SIAM J. MATRIX ANAL. APPL.
Vol. 33, No. 1, pp. 22–51

dqds WITH AGGRESSIVE EARLY DEFLATION[*]

YUJI NAKATSUKASA[†], KENSUKE AISHIMA[‡], AND ICHITARO YAMAZAKI[§]

Abstract. The dqds algorithm computes all the singular values of an $n \times n$ bidiagonal matrix to high relative accuracy in $O(n^2)$ cost. Its efficient implementation is now available as a LAPACK subroutine and is the preferred algorithm for this purpose. In this paper we incorporate into dqds a technique called aggressive early deflation, which has been applied successfully to the Hessenberg QR algorithm. Extensive numerical experiments show that aggressive early deflation often reduces the dqds runtime significantly. In addition, our theoretical analysis suggests that with aggressive early deflation, the performance of dqds is largely independent of the shift strategy. We confirm through experiments that the zero-shift version is often as fast as the shifted version. We give a detailed error analysis to prove that with our proposed deflation strategy, dqds computes all the singular values to high relative accuracy.

Key words. aggressive early deflation, dqds, singular values, bidiagonal matrix

AMS subject classifications. 65F15, 15A18

DOI. 10.1137/110821330

1. Introduction.

The differential quotient difference with shifts (dqds) algorithm computes all the singular values of an $n \times n$ bidiagonal matrix to high relative accuracy in $O(n^2)$ cost [11]. Its efficient implementation has been developed and is now available as a LAPACK subroutine DLASQ [30]. Because of its guaranteed relative accuracy and efficiency, dqds has now replaced the QR algorithm [7], which had been the default algorithm to compute the singular values of a bidiagonal matrix. The standard way of computing the singular values of a general matrix is to first apply suitable orthogonal transformations to reduce the matrix to bidiagonal form, then use dqds [6]. dqds is also a major computational kernel in the MRRR algorithm for computing orthogonal eigenvectors of a symmetric tridiagonal matrix [8, 9, 10] and the singular value decomposition (SVD) of a bidiagonal matrix [35] in $O(n^2)$ cost.

The aggressive early deflation strategy, introduced in [5], is known to greatly improve the performance of the Hessenberg QR algorithm for computing the eigenvalues of a general square matrix by deflating converged eigenvalues long before a conventional deflation strategy does. The primary contribution of this paper is the proposal of two deflation strategies for dqds based on aggressive early deflation. The first strategy is a direct specialization of aggressive early deflation to dqds. The second strategy, which takes full advantage of the bidiagonal structure of the matrix, is computationally more efficient. We present a detailed mixed forward-backward stability analysis that proves the second strategy guarantees high relative accuracy

[*]Received by the editors January 18, 2011; accepted for publication (in revised form) by J. L. Barlow October 18, 2011; published electronically January 5, 2012.
http://www.siam.org/journals/simax/33-1/82133.html

[†]Department of Mathematics, University of California, Davis, CA 95616 (ynakatsukasa@ucdavis.edu). Current address: School of Mathematics, The University of Manchester, Manchester, M13 9PL, UK (yuji.nakatsukasa@manchester.ac.uk).

[‡]Graduate School of Information Science and Technology, University of Tokyo, Tokyo 113-8656, Japan (kensuke_aishima@mist.i.u-tokyo.ac.jp). This author's work was supported in part by the Global 21 Center of Excellence "The research and training center for new development in mathematics."

[§]Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720 (ic.yamazaki@gmail.com).

of all the computed singular values. The results of extensive numerical experiments demonstrate that performing aggressive early deflation significantly reduces the solution time of dqds in many cases. We observed speedups of up to a factor 50, and in all our experiments the second strategy was at least as fast as DLASQ for any matrix larger than 3000.

With a conventional deflation strategy, the zero-shift version dqd is too slow to be practical. Hence, DLASQ adopts a sophisticated shift strategy to improve the convergence of the bottom off-diagonal element [30]. We demonstrate both theoretically and experimentally that when aggressive early deflation is incorporated, dqd is often as fast as dqds. Besides making it unnecessary to compute a shift, this observation provides a potential to parallelize our algorithm by running multiple dqd iterations in a pipelined fashion, which has been successfully done in the QR algorithm [3, 24, 34].

The structure of this paper is as follows. In section 2, we briefly review dqds and aggressive early deflation. Sections 3 and 4 describe our two aggressive early deflation strategies for dqds. Then in section 5 we present convergence analysis of dqds with aggressive early deflation, which explains its speedup and motivates the use of dqd. Numerical results are presented in section 6 to illustrate the efficiency and accuracy of our algorithms.

Notation. For a general matrix X, $\sigma_i(X)$ is its ith largest singular value, $\sigma_{\min}(X)$ and $\sigma_{\max}(X)$ are X's smallest and largest singular values, respectively, and $\lambda_i(X)$ is X's ith eigenvalue, arranged in descending order of magnitude. When the matrix is clear we just write σ_i or λ_i. We use MATLAB notation, in which $V(i, j : k)$ denotes the jth to kth elements of the ith row of V, and $V(:, \mathrm{end})$ is the last column of V. $v(k)$ denotes the kth element of a vector v, ϵ denotes the machine precision, and ϵ_x with any subscript represents a number such that $|\epsilon_x| \le \epsilon$.

2. Backgrounds.

In this section we briefly summarize the two key components of the paper, the dqds algorithm and aggressive early deflation.

2.1. The dqds algorithm.

The dqds algorithm introduced in [11] computes the singular values of a bidiagonal matrix B of the following form:

$$B = \mathrm{bidiag}\begin{pmatrix} & \sqrt{e_1} & & \cdot & & \cdot & & \sqrt{e_{n-2}} & & \sqrt{e_{n-1}} & \\ \sqrt{q_1} & & \cdot & & \cdot & & \cdot & & \sqrt{q_{n-1}} & & \sqrt{q_n} \end{pmatrix}, \tag{2.1}$$

which denotes a bidiagonal matrix whose diagonal elements are $\sqrt{q_1}, \dots, \sqrt{q_n}$ and whose off-diagonal elements are $\sqrt{e_1}, \dots, \sqrt{e_{n-1}}$. dqds is mathematically equivalent to the Cholesky LR algorithm on $B^T B$ with shifts, expressed as $(B^{(m+1)})^T B^{(m+1)} = B^{(m)}(B^{(m)})^T - s^{(m)} I$, where $B^{(m)}$ is the bidiagonal matrix of the form (2.1) obtained after m dqds iterations.

It is known [1] that if the shifts $s^{(m)}$ are taken such that $0 \le s^{(m)} < (\sigma_{\min}(B^{(m)}))^2$ at each iteration, then as $m \to \infty$, $B^{(m)}$ converges to the diagonal matrix

$$\lim_{m\to\infty} B^{(m)} = \mathrm{bidiag}\begin{pmatrix} & 0 & & \cdot & & 0 & \\ \sqrt{{\sigma_1}^2 - S} & & \cdot & & \cdot & & \sqrt{{\sigma_n}^2 - S} \end{pmatrix},$$

where $S = \sum_{m=0}^{\infty} s^{(m)}$ is the sum of the previously applied shifts. Moreover, the asymptotic convergence rate of the off-diagonal elements is described by

$$\lim_{m\to\infty} \frac{e_i^{(m+1)}}{e_i^{(m)}} = \frac{{\sigma_{i+1}}^2 - S}{{\sigma_i}^2 - S} < 1 \quad \text{for} \quad i = 1, \dots, n-1. \tag{2.2}$$

Therefore, the convergence of $e_i^{(m)}$ for $1 \leq i \leq n-2$ is linear, while the bottom off-diagonal $e_{n-1}^{(m)}$ converges superlinearly if $\sigma_n^2 - \sum_{m=0}^{\infty} s^{(m)} = 0$. In view of this, practical deflation strategies, such as that adopted in DLASQ, check whether any of the off-diagonal elements, particularly the bottom one, is small enough to be deflated [30, 2]. For detailed descriptions and analyses of dqds, see [1, 11, 30].

For notational simplicity, in the following we omit the superscript m.

2.2. Aggressive early deflation. Aggressive early deflation [5] aims to deflate eigenvalues long before conventional deflating strategies do by looking for converged eigenvalues in a $k \times k$ deflation window. This is in contrast to conventional deflating strategies, which typically look only at the bottom off-diagonal element (or two consecutive off-diagonals as described in [12, 13]). In this section, we briefly review aggressive early deflation, and discuss its specialization to the symmetric tridiagonal case.

2.2.1. Nonsymmetric case. Let H be an irreducible $n \times n$ Hessenberg matrix obtained after several Hessenberg QR iterations. H can be partitioned as

$$H = \begin{array}{c|ccc} & {\scriptstyle n-k-1} & {\scriptstyle 1} & {\scriptstyle k} \\ \hline {\scriptstyle n-k-1} & H_{11} & H_{12} & H_{13} \\ {\scriptstyle 1} & H_{21} & H_{22} & H_{23} \\ {\scriptstyle k} & 0 & H_{32} & H_{33} \end{array}, \tag{2.3}$$

where $k \geq 1$ is the window size for aggressive early deflation. To perform aggressive early deflation, one computes a Schur decomposition $H_{33} = VTV^H$, and considers the unitary transformation

$$\begin{bmatrix} I & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & V \end{bmatrix}^H \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ 0 & H_{32} & H_{33} \end{bmatrix} \begin{bmatrix} I & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & V \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} & H_{13}V \\ H_{21} & H_{22} & H_{23}V \\ 0 & t & T \end{bmatrix}, \tag{2.4}$$

where t is a $k \times 1$ vector, referred to as the *spike vector*. In practice, many of the trailing elements of t are negligibly small so that they can be set to zero. If k_ℓ elements of t are set to zero, then the corresponding k_ℓ eigenvalues are effectively deflated. In practice, more deflatable eigenvalues may be found by recomputing the Schur decomposition with a different eigenvalue ordering.

The leading $(n-k_\ell) \times (n-k_\ell)$ submatrix of (2.4) is then reduced to a Hessenberg form, then the process is repeated of applying multiple QR iterations and executing aggressive early deflation.

This process of aggressive early deflation often drastically improves the performance of the QR algorithm. In [19] it is shown that the process can be regarded as extracting converged Ritz vectors by the Krylov–Schur algorithm.

In [5] it is shown that $|t_\ell|$, the ℓth element of t, has the expression

$$|t_\ell| = \frac{\left|\prod_{i=n-k}^{n-1} h_{i+1,i}\right|}{\left|\prod_{i \neq \ell}(\mu_i - \mu_\ell)\right| |x_{k,\ell}|}, \tag{2.5}$$

where μ_i $(1 \leq i \leq k)$ is the ith diagonal of T and $x_{k,\ell}$ is the last element of the eigenvector x corresponding to μ_ℓ. The expression (2.5) partially explains why $|t_\ell|$ can be negligibly small even when none of the subdiagonal elements $h_{i+1,i}$ is.

2.2.2. Symmetric case. Since aggressive early deflation is so effective for the Hessenberg QR algorithm, a similar improvement can be expected in the symmetric tridiagonal case. Here we consider aggressive early deflation applied to the symmetric tridiagonal QR algorithm. Let A be a symmetric tridiagonal matrix, defined by

$$A = \text{tridiag}\left\{\begin{array}{ccccccccc} & b_1 & & b_2 & . & b_{n-2} & & b_{n-1} & \\ a_1 & & a_2 & & . \quad . & & a_{n-1} & & a_n \\ & b_1 & & b_2 & . & b_{n-2} & & b_{n-1} & \end{array}\right\}, \tag{2.6}$$

whose diagonals are $a_1, \dots, a_n$ and whose off-diagonals are $b_1, \dots, b_{n-1}$. We assume without loss of generality that b_i are positive.

Let $A_2 = VDV^T$ be an eigendecomposition of A's lower-right $k \times k$ submatrix A_2, where the diagonals of D are in decreasing order of magnitude. Then, we have

$$\begin{bmatrix} I & \\ & V \end{bmatrix}^T A \begin{bmatrix} I & \\ & V \end{bmatrix} = \begin{bmatrix} A_1 & u_{n-k}t^T \\ tu_{n-k}^T & D \end{bmatrix}, \tag{2.7}$$

where A_1 is the upper-left $(n-k) \times (n-k)$ submatrix of A, $u_{n-k} = [0, 0, \dots, 1]^T \in \mathbb{R}^{(n-k)\times 1}$ and the spike vector $t = [t_1, \dots, t_k]^T$ is given by $t = b_{n-k}V(1,:)^T$. If k_ℓ elements of t are smaller than a tolerance τ, for example, $\tau = \epsilon\|A\|_2$, then Weyl's theorem [27] ensures that the k_ℓ corresponding diagonal elements of D approximate the eigenvalues of A with errors bounded by τ. Hence, we deflate these elements as converged eigenvalues and obtain the symmetric matrix of size $n - k_\ell$ of the form

$$\begin{bmatrix} A_1 & u_{n-k}\tilde{t}^T \\ \tilde{t}u_{n-k}^T & D \end{bmatrix},$$

where $\tilde{t} = [t_1, \dots, t_{k-k_\ell}]^T$ and $\tilde{D} = \text{diag}(d_1, \dots, d_{k-k_\ell})$. Now, the bottom-right $(k - k_\ell + 1) \times (k - k_\ell + 1)$ arrowhead matrix needs to be tridiagonalized before we proceed to the next QR iteration. This tridiagonalization can be done in $O(k^2)$ flops by the algorithms in [25, 36].

Contrary to the nonsymmetric case, in the symmetric case there is no need to consider another eigendecomposition of A_2 with a different eigenvalue ordering, because it does not change the number of deflatable eigenvalues. The QR algorithm is known to be backward stable, although it can be forward unstable [29]. In the symmetric case the backward stability of the QR algorithm implies the computed eigenvalues are correct to $\epsilon\|A\|_2$, so they are accurate in the absolute sense. For bidiagonal matrices the dqds algorithm computes singular values with high relative accuracy, so in our algorithm development in this paper we ensure relative accuracy is maintained when aggressive early deflation is incorporated into dqds.

3. Aggressive early deflation for dqds—version 1: Aggdef(1).

In this section, we describe our first aggressive early deflation strategy for dqds, which is referred to as Aggdef(1) and is more or less a direct application of aggressive early deflation to the bidiagonal case.

3.1. Algorithm. Let $B_2 = U\Sigma V^T$ be the SVD of the lower-right $k \times k$ submatrix B_2 of a bidiagonal matrix B as in (2.1), where the singular values appear in decreasing order of magnitude. Then, we compute the orthogonal transformation

$$\begin{bmatrix} I & \\ & U^T \end{bmatrix} B \begin{bmatrix} I & \\ & V \end{bmatrix} = \begin{bmatrix} B_1 & u_{n-k}t^T \\ & \Sigma \end{bmatrix}, \tag{3.1}$$

where B_1 is the top-left $(n-k)\times(n-k)$ submatrix of B, and the right-hand side matrix has the nonzero pattern

$$\begin{bmatrix} \ddots & \ddots & & & & \\ & * & * & & & \\ & & * & * & * & * \\ & & & * & & \\ & & & & * & \\ & & & & & * \end{bmatrix}. \tag{3.2}$$

We now look for elements of the spike vector $t^T = \sqrt{e_{n-k}}V(1,:)$ that are small enough to be neglected, and deflate the corresponding diagonals of Σ to obtain the reduced $(n-k_\ell)\times(n-k_\ell)$ matrix, where k_ℓ is the number of negligible elements in t. This matrix needs to be rebidiagonalized in order to return to the dqds iterations. Algorithm 1 shows the pseudocode of this aggressive deflation strategy, which we call Aggdef(1).

Algorithm 1. Aggressive early deflation—version 1: Aggdef(1).

```
Inputs: Bidiagonal $B$, window size $k$, sum of previous shifts $S$
1: Compute the singular values of $B_2$, the lower-right $k \times k$ submatrix of $B$.
2: Compute the spike vector $t$ in (3.1).
3: Find negligible elements in $t$ and deflate converged singular values.
4: Bidiagonalize matrix of form (3.2).
```

Below we discuss the details of each step of Aggdef(1).

Computing the singular values of B_2. On line 1 of Aggdef(1), we use standard dqds (without aggressive early deflation) to compute the singular values of B_2. This generally requires $O(k^2)$ flops.

Computing the spike vector. To compute the spike vector t on line 2, the first elements of the right singular vectors V of B_2 need to be computed. This can be done by computing the full SVD of B_2, which requires at least $O(k^2)$ flops. We can reduce the cost by noting that only the first element of each singular vector is needed to compute t. This corresponds to the Gauss quadrature, whose computational algorithms are discussed in [15, 14]. However, this approach generally still requires $O(k^2)$ flops.

When to neglect elements of the spike vector. Basic singular value perturbation theory [33, p. 69] tells us that the perturbation on the computed singular values caused by neglecting the ℓth element t_ℓ of t is bounded by $|t_\ell|$. Since the unconverged singular values are greater than $\sqrt{S}$ where S is the sum of previous shifts, we can safely neglect elements of t that are smaller than $\sqrt{S}\epsilon$ (ϵ is the machine precision) without causing loss of relative accuracy of any singular value.

Rebidiagonalization process. Since the upper-left part of the matrix is already bidiagonal, we need only bidiagonalize the bottom-right $(k_0+1)\times(k_0+1)$ part of the matrix of the form (3.2), where $k_0 = k - k_\ell$.

We use a 4×4 ($k_0 = 3$) example to illustrate our bidiagonalization process, which is based on a sequence of Givens rotations:

$$\begin{bmatrix} * & * & * & * \\ & * & & \\ & & * & \\ & & & * \end{bmatrix} \xrightarrow{G_R(3,4)} \begin{bmatrix} * & * & * & 0 \\ & * & & \\ & & * & + \\ & & + & * \end{bmatrix} \xrightarrow{G_L(3,4)} \begin{bmatrix} * & * & * & \\ & * & & \\ & & * & * \\ & & 0 & * \end{bmatrix} \xrightarrow{G_R(2,3)} \begin{bmatrix} * & * & 0 & \\ & * & + & \\ & + & * & * \\ & & & * \end{bmatrix}$$

$$\xrightarrow{G_L(2,3)} \begin{bmatrix} * & * & & \\ & * & * & + \\ & 0 & * & * \\ & & & * \end{bmatrix} \xrightarrow{G_R(3,4)} \begin{bmatrix} * & * & & \\ & * & * & 0 \\ & & * & * \\ & & + & * \end{bmatrix} \xrightarrow{G_L(3,4)} \begin{bmatrix} * & * & & \\ & * & * & \\ & & * & * \\ & & 0 & * \end{bmatrix}.$$

Here, $G_L(i, j)$ (or $G_R(i, j)$) above the arrow indicates the application of a Givens rotation from the left (or right) to the ith and jth rows (or columns). "0" indicates an element that was zeroed out by the rotation, and "+" is a nonzero that was newly created. By counting the number of rotations, we can show that the total flops required for this process is at most $18k_0^2$, which is generally $O(k^2)$. We note that this process can be regarded as a bidiagonal version of the tridiagonalization algorithm of an arrowhead matrix discussed in [25, 36].

Maintaining high relative accuracy. The computation of the spike vector t and the bidiagonalization process described above can cause errors of order $\epsilon\|B_2\|_2$ in finite precision arithmetic. This may result in loss of relative accuracy for small singular values. To avoid this, we dynamically adjust the deflation window size (shrink from input size k) so that B_2 does not contain elements that are larger than $c\sqrt{S}$, where S is the sum of previous shifts and c is a modest constant. In our experiments we let $c = 1.0$.

4. Aggressive early deflation for dqds—version 2: Aggdef(2).

Numerical experiments in section 6 illustrate that Aggdef(1) described above significantly reduces the number of dqds iterations in many cases. However, computing the spike vector t and rebidiagonalizing the matrix generally require at least $O(k^2)$ flops, which can be expensive. Furthermore, Aggdef(1) requires the computation of the square roots of q_i and e_i, and it needs to consider a safe window size to guarantee the high relative accuracy. In this section, we discuss an alternative deflation strategy, Aggdef(2), which addresses these issues by seeking one deflatable singular value at a time.

4.1. Process to deflate one singular value.

To introduce Aggdef(2) we first describe Aggdef(2)-1, a simplified process to deflate one smallest singular value. As before, B_2 is the lower-right $k \times k$ submatrix of B.

Algorithm 2. Aggdef(2)-1, process for deflating one singular value.

```
Inputs: Bidiagonal $B$, window size $k$, sum of previous shifts $S$
1: Compute $s = (\sigma_{\min}(B_2))^2$.
2: Compute $\widehat{B}_2$ such that $\widehat{B}_2^T\widehat{B}_2 = B_2^TB_2 - sI$ by dstqds. Set $\widehat{B}_2(\text{end},\text{end}) \leftarrow 0$ if it
   is negligible (see (4.7)), otherwise exit.
3: Compute $\breve{B}_2 = \widehat{B}_2\prod_{i=1}^{i_0} G_R(k-i,k)$ for $i_0 = 1, \dots, k-2$ until $w$ as in (4.1)
   becomes negligible (see (4.9)), then $w \leftarrow 0$. Exit if $w$ never becomes negligible.
4: Compute $\widetilde{B}_2$ such that $\widetilde{B}_2^T\widetilde{B}_2 = \breve{B}_2^T\breve{B}_2 + sI$ by dstqds and update $B$ by replacing
   $B_2$ with $\widetilde{B}_2$.
```

On the first line of Aggdef(2)-1, only the smallest singular value of B_2 is computed using dqds, which requires $O(k)$ flops.

On lines 2 and 4, we use the dstqds algorithm [9, 10], which was originally developed to obtain the LDL^T decomposition of a shifted tridiagonal matrix in a mixed forward-backward stable manner in the relative sense. We slightly modify this algorithm to reflect the bidiagonal structure. This allows us to compute the $k \times k$ bidiagonal $\widehat{B}_2$ with $\widehat{q}_{n-k+i} = (\widehat{B}_2(i,i))^2$ and $\widehat{e}_{n-k+i} = (\widehat{B}_2(i,i+1))^2$ from B_2 such that $\widehat{B}_2^T\widehat{B}_2 = B_2^T B_2 - sI$ in about $5k$ flops, without losing relative accuracy of the computed singular values. Algorithm 3 shows the pseudocode of our dstqds algorithm.

Algorithm 3. Differential stationary qds (dstqds).

```
Inputs: $s, q_i = (B(i,i))^2$ $(i = n-k+1,\dots,n), e_i = (B(i,i+1))^2$ $(i = n-k+1,\dots,n-1)$
1: $d = -s$
2: $\widehat{q}_{n-k+1} = q_{n-k+1} + d$
3: for $i := n-k+1,\dots,n-1$ do
4:     $\widehat{e}_i = q_i e_i / \widehat{q}_i$
5:     $d = d e_i / \widehat{q}_i - s$
6:     $\widehat{q}_{i+1} = q_{i+1} + d$
7: end for
```

The bottom diagonal element of $\widehat{B}_2$ is 0 in exact arithmetic. However, in practice its computed value is nonzero, and we safely set it to 0 when it is small enough, as detailed in (4.7). In exact arithmetic, the bidiagonal elements of $\widehat{B}_2$ are all positive except for the bottom zero element. However, in finite precision arithmetic, negative elements could appear. When negative elements exist other than at the bottom diagonal, this indicates a breakdown of the Cholesky factorization. When this happens, we exit Aggdef(2)-1 and return to the dqds iterations.

On line 3 of Aggdef(2)-1, to determine if $\sqrt{s+S}$ can be deflated as a converged singular value, we apply a sequence of i_0 $(\le k-2)$ Givens transformations (note that they are not strictly Givens rotations: we apply matrices of the form $\left[\begin{smallmatrix} c & s \\ s & -c \end{smallmatrix}\right]$ to specified columns, where $c^2+s^2=1$) to $\widehat{B}_2$ on the right to compute $\breve{B}_2 = \widehat{B}_2 \prod_{i=1}^{i_0} G_R(k-i,k)$, where $G_R(k-i,k)$ is the Givens transformation acting on the $(k-i)$th and kth columns. Below we describe this process for the case $k=5$ and $i_0=3$:

$$
\begin{bmatrix} * & * & & & \\ & * & * & & \\ & & * & * & \\ & & & * & * \\ & & & & \end{bmatrix} \to \begin{bmatrix} * & * & & & \\ & * & * & & \\ & & * & * & w \\ & & & * & 0 \\ & & & & \end{bmatrix} \to \begin{bmatrix} * & * & & & \\ & * & * & & w \\ & & * & * & 0 \\ & & & * & 0 \\ & & & & \end{bmatrix} \to \begin{bmatrix} * & * & & & w \\ & * & * & & 0 \\ & & * & * & 0 \\ & & & * & 0 \\ & & & & \end{bmatrix}. \tag{4.1}
$$

Here, "0" represents the element that was zeroed out and "w" is the nonzero that was newly created by the transformation. The Givens transformations are applied so that all but the bottom diagonals of the matrices in (4.1) are positive. We stop applying the transformations once w becomes negligibly small so that (4.9) is satisfied.

Let us denote $x = \sqrt{w}$ and express the effects of the ith Givens transformation in (4.1) as follows:

$$
\begin{bmatrix} * & * & & & \\ & * & \sqrt{\widehat{e}_j} & & \\ & & \sqrt{\widehat{q}_{j+1}} & * & \sqrt{x} \\ & & & * & 0 \\ & & & & \end{bmatrix} \underset{\to}{G_R(k-i,k)} \begin{bmatrix} * & * & & & \\ & * & \sqrt{\breve{e}_j} & & \sqrt{\breve{x}} \\ & & \sqrt{\breve{q}_{j+1}} & * & 0 \\ & & & * & 0 \\ & & & & \end{bmatrix}, \tag{4.2}
$$

where $j = n - i - 1$. The triplet $(\breve{q}_{j+1}, \breve{e}_j, \breve{x})$ can be computed from $(\widehat{q}_{j+1}, \widehat{e}_j, x)$ by

$$\breve{q}_{j+1} = \widehat{q}_{j+1} + x, \quad \breve{e}_j = \frac{\widehat{q}_{j+1}\widehat{e}_j}{\widehat{q}_{j+1} + x}, \quad \breve{x} = \frac{x\widehat{e}_j}{\widehat{q}_{j+1} + x}. \tag{4.3}$$

Hence, Aggdef(2)-1 can be executed without computing square roots. Note that (4.3) provides a decreasing factor of the element x, i.e., $\breve{x} < x\frac{\widehat{e}_j}{\widehat{q}_{j+1}}$. Now, since $\widehat{B}_2$ converges to a diagonal matrix, the diagonal element $\widehat{q}_{j+1}$ is typically larger than the off-diagonal element $\widehat{e}_j$. This suggests that the size of $\breve{x}$ tends to decrease as it is chased up, i.e., $0 < \breve{x} \ll x$ if $\widehat{q}_{j+1} \gg \widehat{e}_j$. In practice, we observed that $\breve{x}$ often becomes negligible long before it reaches the top, that is, $i_0 \ll k - 2$.

4.2. Theoretical justifications. Here we express the key steps of Aggdef(2)-1 in matrix notations when it deflates a singular value. We denote by B and $\widetilde{B}$ the input and output $n \times n$ bidiagonals of Aggdef(2)-1, respectively.

Line 2 of Aggdef(2)-1 computes $\widehat{B}_2$ such that $\widehat{B}_2^T\widehat{B}_2 = B_2^TB_2 - sI$. Then, line 3 postmultiplies $\widehat{B}_2$ by the unitary matrix $\prod_{i=1}^{i_0} G_R(k-i,k) \equiv \left[\begin{smallmatrix}1 & \\ & Q\end{smallmatrix}\right]$, where Q is a $(k-1)\times(k-1)$ unitary matrix. Once w in (4.1) becomes negligible it is set to 0, and we thus obtain the bidiagonal matrix $\breve{B}_2$ such that

$$\begin{aligned}\breve{B}_2^T\breve{B}_2 &= \left(\widehat{B}_2\begin{bmatrix}1 & \\ & Q\end{bmatrix} - \begin{bmatrix} & w \\ & \end{bmatrix}\right)^T\left(\widehat{B}_2\begin{bmatrix}1 & \\ & Q\end{bmatrix} - \begin{bmatrix} & w \\ & \end{bmatrix}\right)\\ &\equiv \begin{bmatrix}1 & \\ & Q^T\end{bmatrix}B_2^TB_2\begin{bmatrix}1 & \\ & Q\end{bmatrix} - sI + E.\end{aligned} \tag{4.4}$$

We will carefully examine the "error matrix" E later in section 4.3.

Finally, line 4 computes $\widetilde{B}_2$ such that

$$\begin{aligned}\widetilde{B}_2^T\widetilde{B}_2 &= \breve{B}_2^T\breve{B}_2 + sI\\ &= \begin{bmatrix}1 & \\ & Q^T\end{bmatrix}B_2^TB_2\begin{bmatrix}1 & \\ & Q\end{bmatrix} + E.\end{aligned}$$

Since denoting $u_1 = [1, 0, 0, \dots, 0]^T \in \mathbb{R}^{k\times 1}$ and $u_{n-k} = [0, 0, 0, \dots, 1]^T \in \mathbb{R}^{n-k\times 1}$ we have

$$B^TB = \begin{bmatrix} B_1^TB_1 & \sqrt{q_{n-k}e_{n-k}}u_{n-k}u_1^T \\ \sqrt{q_{n-k}e_{n-k}}u_1u_{n-k}^T & B_2^TB_2 + e_{n-k}u_1u_1^T\end{bmatrix},$$

and noting that $u_1^T\left[\begin{smallmatrix}1 & \\ & Q\end{smallmatrix}\right] = u_1^T$ we obtain

$$\begin{aligned}\widetilde{B}^T\widetilde{B} &= \begin{bmatrix} B_1^TB_1 & \sqrt{q_{n-k}e_{n-k}}u_{n-k}u_1^T \\ \sqrt{q_{n-k}e_{n-k}}u_1u_{n-k}^T & \widetilde{B}_2^T\widetilde{B}_2 + e_{n-k}u_1u_1^T\end{bmatrix}\\ &= \begin{bmatrix}I_{n-k} & & \\ & 1 & \\ & & Q^T\end{bmatrix}\begin{bmatrix} B_1^TB_1 & \sqrt{q_{n-k}e_{n-k}}u_{n-k}u_1^T \\ \sqrt{q_{n-k}e_{n-k}}u_1u_{n-k}^T & B_2^TB_2 + e_{n-k}u_1u_1^T\end{bmatrix}\begin{bmatrix}I_{n-k} & & \\ & 1 & \\ & & Q\end{bmatrix}\\ &\quad + \begin{bmatrix} & \\ & E\end{bmatrix}\end{aligned} \tag{4.5}$$

$$= \begin{bmatrix}I_{n-k+1} & \\ & Q^T\end{bmatrix}B^TB\begin{bmatrix}I_{n-k+1} & \\ & Q\end{bmatrix} + \begin{bmatrix} & \\ & E\end{bmatrix}. \tag{4.6}$$

Later in section 4.3 we show that the condition (4.9) implies $\|E\|_2$ is small enough to ensure $|\lambda_i(\widetilde{B}^T\widetilde{B}+SI)-\lambda_i(B^TB+SI)| \le 2cS\epsilon$ for a modest constant c, from which we conclude that high relative accuracy of the computed singular values is maintained.

The above arguments tell us that the entire process of Aggdef(2)-1 (which is to peel off the submatrix B_2, "shift" it by sI, multiply a unitary matrix, shift it back, then copy it back to the original B_2) is a valid process only because the unitary matrix $\prod_{i=1}^{i_0} G_R(k-i,k)$ we right-multiply to $\widehat{B}_2$ preserves the first column: multiplying a general unitary matrix destroys the crucial equality $u_1^T \left[\begin{smallmatrix}1 & \\ & Q\end{smallmatrix}\right] = u_1^T$, and is not allowed here.

4.3. When to neglect elements. In this section, we derive conditions that ensure neglecting the bottom diagonal element $\sqrt{\widehat{q}_n}$ of $\widehat{B}_2$ and the error matrix E in (4.4) does not cause loss of relative accuracy of the computed singular values.

We first examine when it is safe to neglect a nonzero computed $\widehat{q}_n$.

First suppose that $\widehat{q}_n$ is positive. Since setting $\widehat{q}_n$ to zero only changes the bottom diagonal of $\widehat{B}_2^T\widehat{B}_2+(s+S)I$ by $\widehat{q}_n$, Weyl's theorem ensures that high relative accuracy of the unconverged singular values is maintained if $\widehat{q}_n < cS\epsilon$ for a modest constant c.

Next consider the case $\widehat{q}_n < 0$. dstqds of Algorithm 3 computes $\widehat{q}_n$ as $\widehat{q}_n = q_n + d$, where d does not depend on q_n. Hence, setting $\widehat{q}_n$ to 0 is equivalent to replacing q_n of the original matrix $B_2^TB_2$ with $q_n - \widehat{q}_n$. Weyl's theorem applied to $B_2B_2^T + SI$ guarantees that high relative accuracy of the singular values is preserved if $|\widehat{q}_n| < cS\epsilon$.

In summary, we can safely neglect $\widehat{q}_n$ if

$$|\widehat{q}_n| \le cS\epsilon. \tag{4.7}$$

We next examine when to neglect $w = \sqrt{x}$ (or equivalently E) when applying the Givens transformations. After setting $\widehat{q}_n$ to zero and applying i_0 Givens transformations to $\widehat{B}_2$, we have $\widehat{B}_2^T\widehat{B}_2 + sI = B_2^TB_2$, where $\widehat{B}_2$ is of the form

$$\widehat{B}_2 = \begin{bmatrix} * & * & & & \\ & \sqrt{\widehat{q}_j} & \sqrt{\breve{e}_j} & & \sqrt{x} \\ & & * & * & \\ & & & * & \\ & & & & 0 \end{bmatrix}, \tag{4.8}$$

where $j = n - i_0 - 1$ is the row index of x. Then, recalling $x = w^2$, we see that E as in (4.4), (4.6) is

$$E = \begin{bmatrix} & & -\sqrt{x\widehat{q}_j} \\ & & -\sqrt{x\breve{e}_j} \\ & & \\ -\sqrt{x\widehat{q}_j} & -\sqrt{x\breve{e}_j} & x \end{bmatrix}.$$

Hence, Weyl's theorem ensures that the perturbation to the eigenvalues of $\widehat{B}_2^T\widehat{B}_2 + (S+s)I$ caused by setting x to zero is bounded by $\|E\|_2 \le \sqrt{x(\widehat{q}_j + \breve{e}_j)} + x$. Therefore, it is safe to neglect x when $\sqrt{x(\widehat{q}_j + \breve{e}_j)} \le cS\epsilon$ and $x \le cS\epsilon$, or equivalently

$$x(\widehat{q}_j + \breve{e}_j) \le (cS\epsilon)^2 \quad \text{and} \quad x \le cS\epsilon. \tag{4.9}$$

In our numerical experiments, we set $c = 1.0$ in both (4.7) and (4.9).

We note that as the dqds iterations proceed, the sum of the previous shifts S typically becomes larger than $\widehat{q}_n, \widehat{q}_j$, and $\breve{e}_j$, so that the three inequalities all become more likely to hold. As a result, more singular values are expected to be deflated.

In the discussion here and in section 3 we use only the Weyl bound. If some information on the gap between singular values is available, a sharper, quadratic perturbation bound can be used [20, 23]. We do not use such bounds here because estimating the gap is a nontrivial task, involving the whole matrix B instead of just B_2 or $\widehat{B}_2$, and experiments suggest that the improvement we get is marginal.

Let us give more details. In practice we are unwilling to spend $O(n)$ flops for estimating the gap, so instead we estimate the gap using only $\widehat{B}_2$. One choice is to estimate a lower bound for the smallest singular value $\sigma_{\min}$ of the top-left $(k-1) \times (k-1)$ submatrix of B_2, and apply the bound in [20] to obtain the bound

$$|\sigma_i(\widehat{B}_2) - \sigma_i(\widehat{B}_{2,0})| \leq \frac{2x}{\sigma_{\min} + \sqrt{\sigma_{\min}^2 + 4x}}, \tag{4.10}$$

where $\widehat{B}_{2,0}$ is the matrix obtained by setting x to 0 in (4.8). We emphasize that (4.10) is not a bound in terms of the entire matrix B, which is what we need to guarantee the desired accuracy. In practice estimating $\sigma_{\min}$ can also be costly, so we attempt to estimate it simply by $\sqrt{\widehat{q}_{n-1}}$. Combining this with (4.10), we tried neglecting the x values when

$$\frac{2x}{\sqrt{\widehat{q}_{n-1}} + \sqrt{\widehat{q}_{n-1} + 4x}} \leq \sqrt{cS\epsilon}. \tag{4.11}$$

We observed through experiments that using this criterion sometimes results in loss of relative accuracy. Moreover, there was no performance gain on average, no particular case giving more than 5% speedup. To guarantee relative accuracy while using a quadratic perturbation bound we need a more complicated and restrictive criterion than (4.11), which is unlikely to provide a faster implementation. Therefore we decide to use the simple and safe criterion (4.9) using Weyl's theorem.

4.4. High relative accuracy of Aggdef(2)-1 in floating point arithmetic. Here we show that Aggdef(2)-1 preserves high relative accuracy of singular values. We use the standard model of floating point arithmetic

$$fl(x \circ y) = (x \circ y)(1 + \delta) = (x \circ y)/(1 + \eta),$$

where $\circ \in \{+, -, \times, \div\}$ and $\delta,\ \eta$ satisfy

$$(1 + \epsilon)^{-1}(x \circ y) \leq fl(x \circ y) \leq (1 + \epsilon)(x \circ y).$$

For the error analysis below, we need to define $\widehat{B}_2$ clearly. In this subsection, we let $\widehat{B}_2$ be the first bidiagonal matrix in (4.1). In other words, $\widehat{B}_2$ is obtained by computing the dstqds transform from B_2 in floating point arithmetic, then setting the bottom element $\widehat{q}_n$ to 0, supposing that (4.7) is satisfied.

First we show that high relative accuracy of singular values of the lower-right submatrices B_2 is preserved. We do this by using direct mixed stability analysis with respect to B_2, $\widehat{B}_2$, $\breve{B}_2$, $\widetilde{B}_2$, using an argument similar to that in [11, sect. 7.2].

Let us first analyze the transformation from B_2 to $\widehat{B}_2$. We introduce two ideal matrices $\dot{B}_2$, $\ddot{\widehat{B}}_2$ satisfying $\ddot{\widehat{B}}_2^T \ddot{\widehat{B}}_2 = \dot{B}_2^T \dot{B}_2 - sI$ for all but the bottom element $\ddot{\widehat{q}}_n$ of $\ddot{\widehat{B}}_2$, which is set to 0 (note that this is equivalent to setting $\widehat{q}_n$ to 0). We seek such $\dot{B}_2$ and $\ddot{\widehat{B}}_2$ so that $\dot{B}_2$ is a small relative entrywise perturbation of B_2 and $\ddot{\widehat{B}}_2$ is a small relative entrywise perturbation of $\widehat{B}_2$. In this subsection, we use a *dot* to denote backward-type ideal matrices, and a *double dot* to denote forward-type ideal matrices. The ith main and off-diagonals of $\dot{B}_2$ are $\dot{q}_i$ and $\dot{e}_i$, and those of $\ddot{\widehat{B}}_2$ are $\ddot{\widehat{q}}_i$ and $\ddot{\widehat{e}}_i$.

All the results in this subsection state errors in terms of the relative error, and we use the statement "$\dot{q}_i$ differs from q_i by $\alpha\epsilon$" to mean $(1+\epsilon)^{-\alpha}\dot{q}_i \le q_i \le (1+\epsilon)^{\alpha}\dot{q}_i$ ($\simeq (1+\alpha\epsilon)\dot{q}_i$). Below we specify the values of d as in Algorithm 3 and x as in (4.3) by denoting them with subscripts d_i and x_i.

LEMMA 4.1. *Concerning the mixed stability analysis in the transformation from B_2 to $\widehat{B}_2$, $\dot{q}_i$ differs from q_i by ϵ and $\dot{e}_i$ differs from e_i by 3ϵ, and $\widehat{q}_i$ differs from $\ddot{\widehat{q}}_i$ by 2ϵ and $\widehat{e}_i$ differs from $\ddot{\widehat{e}}_i$ by 2ϵ.*

Proof. From the dstqds transform, we have

$$\begin{aligned}
&\widehat{e}_i = (q_i e_i/\widehat{q}_i)(1+\epsilon_{i,*1})(1+\epsilon_{i,/}),\\
&d_{i+1} = ((d_i e_i/\widehat{q}_i)(1+\epsilon_{i+1,*2})(1+\epsilon_{i,/}) - s)(1+\epsilon_{i+1,-}),\\
&\widehat{q}_{i+1} = (q_{i+1}+d_{i+1})(1+\epsilon_{i+1,+}).
\end{aligned}$$

From these equalities for d_{i+1} and $\widehat{q}_{i+1}$, we have

$$\frac{d_{i+1}}{1+\epsilon_{i+1,-}} = \frac{d_i e_i(1+\epsilon_{i+1,*2})(1+\epsilon_{i,/})}{(q_i+d_i)(1+\epsilon_{i,+})} - s.$$

This tells us how to define $\dot{B}_2$. We let them be

$$\dot{d}_{i+1} = d_{i+1}/(1+\epsilon_{i+1,-}), \tag{4.12}$$
$$\dot{q}_{i+1} = q_{i+1}/(1+\epsilon_{i+1,-}), \tag{4.13}$$
$$\dot{e}_i = e_i(1+\epsilon_{i+1,*2})(1+\epsilon_{i+1,/})/(1+\epsilon_{i,+}). \tag{4.14}$$

Then we see that

$$\dot{d}_{i+1} = \dot{d}_i\dot{e}_i/(\dot{q}_i+\dot{d}_i) - s,$$

so the recurrence for $\dot{d}_{i+1}$ of the dstqds transformation is satisfied. We then define the elements of the ideal $\ddot{\widehat{B}}_2$ as

$$\ddot{\widehat{q}}_{i+1} = \widehat{q}_{i+1}/(1+\epsilon_{i+1,+})(1+\epsilon_{i+1,-}), \tag{4.15}$$
$$\ddot{\widehat{e}}_i = \widehat{e}_i(1+\epsilon_{i+1,*2})/(1+\epsilon_{i,*1}). \tag{4.16}$$

Then the dstqds transformation from $\dot{B}_2$ to $\ddot{\widehat{B}}_2$, expressed in matrix form as $\ddot{\widehat{B}}_2^T\ddot{\widehat{B}}_2 = \dot{B}_2^T\dot{B}_2 + sI$, is satisfied. ∎

We next prove two lemmas regarding the connections between $\widehat{B}_2$, $\breve{B}_2$, and $\widetilde{B}_2$, and their corresponding ideal matrices denoted with dots. Similarly to $\widehat{B}_2$, the bidiagonal matrix $\breve{B}_2$ is here defined as the $(k-1)\times(k-1)$ deflated matrix obtained after applying the Givens transformations and setting x to 0.

LEMMA 4.2. *Concerning the mixed stability analysis in the transformation from $\widehat{B}_2$ to $\breve{B}_2$, we have $\dot{\widehat{q}}_i = \widehat{q}_i$, and $\dot{\widehat{e}}_i$ differs from $\widehat{e}_i$ by 3ϵ, $\breve{q}_i$ differs from $\ddot{\breve{q}}_i$ by ϵ, and $\breve{e}_i$ differs from $\ddot{\breve{e}}_i$ by 2ϵ.*

$$\begin{tikzcd}
B_2 \arrow[r, "\text{Step (a)}", "\text{computed}"'] \arrow[d, "\begin{array}{l}\text{change}\\ q_{n-k+i}\\ \text{by } \epsilon\\ e_{n-k+i}\\ \text{by } 3\epsilon\end{array}"] & \widehat{B}_2\ (\grave{B}_2)\ (\grave{B}_2)\widehat{B}_2 \arrow[r, "\text{Step (b)}", "\text{computed}"'] \arrow[d, shift left=8, "\begin{array}{l}\text{change}\\ \widehat{q}_{n-k+i}\\ \text{by } 0\\ \widehat{e}_{n-k+i}\\ \text{by } 3\epsilon\end{array}"] & \breve{B}_2\ (\acute{B}_2)\ (\acute{B}_2)\breve{B}_2 \arrow[r, "\text{Step (c)}", "\text{computed}"'] \arrow[d, shift left=8, "\begin{array}{l}\text{change}\\ \breve{q}_{n-k+i}\\ \text{by } \epsilon\\ \breve{e}_{n-k+i}\\ \text{by } 3\epsilon\end{array}"] & \widetilde{B}_2 \\
\dot{B}_2 \arrow[r, "\text{exact}"'] & \ddot{\widehat{B}}_2\ (\ddot{\grave{B}}_2)\ (\dot{\grave{B}}_2)\dot{\widehat{B}}_2 \arrow[r, "\text{exact}"'] \arrow[u, shift left=8, "\begin{array}{l}\text{change}\\ \widehat{q}_{n-k+i}\\ \text{by } 2\epsilon\\ \widehat{e}_{n-k+i}\\ \text{by } 2\epsilon\end{array}"'] & \ddot{\breve{B}}_2\ (\ddot{\acute{B}}_2)\ (\dot{\acute{B}}_2)\dot{\breve{B}}_2 \arrow[r, "\text{exact}"'] \arrow[u, shift left=8, "\begin{array}{l}\text{change}\\ \breve{q}_{n-k+i}\\ \text{by } \epsilon\\ \breve{e}_{n-k+i}\\ \text{by } 2\epsilon\end{array}"'] & \ddot{\widetilde{B}}_2 \arrow[u, "\begin{array}{l}\text{change}\\ \widetilde{q}_{n-k+i}\\ \text{by } 2\epsilon\\ \widetilde{e}_{n-k+i}\\ \text{by } 2\epsilon\end{array}"']
\end{tikzcd}$$

FIG. 4.1. *Effect of roundoff.*

Proof. Recalling (4.3) we have

$$\begin{aligned}
\breve{q}_{i+1} &= (\widehat{q}_{i+1} + x_{i+1})(1 + \epsilon_{i+1,+}),\\
\breve{e}_i &= \frac{\widehat{q}_{i+1}\widehat{e}_i(1 + \epsilon_{i+1,*1})(1 + \epsilon_{i+1,/})}{(\widehat{q}_{i+1} + x_{i+1})(1 + \epsilon_{i+1,+})},\\
x_i &= \frac{x_{i+1}\widehat{e}_i(1 + \epsilon_{i+1,*2})(1 + \epsilon_{i+1,/})}{(\widehat{q}_{i+1} + x_{i+1})(1 + \epsilon_{i+1,+})}.
\end{aligned}$$

Hence, we define the variables for the ideal matrix $\dot{\widehat{B}}_2$ as

$$\begin{aligned}
\dot{x}_i &= x_i,\\
\dot{\widehat{q}}_{i+1} &= \widehat{q}_{i+1},\\
\dot{\widehat{e}}_i &= \widehat{e}_i(1 + \epsilon_{i+1,*2})(1 + \epsilon_{i+1,/})/(1 + \epsilon_{i+1,+}).
\end{aligned}$$

Then it follows that

$$\dot{x}_i = \frac{\dot{x}_{i+1}\dot{\widehat{e}}_i}{\dot{\widehat{q}}_{i+1} + \dot{x}_{i+1}},$$

so the recurrence for $\dot{x}_i$ is satisfied.

Similarly, we define the variables for the ideal matrix $\ddot{\breve{B}}_2$ as

$$\begin{aligned}
\ddot{\breve{q}}_{i+1} &= \breve{q}_{i+1}/(1 + \epsilon_{i+1,+}),\\
\ddot{\breve{e}}_i &= \breve{e}_i(1 + \epsilon_{i+1,*2})/(1 + \epsilon_{i+1,*1}).
\end{aligned}$$

Then the transformation from $\dot{\widehat{B}}_2$ to $\ddot{\breve{B}}_2$ is realized in exact arithmetic. ☐

LEMMA 4.3. *Concerning the mixed stability analysis in the transformation from $\breve{B}_2$ to $\widetilde{B}_2$, $\dot{\breve{q}}_i$ differs from $\breve{q}_i$ by ϵ, $\dot{\breve{e}}_i$ differs from $\breve{e}_i$ by 3ϵ, $\widetilde{q}_i$ differs from $\ddot{\widetilde{q}}_i$ by 2ϵ, and $\widetilde{e}_i$ differs from $\ddot{\widetilde{e}}_i$ by 2ϵ.*

Proof. The proof is the same as in Lemma 4.1. ☐

The above results are summarized in Figure 4.1. We will discuss the matrices $\grave{B}_2$, $\acute{B}_2$ shortly.

Combining Lemma 4.1, (4.7), and a result by Demmel and Kahan [7, Corollary 2] that shows that the relative perturbation of bidiagonal elements produces only small relative perturbation in the singular values, we see that relative accuracy of the deflated singular value is preserved. We next show that Aggdef(2)-1 preserves high relative accuracy of all singular values of the whole bidiagonal matrix B.

The key idea of the proof below is to define bidiagonal matrices $\grave{B}_2$ and $\acute{B}_2$ satisfying $\grave{B}_2^T\grave{B}_2 = \widehat{B}_2^T\widehat{B}_2 + sI$ and $\acute{B}_2^T\acute{B}_2 = \breve{B}_2^T\breve{B}_2 + sI$ in exact arithmetic, so that we can discuss solely in terms of matrices that are not shifted by $-sI$. We first consider the bidiagonal matrices $\grave{B}_2$ and $\ddot{\grave{B}}_2$ satisfying $\grave{B}_2^T\grave{B}_2 = \widehat{B}_2^T\widehat{B}_2 + sI$ and $\ddot{\grave{B}}_2^T\ddot{\grave{B}}_2 = \ddot{\widehat{B}}_2^T\ddot{\widehat{B}}_2 + sI$. We have the following lemma.

LEMMA 4.4. *Concerning the relative errors between* $\grave{B}_2$ *and* $\ddot{\grave{B}}_2$, $\grave{q}_{n-k+i}$ *differs from* $\ddot{\grave{q}}_{n-k+i}$ *by* $4i\epsilon$ *and* $\grave{e}_{n-k+i}$ *differs from* $\ddot{\grave{e}}_{n-k+i}$ *by* $4(i+1)\epsilon$ *for* $i = 1, \dots, k$.

Proof. The dstqds transformation from $\widehat{B}_2$ to $\grave{B}_2$ gives

$$\grave{e}_i = \widehat{q}_i\widehat{e}_i/\grave{q}_i, \quad \grave{d}_{i+1} = \grave{d}_i\widehat{e}_i/\grave{q}_i + s, \quad \grave{q}_{i+1} = \widehat{q}_{i+1} + \grave{d}_{i+1}. \tag{4.17}$$

Hence

$$\grave{d}_{i+1} = \frac{\grave{d}_i\widehat{e}_i}{\widehat{q}_i + \grave{d}_i} + s = \frac{\widehat{e}_i}{\widehat{q}_i/\grave{d}_i + 1} + s. \tag{4.18}$$

Regarding the variables of $\ddot{\widehat{B}}$, by Lemma 4.1 the relative perturbations of $\widehat{q}_i$, $\widehat{e}_i$ from $\ddot{\widehat{q}}_i$, $\ddot{\widehat{e}}_i$ are both 2ϵ, that is,

$$(1+\epsilon)^{-2}\widehat{q}_i \le \ddot{\widehat{q}}_i \le (1+\epsilon)^2\widehat{q}_i, \tag{4.19}$$

$$(1+\epsilon)^{-2}\widehat{e}_i \le \ddot{\widehat{e}}_i \le (1+\epsilon)^2\widehat{e}_i. \tag{4.20}$$

Moreover, similar to (4.18) we have

$$\ddot{\grave{d}}_{i+1} = \frac{\ddot{\widehat{e}}_i}{\ddot{\widehat{q}}_i/\ddot{\grave{d}}_i + 1} + s.$$

Note that in the computation of $\grave{d}_i$, $\ddot{\grave{d}}_i$, subtraction is not involved and $\grave{d}_{n-k+1} = \ddot{\grave{d}}_{n-k+1} = s$. We claim that the relative perturbation of $\grave{d}_{n-k+i}$ of $\widehat{B}_2$ from $\ddot{\grave{d}}_{n-k+i}$ of $\ddot{\widehat{B}}_2$ is less than $4i\epsilon$:

$$(1+\epsilon)^{-4i}\grave{d}_{n-k+i} \le \ddot{\grave{d}}_{n-k+i} \le (1+\epsilon)^{4i}\grave{d}_{n-k+i} \tag{4.21}$$

for $i = 1, \dots, k$. We can prove (4.21) by backward induction on i. For $i = k$ it is obvious. Next, if (4.21) holds for $i = j-1$, then for $i = j$ we have

$$\begin{aligned}
\ddot{\grave{d}}_{n-k+j} &= \frac{\ddot{\widehat{e}}_{n-k+j-1}}{\ddot{\widehat{q}}_{n-k+j-1}/\ddot{\grave{d}}_{n-k+j-1} + 1} + s \\
&\le \frac{\widehat{e}_{n-k+j-1}(1+\epsilon)^2}{\widehat{q}_{n-k+j-1}(1+\epsilon)^{-2}/\grave{d}_{n-k+j-1}(1+\epsilon)^{4(j-1)} + 1} + s \\
&\le \frac{\grave{d}_{n-k+j-1}\widehat{e}_{n-k+j-1}(1+\epsilon)^{4j}}{\widehat{q}_{n-k+j-1}/\grave{d}_{n-k+j-1} + (1+\epsilon)^{4j-2}} + s \\
&\le \frac{\grave{d}_{n-k+j-1}\widehat{e}_{n-k+j-1}(1+\epsilon)^{4j}}{\widehat{q}_{n-k+j-1}/\grave{d}_{n-k+j-1} + 1} + (1+\epsilon)^{4j}s \\
&= (1+\epsilon)^{4j}\grave{d}_{n-k+j}.
\end{aligned}$$

The first inequality in (4.21) can be shown similarly. Using (4.17), (4.20), and (4.21) we get

$$\begin{aligned}(1+\epsilon)^{-4i}\grave{q}_{n-k+i} &= (1+\epsilon)^{-4i}(\widehat{q}_{n-k+i}+\grave{d}_{n-k+i})\\ &\le \ddot{\widehat{q}}_{n-k+i}+\ddot{\grave{d}}_{n-k+i}(=\ddot{\grave{q}}_{n-k+i})\\ &\le (1+\epsilon)^{4i}(\widehat{q}_{n-k+i}+\grave{d}_{n-k+i})\\ &= (1+\epsilon)^{4i}\grave{q}_{n-k+i}.\end{aligned}$$

Therefore,

$$(1+\epsilon)^{-4i}\grave{q}_{n-k+i} \le \ddot{\grave{q}}_{n-k+i} \le (1+\epsilon)^{4i}\grave{q}_{n-k+i} \tag{4.22}$$

for $i = 1, \dots, k$. Therefore, the relative error between $\grave{q}_{n-k+i}$ and $\ddot{\grave{q}}_{n-k+i}$ is $4i\epsilon$. Similarly, we estimate the relative error between $\grave{e}_{n-k+i}$ and $\ddot{\grave{e}}_{n-k+i}$. We see that

$$\begin{aligned}(1+\epsilon)^{-4(i+1)}\grave{e}_{n-k+i} &= (1+\epsilon)^{-4(i+1)}\widehat{q}_{n-k+i}\widehat{e}_{n-k+i}/\grave{q}_{n-k+i}\\ &\le \ddot{\widehat{q}}_{n-k+i}\ddot{\widehat{e}}_{n-k+i}/\ddot{\grave{q}}_{n-k+i}(=\ddot{\grave{e}}_{n-k+i})\\ &\le (1+\epsilon)^{4(i+1)}\widehat{q}_{n-k+i}\widehat{e}_{n-k+i}/\grave{q}_{n-k+i}\\ &= (1+\epsilon)^{4(i+1)}\grave{e}_{n-k+i},\end{aligned}$$

where we used (4.17), (4.20), (4.21), and (4.22). Therefore, the relative error between $\grave{e}_{n-k+i}$ and $\ddot{\grave{e}}_{n-k+i}$ is $4(i+1)\epsilon$ for $i = 1, \dots, k-1$. ◻

Now, from the $n \times n$ bidiagonal matrix B we define a new $n \times n$ bidiagonal matrix $\dot{B}$ obtained by replacing the lower right $k \times k$ part by $\dot{B}_2$, and let $\dot{\sigma}_i$ be its ith singular value. In general, in this subsection we denote by $\vec{B}$ (in which $\to$ represents any accent) an $n \times n$ bidiagonal matrix obtained by replacing the lower right $k \times k$ part of B by $\vec{B}_2$, and denote by $\vec{\sigma}_i$ the singular values of $\vec{B}$.

By Lemma 4.1 and Demmel–Kahan's result [7, Corollary 2], we have

$$\prod_{i=1}^{k}\sqrt{(1+\epsilon)^{-3}}\prod_{i=1}^{k-1}\sqrt{(1+\epsilon)^{-1}}\sigma_i \le \dot{\sigma}_i \le \prod_{i=1}^{k}\sqrt{(1+\epsilon)^{3}}\prod_{i=1}^{k-1}\sqrt{(1+\epsilon)}\sigma_i,$$

where $\dot{\sigma}_i$ denotes the ith singular value of $\dot{B}$. Here the square roots come from the facts $B_{i,i} = \sqrt{q_i}$ and $B_{i,i+1} = \sqrt{e_i}$. Using $\prod_{i=1}^{k}\sqrt{(1+\epsilon)^3}\prod_{i=1}^{k-1}\sqrt{(1+\epsilon)} \le \prod_{i=1}^{k}(1+\epsilon)^2 \le \exp(2k\epsilon)$ and an analogous inequality for the lower bound we get

$$\exp(2k\epsilon)^{-1}\sigma_i \le \dot{\sigma}_i \le \exp(2k\epsilon)\sigma_i \quad \text{for} \quad i = 1, \dots, n. \tag{4.23}$$

Similarly, regarding $\grave{B}$ and $\ddot{\grave{B}}$ (whose lower right submatrix is replaced by $\grave{B}_2$ and $\ddot{\grave{B}}_2$), by Lemma 4.4 and Demmel–Kahan's result we have

$$\exp((2k^2+4k)\epsilon)^{-1}\ddot{\grave{\sigma}}_i \le \grave{\sigma}_i \le \exp((2k^2+4k)\epsilon)\ddot{\grave{\sigma}}_i \quad \text{for} \quad i = 1, \dots, n, \tag{4.24}$$

where $\ddot{\grave{\sigma}}_i$ and $\grave{\sigma}_i$ are the singular values of $\ddot{\grave{B}}$ and $\grave{B}$. This analysis is summarized in Figure 4.2.

Recall that assuming (4.7) is satisfied, we set $\widehat{q}_n$ and $\ddot{\widehat{q}}_n$ to 0. We next bound the effect of the operation $\ddot{\widehat{q}}_n \leftarrow 0$ on the singular values of $\dot{B}_2$ and $\ddot{\grave{B}}_2$. Noting that the bounds in Lemma 4.1 hold for the bottom elements $\widehat{q}_n$ and $\ddot{\widehat{q}}_n$ even before setting

$$\begin{tikzcd}
\widehat{B}_2 \arrow[r, "\text{exact}"'] & \grave{B}_2 \arrow[d, leftrightarrow, "\begin{array}{l}\text{change}\\ \grave{q}_{n-k+i}\\ \text{by } 4i\epsilon\\ \grave{e}_{n-k+i}\\ \text{by } 4(i+1)\epsilon\end{array}"] & \grave{B} \arrow[d, leftrightarrow, "\begin{array}{l}\text{perturbation of}\\ \text{singular values are}\\ \text{less than } (2k^2+4k)\epsilon\end{array}"] \\
\ddot{\widehat{B}}_2 \arrow[u, "\begin{array}{l}\text{change}\\ \widehat{q}_{n-k+i}\\ \text{by } 2\epsilon\\ \widehat{e}_{n-k+i}\\ \text{by } 2\epsilon\end{array}"'] \arrow[r, "\text{exact}"'] & \ddot{\grave{B}}_2 & \ddot{\grave{B}}
\end{tikzcd}$$

FIG. 4.2. *Effect of roundoff for singular values of* $\ddot{\grave{B}}$ *and* $\grave{B}$.

them to 0, by the argument leading to (4.7) we obtain (recall that $\sqrt{\sigma_i^2+S}$ are the singular values to be computed)

$$(1-c(1+\epsilon)^2\epsilon)(\acute{\sigma}_i^2+S) \le \ddot{\grave{\sigma}}_i^2+S \le (1+c(1+\epsilon)^2\epsilon)(\acute{\sigma}_i^2+S) \quad \text{for} \quad i=1,\dots,n. \tag{4.25}$$

For simplicity we rewrite (4.25) as

$$\exp(c'\epsilon)^{-1}\sqrt{\acute{\sigma}_i^2+S} \le \sqrt{\ddot{\grave{\sigma}}_i^2+S} \le \exp(c'\epsilon)\sqrt{\acute{\sigma}_i^2+S} \quad \text{for} \quad i=1,\dots,n, \tag{4.26}$$

where $c'(\approx\ c/2)$ is a suitable constant such that the original inequality (4.25) is satisfied.

Since $S \ge 0$, we see that (4.23) implies $\exp(2k\epsilon)^{-1}\sqrt{\sigma_i^2+S} \le \sqrt{\acute{\sigma}_i^2+S} \le \exp(2k\epsilon)\sqrt{\sigma_i^2+S}$, and an analogous inequality holds for (4.24). Combining the three bounds we obtain a bound for the relative error in step (a) in Figure 4.1:

$$\exp((2k^2+6k+c')\epsilon)^{-1}\sqrt{\sigma_i^2+S} \le \sqrt{\grave{\sigma}_i^2+S} \le \exp((2k^2+6k+c')\epsilon)\sqrt{\sigma_i^2+S} \tag{4.27}$$

for $i=1,\dots,n$.

We next discuss step (b) in Figure 4.1. Similarly to the above discussion, we define a bidiagonal matrix $\dot{\grave{B}}_2$ satisfying $\dot{\grave{B}}_2^T\dot{\grave{B}}_2 = \dot{\widehat{B}}_2^T\dot{\widehat{B}}_2 + sI$ in exact arithmetic.

LEMMA 4.5. *$\dot{\grave{q}}_{n-k+i}$ differs from $\grave{q}_{n-k+i}$ by $3i\epsilon$ and $\dot{\grave{e}}_{n-k+i}$ differs from $\grave{e}_{n-k+i}$ by $3(i+1)\epsilon$ for $i=1,\dots,k$.*

Proof. Similarly to (4.19) and (4.20), by Lemma 4.2 we have

$$\dot{\widehat{q}}_i = \widehat{q}_i, \tag{4.28}$$

$$(1+\epsilon)^{-3}\widehat{e}_i \le \dot{\widehat{e}}_i \le (1+\epsilon)^3\widehat{e}_i. \tag{4.29}$$

Therefore, similarly to (4.21), we have

$$(1+\epsilon)^{-3i}\grave{d}_{n-k+i} \le \dot{\grave{d}}_{n-k+i} \le (1+\epsilon)^{3i}\grave{d}_{n-k+i}, \tag{4.30}$$

so the proof is completed as in Lemma 4.4. ∎

Therefore, we have

$$\exp((3k^2/2+3k)\epsilon)^{-1}\grave{\sigma}_i \le \dot{\grave{\sigma}}_i \le \exp((3k^2/2+3k)\epsilon)\grave{\sigma}_i \quad \text{for} \quad i=1,\dots,n. \tag{4.31}$$

We next define and compare the bidiagonal matrices $\acute{B}_2$ and $\ddot{\acute{B}}_2$ satisfying $\acute{B}_2^T\acute{B}_2 = \breve{B}_2^T\breve{B}_2 + sI$ and $\ddot{\acute{B}}_2^T\ddot{\acute{B}}_2 = \ddot{\breve{B}}_2^T\ddot{\breve{B}}_2 + sI$ in exact arithmetic.

Lemma 4.6. *$\acute{q}_{n-k+i}$ differs from $\ddot{\acute{q}}_{n-k+i}$ by $3i\epsilon$ and $\acute{e}_{n-k+i}$ differs from $\ddot{\acute{e}}_{n-k+i}$ by $3(i+1)\epsilon$ for $i = 1, \dots, k$.*

Proof. By Lemma 4.2 we have

$$(1+\epsilon)^{-1}\breve{q}_i \le \ddot{\breve{q}}_i \le (1+\epsilon)\breve{q}_i,$$
$$(1+\epsilon)^{-2}\breve{e}_i \le \ddot{\breve{e}}_i \le (1+\epsilon)^2\breve{e}_i.$$

Therefore, similarly to (4.21), we have

$$(1+\epsilon)^{-3i}\acute{d}_{n-k+i} \le \ddot{\acute{d}}_{n-k+i} \le (1+\epsilon)^{3i}\acute{d}_{n-k+i}. \tag{4.32}$$

The same argument as in Lemma 4.4 completes the proof. ▯

Therefore, we have

$$\exp((3k^2/2+3k)\epsilon)^{-1}\ddot{\acute{\sigma}}_i \le \acute{\sigma}_i \le \exp((3k^2/2+3k)\epsilon)\ddot{\acute{\sigma}}_i \quad \text{for} \quad i = 1, \dots, n. \tag{4.33}$$

Recall that the kth column of $\breve{B}_2$ is set to the zero vector when (4.9) is satisfied, and hence by Lemma 4.2 we see that

$$(1-2c(1+\epsilon)\epsilon)(\dot{\acute{\sigma}}_i^2 + S) \le (\ddot{\acute{\sigma}}_i^2 + S) \le (1+2c(1+\epsilon)\epsilon)(\dot{\acute{\sigma}}_i^2 + S) \quad \text{for} \quad i = 1, \dots, n.$$

For simplicity, we rewrite the inequalities using a suitable constant $c''(\approx c)$ as

$$\exp(c''\epsilon)^{-1}\sqrt{\dot{\acute{\sigma}}_i^2 + S} \le \sqrt{\ddot{\acute{\sigma}}_i^2 + S} \le \exp(c''\epsilon)\sqrt{\dot{\acute{\sigma}}_i^2 + S} \quad \text{for} \quad i = 1, \dots, n. \tag{4.34}$$

Combining (4.31), (4.33), and (4.34) we get

$$\exp((3k^2+6k+c'')\epsilon)^{-1}\sqrt{\grave{\sigma}_i^2 + S} \le \sqrt{\acute{\sigma}_i^2 + S} \le \exp((3k^2+6k+c'')\epsilon)\sqrt{\grave{\sigma}_i^2 + S} \tag{4.35}$$

for $i = 1, \dots, n$.

Finally, we bound the relative error caused in step (c) of Figure 4.1. Let $\dot{\acute{B}}_2$ be a bidiagonal matrix satisfying $\dot{\acute{B}}_2^T\dot{\acute{B}}_2 = \dot{\breve{B}}_2^T\dot{\breve{B}}_2 + sI$ in exact arithmetic (note that we have exactly $\dot{\acute{B}}_2 = \ddot{\tilde{B}}_2$). We have the following lemma comparing $\acute{B}_2$ and $\dot{\acute{B}}_2$.

Lemma 4.7. *$\acute{q}_{n-k+i}$ differs from $\dot{\acute{q}}_{n-k+i}$ by $4i\epsilon$ and $\acute{e}_{n-k+i}$ differs from $\dot{\acute{e}}_{n-k+i}$ by $4(i+1)\epsilon$ for $i = 1, \dots, k$.*

Proof. By Lemma 4.3 we have

$$(1+\epsilon)^{-1}\breve{q}_i \le \dot{\breve{q}}_i \le (1+\epsilon)\breve{q}_i,$$
$$(1+\epsilon)^{-3}\breve{e}_i \le \dot{\breve{e}}_i \le (1+\epsilon)^3\breve{e}_i.$$

Therefore, similar to (4.21), we have

$$(1+\epsilon)^{-4i}\acute{d}_{n-k+i} \le \dot{\acute{d}}_{n-k+i} \le (1+\epsilon)^{4i}\acute{d}_{n-k+i}. \tag{4.36}$$

The same argument as in Lemma 4.4 completes the proof. ▯

From this lemma, we get

$$\exp((2k^2+4k)\epsilon)^{-1}\acute{\sigma}_i \le \dot{\acute{\sigma}}_i \le \exp((2k^2+4k)\epsilon)\acute{\sigma}_i \quad \text{for} \quad i = 1, \dots, n,$$

with the aid of Demmel–Kahan's result. Moreover, using Lemma 4.3 we get

$$\exp(2k\epsilon)^{-1}\dot{\sigma}_i \le \widetilde{\sigma}_i \le \exp(2k\epsilon)\dot{\sigma}_i \quad \text{for} \quad i = 1,\dots,n.$$

Therefore, we obtain

$$\exp((2k^2+6k)\epsilon)^{-1}\acute{\sigma}_i \le \widetilde{\sigma}_i \le \exp((2k^2+6k)\epsilon)\acute{\sigma}_i \tag{4.37}$$

for $i = 1,\dots,n$.

Now we present the main result of this subsection. Note that $\widetilde{B}$ is the output of Aggdef(2) (recall that $\widetilde{B}$ is obtained by replacing the lower right $k \times k$ part of B by $\widetilde{B}_2$).

THEOREM 4.8. *Aggdef(2)-*1 *preserves high relative accuracy. The singular values* $\sigma_1 > \cdots > \sigma_n$ *of* B *and* $\widetilde{\sigma}_1 > \cdots > \widetilde{\sigma}_n$ *of* $\widetilde{B}$ *and the sum of previous shifts* S *satisfy*

$$\exp((7k^2+18k+C)\epsilon)^{-1}\sqrt{\sigma_i^2+S} \le \sqrt{\widetilde{\sigma}_i^2+S} \le \exp((7k^2+18k+C)\epsilon)\sqrt{\sigma_i^2+S} \tag{4.38}$$

for $i = 1,\dots,n$, *where* $C = c' + c''$, *where* $c'(\approx c/2)$ *and* $c''(\approx c)$ *are constants as defined in* (4.26) *and* (4.34).

Proof. Combine (4.27), (4.35), and (4.37). ∎

We note that in practice we always let the window size k be $k \le \sqrt{n}$ (see section 6.2), so the bound (4.38) gives a relative error bound of order $O(n\epsilon)$, which has the same order as the bound for one dqds iteration derived in [11]. Also note that in our experiments we let $c = 1.0$ so $C \simeq 1.5$. We conclude that executing Aggdef(2)-1 does not affect the relative accuracy of dqds.

As discussed below, in Aggdef(2) we execute Aggdef(2)-1 repeatedly to deflate $\ell(>1)$ singular values. In this case we have

$$\exp((7k^2+18k+C)\ell\epsilon)^{-1}\sqrt{\sigma_i^2+S} \le \sqrt{\widetilde{\sigma}_i^2+S} \le \exp((7k^2+18k+C)\ell\epsilon)\sqrt{\sigma_i^2+S}$$

for $i = 1,\dots,n$, where ℓ is the number of deflated singular values by Aggdef(2).

4.5. Overall algorithm Aggdef(2).

As mentioned above, Aggdef(2)-1 deflates only one singular value. To deflate $\ell(>1)$ singular values we execute Aggdef(2), which is mathematically equivalent to ℓ runs of Aggdef(2)-1, but is cheaper saving ℓ calls of dstqds. Algorithm 4 is its pseudocode.

Algorithm 4. Aggressive early deflation—version 2: Aggdef(2).

Inputs: Bidiagonal B, window size k, sum of previous shifts S

```
1: $C = B_2$, $\ell = 0$.
2: Compute $s_{\ell+1} = (\sigma_{\min}(C))^2$.
3: Compute $\widehat{B}_2$ such that $\widehat{B}_2^T\widehat{B}_2 = C^TC - s_{\ell+1}I$ by dstqds. Set $\widehat{B}_2(\text{end},\text{end}) \leftarrow 0$ if
   (4.7) holds, otherwise go to line 6.
4: Compute $\breve{B}_2 = \widehat{B}_2\prod_{i=1}^{i_0} G_R(k-i,k)$ for $i_0 = 1,\dots,k-2$ until (4.9) holds. Go to
   line 6 if (4.9) never holds.
5: $C := \breve{B}_2(1:\text{end}-1, 1:\text{end}-1)$, $\ell \leftarrow \ell+1$, $k \leftarrow k-1$, go to line 2.
6: Compute $\widetilde{B}_2$ such that $\widetilde{B}_2^T\widetilde{B}_2 = C^TC + \sum_{i=1}^{\ell} s_iI$ by dstqds and update $B$ by
   replacing $B_2$ with $\mathrm{diag}(\tilde{B}_2, \mathrm{diag}(\sqrt{\sum_{j=1}^{\ell} s_j},\dots,\sqrt{\sum_{j=1}^{2} s_j},\sqrt{s_1}))$.
```

4.6. Relation between Aggdef(2) and other methods. In this subsection, we examine the relation between Aggdef(2) and previously proposed methods including Aggdef(1).

4.6.1. Comparison with Aggdef(1). First, it should be stressed that Aggdef(2) is computationally more efficient than Aggdef(1). Specifically, in contrast to Aggdef(1), which always needs $O(k^2)$ flops, Aggdef(2) requires $O(k\ell)$ flops when it deflates ℓ singular values. Hence, even when only a small number of singular values are deflated (when $\ell \ll k$), Aggdef(2) wastes very little work. In addition, as we saw above, unlike Aggdef(1), Aggdef(2) preserves high relative accuracy of the computed singular values, regardless of the window size k.

However, it is important to note that Aggdef(1) and Aggdef(2) are not mathematically equivalent, although closely related. To see the relation between the two, let us define the $k \times k$ unitary matrices $Q_i = \prod_{j=1}^{i} G_R(k-j,k)$ for $i = 1, \ldots, k-2$. After i Givens transformations are applied on line 4 of Aggdef(2), $Q_i(k-i:k,k)$ (the $i+1$ bottom part of the last column) is parallel to the corresponding part of $v = [v_1, \ldots, v_k]^T$, the null vector of $\widehat{B}_2$. This can be seen by recalling that $\widehat{B}_2$ is upper-bidiagonal and the bottom $i+1$ elements of $\widehat{B}_2 Q_i(:,k)$ are all zeros. Note that v is also the right-singular vector corresponding to $\sigma_{\min}(B_2)$.

Hence in particular, after $k-2$ (the largest possible number) Givens transformations have been applied we have $Q_{k-2}(2:k,k) = v(2:k)/\sqrt{1-v_1^2}$. It follows that w in (4.1) is $w = \sqrt{\widehat{e}_{n-k+1}}v_2/\sqrt{1-v_1^2} = -\sqrt{\widehat{q}_{n-k+1}}v_1/\sqrt{1-v_1^2}$, where we used the fact $\sqrt{\widehat{q}_{n-k+1}}v_1 + \sqrt{\widehat{e}_{n-k+1}}v_2 = 0$. Hence recalling (4.9) and $x = w^2$, we conclude that Aggdef(2) deflates $\sqrt{S+s}$ as a converged singular value if $\frac{|v_1|}{\sqrt{1-|v_1|^2}} < \min\{S\epsilon/\sqrt{\widehat{q}_{n-k+1}(\widehat{q}_{n-k+1}+\widehat{e}_{n-k+1})}, \sqrt{S\epsilon/\widehat{q}_{n-k+1}}\}$. On the other hand, as we discussed in section 3.1, Aggdef(1) deflates $\sqrt{S+s}$ if $|v_1| < \sqrt{S}\epsilon/\sqrt{e_{n-k+1}}$. We conclude that Aggdef(1) and Aggdef(2) are similar in the sense that both deflate the smallest singular value of B_2 when the first element of its right-singular vector v_1 is small enough, albeit with different tolerances.

The fundamental difference between Aggdef(1) and Aggdef(2) is that Aggdef(1) deflates all the deflatable singular values at once, while Aggdef(2) attempts to deflate singular values one by one from the smallest ones. As a result, Aggdef(2) deflates only the smallest singular values of B_2, while Aggdef(1) can detect the converged singular values that are not among the smallest. Consequently, sometimes fewer singular values are deflated by Aggdef(2). However, this is not a serious defect of Aggdef(2), since, as we show in section 5.1, smaller singular values are more likely to be deflated via aggressive early deflation. The numerical results in section 6 also show that the total numbers of singular values deflated by the two strategies are typically about the same.

4.6.2. Relation with Sorensen's deflation strategy. A deflation strategy closely related to Aggdef(2) is that proposed by Sorensen ([32], [4, sect. 4.5.7]) for restarting the Arnoldi or Lanczos algorithm. Just like Aggdef(2), the strategy attempts to deflate one converged eigenvalue at a time. Its idea can be readily applied for deflating a converged eigenvalue in a $k \times k$ bottom-right submatrix A_2 of an $n \times n$ symmetric tridiagonal matrix A as in (2.6). An outline of the process is as follows: Let (λ, v) be an eigenpair of A_2 with $v(k) \neq 0$. Sorensen defines the special $k \times k$ orthogonal matrix $Q_S = L + vu_k^T$, where $u_k = [0, \ldots, 1]$ and L is lower triangular with nonnegative diagonals except $L(k,k) = 0$ (see [32, 4] for a precise formulation of L). L also has the property that for $i = 1, \ldots, k-1$, the below-diagonal part of the ith column $L(i+1:k, i)$ is parallel to $v(i+1:k)$. In [32] it is shown that $\operatorname{diag}(I_{n-k}, Q_S^T) A \operatorname{diag}(I_{n-k}, Q_S) = \left[\begin{smallmatrix} A_1 & t^T \\ t & \widehat{A}_2 \end{smallmatrix}\right]$ for $\widehat{A}_2 = \operatorname{diag}(T, \lambda)$ and $t = [\widehat{b}_{n-k}, 0, \ldots, 0, b_{n-k}v(1)]^T$, where T is a $(k-1) \times (k-1)$ tridiagonal matrix. Therefore, λ can be deflated if $b_{n-k}v(1)$ is negligibly small.

Now we discuss the close connection between Sorensen's deflation strategy and Aggdef(2). Recall the definition $Q_i = \prod_{j=1}^{i} G_R(n-j, n)$. We shall see that Q_{k-2} and Q_S are closely related. To do so, we first claim that a unitary matrix with the properties of Q_S is uniquely determined by the vector v. To see this, note that [16] shows that for any vector v there exists a unique unitary upper Hessenberg matrix expressed as a product of $n-1$ Givens transformations, whose last column is v. We also note that such unitary Hessenberg matrices is discussed in [26]. Now, by permuting the columns of Q_S by right-multiplying the permutation matrix P such that $P(i, i+1) = 1$ for $i = 1, \ldots, k-1$ and $P(k, 1) = 1$, and taking the transpose we get a unitary upper Hessenberg matrix $P^T Q_S^T$ whose last column is v. Therefore, we conclude that such Q_S is unique. Recalling that for $i = 1, \ldots, k-2$ the last column of Q_i is parallel to $[0, \ldots, 0, v(k-i:k)^T]^T$, and noting that the irreducibility of B_2 ensures $v(k) \neq 0$ and that the diagonals of Q_i are positive (because the diagonals of (4.1) are positive), we conclude that $Q_S = Q_{k-2} \cdot G_R(n-k+1, n)$. Here, the Givens transformation $G_R(n-k+1, n)$ zeros the top-right w if applied to the last matrix in (4.1).[1] Conversely, Q_i can be obtained in the same way as Q_S, by replacing v with a unit vector parallel to $[0, 0, \ldots, 0, v_{k-i}, \ldots, v_k]^T$. Therefore, recalling (4.6), we see that performing Aggdef(2) can be regarded as successively and implicitly forming $\operatorname{diag}(I_{n-k}, Q_i^T) B^T B \operatorname{diag}(I_{n-k}, Q_i)$, where Q_i is a truncated version of Q_S.

Table 4.1 summarizes the relation between aggressive early deflation (AED) as in [5], Sorensen's strategy, Aggdef(1), and Aggdef(2).

TABLE 4.1
Summary of deflation strategies.

	Hessenberg	Bidiagonal
Deflate all at once	AED [5]	Aggdef(1)
Deflate one at a time	Sorensen [32]	[18], Aggdef(2)

While being mathematically nearly equivalent to Sorensen's strategy, Aggdef(2) achieves significant improvements in both efficiency and stability. To emphasize this point, we compare Aggdef(2) with another, more straightforward extension of Sorensen's deflation strategy for the bidiagonal case, described in [18]. The authors in [18] use both the left and right singular vectors of a target singular value to form two unitary matrices Q_S and P_S (determined by letting y be the singular vectors that determines the unitary matrix) such that $\operatorname{diag}(I_{n-k}, P_S^T) \cdot B \cdot \operatorname{diag}(I_{n-k}, Q_S)$ is bidiagonal except for the nonzero $(n-k, n)$th element, and its bottom diagonal is "isolated." Computing this orthogonal transformation requires at least $O(k^2)$ flops. It can also cause loss of relative accuracy of the computed singular values. In contrast, Aggdef(2) completely bypasses P_S (whose effect is implicitly "taken care of" by the two dstqds transformations) and applies the truncated version of Q_S without forming it. The resulting rewards are immense: the cost is reduced to $O(k)$ and high relative accuracy is guaranteed.

[1]We do not allow applying the transformation $G_R(n-k+1, n)$ in Aggdef(2) for the reason mentioned in section 4.2.

5. Convergence analysis. In this section we develop convergence analyses of dqds with aggressive early deflation. Specifically, we derive convergence factors of the x elements in (4.2), which explain why aggressive early deflation improves the performance of dqds. Our analyses also show that aggressive early deflation makes a sophisticated shift strategy less vital for convergence speed, making the zero-shift variant dqd a competitive alternative to dqds.

Our analysis focuses on Aggdef(2), because it is more efficient and always stable, and outperforms Aggdef(1) in all our experiments. In addition, as we discussed in section 4.6.1, the two strategies are mathematically closely related. We start by estimating the value of the x elements as in (4.2) in Aggdef(2). Then in section 5.2 we study the impact on x of one dqds iteration.

5.1. Bound for *x* elements. As reviewed in section 2.1, in the dqds algorithm the diagonals $\sqrt{q_i}$ of B converge to the singular values in descending order of magnitude, and the ith off-diagonal element $\sqrt{e_i}$ converge to zero with the convergence factor $\frac{{\sigma_{i+1}}^2-S}{{\sigma_i}^2-S}$ [1]. In view of this, here we assume that q_i are roughly ordered in descending order of magnitude, and that the off-diagonals e_i are small so that $e_i \ll q_i$.

Under these assumptions we claim that the dstqds step changes the matrix B_2 only slightly, except for the bottom element $\sqrt{q_n}$ which is mapped to 0. To see this, note that since $s < q_n$, we have $\widehat{q}_{n-k+1} = q_{n-k+1} - s \simeq q_{n-k+1}$, which also implies $\widehat{e}_{n-k+1} \simeq e_{n-k+1}$. Now since by assumption we have $e_i \ll q_i \simeq \widehat{q}_i$, so $d \simeq -s$ throughout the dstqds transformation. Therefore, the claim follows.

Now consider the size of the x element in Aggdef(2)-1 when it is chased up to the top, $(n-k+1, n)$ element. For definiteness here we denote by x_i the value of x after i transformations are applied. Initially we have $x_0 = \widehat{e}_{n-1} \simeq e_{n-1}$. Then the Givens transformations are applied, and as seen in (4.3), the ith transformation reduces x by a factor $\frac{\widehat{e}_{n-i-1}}{\widehat{q}_{n-i}+x_i}$. Therefore, after the application of $k-2$ transformations x becomes

$$x = \widehat{e}_{n-1} \prod_{i=1}^{k-2} \frac{\widehat{e}_{n-i-1}}{\widehat{q}_{n-i} + x_i} \le \widehat{e}_{n-1} \prod_{i=1}^{k-2} \frac{\widehat{e}_{n-i-1}}{\widehat{q}_{n-i}} \simeq e_{n-1} \prod_{i=1}^{k-2} \frac{e_{n-i-1}}{q_{n-i}}. \tag{5.1}$$

This is easily seen to be small when $q_{n-i} \gg e_{n-i-1}$ for $i = 1, \dots, k-2$, which necessarily holds in the asymptotic stage of dqds convergence. Note that asymptotically we have $x_i \ll \widehat{q}_{n-i}$, so that $\widehat{q}_{n-i} + x_i \simeq \widehat{q}_{n-i}$ for $i = 1, \dots, k-2$, and so all the inequalities and approximations in (5.1) become an equality.

Now we consider deflating the $\ell(\ge 2)$th smallest singular value of B_2 via the ℓth run of Aggdef(2)-1 in Aggdef(2), assuming that the smallest $\ell-1$ singular values have been deflated. Here we denote by x_ℓ the x element after the Givens transformations. Under the same asymptotic assumptions as above we see that after the maximum $(k-\ell-1)$ transformations the x_ℓ element is

$$x_\ell = \widehat{e}_{n-\ell} \prod_{i=1}^{k-\ell-1} \frac{\widehat{e}_{n-i-\ell}}{\widehat{q}_{n-i-\ell+1} + x_i} \le \widehat{e}_{n-\ell} \prod_{i=1}^{k-\ell-1} \frac{\widehat{e}_{n-i-\ell}}{\widehat{q}_{n-i-\ell+1}} \simeq e_{n-\ell} \prod_{i=1}^{k-\ell-1} \frac{e_{n-i-\ell}}{q_{n-i-\ell+1}}. \tag{5.2}$$

Several remarks regarding (5.2) are in order.

- While the analyses in [5, 19] are applicable to the more general Hessenberg matrix, our result exhibits several useful simplifications by specializing in the bidiagonal case. Our bound (5.2) on x_ℓ involves only the elements of B_2. On the other hand, the bound (2.5) in [5], which bound the spike vector elements in Aggdef(1) (after interpreting the problem in terms of computing the eigenvalues of B^TB), is difficult to use in practice because it requires information about the eigenvector.

- By (5.2) we see that x_ℓ is typically larger for large ℓ. This is because for large ℓ, fewer Givens transformations are applied, and $e_{n-\ell}$ tends to be smaller for small ℓ, because as can be seen by (2.2), e_{n-i} typically converges via the dqds iterations with a smaller convergence factor for small i. This suggests that Aggdef(2) detects most of the deflatable singular values of B_2, because it looks for deflatable singular values from the smallest ones. Together with the discussion in section 4.6.1, we argue that the numbers of singular values deflated by Aggdef(1) and Aggdef(2) are expected to be similar. Our numerical experiments confirm that this is true in most cases.
- Equation (5.2) indicates that x_ℓ can be regarded as converged when $e_{n-\ell}\prod_{i=1}^{k-\ell-1}\frac{e_{n-i-\ell}}{q_{n-i-\ell+1}}$ is small. This is essentially proportional to the product of the off-diagonal elements $e_{n-i-\ell}$ for $i=0,1,\dots,k-\ell-1$, because once convergence reaches the asymptotic stage the denominators $q_{n-i-\ell+1}$ converge to the constants $\sigma_{n-i-\ell+1}$. Hence, (5.2) shows that x_ℓ can be deflated when the product $\prod_{i=0}^{k-\ell-1}e_{n-i-\ell}$ is negligibly small, which can be true even if none of $e_{n-i-\ell}$ is.

5.2. Impact of one dqd iteration. Here we study the convergence factor of x_ℓ when one dqd (without shift; we discuss the effect of shifts shortly) iteration is run. As we reviewed in the introduction, in the asymptotic stage we have $q_i \simeq \sigma_i^2$ and $e_i \to 0$ with the convergence factor $\sigma_{i+1}^2/\sigma_i^2$ [1]. This is an appropriate measure for the convergence factor of the dqd(s) algorithm with a conventional deflation strategy. On the other hand, when Aggdef(2) is employed, the convergence factor of x_ℓ is a more natural way to measure convergence. In this section, we discuss the impact of running one dqd iteration on x_ℓ.

In this subsection we denote by $\widetilde{B}$ the bidiagonal matrix obtained by running one dqd step on B, and let $\widetilde{q}_i, \widetilde{e}_i$ be the bidiagonal elements of $\widetilde{B}$. Similarly denote by $\widetilde{x}$ the value of x when Aggdef(2) is applied to $\widetilde{B}$. Then, in the asymptotic stage we have $\widetilde{q}_i \simeq q_i \simeq \sigma_i^2$ and $\widetilde{e}_i \simeq \frac{\sigma_{i+1}^2}{\sigma_i^2}e_i$. Then by the estimate (5.2) we have

$$\widetilde{x}_\ell \simeq \widetilde{e}_{n-\ell}\prod_{i=1}^{k-\ell-1}\frac{\widetilde{e}_{n-i-\ell}}{\widetilde{q}_{n-i-\ell+1}}.$$

It follows that

$$\begin{aligned}\frac{\widetilde{x}_\ell}{x_\ell} &\simeq \frac{\prod_{i=0}^{k-\ell-1}\widetilde{e}_{n-i-\ell}}{\prod_{i=0}^{k-\ell-1}e_{n-i-\ell}}\cdot\frac{\prod_{i=1}^{k-\ell-1}\widetilde{q}_{n-i-\ell+1}}{\prod_{i=1}^{k-\ell-1}\widetilde{q}_{n-i-\ell+1}}\\ &\simeq \frac{\prod_{i=0}^{k-\ell-1}e_{n-i-\ell}(\sigma_{n-i-\ell+1}^2/\sigma_{n-i-\ell}^2)}{\prod_{i=0}^{k-\ell-1}e_{n-i-\ell}}\cdot\frac{\prod_{i=1}^{k-\ell-1}\widetilde{q}_{n-i-\ell+1}}{\prod_{i=1}^{k-\ell-1}\widetilde{q}_{n-i-\ell+1}}\\ &= \prod_{i=0}^{k-\ell-1}\frac{\sigma_{n-i-\ell+1}^2}{\sigma_{n-i-\ell}^2} = \frac{\sigma_{n-\ell+1}^2}{\sigma_{n-k+1}^2}.\end{aligned} \tag{5.3}$$

The estimate (5.3) suggests that asymptotically one dqd iteration reduces the magnitude of x_ℓ by an approximate factor $\frac{\sigma_{n-\ell+1}^2}{\sigma_{n-k+1}^2}$. This is generally much smaller than

$\frac{\sigma^2_{n-\ell+1}}{\sigma^2_{n-\ell}}$, the asymptotic convergence factor of $e_{n-\ell}$ (which needs to be small to deflate ℓ smallest singular values by a conventional deflation strategy). We note that the estimate (5.3) is still sharp in the asymptotic sense.

Simple example. The following example illustrates the sharpness of the convergence estimate (5.3). Consider the bidiagonal matrix B of size $n = 1000$, defined by $q_i = n + 1 - i, e_i = 0.1$ for $i = 1, \dots, n-1$, that is,

$$B = \text{bidiag}\left(\begin{array}{cccccccc} & \sqrt{0.1} & & . & & \sqrt{0.1} & & \sqrt{0.1} & \\ \sqrt{1000} & & . & & . & & \sqrt{2} & & \sqrt{1} \end{array} \right). \tag{5.4}$$

To see the sharpness of the estimate (5.3), we set the window size to $k = 30$, and ran a dqd iteration (without shifts; an experiment with shifts will appear shortly) on B, then computed $\frac{\widehat{x}_\ell}{x_\ell} \Big/ \frac{\sigma^2_{n-\ell+1}}{\sigma^2_{n-k+1}}$. We observed that[2]

$$1 + 6.79 \times 10^{-3} \leq \frac{\widehat{x}_\ell}{x_\ell} \Bigg/ \frac{\sigma^2_{n-\ell+1}}{\sigma^2_{n-k+1}} \leq 1 + 5.996 \times 10^{-2} \tag{5.5}$$

for $\ell = 1, \dots, k-2$. This suggests that (5.3) is a sharp estimate of the convergence factor $\widehat{x}_\ell / x_\ell$. This behavior is not specific to the particular choice of B, and similar results were obtained generally with any graded diagonally dominant bidiagonal matrix.

5.3. Effect of shifts and motivations for dqd. The estimate of $\widetilde{x}_\ell / x_\ell$ in (5.3) suggests that the introduction of shifts may have little effect on the number of deflatable eigenvalues in dqds with aggressive early deflation. Specifically, when a shift s $(< \sigma_n^2)$ is applied to dqds, the convergence factor estimate (5.3) becomes $\frac{\sigma^2_{n-\ell+1} - s}{\sigma^2_{n-k+1} - s}$. When a conventional deflation strategy is used, we can regard $k = 2$ and $\ell = 1$, in which case this factor is greatly reduced with an appropriate shift $s \simeq \sigma_n^2$. In fact, the zero-shift strategy dqd results in prohibitively slow convergence of the bottom off-diagonal element, so a sophisticated shift strategy is imperative. However, when aggressive early deflation is adopted so that $k > 2$, we see that the estimate (5.3) is close to that of dqd, that is,

$$\frac{\sigma^2_{n-\ell+1} - s}{\sigma^2_{n-k+1} - s} \simeq \frac{\sigma^2_{n-\ell+1}}{\sigma^2_{n-k+1}} \tag{5.6}$$

for $\ell = 2, 3, \dots, k-2$, because during a typical run of dqds we have $\sigma_{n-\ell+1} \gg \sigma_n > s$ for such ℓ. Hence, when dqds is equipped with aggressive early deflation, shifts may not be essential for the performance. This observation motivates the usage of dqd instead of dqds.

Using dqd instead of dqds has many advantages: dqd has a smaller chance of an overflow or underflow [30] and smaller computational cost per iteration, not to mention the obvious fact that computing the shifts is unnecessary.[3] Furthermore, because the shifts are prescribed to be zero, dqd can be parallelized by running multiple dqd in

[2]Assuming $S = 1$, about ten singular values were deflated by Aggdef(2), so only $\ell \leq 10$ will appear in actual computation. To fully investigate the convergence behavior of x here, we kept running Aggdef(2) regardless of whether (4.9) was satisfied.

[3]Choosing the shift is a delicate task because a shift larger than the smallest singular value results in breakdown of the Cholesky factorization that is implicitly computed by dqds. In DLASQ, whenever a shift violates the positivity, the dqds iteration is rerun with a smaller shift.

a pipelined manner, just as multiple steps of the QR algorithm can be executed in parallel when multiple shifts are chosen in advance [3, 34, 24]. We note that it has been difficult to parallelize dqds in such a way, since an effective shift usually cannot be determined until the previous dqds iteration is completed.

Simple example. To justify the above observation, we again use our example matrix (5.4), and run five dqds iterations with the Johnson shift [17] to obtain $\widetilde{B}$, and compute the values $\widetilde{x}_\ell$ by running Aggdef(2). Similarly we obtain $\widehat{B}$ by running five dqd iterations and compute $\widehat{x}_\ell$. Figure 5.1 shows plots of $x_\ell, \widehat{x}_\ell$ and $\widetilde{x}_\ell$ for $\ell = 1, \ldots, 15$.

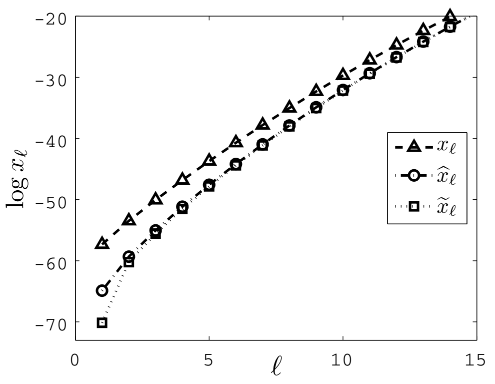

FIG. 5.1. *ℓ-$\log x_\ell$ plots for matrix B in* (5.4). *$\widehat{x}_\ell$ and $\widetilde{x}_\ell$ are obtained from matrices after running five dqd and dqds iterations, respectively.*

We make two remarks on Figure 5.1. First, running Aggdef(2) on the original B already lets us deflate about nine singular values (since we need $x_\ell \simeq 10^{-30}$ to satisfy (4.9)). This is because B has a graded and diagonally dominant structure that typically arises in the asymptotic stage of dqds convergence, which is favorable for Aggdef(2). Running dqd (or dqds) iterations generally reduces the magnitude of x_ℓ, and for $\widehat{B}$ and $\widetilde{B}$ we can deflate one more singular value by Aggdef(2).

Second, the values of $\widehat{x}_\ell$ and $\widetilde{x}_\ell$ are remarkably similar for all ℓ but $\ell = 1$. This reflects our above estimates (5.3) and (5.6), which suggest shifts can help the convergence only of the smallest singular value. Furthermore, as can be seen in Figure 5.1, the smallest singular value tends to be already converged in the context of Aggdef(2), so enhancing its convergence is not necessary. Therefore, we conclude that shifts may have little or no effect on the overall deflation efficiency of Aggdef(2), suggesting that a zero shift is sufficient and preferred for parallelizability.

6. Numerical experiments.

This section shows results of numerical experiments to compare the performance of different versions of dqds.

6.1. Pseudocodes.

Algorithm 5 shows the pseudocode of dqds with aggressive early deflation.

Algorithm 5. dqds with aggressive early deflation.

```
Inputs: Bidiagonal matrix $B \in \mathbb{R}^{n \times n}$, deflation frequency $p$
1: while size of $B$ is larger than $\sqrt{n}$ do
2:    run $p$ iterations of dqds
3:    perform aggressive early deflation
4: end while
5: run dqds until all singular values are computed
```

On the third line, either Aggdef(1) or Aggdef(2) may be invoked. After the matrix size is reduced to smaller than $\sqrt{n}$ we simply use the standard dqds algorithm because the remaining part needs only $O(n)$ flops, the same as one dqds iteration for the original matrix. Along with aggressive early deflation we invoke the conventional deflation strategy after each dqds iteration, just like in the Hessenberg QR case [5].

As motivated in the previous section, we shall also examine the performance of the zero-shift version, dqd with aggressive early deflation. Algorithm 6 is its pseudocode.

Algorithm 6. dqd with aggressive early deflation.

```
Inputs: Bidiagonal matrix $B \in \mathbb{R}^{n \times n}$, deflation frequency $p$
1: while size of $B$ is larger than $\sqrt{n}$ do
2:    run one iteration of dqds, followed by $p-1$ iterations of dqd
3:    perform aggressive early deflation
4: end while
5: run dqds until all singular values are computed
```

Note that on line 2 of Algorithm 6, one dqds iteration is run prior to the dqd iterations. This can be done even if we run the p iterations (1 dqds and $p-1$ dqd) in parallel, because this requires only the first shift. The single dqds iteration is important for efficiency because typically a large shift s can be taken after a significant number of singular values are deflated by aggressive early deflation.

6.2. Choice of parameters. Two parameters need to be specified when executing Aggdef in Algorithm 5 or 6: the frequency p with which we invoke Aggdef, and the window size k.

We first discuss our choice of the frequency p. It is preferable to set p small enough to take full advantage of aggressive early deflation. This is the case especially for Aggdef(2), because each execution requires only $O(n\ell)$ flops, and experiments indicate that the execution of Aggdef(2) typically takes less than 4% of the overall runtime. In our experiments we let $p = 16$. This choice is based on the fact that when we run dqd iterations in parallel, we need $p \geq n_p$, where n_p is the number of processors run in a pipelined fashion. Experiments suggest that setting p too large (say $p > 300$) can noticeably deteriorate the performance on a sequential implementation, especially for Algorithm 6 (dqd with Aggdef). For example, for the twelve test matrices described in the next subsection, the runtime of dqd with Aggdef(2) with $p = 300$ was on average almost twice that of $p = 16$. When shifts are used (dqds) the performance depends less on p; dqds with Aggdef(2) with $p = 300$ was about just 10% slower than choosing $p = 16$. More study is needed for a good choice of p on a parallel implementation of dqd with Aggdef(2).

In practice, when a significant number of singular values are deflated by Aggdef, the performance can often be further improved by performing another aggressive early deflation before starting the next dqds iteration. In our experiments we performed another aggressive early deflation when three or more singular values were deflated by Aggdef. A similar strategy is suggested in [5] for the Hessenberg QR algorithm.

We now discuss our choice of the window size k. The idea is to choose k flexibly, using the information of the bottom-right part of B. From the estimate of x_ℓ in (5.1) we see that increasing k reduces the size of x as long as $e_{n-k+1} < q_{n-k+2}$ holds. Hence we compare e_{n-i+1} and q_{n-i+2} for $i = 1, 2, \ldots$, and set k to be the largest i such that $e_{n-i+1} < q_{n-i+2}$. When this results in $k \le 10$, we skip Aggdef and go on to the next dqds iteration to avoid wasting effort. The choice sometimes makes k too large (e.g., $k = n$ when B is diagonally dominant), so we set a safeguard upper bound $k \le \sqrt{n}$. In addition, from (4.9) and (5.1) we see that a singular value can be deflated once $\prod_{i=1}^{k-2} \frac{e_{n-i-1}}{q_{n-i}}$ is negligible, so we compute the products $\prod_{i=11}^{k-2} \frac{e_{n-i-1}}{q_{n-i}}$ (we start taking the product from $i = 11$ because we want to deflate more than one singular value; in view of (5.2), with $i = 11$ we can expect $\simeq 10$ deflations to occur) and decide to stop increasing k once the product becomes smaller than ϵ^2.

Through experiments we observed that the above choice of p and k is effective, achieving speedups of on average about 25% for matrices $n \ge 10000$, compared with any static choice such as $p = k = \sqrt{n}$.

6.3. Experiment details. We compare the performance of the following four algorithms.[4]

1. DLASQ: dqds subroutine of LAPACK version 3.2.2.
2. dqds+agg1: Algorithm 5, call Aggdef(1) on line 3.
3. dqds+agg2: Algorithm 5, call Aggdef(2) on line 3.
4. dqd+agg2: Algorithm 6, call Aggdef(2) on line 3.

We implemented our algorithms in FORTRAN by incorporating our deflation strategies into the LAPACK routines dlasqx.f (x ranges from 1 to 6). Hence, our codes perform the same shift and splitting strategies implemented in DLASQ.[5] When running dqds+agg1, we used the LAPACK subroutine DBDSQR to compute the singular values of B_2 and the spike vector t in (3.1). All experiments were conducted on a single core of a desktop machine with a quad core, Intel Core i7 2.67GHz Processor and 12GB of main memory. For compilation we used the f95 compiler and the optimization flag $-$O3, and linked the codes to BLAS and LAPACK.

Table 6.1 gives a brief description of our test matrices. Matrix 1 is "nearly diagonal," for which aggressive early deflation is particularly effective. Matrix 2 is a "nicely graded" matrix [11], for which DLASQ needs relatively few ($\ll 4n$) iterations. Matrix 3 is a Toeplitz matrix [11], which has uniform diagonals and off-diagonals. Matrix 4 has highly oscillatory diagonal entires. Matrix 5 is a perversely graded matrix, designed to be difficult for DLASQ. Matrices 6–10 are the Cholesky factors of test tridiagonal matrices taken from [22]. For matrices 8–10, we applied an appropriate shift to make the tridiagonal matrix positive definite before computing the Cholesky factor. Matrices 11 and 12 have the property that some of the singular values are tightly clustered.

6.4. Results. Results are shown in Figures 6.1–6.4, which compare the total runtime, number of dqds iterations, percentage of singular values deflated via aggres-

[4]dqd with Aggdef(1) can be implemented, but we do not present its results because dqd+agg2 was faster in all our experiments.

[5]It is possible that when equipped with aggressive early deflation, a different shift strategy for dqds is more efficient than that used in DLASQ. This is a possible topic of future study.

Table 6.1
Test bidiagonal matrices.

	n	Description of the bidiagonal matrix B	Source
1	30000	$\sqrt{q_i} = n+1-i,\ \sqrt{e_i} = 1$	
2	30000	$\sqrt{q_{i-1}} = \beta\sqrt{q_i},\ \sqrt{e_i} = \sqrt{q_i},\ \beta = 1.01$	[11]
3	30000	Toeplitz: $\sqrt{q_i} = 1,\ \sqrt{e_i} = 2$	[11]
4	30000	$\sqrt{q_{2i-1}} = n+1-i,\ \sqrt{q_{2i}} = i,\ \sqrt{e_i} = (n-i)/5$	[28]
5	30000	$\sqrt{q_{i+1}} = \beta\sqrt{q_i}\ (i \ge n/2),\ \sqrt{q_{n/2}} = 1,$ $\sqrt{q_{i-1}} = \beta\sqrt{q_i}\ (i \le n/2),\ \sqrt{e_i} = 1,\ \beta = 1.01$	
6	30000	Cholesky factor of tridiagonal $(1,2,1)$ matrix	[22, 30]
7	30000	Cholesky factor of Laguerre matrix	[22]
8	30000	Cholesky factor of Hermite recurrence matrix	[22]
9	30000	Cholesky factor of Wilkinson matrix	[22]
10	30000	Cholesky factor of Clement matrix	[22]
11	13786	matrix from electronic structure calculations	[31]
12	16023	matrix from electronic structure calculations	[31]

sive early deflation, and the percentage of the time spent performing aggressive early deflation relative to the overall runtime. We executed ten runs and took the average. The numbers in parentheses show the performance of DLASQ for each matrix: the runtime in seconds in Figure 6.1 and the iteration counts divided by the matrix size n in Figure 6.2. Although not shown in the figures, in all our experiments we confirmed the singular values are computed to high relative accuracy. Specifically, the maximum elementwise relative difference of the singular values computed by our algorithms from those computed by DLASQ was smaller than both 1.5×10^{-13} and $n\epsilon$ for each problem.

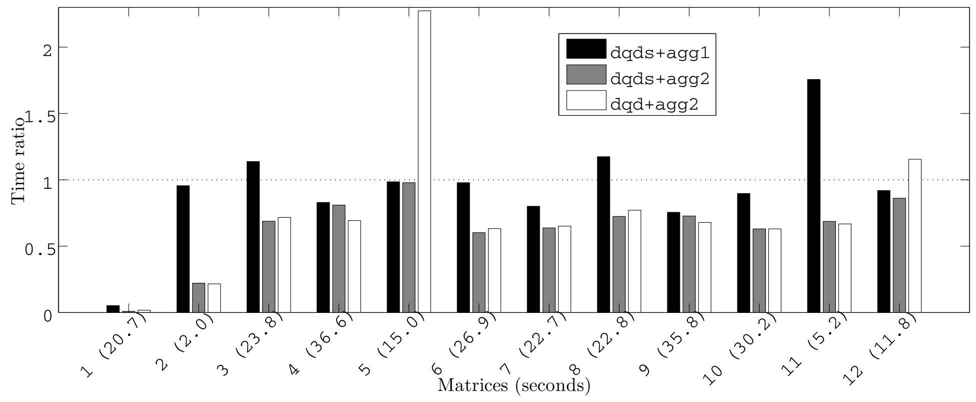

Fig. 6.1. *Ratio of time/DLASQ time.*

The results show that aggressive early deflation, particularly Aggdef(2), can significantly reduce both the runtime and iteration count of DLASQ. We obtained speedups of up to a factor 50 with dqds+agg2 and dqd+agg2.

dqds+agg2 was notably faster than DLASQ in most cases, and never slower. There was no performance gain for the "difficult" matrix 5, for which many dqds iterations are needed before the iteration reaches the asymptotic stage where the matrix is graded and diagonally dominant, after which Aggdef(2) becomes effective. dqds+agg2 was also at least as fast as dqds+agg1 in all our experiments. This is because as discussed in section 4.1, Aggdef(1) requires at least $O(k^2)$ flops, while Aggdef(2) needs only $O(k\ell)$ flops when it deflates $\ell \le k$ singular values.

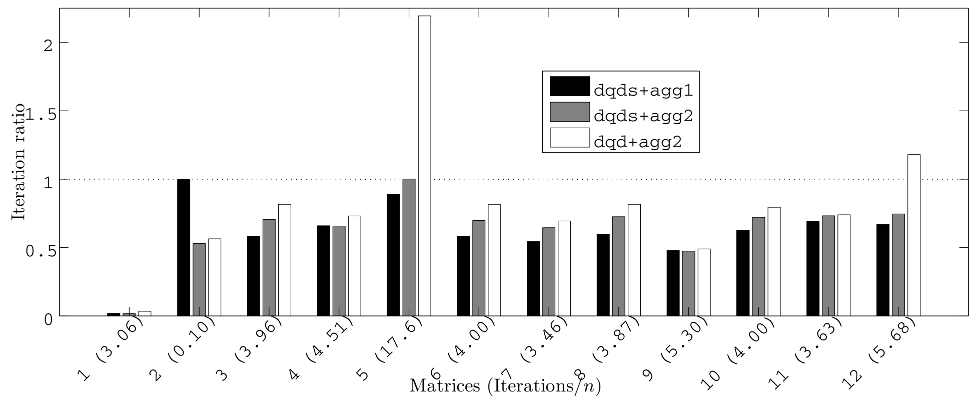

FIG. 6.2. *Ratio of iteration/DLASQ iteration.*

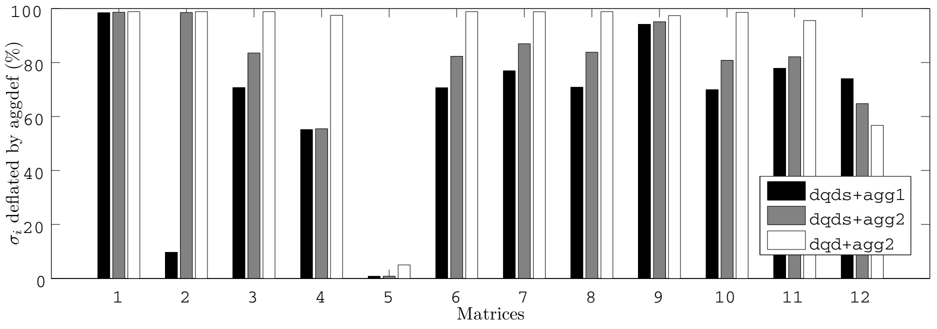

FIG. 6.3. *Percentage of singular values deflated by aggressive early deflation.*

We see from Figures 6.2 and 6.3 that dqds+agg1 and dqds+agg2 usually require about the same number of iterations and deflate similar numbers of singular values by Aggdef. The exception in matrix 2 is due to the fact that the safe window size enforced in Aggdef(1) (described in section 3.1) is often much smaller than k (determined as in section 6.2), making Aggdef(1) less efficient. For dqd+agg2, usually most of the singular values are deflated by Aggdef(2). This is because with zero shifts the bottom off-diagonal converges much slower, making the conventional deflation strategy ineffective.

Finally, in many cases dqd+agg2 was the fastest algorithm requiring comparable numbers of iterations to dqds+agg2, except for problems that are difficult (iteration $\geq 5n$) for DLASQ. As we mentioned earlier, this is in major contrast to dqd with a conventional deflation strategy, which is impractical due to the slow convergence of each off-diagonal element. Furthermore, as can be seen in Figure 6.4, with Aggdef(2) the time spent executing aggressive early deflation is typically less than 4%.[6] This observation makes the parallel implementation of dqd+agg2 particularly promising since it is already the fastest of the tested algorithms in many cases, and its parallel implementation is expected to speed up the dqd runs, which are essentially taking up

[6]Exceptions are in "easy" cases, such as matrices 1 and 2, where dqd+agg2 requires many fewer iterations than $4n$. In such cases dqd+agg2 spends relatively more time executing Aggdef(2) recursively. This is by no means a pathological case, because dqd+agg2 is already very fast with a sequential implementation.

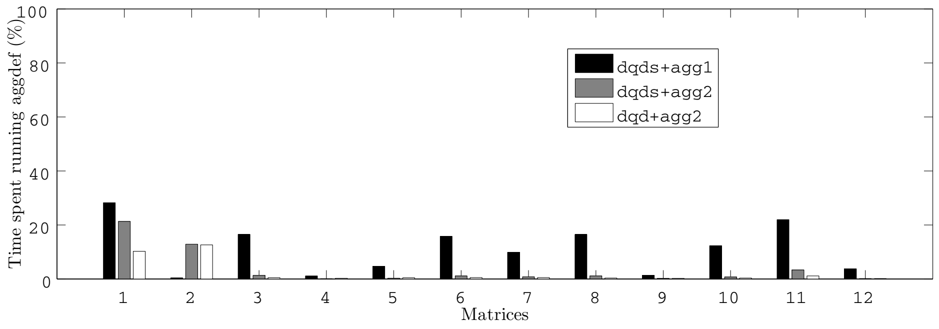

FIG. 6.4. *Percentage of time spent executing aggressive early deflation.*

more than 95% of the time.

We also tested with more than 500 other bidiagonal matrices, including the 405 bidiagonal matrices from the tester in the development of DLASQ [21], nine test matrices from [30], and six matrices that arise in electronic structure calculations [31]. We show in Figure 6.5 a scatter plot of the runtime ratio over DLASQ against the matrix size for dqds+agg2 and dqd+agg2. To keep the plot simple we do not show dqds+agg1, which was never faster than dqds+agg2. Also, to ensure that the computed time is measured reliably, we show the results only for 285 matrices for which DLASQ needed more than 0.01 second.

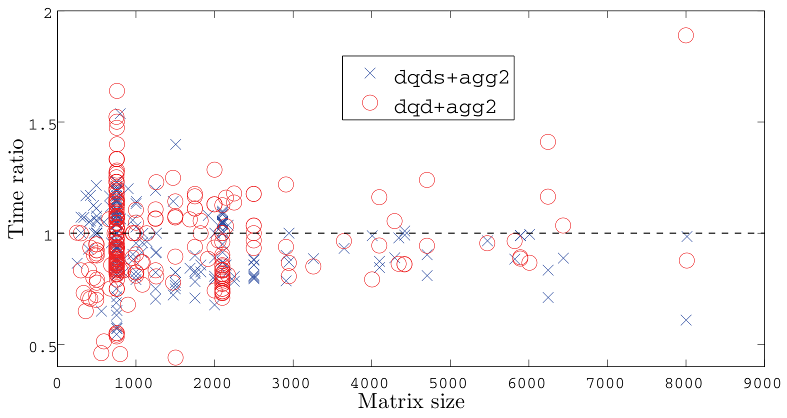

FIG. 6.5. *Ratio of time/DLASQ time for test matrices.*

We note that many of these matrices represent "difficult" cases (DLASQ needs more than $5n$ iterations), as they were generated for checking the algorithm robustness. In such cases, many dqd(s) iterations are needed for the matrix to reach the asymptotic graded structure, during which using Aggdef(2) may not be of much help. Nonetheless, dqds+agg2 was always at least as fast as DLASQ for all matrices larger than 3000. Moreover, dqds+agg2 was never slower than DLASQ by more than 0.016 second, so we argue that in practice it is never slower. The speed of dqd+agg2 varied more depending on the matrices, taking up to 0.15 second more than or 1.9 times as much time as DLASQ.

7. Conclusion. We proposed two algorithms dqds+agg1 and dqds+agg2 to incorporate aggressive early deflation into dqds for computing the singular values of bidiagonal matrices to high relative accuracy. We presented numerical results to demonstrate that aggressive early deflation can significantly speed up dqds. In particular, dqds+agg2 is at least as fast as the LAPACK implementation of dqds, and is often much faster. The zero-shifting strategy exhibits even more promising results with the potential to be parallelized. We plan to report the implementation and performance of a parallel version of dqd+agg2 in a future work.

Acknowledgments. We are extremely grateful to Professor Beresford Parlett for his numerous and insightful comments. His suggestions inspired us to look into a root-free deflation strategy (which led to Aggdef(2)), and let us realize the connection between Aggdef(2) and Sorensen's deflation strategy. We thank Dr. Osni Marques and Matthias Petschow for providing the test matrices. We thank the referees for their helpful suggestions, which helped us improve this paper considerably.

REFERENCES

[1] K. Aishima, T. Matsuo, K. Murota, and M. Sugihara, *On convergence of the DQDS algorithm for singular value computation*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 522–537.
[2] K. Aishima, T. Matsuo, K. Murota, and M. Sugihara, *Superquadratic convergence of DLASQ for computing matrix singular values*, J. Comput. Appl. Math., 234 (2010), pp. 1179–1187.
[3] Z. Bai and J. Demmel, *On a block implementation of Hessenberg multishift QR iteration*, Int. J. High Speed Comput., 1 (1989), pp. 97–112.
[4] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, SIAM, Philadelphia, 2000.
[5] K. Braman, R. Byers, and R. Mathias, *The multishift QR algorithm.* II. *Aggressive early deflation*, SIAM J. Matrix Anal. Appl., 23 (2002), pp. 948–973.
[6] J. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, 1997.
[7] J. Demmel and W. Kahan, *Accurate singular values of bidiagonal matrices*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 873–912.
[8] I. S. Dhillon, *A New $O(n^2)$ Algorithm for the Symmetric Tridiagonal Eigenvalue/Eigenvector Problem*, Ph.D. thesis, University of California, Berkeley, CA, 1997.
[9] I. S. Dhillon and B. N. Parlett, *Multiple representations to compute orthogonal eigenvectors of symmetric tridiagonal matrices*, Linear Algebra Appl., 387 (2004), pp. 1–28.
[10] I. S. Dhillon and B. N. Parlett, *Orthogonal eigenvectors and relative gaps*, SIAM J. Matrix Anal. Appl., 25 (2004), pp. 858–899.
[11] K. V. Fernando and B. N. Parlett, *Accurate singular values and differential qd algorithms*, Numer. Math., 67 (1994), pp. 191–229.
[12] J. G. F. Francis, *QR transformation: A unitary analogue to the LR transformation.* I, Comput. J., 4 (1961), pp. 265–271.
[13] J. G. F. Francis, *The QR transformation.* II, Comput. J., 4 (1962), pp. 332–345.
[14] G. H. Golub and G. Meurant, *Matrices, Moments and Quadrature with Applications*, Princeton Ser. Appl. Math., Princeton University Press, Princeton, NJ, 2010.
[15] G. H. Golub and J. H. Welsch, *Calculation of Gauss quadrature rules*, Math. Comp., 23 (1969), pp. 221–230.
[16] W. B. Gragg, *The QR algorithm for unitary Hessenberg matrices*, J. Comput. Appl. Math., 16 (1986), pp. 1–8.
[17] C. R. Johnson, *A Gersgorin-type lower bound for the smallest singular value*, Linear Algebra Appl., 112 (1989), pp. 1–7.
[18] E. Kokiopoulou, C. Bekas, and E. Gallopoulos, *Computing smallest singular triplets with implicitly restarted Lanczos bidiagonalization*, Appl. Numer. Math., 49 (2004), pp. 39–61.
[19] D. Kressner, *The effect of aggressive early deflation on the convergence of the QR algorithm*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 805–821.
[20] C.-K. Li and R.-C. Li, *A note on eigenvalues of perturbed Hermitian matrices*, Linear Algebra Appl., 395 (2005), pp. 183–190.
[21] O. A. Marques, *private communication*, 2010.

[22] O. A. Marques, C. Voemel, J. W. Demmel, and B. N. Parlett, *Algorithm* 880*: A testing infrastructure for symmetric tridiagonal eigensolvers*, ACM Trans. Math. Softw., 35 (2008), 8.

[23] R. Mathias, *Quadratic residual bounds for the Hermitian eigenvalue problem*, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 541–550.

[24] T. Miyata, Y. Yamamoto, and S.-L. Zhang, *A fully pipelined multishift QR algorithm for parallel solution of symmetric tridiagonal eigenproblems*, IPSJ Trans. Advanced Computing Systems, 1 (2008), pp. 14–27.

[25] S. Oliveira, *A new parallel chasing algorithm for transforming arrowhead matrices to tridiagonal form*, Math. Comp., 67 (1998), pp. 221–235.

[26] B. Parlett and E. Barszcz, *Another orthogonal matrix*, Linear Algebra Appl., 417 (2006), pp. 342–346.

[27] B. N. Parlett, *The Symmetric Eigenvalue Problem*, SIAM, Philadelphia, 1998.

[28] B. N. Parlett, *private communication*, 2010.

[29] B. N. Parlett and J. Le, *Forward instability of tridiagonal QR*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 279–316.

[30] B. N. Parlett and O. A. Marques, *An implementation of the dqds algorithm (positive case)*, Linear Algebra Appl., 309 (2000), pp. 217–259.

[31] M. Petschow, *private communication*, 2010.

[32] D. Sorensen, *Deflation for Implicitly Restarted Arnoldi Methods*, Technical Report 98-12, CAAM, Rice University, Houston, TX, 1998.

[33] G. W. Stewart, *Matrix Algorithms. Vol.* I: *Basic Decompositions*, SIAM, Philadelphia, 1998.

[34] R. A. van de Geijn, *Deferred shifting schemes for parallel QR methods*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 180–194.

[35] P. Willems, *On MR^3-type Algorithms for the Tridiagonal Symmetric Eigenproblem and the Bidiagonal SVD*, Ph.D. thesis, University of Wuppertal, Wuppertal, Germany, 2010.

[36] H. Zha, *A two-way chasing scheme for reducing a symmetric arrowhead matrix to tridiagonal form*, J. Numer. Linear Algebra Appl., 1 (1992), pp. 49–57.