\Big) \le |\beta_k|\Big(|\beta_k| + |\alpha_k|\Big). \end{aligned}$$

The eigenvalue equations (23,24) as well as equation (25) still apply for the new perturbation matrices. Consequently, we are led to the numerical deflation criterion

$$\sigma^2 + |\beta_k|\Big(|\beta_k| + \min(|\alpha_k|, |\alpha_{k+1}|)\Big) = \sigma^2 \tag{28}$$

for neglecting an off-diagonal element β_k. The equivalent criterion for an upper bidiagonal matrix U reads

$$\sigma^2 + |\delta_k|\Big(|\delta_k| + \min(|\gamma_k|, |\gamma_{k+1}|)\Big) = \sigma^2. \tag{29}$$

The deflation criterions (26,27,28,29) ensure the high relative accuracy of the computed singular values. If we also compute the left singular vectors additional conditions need to be satisfied. This issue is discussed in the next section.

11. Singular Vectors. If we accumulate the orthogonal transformations in the orthogonal qd-algorithm we can compute the decomposition (16). In most cases, however, we would like to get the singular value decomposition

$$B = U\Sigma V^{\mathrm{T}} \tag{30}$$

of the n-by-n lower bidiagonal matrix B, where U and V denote orthogonal n-by-n matrices. We will identify the matrix Q from (16) with the matrix V in (30). On the other hand we can recover the left singular vectors as a part of the matrix P. In order to see this we partition P into four n-by-n submatrices as follows:

$$P = \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix}.$$

Then equation (16) is equivalent to

$$\begin{bmatrix} B \\ 0 \end{bmatrix} = \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} \begin{bmatrix} 0 \\ \Sigma \end{bmatrix} Q^{\mathrm{T}}.$$

If Σ is nonsingular this implies $P_{11} = P_{22} = 0$, and P_{12} is an orthogonal matrix with the left singular vectors. We will now show that we can get the same result in the presence of rounding errors, even if B has small or zero singular values.

Let us first analyse the effects of deflation. We assume that we have a lower bidiagonal matrix L which satisfies the equation

$$\begin{bmatrix} B \\ 0 \end{bmatrix} = P \begin{bmatrix} L \\ \sigma I \end{bmatrix} Q^{\mathrm{T}},$$

where P and Q are orthogonal matrices. In order to express deflation in L we write L as

$$L = \hat{L} + \Delta L,$$

where $\hat{L}$ denotes the deflated matrix, and ΔL is a rank-one matrix consisting of the matrix entry that has been removed from L. If we assume that all the calculations after the deflation are executed exactly we get the decomposition

$$\begin{bmatrix} B \\ 0 \end{bmatrix} = P \begin{bmatrix} \hat{L} \\ \sigma I \end{bmatrix} Q^{\mathrm{T}} + P \begin{bmatrix} \Delta L \\ 0 \end{bmatrix} Q^{\mathrm{T}} = U \begin{bmatrix} 0 \\ \Sigma \end{bmatrix} V^{\mathrm{T}} + P \begin{bmatrix} \Delta L \\ 0 \end{bmatrix} Q^{\mathrm{T}}.$$

Again U and V are orthogonal matrices. If we partition U and P into n-by-n submatrices we get

$$\begin{bmatrix} BV \\ 0 \end{bmatrix} = \begin{bmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{bmatrix} \begin{bmatrix} 0 \\ \Sigma \end{bmatrix} + \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} \begin{bmatrix} \Delta L \\ 0 \end{bmatrix} Q^{\mathrm{T}} V. \tag{31}$$

This implies that

$$0 = U_{22}\Sigma + P_{21}\Delta L Q^{\mathrm{T}} V.$$

Note that all the singular values of Σ are greater than σ. Consequently, we can write U_{22} as

$$U_{22} = -P_{21}\Delta L Q^{\mathrm{T}} V \Sigma^{-1},$$

and the norm of U_{22} can be bounded by

$$\|U_{22}\|_2 \leq \frac{\|\Delta L\|_2}{\sigma}.$$

We conclude that U_{11} and U_{22} are zero numerically if $\|\Delta L\|_2 \leq \varepsilon\sigma$, where ε denotes the unit roundoff of the computer. This condition is met by the deflation criterions

$$|\alpha_k| \leq \varepsilon\sigma \tag{32}$$

and

$$|\beta_k| \leq \varepsilon\sigma. \tag{33}$$

The equivalent criterions for an upper bidiagonal matrix are

$$|\gamma_k| \leq \varepsilon\sigma \tag{34}$$

and

$$|\delta_k| \leq \varepsilon\sigma. \tag{35}$$

Note that the conditions (32,33,34,35) only ensure that the matrix U_{12} in (31) will be numerically orthogonal. We still need the conditions (26,27,28,29) to guarantee the accuracy of the singular values.

The deflation criterions (32,33,34,35) only work properly if σ is sufficiently large. In particular, we require that

$$\sigma > \frac{m}{\varepsilon},$$

where m denotes the underflow threshold. If B has some tiny or zero singular values this condition will not be satisfied right away. In this case we use zero-shift qd-steps to determine the tiny singular values. We may expect these initial zero-shift qd-steps to converge rapidly. Our implementation sets an off-diagonal entry β_k or δ_k to zero as soon as the deflation criterion 1 by Demmel and Kahan [2, p. 889] is met. Note that two zero-shift qd-steps are equivalent to one zero-shift QR iteration.

12. Shifts. The performance of the orthogonal qd-algorithm mainly depends on the choice of the shift s in each step. In this section we present two different shift strategies based on Newton's and Laguerre's method to compute the zeros of a polynomial.

Laguerre's method leads to a cubically convergent process. However, it is quite expensive to compute and it must also be carefully implemented to avoid numerical overflow problems. Unlike Wilkinson's shift, Laguerre's method will always compute a lower bound for the smallest singular value of the bidiagonal matrix.

In what follows we will compute Newton's and Laguerre's shift for an upper bidiagonal matrix U. If we have to compute these shifts for a lower bidiagonal matrix L, we simply transpose it since this does not change the singular values.

We may assume that U does not decouple, i.e. $\gamma_i \neq 0$ and $\delta_i \neq 0$. Otherwise we could execute a deflation step and reduce the size of the problem.

12.1. Newton's and Laguerre's Shift. The zeros of the characteristic polynomial

$$p(\lambda) := \det(U^{\mathrm{T}}U - \lambda I).$$

are the eigenvalues of $U^{\mathrm{T}}U$, which are equal to the squares of the singular values of U. Thus if we use Newton's or Laguerre's method to approximate the smallest zero of $p(\lambda)$ we can also get an approximation for the smallest singular value of U.

Newton's method can be described by the iteration

$$\lambda_{k+1} = \lambda_k - r\frac{p(\lambda_k)}{p'(\lambda_k)},$$

and in the case of Laguerre's method we have

$$\lambda_{k+1} = \lambda_k - \frac{p(\lambda_k)}{p'(\lambda_k)}\frac{n}{1 \pm \sqrt{\frac{n-r}{r}\left(n\frac{p'(\lambda_k)^2 - p(\lambda_k)p''(\lambda_k)}{p'(\lambda_k)^2} - 1\right)}}.$$

The value of r denotes the multiplicity of the zero that is being approached by the iteration (see also [8, 9, 10, 13, 16, 17] and [27, pp. 441–445]).

If all the entries on the two diagonals of U are nonzero, then the eigenvalues of $U^{\mathrm{T}}U$ must be distinct [18, p. 124]. Consequently, $p(\lambda)$ has n distinct positive zeros, and we can always set $r = 1$. However, some of these zeros may lie so closely together that they seem to be a multiple zero numerically. Unfortunately, the corresponding numerical multiplicity r is usually not known a priori. This is another reason why we will always set $r = 1$ in this section.

We choose $\lambda_0 = 0$ as our initial value. It is well-known (cf. [8, 10, 16] and [27, pp. 443–445]) that both methods will then converge monotonically to the smallest zero of $p(\lambda)$. In particular we have

$$0 = \lambda_0 \leq \lambda_1 \leq \lambda_{\min}(U^{\mathrm{T}}U).$$

Laguerre's method enjoys cubic convergence provided that the multiplicity r is chosen properly (cf. [16, pp. 353–362] and [27, pp. 443–445]). On the other hand Newton's method will converge only quadratically [27, p. 441].

We intend to use $\sqrt{\lambda_1}$ as the shift in an orthogonal qd-step. Thus we define Newton's shift and Laguerre's shift as follows:

$$s_{\mathrm{Newton}} := \sqrt{-r\frac{p(0)}{p'(0)}}, \tag{36}$$

$$s_{\mathrm{Laguerre}} := \sqrt{-\frac{p(0)}{p'(0)}}\sqrt{\frac{n}{1+\sqrt{\frac{n-r}{r}\left(n\frac{p'(0)^2-p(0)p''(0)}{p'(0)^2}-1\right)}}}. \tag{37}$$

For $r = 1$, we can also view s_{Laguerre} as an enlarged Newton step, and the second term in (37) serves as an acceleration factor for Newton's method.

In the rest of this section we will be concerned with the numerically stable evaluation of these shifts. In [12] T. Y. Li and Z. Zeng discuss the same problem when they evaluate Laguerre's shift for the symmetric tridiagonal eigenproblem. They avoid the calculation of the characteristic polynomial p and its derivatives since these values are likely to underflow or overflow. Fortunately, the key quantities in (36) and (37) are given by

$$f := \sqrt{-\frac{p(0)}{p'(0)}} = \left(\sum_{k=1}^{n}\frac{1}{\sigma_k^2}\right)^{-1/2} = \left(\mathrm{trace}(UU^{\mathrm{T}})^{-1}\right)^{-1/2},$$

$$g := \frac{p'(0)^2-p(0)p''(0)}{p'(0)^2} = \frac{\displaystyle\sum_{k=1}^{n}\frac{1}{\sigma_k^4}}{\left(\displaystyle\sum_{k=1}^{n}\frac{1}{\sigma_k^2}\right)^2} = \frac{\mathrm{trace}(UU^{\mathrm{T}})^{-2}}{\left(\mathrm{trace}(UU^{\mathrm{T}})^{-1}\right)^2}.$$

It is easy to see that f and g can be bounded as follows:

$$0 \le f \le \sigma_{\min}(U),$$
$$\frac{1}{n} \le g \le 1.$$

Obviously, both f and g are well-scaled and are not in danger of numerical overflow. However, the value of f may underflow if U has tiny singular values close to or smaller than the underflow threshold. If this happens we choose $s = 0$ as our shift.

The following lemma holds the key for the stable evaluation of f and g.

Lemma 12.1. *Let U be a nonsingular n-by-n upper bidiagonal matrix, and let P and Q be orthogonal matrices such that*

$$UP\mathbf{e}_k = r_k\mathbf{e}_k, \tag{38}$$

$$P^{\mathrm{T}}U^{\mathrm{T}}Q\mathbf{e}_k = s_k\mathbf{e}_k. \tag{39}$$

Then we have

$$\mathbf{e}_k^{\mathrm{T}}(UU^{\mathrm{T}})^{-1}\mathbf{e}_k = \frac{1}{r_k^2}, \tag{40}$$

$$\mathbf{e}_k^{\mathrm{T}}(UU^{\mathrm{T}})^{-2}\mathbf{e}_k = \frac{1}{r_k^2 s_k^2}. \tag{41}$$

Proof. Since U is nonsingular r_k must be nonzero, and equation (38) is equivalent to

$$P\mathbf{e}_k = r_k U^{-1}\mathbf{e}_k.$$

But this implies (40). Similarly s_k is nonzero, and equation (39) is equivalent to

$$Q\mathbf{e}_k = s_k U^{-\mathrm{T}} P\mathbf{e}_k = r_k s_k U^{-\mathrm{T}} U^{-1}\mathbf{e}_k,$$

from which (41) follows. □

If we can construct a sequence of orthogonal matrices P and Q such that (38) and (39) hold for $k = 1, \ldots, n$, then we would have

$$\begin{aligned} \operatorname{trace}(UU^{\mathrm{T}})^{-1} &= \sum_{k=1}^{n} \frac{1}{r_k^2}, \\ \operatorname{trace}(UU^{\mathrm{T}})^{-2} &= \sum_{k=1}^{n} \frac{1}{r_k^2 s_k^2}. \end{aligned}$$

We will now show how this can be accomplished.

First it is necessary to apply an orthogonal right lu-step to U^{T}. This means that we determine an orthogonal matrix W such that

$$R = U^{\mathrm{T}} W,$$

where

$$R = \begin{bmatrix} \tilde{\gamma}_1 & \tilde{\delta}_1 & & & \\ & \tilde{\gamma}_2 & \tilde{\delta}_2 & & \\ & & \ddots & \ddots & \\ & & & \ddots & \tilde{\delta}_{n-1} \\ & & & & \tilde{\gamma}_n \end{bmatrix}$$

is an n-by-n upper bidiagonal matrix. Then we obviously have

$$\begin{aligned} U\mathbf{e}_1 &= \gamma_1 \mathbf{e}_1, \\ U^{\mathrm{T}} W\mathbf{e}_1 &= \tilde{\gamma}_1 \mathbf{e}_1, \end{aligned}$$

such that $r_1 = \gamma_1$ and $s_1 = \tilde{\gamma}_1$.

In order to compute r_2 we postmultiply U by the first Givens rotation G_1 from a right ul-step:

$$UG_1^{\mathrm{T}} = \begin{bmatrix} \alpha_1 & & & & \\ \beta_1 & \tilde{\alpha}_2 & \delta_2 & & \\ & & \gamma_3 & \delta_3 & \\ & & & \gamma_4 & \ddots \\ & & & & \ddots \end{bmatrix}.$$

Consequently, we have $r_2 = \tilde{\alpha}_2$. Because of Lemma 12.1 we also must premultiply R by G_1:

$$G_1 R = \begin{bmatrix} m_{11} & m_{12} & m_{13} & & \\ m_{21} & m_{22} & m_{23} & & \\ & & \tilde{\gamma}_3 & \tilde{\delta}_3 & \\ & & & \tilde{\gamma}_4 & \ddots \\ & & & & \ddots \end{bmatrix}.$$

The next two Givens rotations are used to zero the entries m_{12} and m_{13}. We get

$$G_1 R G_2^{\mathrm{T}} = \begin{bmatrix} m'_{11} & & m_{13} & & \\ m'_{21} & m'_{22} & m_{23} & & \\ & & \tilde{\gamma}_3 & \tilde{\delta}_3 & \\ & & & \tilde{\gamma}_4 & \ddots \\ & & & & \ddots \end{bmatrix}$$

and

$$G_1 R G_2^{\mathrm{T}} G_3^{\mathrm{T}} = \begin{bmatrix} m''_{11} & & & & \\ m''_{21} & m'_{22} & m'_{23} & & \\ m_{31} & & m_{33} & \tilde{\delta}_3 & \\ & & & \tilde{\gamma}_4 & \ddots \\ & & & & \ddots \end{bmatrix}.$$

At this point we have $s_2 = m'_{22}$.

The rest of the sequences $\{r_k\}$ and $\{s_k\}$ can be determined by the same technique. In each step we need one Givens rotation to compute r_k and two additional rotations to get s_k. Consequently, the sequences $\{r_k\}$ and $\{s_k\}$ can be computed efficiently with $O(n)$ operations.

Now we can express the quantities f and g in terms of the sequences $\{r_k\}$ and $\{s_k\}$ as follows:

$$f = \left(\sum_{k=1}^{n} \frac{1}{r_k^2}\right)^{-1/2}, \tag{42}$$

$$g = \frac{\displaystyle\sum_{k=1}^{n} \frac{1}{r_k^2 s_k^2}}{\left(\displaystyle\sum_{k=1}^{n} \frac{1}{r_k^2}\right)^2}. \tag{43}$$

In order to avoid problems with numerical underflow or overflow we must scale the expressions in (42) and (43). We propose the following two scaling factors:

$$\rho_k := \min_{i \le k} |r_i|,$$
$$\tau_k := \min_{i \le k} \sqrt{|r_i|}\sqrt{|s_i|}.$$

We also introduce two sequences of scaled partial sums:

$$\mu_k := \rho_k^2 \sum_{i=1}^{k} \frac{1}{r_i^2},$$
$$\nu_k := \tau_k^4 \sum_{i=1}^{k} \frac{1}{r_i^2 s_i^2}.$$

It is important to note that we always have

$$1 \le \mu_k \le k,$$
$$1 \le \nu_k \le k.$$

These two sequences can be evaluated recursively as follows:

$$\mu_1 = 1,$$
$$\mu_k = \begin{cases} \left(\dfrac{\rho_k}{\rho_{k-1}}\right)^2 \mu_{k-1} + 1, & \text{if } |r_k| < \rho_{k-1}, \\ \mu_{k-1} + \left(\dfrac{\rho_k}{r_k}\right)^2, & \text{if } |r_k| \ge \rho_{k-1}, \end{cases}$$
$$\nu_1 = 1,$$
$$\nu_k = \begin{cases} \left(\dfrac{\tau_k}{\tau_{k-1}}\right)^4 \nu_{k-1} + 1, & \text{if } \sqrt{|r_k|}\sqrt{|s_k|} < \tau_{k-1}, \\ \nu_{k-1} + \left(\dfrac{\tau_k}{\sqrt{|r_k|}\sqrt{|s_k|}}\right)^4, & \text{if } \sqrt{|r_k|}\sqrt{|s_k|} \ge \tau_{k-1}. \end{cases}$$

Now the values of f and g can be written as

$$f = \frac{\rho_n}{\sqrt{\mu_n}},$$
$$g = \left(\frac{\rho_n}{\tau_n}\right)^4 \frac{\nu_n}{\mu_n^2},$$

and Newton's shift and Laguerre's shift are given by

$$s_{\text{Newton}} = f,$$
$$s_{\text{Laguerre}} = f\sqrt{\frac{n}{1 + \sqrt{(n-1)\big(ng - 1\big)}}}.$$

TABLE 2
Test Cases with Graded Matrices.

Test Case	n	c	$\sigma_{\min}$	$\sigma_{\max}$
1	50	2	$8.325 \cdot 10^{-1}$	$6.450 \cdot 10^{14}$
2	50	4	$9.662 \cdot 10^{-1}$	$3.273 \cdot 10^{29}$
3	50	0.5	$2.189 \cdot 10^{-16}$	$1.467 \cdot 10^{0}$
4	50	0.25	$4.326 \cdot 10^{-31}$	$1.426 \cdot 10^{0}$
5	100	2	$8.325 \cdot 10^{-1}$	$7.262 \cdot 10^{29}$
6	100	0.5	$1.370 \cdot 10^{-31}$	$1.467 \cdot 10^{0}$
7	500	1.1875	$3.475 \cdot 10^{-1}$	$2.637 \cdot 10^{37}$
8	500	0.875	$2.510 \cdot 10^{-31}$	$1.672 \cdot 10^{0}$

12.2. Bisection Shift. In theory an orthogonal qd-step with Newton's or Laguerre's shift s must always succeed since s is a lower bound for the singular values of the bidiagonal matrix. In floating point arithmetic, however, an orthogonal qd-step may occasionally fail due to rounding errors. The chance of failure increases as the shift s approaches the smallest singular value from below.

If such a breakdown occurs we divide s by two and restart the orthogonal qd-step. We call this a bisection step. In unusual circumstances it may be necessary to repeat this procedure several times.

13. Numerical Results. We will now report on the numerical results obtained from the orthogonal qd-algorithm. Our analysis is based on the performance of our implementation for two classes of test matrices, graded matrices and Toeplitz matrices.

Let us first consider the class of graded matrices

$$B_{\text{graded}} := \begin{bmatrix} 1 & & & & \\ 1 & c & & & \\ & c & c^2 & & \\ & & c^2 & \ddots & \\ & & & \ddots & c^{n-2} & \\ & & & & c^{n-2} & c^{n-1} \end{bmatrix},$$

which are also used as test matrices by Demmel and Kahan [2, Section 7] and by Fernando and Parlett [4, Section 9.3]. The singular values σ_i of these matrices are of a vastly different size. The approximation $\sigma_i \approx c^{i-1}$ gives a rough estimate of the order of magnitude of σ_i.

In Table 2 the different values of n and c are summarized that we have used for our test matrices. We have run all the test cases in single precision on a DECstation 3100 with a relative machine precision of $\varepsilon = 2^{-24} \approx 5.9605 \cdot 10^{-8}$. The smallest positive normalized number (underflow threshold) is given by $m = 2^{-126} \approx 1.1755 \cdot 10^{-38}$, and the largest number (overflow threshold) is $M = 2^{128}(1 - \varepsilon) \approx 3.4028 \cdot 10^{38}$.

In Tables 3 and 4 we compare the orthogonal qd-algorithm with the subroutine `sbdsqr` from the LAPACK library [1]. This subroutine represents an implementation of the work of Demmel and Kahan [2]. We have also used the procedure `dbdsqr`, which is an implementation of `sbdsqr` in double precision, to assess the accuracy of the

TABLE 3
Accuracy and Sweeps for Graded Matrices.

Test Case	Orthogonal qd-Algorithm				sbdsqr	
	Error		Sweeps		Error	Sweeps
	no Vectors	Vectors	no Vectors	Vectors		
1	$5.399 \cdot 10^{-7}$	$4.359 \cdot 10^{-7}$	0.840	0.820	$4.359 \cdot 10^{-7}$	0.220
2	$2.702 \cdot 10^{-7}$	$2.702 \cdot 10^{-7}$	0.580	0.600	$1.192 \cdot 10^{-7}$	0.120
3	$8.497 \cdot 10^{-7}$	$8.497 \cdot 10^{-7}$	0.800	0.780	$6.739 \cdot 10^{-7}$	0.200
4	$3.898 \cdot 10^{-7}$	$3.898 \cdot 10^{-7}$	0.540	0.560	$5.168 \cdot 10^{-7}$	0.100
5	$4.359 \cdot 10^{-7}$	$5.399 \cdot 10^{-7}$	0.500	0.490	$4.359 \cdot 10^{-7}$	0.110
6	$7.321 \cdot 10^{-7}$	$6.145 \cdot 10^{-7}$	0.490	0.490	$6.739 \cdot 10^{-7}$	0.100
7	$1.659 \cdot 10^{-6}$	$1.479 \cdot 10^{-6}$	0.310	0.312	$8.486 \cdot 10^{-7}$	0.084
8	$2.052 \cdot 10^{-6}$	$2.289 \cdot 10^{-6}$	0.376	0.378	$2.204 \cdot 10^{-6}$	0.108

TABLE 4
CPU-Times for Graded Matrices.

Test Case	Orthogonal qd-Algorithm		sbdsqr	
	no Vectors	Vectors	no Vectors	Vectors
1	0.094 sec	0.539 sec	0.031 sec	0.152 sec
2	0.062 sec	0.402 sec	0.020 sec	0.145 sec
3	0.098 sec	0.539 sec	0.027 sec	0.141 sec
4	0.066 sec	0.383 sec	0.016 sec	0.094 sec
5	0.199 sec	2.176 sec	0.066 sec	0.957 sec
6	0.176 sec	2.129 sec	0.062 sec	0.484 sec
7	3.211 sec	143.284 sec	1.293 sec	97.240 sec
8	3.769 sec	175.290 sec	1.363 sec	87.533 sec

computed singular values. The column labelled "Error" in Table 3 gives the maximal relative error in the computed singular values $\tilde{\sigma}_i$, i.e.

$$\text{Error} := \max_i \frac{|\tilde{\sigma}_i - \sigma_i|}{\sigma_i}.$$

The column "Sweeps" indicates the average number of orthogonal qd-steps per singular value in the orthogonal qd-algorithm and the average number of QR-sweeps in the procedure `sbdsqr`.

The CPU-times needed to solve these test cases are given as Table 4. There are two columns per algorithm depending on whether the singular vectors are computed or not.

As our second class of test matrices we consider the Toeplitz matrices

$$B_{\text{Toeplitz}} := \begin{bmatrix} c & & & \\ 1 & \ddots & & \\ & \ddots & \ddots & \\ & & 1 & c \end{bmatrix}.$$

For $|c| \ll 1$, these matrices have $n-1$ singular values on the order of 1 and one tiny singular value $\sigma_n \approx c^n$.

In Table 5 we give an overview of the test parameters used for this class of matrices. It should be noted that all the values of the different c's can be represented exactly

TABLE 5
Test Cases with Toeplitz Matrices.

Test Case	n	c	$\sigma_{\min}$	$\sigma_{\max}$
9	50	0.5	$6.661 \cdot 10^{-16}$	$1.499 \cdot 10^{0}$
10	50	0.25	$7.396 \cdot 10^{-31}$	$1.250 \cdot 10^{0}$
11	100	0.75	$1.403 \cdot 10^{-13}$	$1.750 \cdot 10^{0}$
12	100	0.5	$5.916 \cdot 10^{-31}$	$1.500 \cdot 10^{0}$
13	500	0.875	$2.366 \cdot 10^{-30}$	$1.875 \cdot 10^{0}$
14	500	2	$1.000 \cdot 10^{0}$	$3.000 \cdot 10^{0}$

TABLE 6
Accuracy and Sweeps for Toeplitz Matrices.

Test Case	Orthogonal qd-Algorithm				sbdsqr	
	Error		Sweeps		Error	Sweeps
	no Vectors	Vectors	no Vectors	Vectors		
9	$5.061 \cdot 10^{-7}$	$1.337 \cdot 10^{-6}$	4.160	5.640	$8.620 \cdot 10^{-7}$	1.660
10	$7.016 \cdot 10^{-7}$	$1.468 \cdot 10^{-6}$	4.180	5.680	$6.677 \cdot 10^{-7}$	1.660
11	$1.333 \cdot 10^{-6}$	$2.425 \cdot 10^{-6}$	4.140	5.790	$7.303 \cdot 10^{-7}$	1.610
12	$7.701 \cdot 10^{-7}$	$2.866 \cdot 10^{-6}$	4.180	5.890	$1.433 \cdot 10^{-6}$	1.650
13	$4.007 \cdot 10^{-6}$	$1.039 \cdot 10^{-5}$	4.042	5.574	$4.625 \cdot 10^{-6}$	1.446
14	$3.426 \cdot 10^{-6}$	$1.230 \cdot 10^{-5}$	4.060	5.524	$4.063 \cdot 10^{-6}$	1.452

by machine numbers; no truncation errors happen while reading the matrix B_{Toeplitz}. The corresponding results are shown as Tables 6 and 7. Note that in the case of the orthogonal qd-algorithm the number of sweeps is usually higher if the singular vectors are also computed. This is due to the fact that the additional deflation criterions of Section 11 also need to be satisfied.

The reader will have noticed that the orthogonal qd-algorithm computes all the singular values for both classes of test matrices to high relative precision. In this respect our algorithm compares well with the procedure `sbdsqr` by Demmel and Kahan.

Demmel and Kahan use Wilkinson's shift, which is easy to compute and leads to rapid convergence. Unfortunately, it can only be applied in conjunction with the QR-algorithm since it does not compute a lower bound for the smallest singular value.

On the other hand, Laguerre's shift is more expensive to evaluate. If n is the size of the matrix, the calculation of Laguerre's shift needs on the order of $O(n)$ operations, whereas Wilkinson's shift only needs $O(1)$ operations. As we can see from Tables 3 and 6 the number of sweeps per singular value is also larger. However Laguerre's shift always gives us a lower bound on the smallest singular value of the bidiagonal matrix, which is essential for the orthogonal qd-algorithm.

Finally, Fernando and Parlett [4] also report significant speedups of their algorithm over the LINPACK-routine `dsvdc` [3]. Their performance advantage can be partly attributed to the fact that they use a root-free algorithm. However, this approach limits the domain of matrices to which it can be applied. Whether this is acceptable or not depends on the application.

14. Conclusions. We have presented the orthogonal qd-algorithm to compute the singular values of a bidiagonal matrix to high relative accuracy. Our approach differs from the qd-algorithm by Fernando and Parlett as we do not transpose the

TABLE 7
CPU-Times for Toeplitz Matrices.

Test Case	Orthogonal qd-Algorithm		sbdsqr	
	no Vectors	Vectors	no Vectors	Vectors
9	0.395 sec	3.000 sec	0.105 sec	0.551 sec
10	0.410 sec	2.765 sec	0.109 sec	0.684 sec
11	1.984 sec	18.893 sec	0.445 sec	3.586 sec
12	1.492 sec	19.190 sec	0.414 sec	4.605 sec
13	32.654 sec	2068.582 sec	8.164 sec	579.436 sec
14	32.994 sec	2074.473 sec	7.824 sec	555.707 sec

bidiagonal matrix in each step. This enables us to accumulate the orthogonal transformations and thus obtain the singular vectors.

It would be fairly straightforward to modify the orthogonal qd-algorithm to compute only the k smallest singular values and their corresponding singular vectors. Special attention would be required if the bidiagonal matrix decouples. But this approach would enable us to compute the k smallest singular values in $O(kn)$ operations. If the singular vectors are also needed this operation count would increase to $O(kn^2)$. This represents a significant savings compared to $O(n^3)$ which is the computational complexity of a complete singular value decomposition.

We use Laguerre's method to compute the shifts for the orthogonal qd-steps. Special attention is given to the numerically stable evaluation. Although Laguerre's shift does not quite attain the efficiency of Wilkinson's shift it has the advantage that it always computes a lower bound on the smallest singular value of the bidiagonal matrix.

We have also presented two generalizations of the Givens transformation. They come in very handy in the context of the implicit Cholesky decomposition and the orthogonal qd-algorithm, but they should also be useful in other applications. A detailed error analysis of the generalized Givens transformation and its differential form will be the subject of a future paper.

Acknowledgments. Our thanks go to W. Gander, G. H. Golub, and J. Waldvogel for their support. The author also thanks G. W. Stewart for his helpful comments.

REFERENCES

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov and D. Sorensen, *LAPACK Users' Guide*, SIAM Publications, Philadelphia, 1992.

[2] J. Demmel and W. Kahan, *Accurate Singular Values of Bidiagonal Matrices*, SIAM J. Sci. Stat. Comput., 11 (1990), pp. 873–912.

[3] J. J. Dongarra, C. B. Moler, J. R. Bunch and G. W. Stewart, *LINPACK Users' Guide*, SIAM Publications, Philadelphia, 1979.

[4] K. V. Fernando and B. N. Parlett, *Accurate singular values and differential qd algorithms*, Numer. Math., 67 (1994), pp. 191–229.

[5] W. Gander, L. Molinari and H. Švecová, *Numerische Prozeduren aus Nachlass und Lehre von Prof. Heinz Rutishauser*, Internat. Ser. Numer. Math., Vol. 33, Birkhäuser, Basel, 1977.

[6] G. H. Golub and W. Kahan, *Calculating the singular values and pseudo-inverse of a matrix*, SIAM J. Numer. Anal., 2 (1965), pp. 205–224.

[7] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Second Edition, The Johns Hopkins University Press, Baltimore, 1989.
[8] E. Hansen and M. Patrick, *A Family of Root Finding Methods*, Numer. Math., 27 (1977), pp. 257–269.
[9] W. Kahan, *Where does Laguerre's method come from?*, in Proceedings of the Fourth Annual Princeton Conference on Information Sciences and Systems, Department of Electrical Engineering, Princeton University, Princeton, 1970, p. 143.
[10] W. Kahan, *Notes on Laguerre's Iteration*, unpublished manuscript, Berkeley, 1992.
[11] C. L. Lawson, R. J. Hanson, D. R. Kincaid and F. T. Krogh, *Basic Linear Algebra Subprograms for Fortran Usage*, ACM Trans. Math. Softw., 5 (1979), pp. 308–325.
[12] T. Y. Li and Z. Zeng, *The Laguerre iteration in solving the symmetric tridiagonal eigenproblem, revisited*, SIAM J. Sci. Comput., 15 (1994), pp. 1145–1173.
[13] H. J. Maehly, *Zur iterativen Auflösung algebraischer Gleichungen*, ZAMP, 5 (1954), pp. 260–263.
[14] R. Mathias and G. W. Stewart, *A Block QR Algorithm and the Singular Value Decomposition*, Linear Algebra Appl., 182 (1993), pp. 91–100.
[15] M. Moonen, P. Van Dooren and F. Vanpoucke, *On the QR Algorithm and Updating the SVD and the URV Decomposition in Parallel*, Linear Algebra Appl., 188/189 (1993), pp. 549–568.
[16] A. M. Ostrowski, *Solution of Equations in Euclidean and Banach Spaces*, Third Edition of "Solution of Equations and Systems of Equations", Academic Press, New York, 1973.
[17] B. N. Parlett, *Laguerre's Method Applied to the Matrix Eigenvalue Problem*, Math. Comp., 18 (1964), pp. 464–485.
[18] B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, 1980.
[19] C. H. Reinsch and F. L. Bauer, *Rational QR Transformation with Newton Shift for Symmetric Tridiagonal Matrices*, Numer. Math., 11 (1968), pp. 264–272.
[20] H. Rutishauser, *Der Quotienten-Differenzen-Algorithmus*, ZAMP, 5 (1954), pp. 233–251.
[21] H. Rutishauser, *Der Quotienten-Differenzen-Algorithmus*, Mitteilungen aus dem Institut für angewandte Mathematik Nr. 7, Birkhäuser, Basel, 1957.
[22] H. Rutishauser, *Über eine kubisch konvergente Variante der LR-Transformation*, ZAMM, 40 (1960), pp. 49–54.
[23] H. Rutishauser, *Les propriétés numériques de l'algorithme quotient-différence*, Rapport EUR 4083f, Communauté Européenne de l'Energie Atomique - EURATOM, Luxembourg, 1968.
[24] H. Rutishauser, *Lectures on Numerical Mathematics*, Birkhäuser, Boston, 1990.
[25] G. W. Stewart, *An Updating Algorithm for Subspace Tracking*, IEEE Trans. Signal Processing, 40 (1992), pp. 1535–1541.
[26] U. von Matt, *Large Constrained Quadratic Problems*, Verlag der Fachvereine, Zürich, 1993.
[27] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965.