Bidiagonal SVD Computation via an Associated Tridiagonal Eigenproblem

Osni Marques*, James Demmel† and Paulo B. Vasconcelos‡
*Lawrence Berkeley National Laboratory, oamarques@lbl.gov
†University of California at Berkeley, demmel@cs.berkeley.edu
‡Faculdade Economia and CMUP, University of Porto, pjv@fep.up.pt

Abstract

In this paper, we present an algorithm for the singular value decomposition (SVD) of a bidiagonal matrix by means of the eigenpairs of an associated symmetric tridiagonal matrix. The algorithm is particularly suited for the computation of a subset of singular values and corresponding vectors. We focus on a sequential version of the algorithm, and discuss special cases and implementation details. We use a large set of bidiagonal matrices to assess the accuracy of the implementation in single and double precision, as well as to identify potential shortcomings. We show that the algorithm can be up to three orders of magnitude faster than existing algorithms, which are limited to the computation of a full SVD. We also show time comparisons of an implementation that uses the strategy discussed in the paper as a building block for the computation of the SVD of general matrices.

I. Introduction

It is well known that the singular value decomposition (SVD) of a matrix $A \in \mathbb{R}^{m\times n}$, namely $A = USV^T$, with left singular vectors $U = [u_1, u_2, \ldots u_m]$, $U \in \mathbb{R}^{m\times m}$, right singular vectors $V = [v_1, v_2, \ldots v_n]$, $V \in \mathbb{R}^{n\times n}$, U and V orthogonal matrices, and singular values $s_1 \geq s_2 \geq \ldots s_n \geq 0$ on the main diagonal of $S \in \mathbb{R}^{m\times n}$, can be obtained through the eigenpairs (λ, x) of the matrices $C_{n\times n} = A^TA$ and $C_{m\times m} = AA^T$, as long as the singular values are distinct. However, if A is square and orthogonal then $C_{n\times n}$ and $C_{m\times m}$ are both the identity and provide little information about the singular vectors, which are not unique: $A = (AQ)I(Q^T)$ is the SVD of A for any orthogonal matrix Q. A potential difficulty for some algorithms (e.g. the one presented in this paper) is large clusters of singular values, as this may have an impact on the orthogonality of the computed singular vectors.

Alternatively, the SVD can be obtained through the eigenpairs of the augmented matrix

$$C = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix}. \tag{1}$$

In this paper we focus on the case $m = n$. Then, one can write [10]

$$C = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} = J \begin{bmatrix} -S & 0 \\ 0 & S \end{bmatrix} J^T, \tag{2}$$

where $J \in \mathbb{R}^{n\times n}$ is defined as

$$J = \begin{bmatrix} U & U \\ -V & V \end{bmatrix} / \sqrt{2}, \tag{3}$$

such that the eigenvalues of C are $\pm s$ and its eigenvectors are mapped into the singular vectors of A (scaled by $\sqrt{2}$) in a very structured manner[1].

In practical calculations, the SVD of a general matrix A involves the reduction of A to bidiagonal form B through orthogonal transformations, i.e. $A = \hat{U}B\hat{V}^T$. The singular values are thus preserved; the singular vectors of B need to be back transformed into those of A.

[1]Although this paper focuses on the real case, the discussion can be easily extended to the complex case.

TABLE I: LAPACK's (bidiagonal, BD) SVD and (tridiagonal, ST) eigensolvers.

routine	usage	algorithm
BDSQR	all s and (opt.) u and/or v	implicit QL or QR
BDSDC	all s and (opt.) u and v	divide-and-conquer
STEQR	all λ's and (opt.) x	implicit QL or QR
STEVX	selected λ's and (opt.) x	bisection & inverse iteration
STEDC	all λ's and (opt.) x	divide-and-conquer
STEMR	selected λ's and (opt.) x	MRRR

If B is an upper bidiagonal matrix with $(a_1, a_2, \ldots a_n)$ on the main diagonal and $(b_1, b_2, \ldots b_{n-1})$ on the superdiagonal, we can replace A with B in (1) to obtain $C = P\,T_{GK}\,P^T$, where T_{GK} is the Golub-Kahan symmetric tridiagonal matrix,

$$T_{GK} = \mathsf{tridiag}\left(\begin{array}{ccccccccccc} & a_1 & & b_1 & & a_2 & & b_2 & \ldots & b_{n-1} & & a_n & \\ 0 & & 0 & & 0 & & 0 & & \ldots & & 0 & & 0 \\ & a_1 & & b_1 & & a_2 & & b_2 & \ldots & b_{n-1} & & a_n & \end{array} \right), \tag{4}$$

with the perfect shuffle $P = [e_{n+1}, e_1, e_{n+2}, e_2, e_{n+3}, \ldots e_{2n}]$, $e_i, i = 1, 2, \ldots 2n$, corresponding to the columns of the identity matrix of dimension $2n$. Then, if the eigenpairs of T_{GK} are $(\pm s, z)$, with $\|z\| = 1$, from (2)-(3) we obtain

$$z = P \left[\begin{array}{c} u \\ \pm v \end{array} \right] \frac{1}{\sqrt{2}},$$

where u and v are the singular vectors associated with s. Thus, we can extract the SVD of B from the eigendecomposition of T_{GK}.

Table I lists the LAPACK [1] subroutines intended for the computation of the SVD of bidiagonal matrices, and eigenvalues and eigenvectors of tridiagonal matrices. The trade-offs (performance, accuracy) of the symmetric tridiagonal (ST) subroutines have been examined in [7]. We are interested in how these subroutines could be applied to T_{GK} in (4), specially for the computation of subsets of eigenpairs, which in turn could reduce the computational costs when a full SVD is not needed (or for the eventual computation of subsets of singular values and vectors in parallel). While STEDC could be potentially redesigned to compute a subset of eigenvectors, saving some work but only at the top level of the recursion of the divide-and-conquer algorithm (see [2]), STEVX and STEMR offer more straightforward alternatives. STEVX performs bisection (subroutine STEBZ in LAPACK) to find selected eigenvalues followed by inverse iteration (subroutine STEIN in LAPACK) to find the corresponding eigenvectors, for an $O(n)$ cost per eigenpair. STEVX may occasionally fail to provide orthogonal eigenvectors when the eigenvalues are too closely clustered. In contrast, STEMR uses a much more sophisticated algorithm called MRRR [8], [9] to guarantee orthogonality. In fact, the computation of the bidiagonal SVD using MRRR, by applying MRRR to the coupled eigenvalue decompositions of B^TB and BB^T, has been discussed in [11], [12], [26]. However, an implementation discussed in [26] and named BDSCR was abandoned: unsatisfactory results (large residuals for the SVD) were obtained in a number of cases, revealing gaps in the theory for the coupled approach. The problem was found to be related to a potential mismatch in twisted factorizations used in MRRR in that context [24]. Subsequently, an improved version of the MRRR algorithm targeting T_{GK} for SVD computations was proposed in [24], [25]; however, our experiments with an implementation given in [25] and named STEXR exposed deficiencies (inaccuracy) for relatively simple matrices. We show one such a case in Appendix A. Therefore, we have decided to adopt STEVX for computing eigenvalues and eigenvectors of (4), even though it also has known failure modes. In particular, in extreme situations of tightly clustered eigenvalues bisection may potentially not converge to the desired accuracy, or not all eigenvalues from the IL-th through the IU-th may be found, or inverse iteration may fail to converge with the allowed number of iterations. Although our exhaustive tests with STEVX (including the ones shown in Section V) have never detected such anomalies, we plan to replace STEVX with a more robust implementation of MRRR for (4) when such an implementation

becomes available[2]. MRRR would dispense with reorthogonalizations that can take a good share of the computing time for matrices with tight clusters of eigenvalues. We emphasize that it is not enough to obtain accurate eigenpairs for (4): we also need the extracted u's and v's to be accurate[3].

The main contribution of this paper is to present and discuss an implementation of an algorithm for the SVD of a bidiagonal matrix obtained from eigenpairs of a tridiagonal matrix T_{GK}. This implementation is called BDSVDX, which was first introduced in LAPACK 3.6.0 [15], with preliminary results reported in [18]. While the associated formulation is not necessarily new, as mentioned above, its actual implementation requires care in order to deal correctly with multiple or tightly clustered singular values. To the best of our knowledge, no such implementation has been done and exhaustively tested. In concert with BDSVDX we have also developed GESVDX (real and complex versions), which takes a general matrix A, reduces it to bidiagonal form B, invokes BDSVDX, and then maps the output of BDSVDX into the SVD of A. In LAPACK, the current counterparts of GESVDX are GESVD and GESDD, which are based on the BD subroutines listed in Table I and can only compute all singular values (and optionally singular vectors). This can be much more expensive if only a few singular values and vectors are desired. For example, our experiments showed that BDSVDX can be 3 orders of magnitude faster than its counterparts, and GESVDX can be 10 times faster than its counterparts.

The rest of the paper is organized as follows. First, we discuss how singular values are mapped into the eigenvalue spectrum. Then, we discuss special cases, the criterion for splitting a bidiagonal matrix, and other implementation details. Next, we show the results of our tests with BDSVDX using a large set of bidiagonal matrices, to assess both its accuracy (in single and double precisions) and its performance with respect to BDSQR and BDSDC[4]. We also compare the performance of GESVDX with those of GESVD and GESDD (in double precision, real and complex), and show how the time is spent in the various phases of GESVDX. Finally, we discuss limitations of the algorithm and opportunities for future work.

II. Mapping singular values into eigenvalues

Similarly to BDSQR and BDSDC, BDSVDX allows the computation of singular values only or singular values and the corresponding singular vectors[5]. Borrowing features from STEVX, BDSVDX can be used in three modes, through a character variable RANGE. If RANGE="A", all singular values will be found: BDSVDX will compute the smallest (negative or zero) n eigenvalues of the corresponding T_{GK}. If RANGE="V", all singular values in the half-open interval [VL,VU) will be found: BDSVDX will compute the eigenvalues of the corresponding T_{GK} in the interval (-VU,-VL]. If RANGE="I", the IL-th through IU-th singular values will be found: the indices IL and IU are mapped into values (appropriate VL and VU) by subroutine STEBZ, which applies bisection to T_{GK}. VL, VU, IL and IU are arguments of BDSVDX (which are mapped into similar arguments for STEVX).

For a bidiagonal matrix B of dimension n, if singular vectors are requested, BDSVDX returns an array $\mathcal{Z}$ of dimension $2n \times p$, where $p \leq n$ is a function of RANGE. Each column of $\mathcal{Z}$ will contain $(u_k^T, v_k^T)^T$ corresponding to singular value s_k, i.e.

$$\mathcal{Z} = \begin{bmatrix} U \\ V \end{bmatrix}, \tag{5}$$

where $U = [u_k, u_{k+1}, \ldots u_{k+p-1}]$, $V = [v_k, v_{k+1}, \ldots v_{k+p-1}]$, with k depending on RANGE and possibly also VL, VU, IL and IU.

STEVX returns eigenvalues (and corresponding vectors) in ascending order, so we target the negative part of the eigenvalue spectrum (i.e. $-S$) in (2). Therefore, the absolute values of the returned eigenvalues

[2]At the time of this writing, [15] lists two issues (bugs) related to STEMR. The issues are related to safeguards mechanisms against NaNs, which may lead to an early termination of STEMR even for some relatively benign matrices.

[3]We note that [3] discusses the computation of singular values by spectrum slicing, but not the corresponding singular vectors.

[4]A modified and potentially faster version of BDSDC exploiting low-rank properties of broken arrow matrices has been proposed in [16]; see also [23].

[5]The returned singular values are the same in both cases since STEVX is invoked in a way that forces the use of bisection, instead of QR or the Pal-Walker-Kahan variant of the QL or QR algorithm, for computing the eigenvalues of T_{GK}.

give us the singular values in the desired order, $s_1 \geq s_2 \geq \ldots s_n \geq 0$. We only need to change the signs of the entries in the eigenvectors that are reloaded to V. We note that BDSVDX inherits the shortcomings of STEVX mentioned in the Introduction (i.e. bisection may fail to converge, not all eigenvalues in a given interval may be found, or inverse iteration may fail to converge).

III. SPLITTING: SPECIAL CASES

Our code must deal properly with cases when one, or more, of the $2n-1$ parameters (a_i, b_i) vanish. Most presentations just take the case when the bidiagonal matrix B is square. However, when $a_j = 0, 1 < j < n$, then B may be written

$$B = \begin{bmatrix} B_1 & 0 \\ 0 & B_2 \end{bmatrix} \tag{6}$$

where B_1 is $(j-1) \times j$ and B_2 is $(n-j+1) \times (n-j)$. Thus, neither B_1 nor B_2 is square. In particular, B_1 must have a normalized column null vector $\hat{v}$, $B_1\hat{v} = 0_{(j-1)\times 1}$, which exhibits the linear dependence of B_1's columns. Likewise, there must be a null vector $\hat{u}$, $\hat{u}^T B_2 = 0_{1\times(n-j)}$, exhibiting the dependence among B_2's rows. In general, B_1's rows and B_2's columns are linearly independent sets if no other diagonal or superdiagonal entries are zero.

The Golub-Kahan matrix $T_{GK}^{(1)}$ for B_1 comes from the $(2j-1) \times (2j-1)$ matrix

$$\begin{bmatrix} 0 & B_1 \\ B_1^T & 0 \end{bmatrix} = P_1 T_{GK}^{(1)} P_1^T$$

where

$$T_{GK}^{(1)} = \mathsf{tridiag}\left(\begin{array}{ccccccccccc} & a_1 & & b_1 & & a_2 & & b_2 & \ldots & a_{j-1} & & b_{j-1} & \\ 0 & & 0 & & 0 & & 0 & & \ldots & & 0 & & 0 \\ & a_1 & & b_1 & & a_2 & & b_2 & \ldots & a_{j-1} & & b_{j-1} & \end{array} \right),$$

and P_1 is the perfect shuffle of appropriate dimension. In this case, $T_{GK}^{(1)}$ is of odd order and must be singular because $T_{GK}^{(1)}$ is similar to $-T_{GK}^{(1)}$, so its nonzero eigenvalues come in plus-minus pairs, and the sum of all the eigenvalues is zero. By the observation above, the corresponding null vector must be

$$P_1^T \begin{bmatrix} 0_{(j-1)\times 1} \\ \hat{v} \end{bmatrix},$$

because

$$T_{GK}^{(1)} P_1^T \begin{bmatrix} 0_{(j-1)\times 1} \\ \hat{v} \end{bmatrix} = P_1^T \begin{bmatrix} 0 & B_1 \\ B_1^T & 0 \end{bmatrix} P_1 P_1^T \begin{bmatrix} 0_{(j-1)\times 1} \\ \hat{v} \end{bmatrix} = P_1^T \begin{bmatrix} 0_{(j-1)\times 1} \\ 0_{j\times 1} \end{bmatrix}$$

since $B_1\hat{v} = 0_{(j-1)\times 1}$.

The expressions for B_2's singular triples (σ, u, v^T) follow a similar pattern, provided that we write the augmented $(2(n-j)+1) \times (2(n-j)+1)$ matrix as

$$\begin{bmatrix} 0 & B_2^T \\ B_2 & 0 \end{bmatrix} = P_2 T_{GK}^{(2)} P_2^T$$

since B_2 has more rows than columns, where

$$T_{GK}^{(2)} = \mathsf{tridiag}\left(\begin{array}{ccccccccc} & a_{j+1} & & b_{j+1} & & a_{j+2} & \ldots & b_{n-1} & & a_n & \\ 0 & & 0 & & 0 & & \ldots & & 0 & & 0 \\ & a_{j+1} & & b_{j+1} & & a_{j+2} & \ldots & b_{n-1} & & a_n & \end{array} \right)$$

and P_2 is the perfect shuffle of appropriate dimension. The row null vector for B_2 must be $(0_{1\times(n-j)}, \hat{u}^T)P_2$, because

$$(0_{1\times(n-j)}, \hat{u}^T) P_2 T_{GK}^{(2)} = (0_{1\times(n-j)}, \hat{u}^T) P_2 P_2^T \begin{bmatrix} 0 & B_2^T \\ B_2 & 0 \end{bmatrix} P_2 = (0_{1\times(n-j)}, 0_{1\times(n-j+1)}) P_2$$

since $\hat{u}^T B_2 = 0_{1\times(n-j)}$.

The other mappings between singular vectors of B_1 and B_2 and eigenvectors of $T_{GK}^{(1)}$ and $T_{GK}^{(2)}$ are the same as in the square case. We include them here for completeness. Consider $B_1 v = u\sigma$, $u^T B_1 = \sigma v^T$, $\sigma > 0$, $\|u\| = \|v\| = 1$. Then,

$$\begin{bmatrix} u \\ \pm v \end{bmatrix} \sigma = \begin{bmatrix} 0 & B_1 \\ B_1^T & 0 \end{bmatrix} \begin{bmatrix} u \\ \pm v \end{bmatrix} = P_1 T_{GK}^{(1)} P_1^T \begin{bmatrix} u \\ \pm v \end{bmatrix},$$

which reveals that

$$P_1^T \begin{bmatrix} u \\ \pm v \end{bmatrix} \frac{1}{\sqrt{2}}$$

are $T_{GK}^{(1)}$'s normalized eigenvectors for eigenvalues $+\sigma$ and $-\sigma$.

In practice, and as mentioned before, we choose to compute only the wanted nonpositive eigenpairs in monotone increasing order because that delivers the wanted singular triples in the conventional monotone decreasing order of singular values. When $T_{GK}^{(1)}$'s spectrum is labeled $\lambda_1 \le \lambda_2 \le \ldots \lambda_{j-1} \le \lambda_j \le \lambda_{j+1} \cdots \le \lambda_{2j-1}$, $\lambda_j = 0$, then $\sigma_1 = |\lambda_{min}| = |\lambda_1|$. Finally, we note that the case $j = 1$ fits the general pattern of (6) with no first row $(0 \;\; B_2)$, and the case $j = n$ fits with no second row $(B_1 \;\; 0)$.

Splitting and the output array $\mathcal{Z}$

As discussed above, if, for a given i, $b_i = 0$ (or it is tiny enough to be set to zero, as discussed later) the matrix B splits and the SVD for each resulting (square) submatrix of B can be obtained independently. In the following, we use small matrices B to illustrate the splitting in the main diagonal and its effect on $\mathcal{Z}$.

a) Zero in the interior: Let us assume that $n = 5$ and $a_3 = 0$. Then, we have the following SVD:

$$B = \begin{bmatrix} a_1 & b_1 & & & \\ & a_2 & b_2 & & \\ & & 0 & b_3 & \\ & & & a_4 & b_4 \\ & & & & a_5 \end{bmatrix} = \begin{bmatrix} U_1 & \\ & U_2 \end{bmatrix} \begin{bmatrix} S_1 & \\ & S_2 \end{bmatrix} \begin{bmatrix} V_1^T & \\ & V_2^T \end{bmatrix},$$

where U_1 and V_2 are 2-by-2, U_2 and V_1 are 3-by-3, S_1 is 2-by-3 (its third column contains only zeros), and S_2 is 3-by-2 (its third row contains only zeros). The first three columns of the eigenvector matrices of $T_{GK}^{(1)}$ and $T_{GK}^{(2)}$ are

$$Z_{5\times3}^{(1)} = \begin{bmatrix} v_{1,1}^{(1)} & v_{1,2}^{(1)} & v_{1,3}^{(1)} \\ u_{1,1}^{(1)} & u_{1,2}^{(1)} & 0 \\ v_{2,1}^{(1)} & v_{2,2}^{(1)} & v_{2,3}^{(1)} \\ u_{2,1}^{(1)} & u_{2,2}^{(1)} & 0 \\ v_{3,1}^{(1)} & v_{3,2}^{(1)} & v_{3,3}^{(1)} \end{bmatrix} D^{-1}, \quad Z_{5\times3}^{(2)} = \begin{bmatrix} u_{1,1}^{(2)} & u_{1,2}^{(2)} & u_{1,3}^{(2)} \\ v_{1,1}^{(2)} & v_{1,2}^{(2)} & 0 \\ u_{2,1}^{(2)} & u_{2,2}^{(2)} & u_{2,3}^{(2)} \\ v_{2,1}^{(2)} & v_{2,2}^{(2)} & 0 \\ u_{3,1}^{(2)} & u_{3,2}^{(2)} & u_{3,3}^{(2)} \end{bmatrix} D^{-1},$$

where $Z_{5\times3}^{(1)}$ and $Z_{5\times3}^{(2)}$ show how the entries of the eigenvectors corresponding to the three smallest eigenvalues of $T_{GK}^{(1)}$, $\lambda_1^{(1)} < \lambda_2^{(1)} < \lambda_3^{(1)}$, and $T_{GK}^{(2)}$, $\lambda_1^{(2)} < \lambda_2^{(2)} < \lambda_3^{(2)}$ relate to the entries of U_1, U_2, V_1 and V_2, where $v_{ij}^{(1)}$ are the entries of V_1 and so on. Note that the left and right singular vectors corresponding to s_3 are in different Z matrices, and D is a diagonal matrix with entries $(\sqrt{2}, \sqrt{2}, 1)$. In this case, the 10-by-5 array Z (see (5)) returned by `BDSVDX` is

$$\mathcal{Z} = P \begin{bmatrix} Z_{5\times2}^{(1a)} & Z_{5\times3}^{(1b)} \\ 0 & Z_{5\times3}^{(2)} \end{bmatrix},$$

where P is the perfect shuffle of appropriate dimension, $Z_{5\times2}^{(1a)}$ contains the first two columns of $Z_{5\times3}^{(1)}$, and $Z_{5\times3}^{(1b)}$ contains zeros in its two first columns and the last column of $Z_{5\times3}^{(1)}$ in its third column.

b) Zero at the top: If $n = 4$ and $a_1 = 0$, then

$$B = \begin{bmatrix} 0 & b_1 & & \\ & a_2 & b_2 & \\ & & a_3 & b_3 \\ & & & a_4 \end{bmatrix} = \begin{bmatrix} \ U \ \end{bmatrix} \begin{bmatrix} 0 & \\ & S \end{bmatrix} \begin{bmatrix} 1 & \\ & V^T \end{bmatrix},$$

where the left singular vector matrix U is 4-by-4, S is 3-by-3, and the right singular vector matrix V is 3-by-3. If we construct a T_{GK} from B, its first row and column will be zero, and the entries of the eigenvectors corresponding to the five smallest eigenvalues of T_{GK} (again, related explicitly to singular values of B) relate to the entries of U and V as follows:

$$Z_{8\times 5} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & u_{1,1} & u_{1,2} & u_{1,3} & u_{1,4} \\ 0 & v_{1,1} & v_{1,2} & v_{1,3} & 0 \\ 0 & u_{2,1} & u_{2,2} & u_{2,3} & u_{2,4} \\ 0 & v_{2,1} & v_{2,2} & v_{2,3} & 0 \\ 0 & u_{3,1} & u_{3,2} & u_{3,3} & u_{3,4} \\ 0 & v_{3,1} & v_{3,2} & v_{3,3} & 0 \\ 0 & u_{4,1} & u_{4,2} & u_{4,3} & u_{4,4} \end{bmatrix} D^{-1},$$

where D is a diagonal matrix with entries $(1, \sqrt{2}, \sqrt{2}, \sqrt{2}, 1)$. In this case, the array $\mathcal{Z}$ (see (5)) returned by `BDSVDX` is formed by taking the last four columns of $Z_{8\times 5}$, and its last column is concatenated with the first column of $Z_{8\times 5}$.

c) Zero at the bottom: If $n = 4$ and $a_4 = 0$, then

$$B = \begin{bmatrix} a_1 & b_1 & & \\ & a_2 & b_2 & \\ & & a_3 & b_3 \\ & & & 0 \end{bmatrix} = \begin{bmatrix} \ U & \\ & 1 \end{bmatrix} \begin{bmatrix} S & \\ & 0 \end{bmatrix} \begin{bmatrix} \ V^T \ \end{bmatrix},$$

where the left singular vector matrix U is 3-by-3, S is 3-by-3, and the right singular vector matrix V is 4-by-4. If we construct a T_{GK} from B, its last row and column will be zero, the entries of the eigenvectors corresponding to the five smallest eigenvalues of T_{GK} (again, related explicitly to singular values of B) relate to the entries of U and V as follows:

$$Z_{8\times 5} = \begin{bmatrix} v_{1,1} & v_{1,2} & v_{1,3} & v_{1,4} & 0 \\ u_{1,1} & u_{1,2} & u_{1,3} & 0 & 0 \\ v_{2,1} & v_{2,2} & v_{2,3} & v_{2,4} & 0 \\ u_{2,1} & u_{2,2} & u_{2,3} & 0 & 0 \\ v_{3,1} & v_{3,2} & v_{3,3} & v_{3,4} & 0 \\ u_{3,1} & u_{3,2} & u_{3,3} & 0 & 0 \\ v_{4,1} & v_{4,2} & v_{4,3} & v_{4,4} & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} D^{-1},$$

where D is a diagonal matrix with entries $(\sqrt{2}, \sqrt{2}, \sqrt{2}, 1, 1)$. In this case, the array $\mathcal{Z}$ (see (5)) returned by `BDSVDX` is formed by taking the first four columns of $Z_{8\times 5}$, and its last column is concatenated with the last column of $Z_{8\times 5}$.

Criterion for splitting

In our implementation, we first form the matrix T_{GK} and then check for splitting in two phases, first the superdiagonal entries of T_{GK} with row indexes $2, 4, \ldots$ (i.e. b_i). If a submatrix is found (or the bottom of the matrix B is reached) we check the superdiagonal entries of T_{GK} with row indices $1, 3, \ldots$ (i.e. a_i). Thus, if the matrix splits in a, the problem can be reduced to one of the three special cases described above. The criterion that `BDSVDX` uses for splitting is the same that is used in `BDSQR`; it is discussed in [6]. We note that the LAPACK subroutine that performs bisection, `STEBZ`, also checks for potential splits, using a different criterion. However, in our tests we have not noticed any additional splits performed in `STEBZ`. Our testing infrastructure contains matrices to trigger the cases of splitting discussed above. The infrastructure also allows the setting of a percentage of entries of B that will be randomly set to 0, thus enabling the testing of a variety of splitting patterns.

IV. Refinement of vectors

As discussed earlier, an eigenvector z_i of T_{GK}, $i \leq 1 \leq n$, corresponds to $z_i = P(u_i^T, -v_i^T)^T/\sqrt{2}$. This means that we could simply create a vector $\hat{u}_i$ with the even entries of z_i and a vector $\hat{v}_i$ with the odd entries of z_i and multiply those vectors by $\sqrt{2}$ in order to obtain u_i and v_i. However, in our implementation we explicitly normalize $\hat{u}_i$ and $\hat{v}_i$. This allows us to check how far the norms of $\hat{u}_i$ and $\hat{v}_i$ are from $\frac{1}{\sqrt{2}}$, which may be the case if z_i is associated with a small eigenvalue. In fact, in [26] the authors observe that there may be a deviation of orthogonality in the singular vectors extracted from the eigenvectors of T_{GK} for some bidiagonals with tight clusters or very small singular values. This is related to the minors of T_{GK} with an odd dimension, which are all singular.

As a safeguard, if necessary, we apply a Gram-Schmidt reorthogonalization to $\hat{u}_i$ and $\hat{v}_i$ against the previous vectors. The test we have implemented to trigger reorthogonalization is based on $|\|\hat{u}\| - \frac{1}{\sqrt{2}}| \geq tol$, similarly for $\hat{v}$, where $tol = \sqrt{\varepsilon}$ and ε is the machine precision. While this refinement seems to work well for most cases, we have found situations for which the test based on $\|\hat{u}\|$ (and $\|\hat{v}\|$) is not sufficient. This is the case of the bidiagonal matrix $B_{8\times 8}$ ($n = 8$) defined as[6]

$$a_i = 10^{-(2i-1)},\ i = 1, 2, \ldots n,$$
$$b_i = 10^{-(2i-2)},\ i = 1, 2, \ldots n-1,$$

whose norm is ≈ 1.005, condition number is $O(10^{22})$, and the three smallest singular values are $s_6 = O(10^{-10})$, $s_7 = O(10^{-12})$ and $s_8 = O(10^{-22})$. For $B_{8\times 8}$, $|\|\hat{u}_i\| - \frac{1}{\sqrt{2}}| < 10\varepsilon$, $i = 1, 2, \ldots n$. Similarly for v_i, $i = 1, 2, \ldots n$. However, taking the output U, S, V of `BDSVDX` in double precision for $B_{8\times 8}$, we obtain $\|U^T B_{8\times 8} V - S\|/(\|B_{8\times 8}\| n\varepsilon) < 1.0$, while $\|I - U^T U\|/(n\varepsilon)$ and $\|I - V^T V\|/(n\varepsilon)$ are $O(10^5)$. To illustrate, Figs. 1a-1b show the orthogonality levels of U and V. If we compare the entries of U and V obtained with `BDSVDX` with the ones obtained with `BDSQR`[7] we observe that the largest differences in those entries are in the 7th entry of u_8 and in the 1st entry of v_8: the output from `BDSVDX` agrees with the output from `BDSQR` in 10 and 9 digits, respectively, in those particular entries.

The eigenvectors z_7 and z_8 of the T_{GK} obtained from $B_{8\times 8}$, i.e. the eigenvectors associated with eigenvalues $-s_7$ and $-s_8$, have small reciprocal condition numbers [1, p. 103], $O(10^{-12})$, leading to singular vectors that are not orthogonal to machine precision. Yet, the eigenvectors z of T_{GK} as computed by `STEVX` are orthogonal to working precision; specifically, $\|I - Z^T Z\|/(2n\varepsilon) = 0.125, Z = [z_1, z_2 \ldots z_n]$. Experiments have shown that if we force the reorthogonalization of u_8 against u_i, $i = 4, \ldots 7$, we obtain $\|I - U^T U\|/(n\varepsilon) < 1.0$. Similarly, if we reorthogonalize v_8 against v_i, $i = 4, \ldots 7$, we obtain $\|I - V^T V\|/(n\varepsilon) < 1.0$. This suggests that the strategy for triggering reorthogonalization needs also to take into account the separation and magnitude of the eigenvalues and not only $\|\hat{u}\|$ and $\|\hat{v}\|$ (as in the current LAPACK implementation) or simply a tighter tol. However, such a strategy remains to be further investigated.

V. Numerical experiments

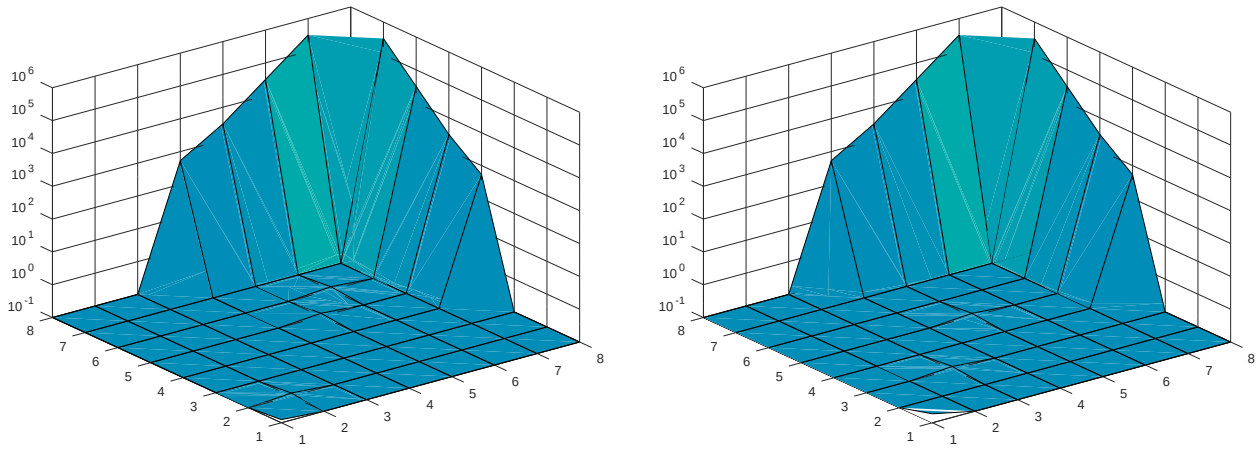
We have used a large set of bidiagonal matrices[8] to test `BDSVDX`, on a computer with a 4-core Intel Core i7-7700K processor at 4.2 GHz, 256 KB of L1 cache (64 KB per core), 1 MB of L2 cache (256 KB per core), 8 MB of L3 cache, 32 GB of memory, in double and single precisions, using the Intel and GNU Fortran compilers. We show results obtained with the Intel compiler [13], with flags `-O2 -fp-model strict`, using Intel's MKL (BLAS) [14], and LAPACK 3.7.0[9]. Most of the test matrices in our testbed are derived from symmetric tridiagonal matrices described in [17] (also used in [7]). In this case, we factor $T - \nu I = LL^T$ (Cholesky) for a proper value of ν (obtained from the Gerschgorin bounds of T),

[6] This matrix is not included in the tests in Section V.

[7] `BDSQR` returns U and V that are orthogonal to machine precision.

[8] All matrices used in our experiments are available upon request. The matrices will be included in a future release of LAPACK.

[9] The results may vary for the more difficult cases, depending on the compiler and compilation flags.



(a) $\left|I - U^T U\right| / (n\varepsilon)$ (b) $\left|I - V^T V\right| / (n\varepsilon)$

Fig. 1: Orthogonality level (surface mesh, log scale) of U and V for $B_{8\times 8}$.

then set $B = L^T$. The testbed also includes random bidiagonal matrices, with entries obtained from a standard normal distribution and a uniform distribution, generated by LAPACK's functions `LARND` and `LARNV`.

To test the accuracy of `BDSVDX`, we compute

$$\begin{aligned} resid &= \|U^T B V - S\|/(\|B\| n\varepsilon) \\ orthU &= \|I - U^T U\|/(n\varepsilon) \\ orthV &= \|I - V^T V\|/(n\varepsilon) \end{aligned}$$

where n is the dimension of B and ε is the machine precision. To test the features `RANGE`="I" and `RANGE`="V" for a given B, we build the corresponding T_{GK} prior to invoking `BDSVDX` and compute its eigenvalues using bisection (i.e. `STEBZ`). Then, for `RANGE`="V" we generate n_V pairs of random indexes IL and IU, map those indexes into the eigenvalues of T_{GK}, perturb the eigenvalues slightly to obtain corresponding pairs VL and VU, and then invoke `BDSVDX` n_V times with the corresponding pair of values. We need VL<VU, a requirement for `STEVX` when `RANGE`="V", motivating the small perturbations in the eigenvalues of T_{GK}. For `RANGE`="I" we simply generate n_I pairs of random indexes IL and IU, and then invoke `BDSVDX` n_I times with the corresponding pair of indexes. We recall that in `BDSVDX` indices IL and IU are mapped into values (as in `STEVX`). This mapping can produce values that differ from the ones obtained by simply perturbing the eigenvalues.

We are also interested in the time `BDSVDX` takes to compute all singular values and vectors and, most importantly, subsets of singular values and vectors. We compare these times with the times taken by `BDSQR` and `BDSDC`. One of our goals is to identify a breakpoint in which the computation of a full SVD would be preferable to a subset. We extend this analysis by comparing `GESVDX`, which is built on top of `BDSVDX`, to its counterparts `GESVD` and `GESDD`, in double and double precision complex, using a set of random matrices with a varying number of rows and columns.

Accuracy in double precision

Fig. 2 shows the accuracy of `BDSVDX` in double precision. Figs. 2a-2c correspond to the computation of all singular values and vectors (`RANGE`="A"), for 250 bidiagonal matrices with dimensions ranging from 9 to 4006. Figs. 2d-2i show the accuracy of `BDSVDX` for the same matrices of `RANGE`="A", but with $n_I = 10$ (random) pairs of IL, IU (`RANGE`="I"), and $n_V = 10$ (random) pairs of VL, VU (`RANGE`="V") for each matrix. In the figures, the matrices (y-axis) are ordered according to their condition numbers (lowest to highest), which range from 1.0 to $> 10^{200}$ (using singular values computed by `BDSQR`). For

the sake of clarity, we use floor and ceiling functions to bound the results in the x-axis, setting its limits to 10^{-1} and 10^4. In other words, for plotting purposes we use $max(10^{-1}, resid)$, and $min(10^4, resid)$ as limits; similarly for $orthU$ and $orthV$. We discuss the cases for which the results are bigger than the axis limit. To assist in the interpretation of the results, Fig. 3 shows a histogram of the symbols displayed in Fig. 2.

For RANGE="A", as can be seen in Figs. 2a-2c and Fig. 3a, the majority of the results are adequately below 1.0. There are three to five cases with results above 10.0, and we consider the outliers to be the ones above 100. Specifically, in Fig. 2a:

a) Matrix 36 is a bidiagonal matrix of dimension 1260 obtained from a tridiagonal matrix computed by running the Lanczos algorithm [5] without reorthogonalization $3 \times n_0$ steps, where n_0 is the dimension of the original (sparse) matrix (BCSSTK07, from the BCSSTRUC1 set in [19]). This strategy leads to tridiagonal matrices with very close eigenvalues: for example, the largest 138 eigenvalues of matrix 36 agree to 12 digits ($4.52093556010 \times 10^{-3}$).
b) Matrix 225 is a bidiagonal matrix of dimension 396 generated from a singular value distribution with large clusters: the largest 41 singular values, for example, agree to 13 digits ($2.311336378236 \times 10^{-2}$).
c) Matrices 248-250 (outliers) are notoriously difficult (they are borrowed from the LAPACK SVD tester, CHKBD): their entries are defined as e^x, where x is chosen uniformly on $[2 \log \varepsilon, -2 \log \varepsilon]$, therefore ranging from ε^{-2} to ε^2. The dimensions of those matrices are 125, 250 and 500, respectively. For $n = 500$, $s_1 \approx O(10^{+31})$ and $s_n \approx O(10^{-284})$ (as computed by BDSQR). For these matrices, $resid$ is $O(10^{-2})$ but $orthU$ and $orthV$ are $O(10^{+13})$.

As expected, the effect of large clusters of singular values of matrices 36 and 225, and the oddities of matrices 248-250 in Figs. 2a-2c, are propagated to Figs. 2d-2i. However, Figs. 2g-2i contain additional results above 10.0. In these figures, case 966 (matrix 97 in Figs. 2a-2c) is a bidiagonal matrix of dimension 2100 obtained from a tridiagonal matrix formed by gluing 100 copies of the Wilkinson matrix W21+[10] with glue factor 10^{-11}, so its eigenvalues (and therefore singular values) are highly clustered. (The values set to VL and VU in case 966 lead to the computation of 1657 singular values and vectors.)

Accuracy in single precision

Fig. 4 shows the accuracy of BDSVDX in single precision. To assist in the interpretation of the results, Fig. 5 shows a histogram of the symbols displayed in Fig. 4. Most of 250 the matrices used in Fig. 2 are read from files and are also used for the experiments in single precision. The matrices that are generated at execution time in Fig. 2 are regenerated in single precision arithmetic.

Figs. 4a-4c correspond to the computation of all singular values and vectors (RANGE="A"). Figs. 4d-4i show the accuracy of BDSVDX for the same matrices of Fig. 4a, with $n_I = 10$ (random) pairs of IL, IU (RANGE="I"), and $n_V = 10$ (random) pairs of VL, VU (RANGE="V") for each matrix. As in the double precision case, the matrices (y-axis) are ordered according to their condition numbers. For convenience, we use floor and ceiling functions to bound the results in the x-axis, setting its limits to 10^{-1} and 10^4.

The matrices that lead to results larger than 1000 in single precision are similar to the ones in double precision. Matrix 129 in Figs. 4a-4c has dimension 1083, and it is another case of a bidiagonal obtained from a tridiagonal matrix computed by running the Lanczos algorithm [5] without reorthogonalization. In double precision, its 139 largest eigenvalues agree to 12 digits, $3.44013410743 \times 10^{-8}$. There are more results larger than 100 in single precision than in double precision for RANGE="I" and RANGE="V". For example, cases 1030 and 1501 in Figs. 4d-4f (matrices 103 and 151 in Figs. 4a-4c) are instances of bidiagonal matrices of dimension 2100 obtained from tridiagonal matrices formed by gluing 100 copies of the Wilkinson matrix W21+. As noted before, these matrices have large clusters of tight singular values, and the single precision version exhibits a slightly different behavior for RANGE="I" and RANGE="V".

[10]Its diagonal entries are $(10, 9, 8, \ldots, 0, \ldots 8, 9, 10)$ and its off-diagonal entries are all 1.

Refinement of vectors

The strategy for refinement of vectors discussed in Section IV was set off for matrices 248-250 in Fig. 2a, but it was not sufficient to produce orthogonal vectors. As mentioned before, the entries of those matrices range from ε^{-2} to ε^2, and we have observed that almost all their singular vectors are perturbations of e_i, the columns of the identity matrix of appropriate dimension. Taking matrix 250 as an example, $n = 500$, $u_{126} \approx e_{80}$ and $u_{144} \approx e_{78}$, and we have verified that these are the only two vectors that are not fully orthogonal: their inner product is $O(10^{-10})$. On the other hand, the strategy for refinement was set off for matrices 108, 242-245, and 249-250 in Fig. 4a. For all these matrices the resulting orthogonality level is smaller than 10. We note that the characteristics of the jth matrix in Fig. 4a may differ significantly from the characteristics of the jth matrix in Fig. 2a due to way the matrices are generated, and also differences in the condition numbers in single and double precisions.

Performance

Fig. 6 shows the times taken by `BDSQR`, `BDSDC` and `BDSVDX` on 12 bidiagonal matrices (a sample from Fig. 2a) with dimensions ranging from 675 to 4006, in double precision. In our experiments, we ask `BDSQR` to compute all left and right singular vectors. In turn, `BDSDC` has an option for returning the left and right singular vectors in a compact form but we ask `BDSDC` to compute the left and right singular vectors explicitly. The matrices are ordered according to their sizes (x-axis), and exhibit a variety of singular value distributions: they are related to applications in power systems and structural engineering (from the `PSADMIT` and `BCSSTRUC1` sets in [19]) and computational chemistry[11]. For `BDSVDX`, we ask for all singular values/vectors, the largest 20% singular values/vectors, the largest 10% singular values/vectors, and the largest 5 singular values/vectors. For each matrix, the timings are normalized with respect to the time taken by `BDSQR`. As somehow expected, `BDSVDX` is not competitive for all or a relatively large set of singular values and vectors. The gains can be noticed for 10% (or less) singular values/vectors; in particular, `BDSVDX` is about 3 orders of magnitude faster than `BDSQR` and 2 orders of magnitude faster than `BDSDC` for the computation of the largest 5 singular values and vectors of the largest matrix.

Note that the computation of the largest 10% singular values/vectors and the largest 5 singular values/vectors for matrix 6 takes about the same time; similarly for matrix 11. Those bidiagonal matrices are obtained from tridiagonal matrices (related to matrices in the `BCSSTRUC1` set in [19]) computed by running the Lanczos algorithm without reorthogonalization, as mentioned before. Matrix 6 in Fig. 6 is matrix 36 in Fig. 2a: its largest 138 eigenvalues agree to 12 digits ($4.52093556010 \times 10^{-3}$). Matrix 11 in Fig. 6 is matrix 129 in Fig. 2a: its largest 325 eigenvalues agree to 13 digits ($1.307880412385 \times 10^{7}$). For these two matrices, the mapping of IL=1 and IU=5 (for the largest 5 singular values) into VL and VU results in intervals with a large number of very close singular values, intervals with 138 and 325 singular values, respectively. The additional computed values and vectors are later discarded (to conform to the input values for IL and IU, as implemented in `STEVX`).

In terms of memory footprint, `BDSDC` is typically the most demanding, requiring more than what is available in the three levels of cache of the computer used for the experiments. The footprint of `BDSQR` and `BDSVDX (A)` are similar, less than 50% of the footprint of `BDSDC`, and fitting in the cache for the three smallest matrices. In contrast, the memory needed by `BDSVDX (5)` can be accommodated in the first two levels of cache for the smallest matrices; it requires a fraction of the third level for some of the largest matrices. The exception are the matrices with very tight clusters of eigenvalues, as mentioned above, since `BDSVDX` may end by computing a large set of vectors.

Fig. 7 shows the performance of `BDSVDX` for the matrices in Fig. 6, for the computation of the largest 20%, the largest 10% and the largest 5 singular values/vectors. The data points in Fig. 7 correspond to the computing time in each of those three scenarios normalized with respect to t_A, where t_A is the time required for the computation of all singular values/vectors. The figure clearly shows how the

[11]Matrices provided to us by George Fann.

performance of BDSVDX may be affected by the distribution of the singular values, as discussed in the previous paragraph. We have used TAU [21] to profile BDSVDX while computing the largest 5 singular values/vectors for a sample of the matrices. For matrices 6 and 11, for example, most of the time is spent with reorthogonalization of the vectors (modified Gram-Schmidt, up to 80% of the total time) followed by solves (inverse iteration). For matrix 9, bisection takes $\approx 10\%$ of the total time while a combination of other operations (e.g. normalizations in BDSVDX, etc.) dominate.

Fig. 8 compares the times taken by GESVD, GESDD and GESVDX in double precision, on random dense $m \times n$ matrices with $m, n = 500, 1000, 1500, 2000$. GESVDX is used to compute the largest 20%, the largest 10% and the largest 5 singular values/vectors. It is consistently faster than its counterparts, which are limited to a full SVD, for 10% or less singular values/vectors. In the double precision case, Fig. 8a, GESVDX is up to 14 times faster than GESVD and 2 times faster than GESDD. (We observe that for all matrices in Fig. 8a BDSVDX is faster than BDSDC since the singular value of those matrices are relatively well separated, in contrast to some of the matrices in Fig. 6.) In the double precision complex case, Fig. 8b, GESVDX is up to 5.6 times faster than GESVD and 1.7 times faster than GESDD. We note that the performance of GESDD may be greatly penalized if a non-optimized BLAS library is used, and recall that in our experiments we use Intel's MKL.

Finally, Fig. 9 shows how the time is spent by GESVDX for the matrices of Fig. 8, for the 5 largest singular values/vectors scenario. The relevant calculations are reduction of the input matrix to bidiagonal form B, computation of the singular values/vectors of B with BDSVDX, and back transformation of the singular vectors of B to those of the input matrix. The bars in the figure are normalized with respect to the time taken by the reduction to bidiagonal form, which typically dominates the costs. As it can be observed, the computation of the singular values/vectors and the back transformation phases are much faster than bidiagonal reduction.

VI. Conclusions

This paper presents an algorithm for the computation of the SVD of a bidiagonal matrix by means of the eigenpairs of an associated tridiagonal matrix. The implementation, BDSVDX (first included in the LAPACK 3.6.0 release), provides for the computation of a subset of singular values/vectors, which is important for many large dimensional problems that do not require the full set. Our experiments revealed that this feature can lead to significant gains in computing times, when compared with existing implementations that are limited to the computation of the full SVD. These gains are transmitted to a higher level routine intended for the computation of a subset of singular values/vectors of a general matrix.

Numerical results on a large set of test matrices substantiated the accuracy of the implementation; the few exceptions are related to matrices with very large condition numbers or highly clustered singular values. We have also identified pathological cases (typically matrices with very small singular values) for which the computed singular vectors may not be orthogonal to machine precision. A more robust strategy to identify and cope with such cases remains to be investigated.

For a parallel implementation of the algorithm presented here, we can built upon the workflow of the parallel subroutines PDSYEVX or PDSYEVR that are implemented in ScaLAPACK [4], [20]. The former is based on bisection and inverse iteration, with the caveat that it does not do reorthogonalization with vectors on distinct processes, so the returned vectors may not be orthogonal in the case of tight clusters of values. The latter is based on the MRRR algorithm and presumably delivers more satisfactory results and scalability [22]. Specific tests will be required (e.g. with cases similar to the difficult ones in Fig. 2) to assess the best alternative. We observe that once a matrix similar to (5) is obtained, the back transformation of the singular vectors of B to those of the input matrix can be then parallelized in different ways.

Acknowledgments: The authors thank Beresford N. Parlett for his comments and suggestions on earlier versions of the manuscript. This work was partially supported by the National Science Foundation under grant No. 1339676 (SI2 SSI: Collaborative Research), Sustained Innovations for Linear Algebra Software (SILAS).

REFERENCES

[1] E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, J. J. Dongarra, J. Du Croz, S. Hammarling, A. Greenbaum, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, third edition, 1999.

[2] T. Auckenthaler, V. Blum, H. J. Bungartz, T. Huckle, R. Johanni, L. Krämer, B. Lang, H. Lederer, and P. R. Willems. Parallel Solution of Partial Symmetric Eigenvalue Problems from Electronic Structure Calculations. *Parallel Comput.*, 37:783–794, 2011.

[3] Å. Björck. *Numerical Methods for Least Squares Problems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1996.

[4] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley. *ScaLAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1997.

[5] J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst. *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000. Bai, Zhaojun (Ed.).

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

[7] J. Demmel, O. Marques, C. Voemel, and B. Parlett. Performance and Accuracy of LAPACK's Symmetric Tridiagonal Eigensolvers. *SIAM J. Sci. Comput.*, 30:1508–1526, 2008.

[8] I. Dhillon and B. Parlett. Multiple Representations to Compute Orthogonal Eigenvectors of Symmetric Tridiagonal Matrices. *Linear Algebra Appl.*, 387:1–28, 2004.

[9] I. Dhillon, B. Parlett, and C. Voemel. The Design and Implementation of the MRRR Algorithm. *ACM Trans. Math. Softw.*, 32:533–560, 2006.

[10] G. Golub and W. Kahan. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics Series B Numerical Analysis*, 2(2):205–224, 1965.

[11] B. Grosser and B. Lang. An $O(n^2)$ Algorithm for the Bidiagonal SVD. *Linear Algebra Appl.*, 358(1):45–70, 2003.

[12] B. Grosser and B. Lang. On Symmetric Eigenproblems Induced by the Bidiagonal SVD. *SIAM. J. Matrix Anal. and Appl.*, 26(3):599–620, 2005.

[13] Intel. Intel(R) Fortran Compiler 2018 Update 1 for Linux, 2017.

[14] Intel. Intel(R) Math Kernel Library 2018 Update 1 for Linux, 2017.

[15] LAPACK. LAPACK - Linear Algebra PACKage. https://github.com/Reference-LAPACK, 2018.

[16] S. Li, M. Gu, L. Cheng, X. Chi, and M. Sun. An Accelerated Divide-and-Conquer Algorithm for the Bidiagonal SVD Problem. *SIAM. J. Matrix Anal. and Appl.*, 35:1038–1057, 2014.

[17] O. Marques, J. Demmel, C. Voemel, and B. Parlett. A Testing Infrastructure for Symmetric Tridiagonal Eigensolvers. *ACM Trans. Math. Softw.*, 35:8:1–8:13, 2008.

[18] O. Marques and P. B. Vasconcelos. Computing the Bidiagonal SVD Through an Associated Tridiagonal Eigenproblem. In *High Performance Computing for Computational Science - VECPAR 2016 - 12th International Conference, Porto, Portugal, June 28-30, 2016, Revised Selected Papers*, pages 64–74, Heildeberg, Germany, 2016. Springer.

[19] MatrixMarket. Matrix Market. http://math.nist.gov/MatrixMarket, 2018.

[20] ScaLAPACK. ScaLAPACK, version 2.0.2. http://www.netlib.org/scalapack, 2012.

[21] TAU. TAU Performance System. https://www.cs.uoregon.edu/research/tau, 2018.

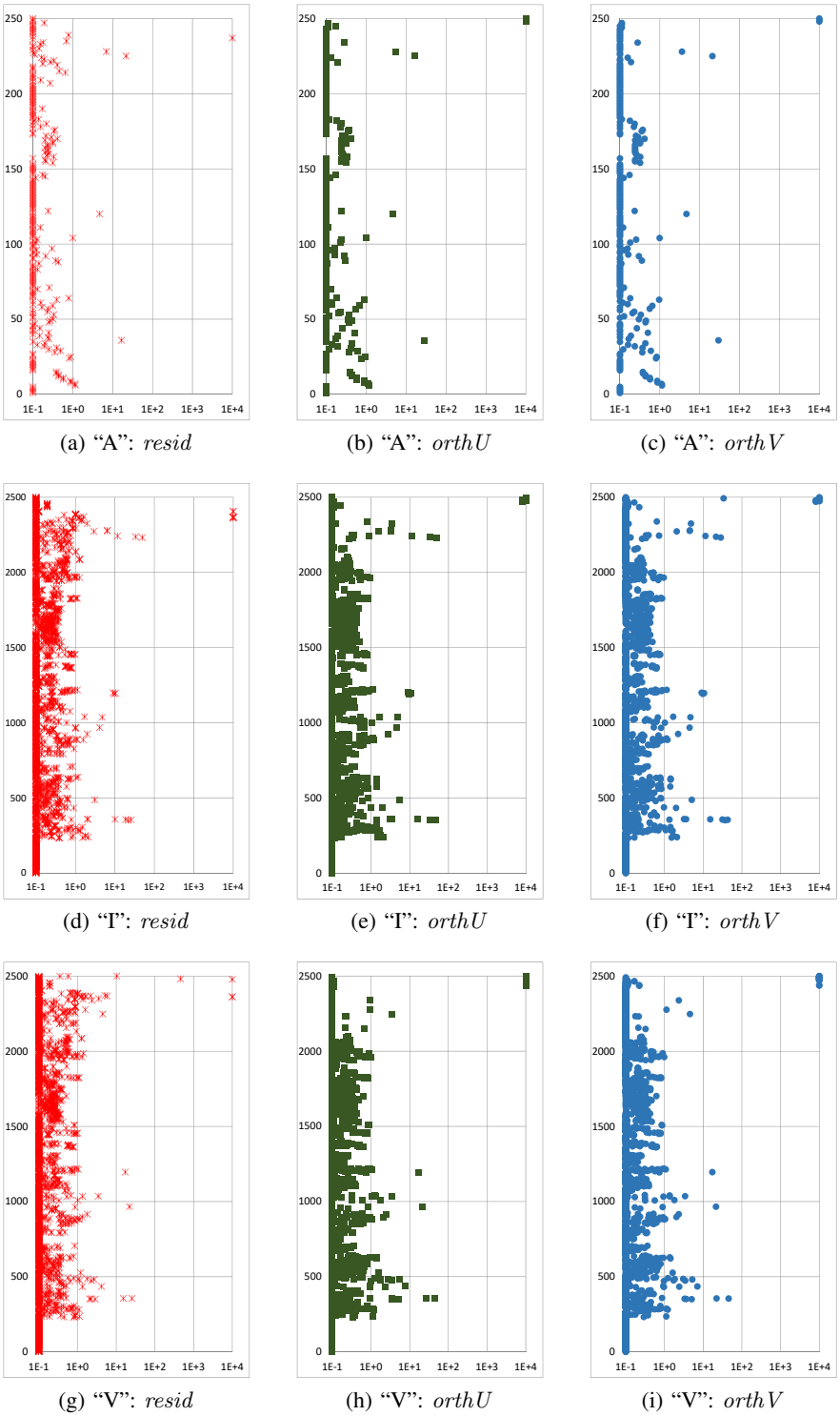
[22] C. Voemel. ScaLAPACK's MRRR algorithm. *ACM Trans. Math. Softw.*, 37:1–35, 2010.

[23] J. Vogel, J. Xia, S. Cauley, and V. Balakrishnan. Superfast Divide-and-Conquer Method and Perturbation Analysis for Structured Eigenvalue Solutions. *SIAM J. Sci. Comput.*, 38(3):A1358–A1382, 2016.

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

[25] P. Willems and B. Lang. A Framework for the MR^3 Algorithm: Theory and Implementation. *SIAM J. Sci. Comput.*, 35:740–766, 2013.

[26] P. Willems, B. Lang, and C. Voemel. Computing the Bidiagonal SVD using Multiple Relatively Robust Representations. *SIAM. J. Matrix Anal. and Appl.*, 28:907–926, 2006.

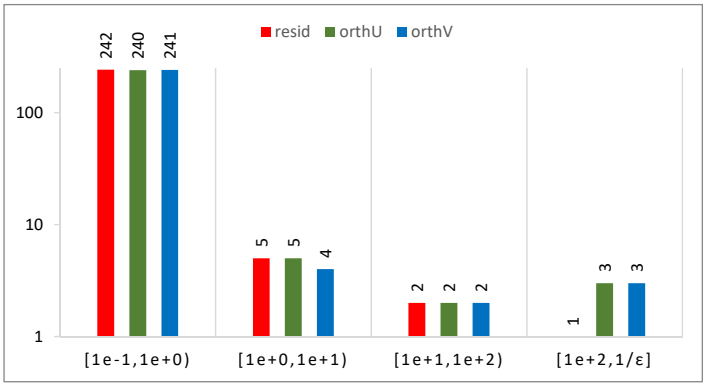


(a) "A": $resid$ (b) "A": $orthU$ (c) "A": $orthV$

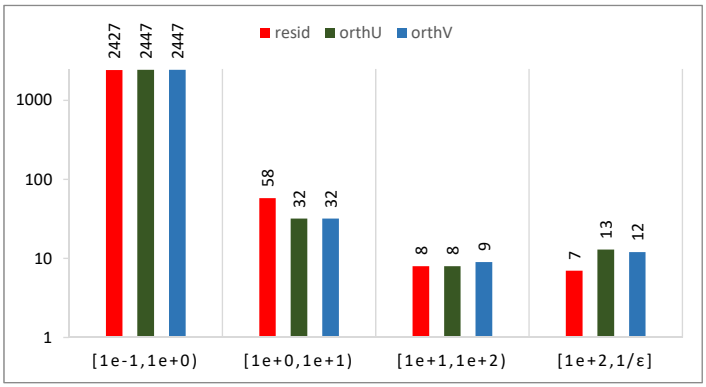
(d) "I": $resid$ (e) "I": $orthU$ (f) "I": $orthV$

(g) "V": $resid$ (h) "V": $orthU$ (i) "V": $orthV$

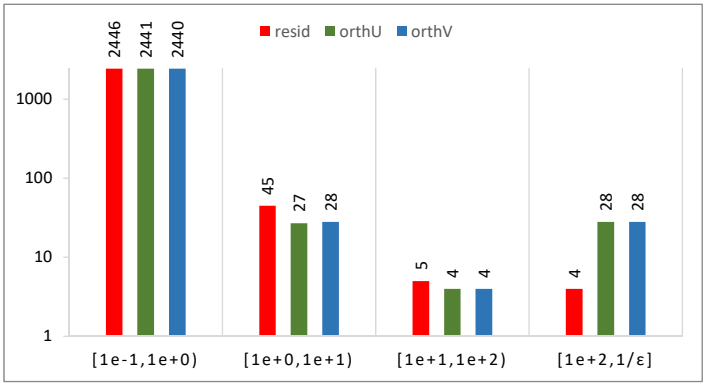
Fig. 2: $resid, orthU, orthV$ (x-axis, log scale) for RANGE="A", "I" and "V", double precision. (2a)-(2c) 250 matrices (y-axis), increasing condition numbers; (2d)-(2f) $n_I = 10$ for each matrix of RANGE="A"; (2g)-(2i) $n_V = 10$ for each matrix of RANGE="A".



(a) "A"

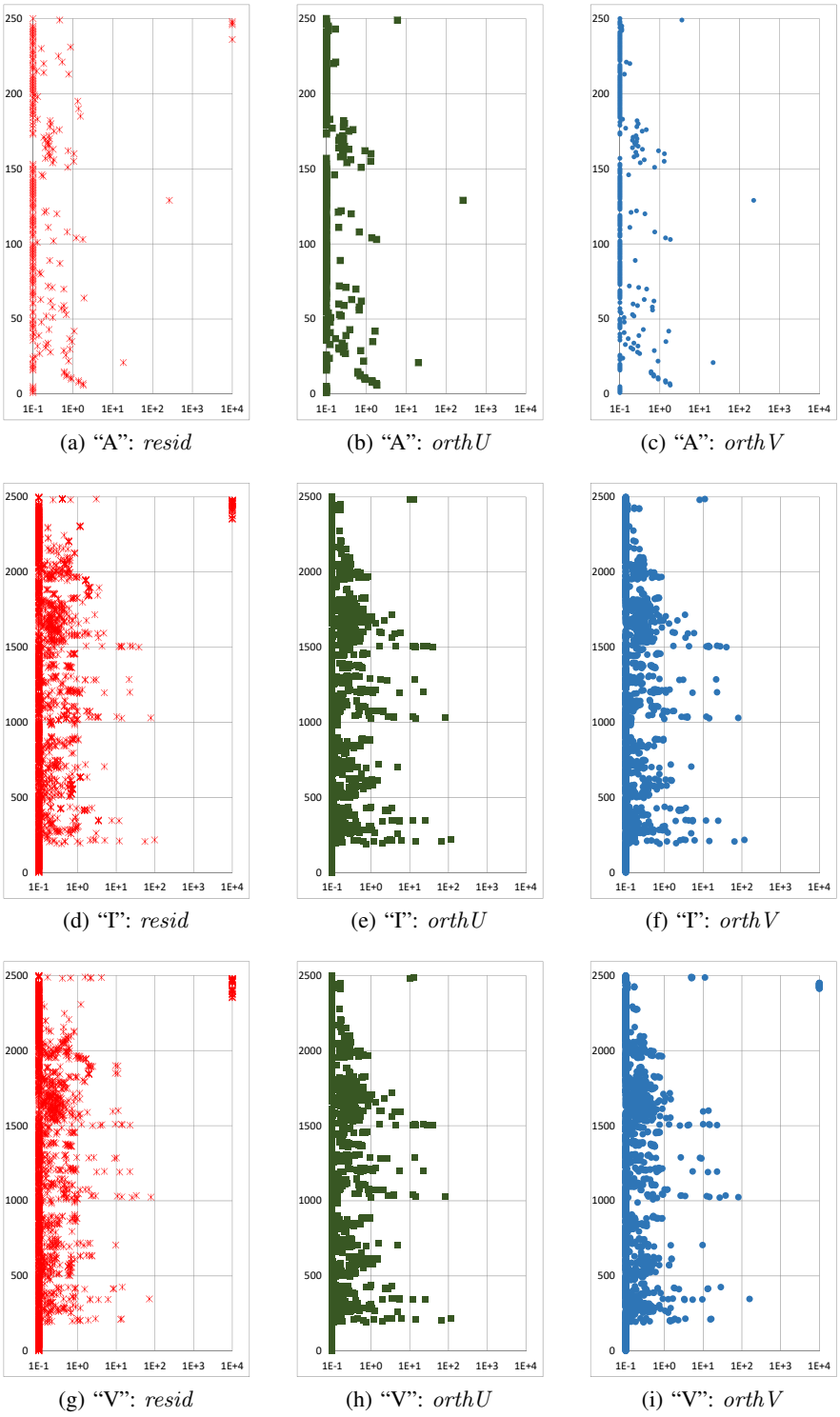


(b) "I"



(c) "V"

Fig. 3: Number of occurrences of $resid, orthU, orthV$ (y-axis, log scale) for RANGE="A","I" and"V", double precision, in the intervals $[10^{-1}, 10^{0})$, $[10^{0}, 10^{1})$, $[10^{1}, 10^{2})$ and $[10^{2}, 1/\varepsilon]$, $\varepsilon \approx 1.11 \times 10^{-16}$. Note that RANGE="I" and"V" have 10 times more data points than RANGE="A", i.e. for each case in (a) there are 10 intervals in (b) and (c).

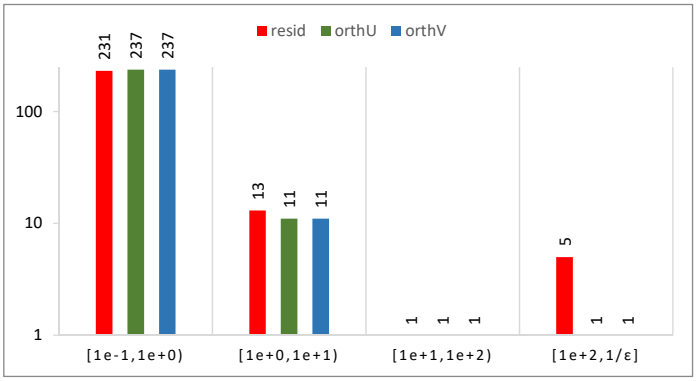


(a) "A": $resid$ (b) "A": $orthU$ (c) "A": $orthV$

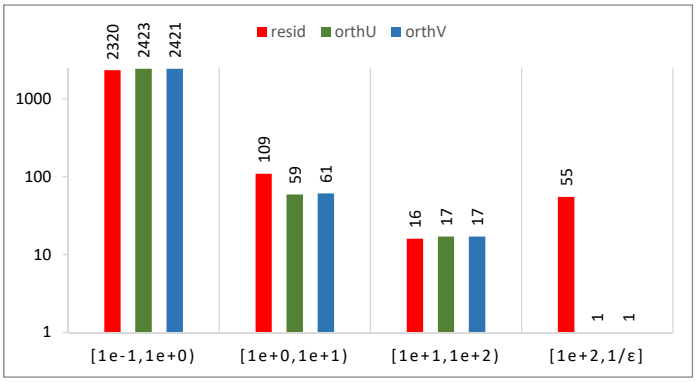
(d) "I": $resid$ (e) "I": $orthU$ (f) "I": $orthV$

(g) "V": $resid$ (h) "V": $orthU$ (i) "V": $orthV$

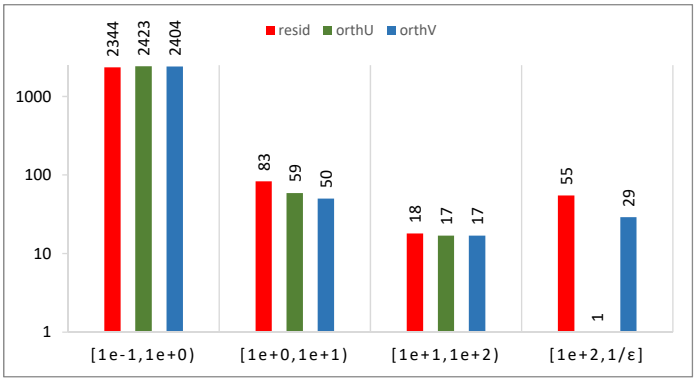
Fig. 4: $resid, orthU, orthV$ (x-axis, log scale) for RANGE="A","I" and"V", single precision. (4a)-(4c) 250 matrices (y-axis), increasing condition numbers; (4d)-(4f) $n_I = 10$ for each matrix of RANGE="A"; (4g)-(4i) $n_V = 10$ for each matrix of RANGE="A".



(a) "A"



(b) "I"



(c) "V"

Fig. 5: Number of occurrences of $resid$, $orthU$, $orthV$ (y-axis, log scale) for RANGE="A", "I" and "V", single precision, in the intervals $[10^{-1}, 10^{0})$, $[10^{0}, 10^{1})$, $[10^{1}, 10^{2})$ and $[10^{2}, 1/\varepsilon]$, $\varepsilon \approx 5.96 \times 10^{-8}$. Note that RANGE="I" and"V" have 10 times more data points than RANGE="A", i.e. for each case in (a) there are 10 intervals in (b) and (c).

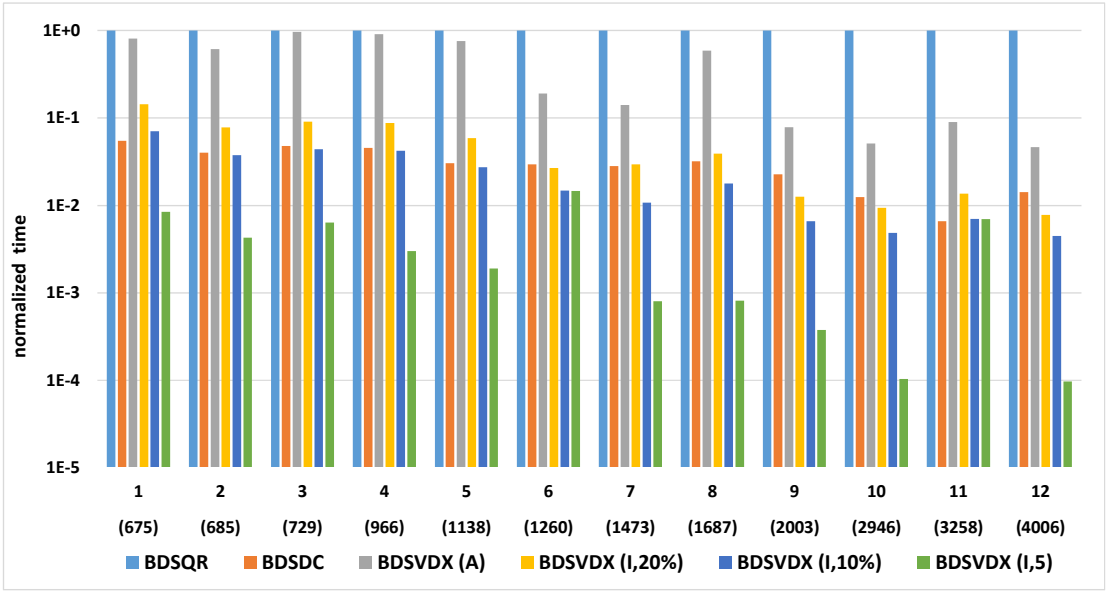


Fig. 6: Timings for `BDSQR`, `BDSDC` and `BDSVDX` on 12 bidiagonal matrices with dimensions ranging from 675 to 4006 (x-axis, dimensions in parentheses), in double precision, average time over 10 runs per matrix. `BDSVDX`: all singular values/vectors, the largest 20%, the largest 10% and the largest 5 singular values/vectors. For each matrix, the timings (y-axis) are normalized with respect to the largest time and are plotted in log scale.

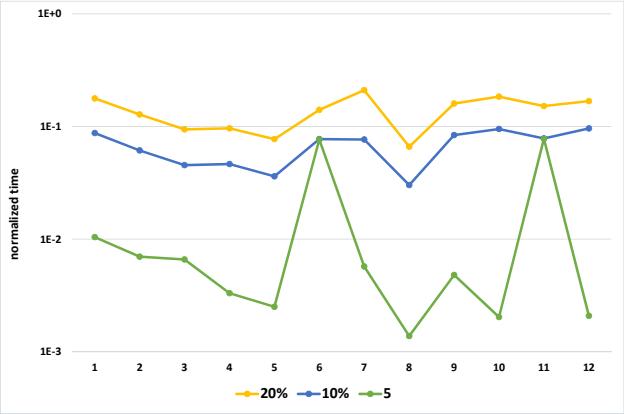
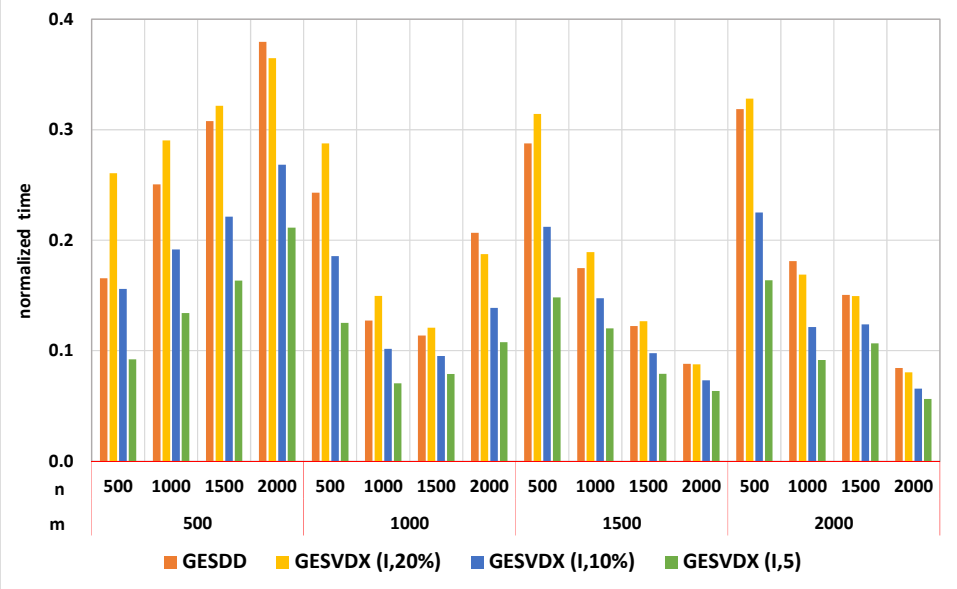
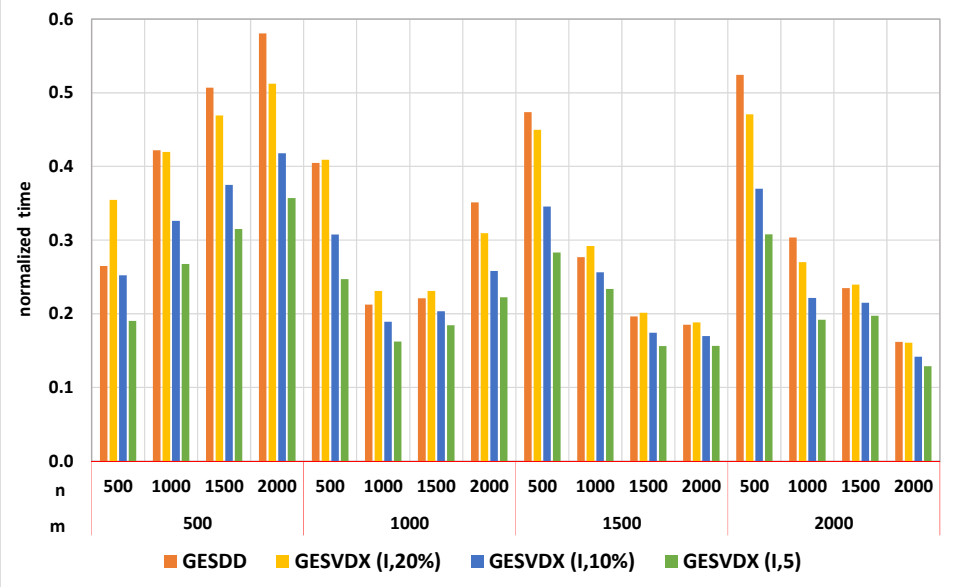


Fig. 7: Performance of `BDSVDX` for the matrices in Fig. 6. The data points (y-axis, log scale) correspond to the computing times for the three subset scenarions (20%, 10%, largest 5) normalized with respect to t_A, where t_A is the time required for the computation of all singular values/vectors.

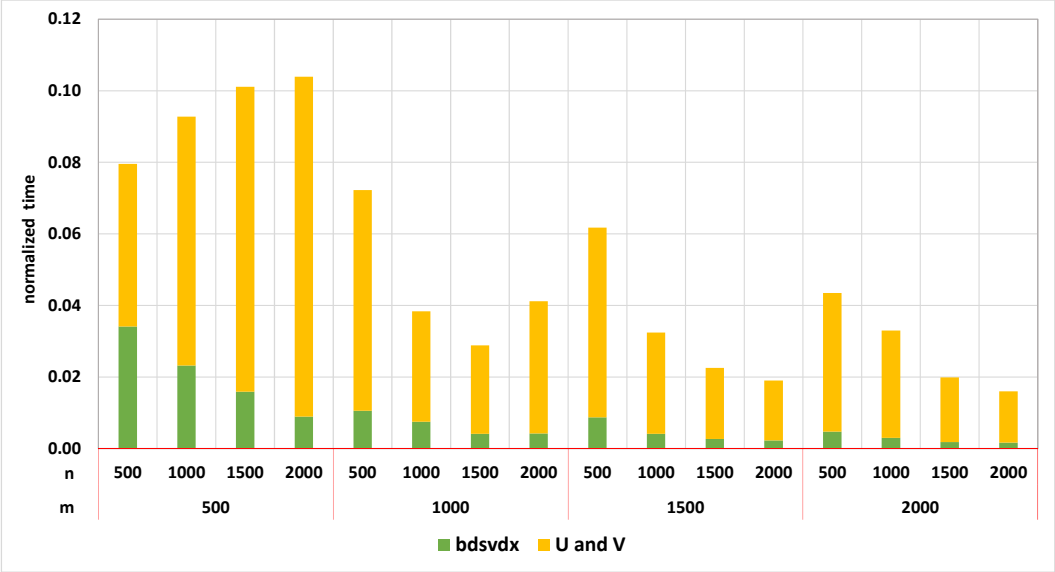


(a) double precision

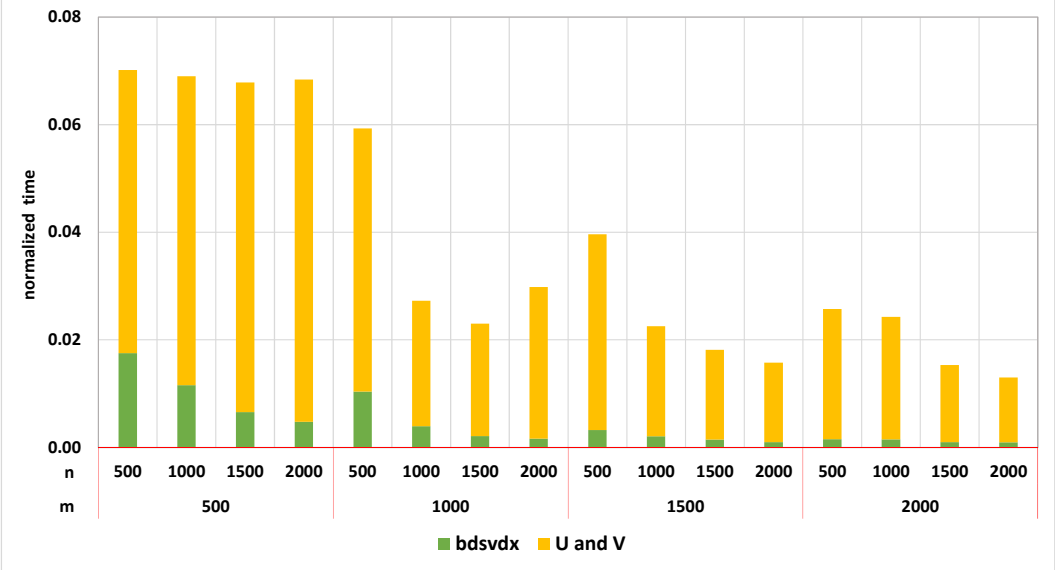


(b) double precision complex

Fig. 8: Timings for GESVD, GESDD and GESVDX on 16 random dense $m \times n$ matrices with $m, n = 500, 1000, 1500, 2000$. GESVDX: the largest 20%, the largest 10% and the largest 5 singular values/vectors. For each matrix, the timings are normalized with respect to the time taken by GESVD. Average time over 10 runs per matrix.



(a) double precision



(b) double precision complex

Fig. 9: Time breakdown for GESVDX, for the matrices used in Figs. 8a and 8b; 5 largest singular values/vectors. *bdsvdx*: computation of the singular values and vectors of B; *U and V*: back transformation of the singular vectors of B to those of the input matrix. The bars are normalized with respect to the time taken by reduction of the input matrix to bidiagonal form B. Average time over 10 runs per matrix.

APPENDIX A
A CASE OF FAILURE IN STEXR

We show here a case of misbehavior of STEXR (double precision) introduced in [25], by using a tridiagonal matrix T generated with the prescribed eigenvalue distribution $\lambda_i = c^{-\frac{(i-1)}{(n-1)}}, c = 1/\sqrt{\varepsilon}, i = 1, 2, \ldots n, n = 10$ (i.e. $\lambda_1 \approx 1.49 \times 10^{-8}, \ldots \lambda_n = 1.00$). We call the LAPACK subroutine LATMS to generate a random symmetric matrix with those eigenvalues followed by SYTRD to tridiagonalize the matrix. Table A.1 lists the entries of T used in the test. Here, we have used the GNU Fortran compiler because the distribution in [25] does not provide a configuration for the Intel Fortran compiler. Although not shown, the eigenvalues of T computed with the eigensolvers listed in Table I and also STEXR are in very good agreement. Specifically, $\|T - Z_{\mathtt{XR}}\Lambda_{\mathtt{XR}}Z_{\mathtt{XR}}^T\|/(\|T\|n\varepsilon) \approx 0.6$, where $\Lambda_{\mathtt{XR}}$ contains the eigenvalues returned by STEXR on its main diagonal, and $Z_{\mathtt{XR}}$ is the matrix of eigenvectors returned by STEXR. However, $\|I - Z_{\mathtt{XR}}^T Z_{\mathtt{XR}}\|/(n\varepsilon) \approx 3.95 \times 10^4$; see Fig. A.1. In contrast, $\|I - Z_{\mathtt{VX}}^T Z_{\mathtt{VX}}\|/(n\varepsilon) \approx 0.9$ and $\|I - Z_{\mathtt{MR}}^T Z_{\mathtt{MR}}\|/(n\varepsilon) \approx 0.1$, where $Z_{\mathtt{VX}}$ and $Z_{\mathtt{MR}}$ are the vectors returned by STEVX and STEMR, respectively.

We have identified other matrices for which STEXR failed to produce orthogonal eigenvectors, for example the Wilkinson matrix W21+ mentioned earlier. Our exhaustive tests revealed that STEMR may also fail for matrices with very close eigenvalues (e.g. matrices formed by gluing Wilkinson-type matrices). To the best of our knowledge, STEXR is no longer maintained, justifying our choice of STEVX for the first implementation of BDSVDX.

TABLE A.1: Entries of T, $\lambda_i = c^{-\frac{(i-1)}{(n-1)}}, c = \frac{1}{\sqrt{\varepsilon}}, i = 1, 2, \ldots n, n = 10$.

i	$t_{i,i}$	$t_{i,i+1} = t_{i+1,i}$
1	1.893161597943482E-01	3.880873104122968E-01
2	8.128005558065539E-01	-3.516122075663728E-02
3	1.258328488738520E-01	3.077875339462724E-02
4	2.448430650126851E-02	-4.746410482563373E-03
5	3.268662131212184E-03	-6.983851144411338E-05
6	2.759036513821439E-04	-1.142712831766173E-04
7	9.443722972151846E-05	6.941905362025514E-06
8	6.149112437832172E-06	-7.426637317219540E-07
9	2.117627370984594E-07	1.892470326809461E-08
10	1.071603546505181E-07	-

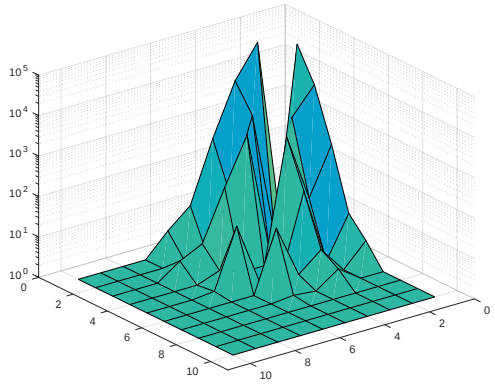


Fig. A.1: Surface plot of $|I - Z_{\mathtt{XR}}^T Z_{\mathtt{XR}}|/(n\varepsilon)$ in log scale, where $Z_{\mathtt{XR}}$ contains the eigenvectors returned by STEXR for the tridiagonal matrix given in Table A.1. The first four columns of $Z_{\mathtt{XR}}$ are linearly dependent: those columns correspond to $\lambda_1 \approx 1.49 \times 10^{-8}$, $\lambda_2 \approx 1.10 \times 10^{-7}$, $\lambda_3 \approx 8.17 \times 10^{-7}$ and $\lambda_4 \approx 6.06 \times 10^{-6}$.