Bidiagonal SVD Computation via an Associated Tridiagonal Eigenproblem

OSNI MARQUES, Lawrence Berkeley National Laboratory
JAMES DEMMEL, University of California at Berkeley
PAULO B. VASCONCELOS, Centro de Matemática & Faculdade Economia, University of Porto

The Singular Value Decomposition (SVD) is widely used in numerical analysis and scientific computing applications, including dimensionality reduction, data compression and clustering, and computation of pseudo-inverses. In many cases, a crucial part of the SVD of a general matrix is to find the SVD of an associated bidiagonal matrix. This article discusses an algorithm to compute the SVD of a bidiagonal matrix through the eigenpairs of an associated symmetric tridiagonal matrix. The algorithm enables the computation of only a subset of singular values and corresponding vectors, with potential performance gains. The article focuses on a sequential version of the algorithm, and discusses special cases and implementation details. The implementation, called BDSVDX, has been included in the LAPACK library. We use a large set of bidiagonal matrices to assess the accuracy of the implementation, both in single and double precision, as well as to identify potential shortcomings. The results show that BDSVDX 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 comparisons of an implementation that uses BDSVDX as a building block for the computation of the SVD of general matrices.

CCS Concepts: • **Mathematics of computing** → **Solvers**; **Computations on matrices**;

Additional Key Words and Phrases: Singular value decomposition, eigenvalues, eigenvectors, LAPACK, numerical software, design, implementation

ACM Reference format:
Osni Marques, James Demmel, and Paulo B. Vasconcelos. 2020. Bidiagonal SVD Computation via an Associated Tridiagonal Eigenproblem. *ACM Trans. Math. Softw.* 46, 2, Article 14 (May 2020), 25 pages.
https://doi.org/10.1145/3361746

1 INTRODUCTION

The computation of the full set of singular values and vectors or a subset of them of a large, general matrix $A \in \mathbb{R}^{m \times n}$, is a demanding task, especially when high accuracy is required. We denote the decomposition $A = USV^T$, 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

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). The third author acknowledges the support from Fundação para a Ciência e a Tecnologia (FCT) through the sabbatical fellowship SFRH/BSAB/142986/2018.

Authors' addresses: O. Marques, Computational Research Division, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, MS 50A3111, Berkeley, California 94720 USA; email: oamarques@lbl.gov; J. Demmel, Computer Science Division, University of California at Berkeley, Berkeley CA 94720 USA; email: demmel@berkeley.edu; P. B. Vasconcelos, Faculdade de Economia da Universidade do Porto, Rua Dr. Roberto Frias 4200-464 Porto, Portugal; email: pjv@fep.up.pt.

0098-3500/2020/05-ART14 $15.00
https://doi.org/10.1145/3361746

values $s_1 \geq s_2 \geq \ldots s_n \geq 0$ on the main diagonal of $S \in \mathbb{R}^{m \times n}$. This article focuses on the real case, but the discussion can be easily extended to the complex case.

Two main approaches exist for obtaining the singular value decomposition (SVD) via an eigenvalue problem formulation—through the eigenpairs of the normal equation matrix or through those of an augmented matrix.

The first approach obtains the SVD through the eigenpairs (λ, x) of matrices $A^T A \in \mathbb{R}^{n \times n}$ and $AA^T \in \mathbb{R}^{m \times m}$, as long as the singular values are distinct. The nonzero eigenvalues of $A^T A$ and AA^T are thus the squares of the nonzero singular values of A. However, if A is square and orthogonal, then both $A^T A$ and AA^T are 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. This approach works well for the separation of large singular values, but brings small singular values into a cluster. Once either the left or the right singular vectors are computed, the others require a multiplication by A or A^T, which can degrade accuracy. A potential difficulty for some algorithms (e.g., the one presented in this article) is large clusters of singular values, as this may have an impact on the orthogonality of the computed singular vectors.

The second approach consists in obtaining the SVD through the eigenpairs of the Jordan-Wielandt matrix[1]

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

with eigenvalues directly related to the singular values, while the left and right singular vectors can be extracted from the corresponding eigenvectors. One advantage of using C is that the singular values are not squared. One disadvantage is that C is larger in size and the smallest singular values are mapped into the interior of the spectrum. In this article, we focus on the case $m = n$. Then, one can write [Golub and Kahan 1965]

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

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

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

which shows 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.

An alternative to the two aforementioned approaches is given by the polar decomposition: the SVD of $A = (UV)SV^T$ is obtained through $A = UH$, where U is orthogonal and H is symmetric positive-semidefinite, with the symmetric eigendecomposition $H = VSV^T$ [Higham and Papadimitriou 1993]. Building upon the work of Nakatsukasa et al. [2010] on the QR-based dynamically weighted Halley algorithm, Nakatsukasa and Higham [2013] proposed improved algorithms based on the polar decomposition. However, we have not explored these developments since our goal was to use functionalities available in LAPACK [Anderson et al. 1999], as discussed below.

In practical calculations, the SVD of a general matrix A involves the reduction of A to a bidiagonal matrix 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.

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 Equation (1) to obtain $C = P\, T_{GK}\, P^T$, where T_{GK}

[1]In Fan and Hoffman [1955], H. Wielandt was credited for this formulation.

Table 1. Current (Bidiagonal, BD) SVD and (Tridiagonal, ST) Eigensolvers Implemented in LAPACK

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
STEDC	all λ's and (opt.) x	divide-and-conquer
STEVX	selected λ's and (opt.) x	bisection and inverse iteration
STEMR	selected λ's and (opt.) x	MRRR

The acronyms follow the naming scheme described in page 14, [Anderson et al. 1999]. The last two letters in the acronyms identify the algorithm.

is the Golub-Kahan symmetric tridiagonal matrix,

$$T_{GK} = \text{tridiag}\begin{pmatrix} & a_1 & & b_1 & & a_2 & & b_2 & & \ldots & & b_{n-1} & & a_n & \\ 0 & & 0 & & 0 & & 0 & & \ldots & & & & 0 & & 0 \\ & a_1 & & b_1 & & a_2 & & b_2 & & \ldots & & b_{n-1} & & a_n & \end{pmatrix}, \tag{4}$$

with the perfect shuffle $P = [e_{n+1}, e_1, e_{n+2}, e_2, e_{n+3}, \ldots e_{2n}]$, with $e_i, i = 1, 2, \ldots 2n$, 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 Equations (2)–(3), we obtain

$$z = P\begin{bmatrix} u \\ \pm v \end{bmatrix}\frac{1}{\sqrt{2}},$$

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}.

The set of LAPACK [Anderson et al. 1999] subroutines intended for the computation of the SVD of bidiagonal matrices, and eigenvalues and eigenvectors of tridiagonal matrices are listed in Table 1. We refer the reader to Demmel et al. [2008] for a discussion on the tradeoffs (performance, accuracy) of the symmetric tridiagonal subroutines. We consider how the last three of those subroutines could be applied to Equation (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).

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 [Auckenthaler et al. 2011]). On the other hand, 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 deliver orthogonal eigenvectors when the eigenvalues are too closely clustered (although STEIN implements a modified Gram-Schmidt reorthogonalization strategy).

STEMR uses a sophisticated algorithm called Multiple Relatively Robust Representations (MRRR) [Dhillon 1997; Dhillon and Parlett 2004; Dhillon et al. 2006] that produces nearly guaranteed orthogonal eigenvectors. 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 Grosser and Lang [2003, 2005] and Willems et al. [2006]. However, an implementation discussed in Willems et al. [2006] 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 by MRRR in that context [Willems 2010, p. 128,149]. Subsequently, an improved version of the MRRR algorithm targeting T_{GK} for SVD computations was proposed in Willems [2010] and Willems and Lang [2013]; however, our experiments with an implementation given in Willems and Lang [2013] and named STEXR exposed deficiencies (inaccuracy) for relatively simple matrices. We show one such case in Appendix B.

Therefore, we have decided to adopt STEVX for computing eigenvalues and eigenvectors of Equation (4), even though it also has known failure modes. In particular, in extreme situations of tightly clustered eigenvalues, the bisection may potentially not converge to the desired accuracy, or not all eigenvalues within an interval defined by indices 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 5) have never detected such anomalies, we plan to replace STEVX with a more robust implementation of MRRR for Equation (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 Equation (4): we also need the extracted u's and v's to be accurate. We note that Björck [1996, Chapter 2] discusses the computation of singular values by spectrum slicing, but not the computation of the corresponding singular vectors.

The main contribution of this article 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}. 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. The implementation is called BDSVDX, which was first introduced in Linear Algebra PACKage (LAPACK) 3.6.0,[3] with preliminary results reported in Marques and Vasconcelos [2016]. Here, along with a detailed discussion of the algorithm, we use a larger set of test cases and present the results more clearly. We elaborate on the refinement of vectors and give an in-depth explanation of the various splitting possibilities. Also, we discuss the outliers, in single and double precision, and examine the performance of the proposed algorithm.

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 1 and can only compute all singular values (and optionally singular vectors). The computation of a full SVD can be significantly more expensive than a partial one. 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 article is organized as follows. First, we discuss how singular values are mapped into the eigenvalue spectrum. Second, we discuss special cases, the criterion for splitting a bidiagonal matrix, and other implementation details. Third, 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 precision) 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. A pseudocode showing the main phases of BDSVDX is given in Appendix A.

[2]At the time of this writing, [LAPACK 2018] 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]https://github.com/Reference-LAPACK
[4]A modified and potentially faster version of BDSDC exploiting low-rank properties of broken arrow matrices has been proposed in [Li et al. 2014]; see also [Vogel et al. 2016].

2 MAPPING SINGULAR VALUES INTO EIGENVALUES

Similar to BDSQR and BDSDC, BDSVDX allows the computation of singular values only or singular values and the corresponding singular vectors. 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}. 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. Therefore, BDSVDX inherits the shortcomings of STEVX mentioned previously (i.e., bisection may fail to converge, not all eigenvalues in a given interval may be found, or inverse iteration may fail to converge). For the mapping of indices into values, bisection could be potentially replaced by dqds [Parlett 1995], possibly leading to gains in performance.

If the singular vectors of a matrix B of dimension n 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.

It is important to note that STEVX returns eigenvalues (and corresponding vectors) in ascending order, so we target the negative part of the eigenvalue spectrum (i.e., $-S$) in Equation (2). As a result, the absolute values of the returned eigenvalues give us the singular values in the desired order, $s_1 \geq s_2 \geq \ldots s_n \geq 0$. Only the signs of the entries in the eigenvectors that are reloaded to V need to be altered.

3 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

$$T_{GK}^{(1)} = \text{tridiag}\begin{pmatrix} & a_1 & & b_1 & & a_2 & & b_2 & \dots & a_{j-1} & & b_{j-1} & \\ 0 & & 0 & & 0 & & 0 & & \dots & & 0 & & 0 \\ & a_1 & & b_1 & & a_2 & & b_2 & \dots & a_{j-1} & & b_{j-1} & \end{pmatrix},$$

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

$$T_{GK}^{(2)} = \text{tridiag}\begin{pmatrix} & a_{j+1} & & b_{j+1} & & a_{j+2} & \dots & b_{n-1} & & a_n & \\ 0 & & 0 & & 0 & & \dots & & 0 & & 0 \\ & a_{j+1} & & b_{j+1} & & a_{j+2} & \dots & b_{n-1} & & a_n & \end{pmatrix}$$

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 $\lambda_1 \leq \lambda_2 \leq \dots \lambda_{j-1} \leq \lambda_j = 0 \leq \lambda_{j+1} \cdots \leq \lambda_{2j-1}$, then $\sigma_1 = |\lambda_{min}| = |\lambda_1|$. Finally, we note that the case $j = 1$ fits the general pattern of Equation (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 $\mathcal{Z}$.

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 Equation (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.

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 Equation (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 $Z_{8\times 5}$.

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 Equation (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 Demmel and Kahan [1990]. 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.

4 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 Willems et al. [2006], 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.

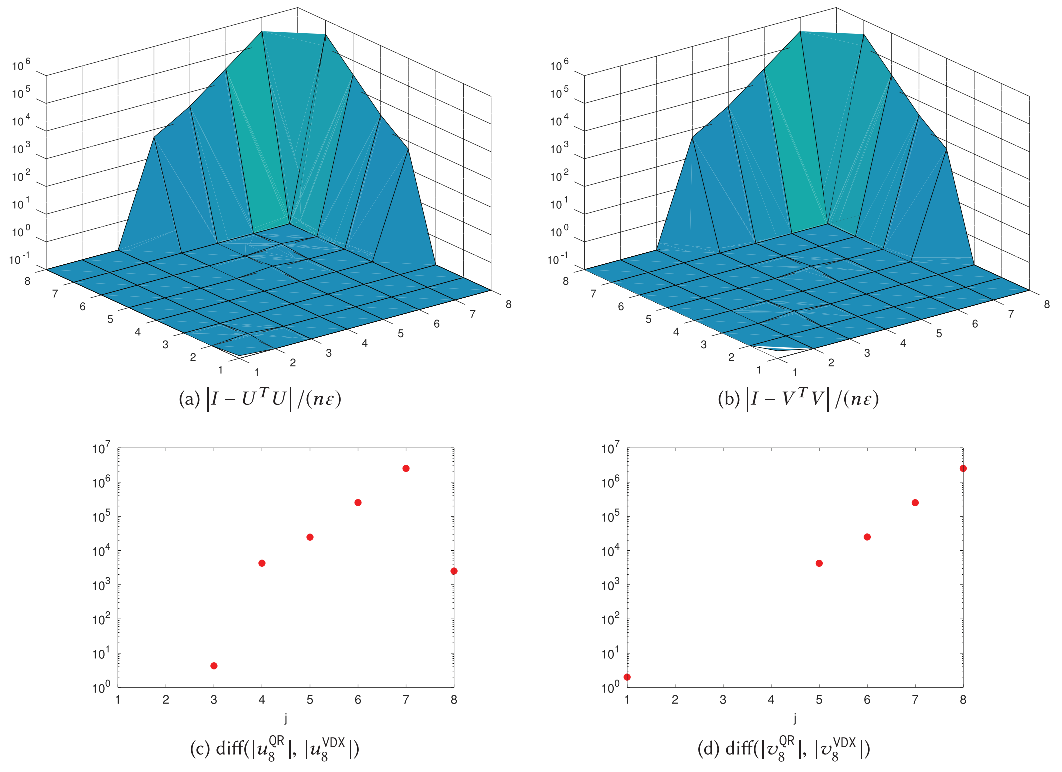
As a safeguard, if necessary, we apply a Gram-Schmidt reorthogonalization to $\hat{u}_i$ and $\hat{v}_i$ against the previous vectors. The test we have implemented to trigger reorthogonalization is based on $|\|\hat{u}\| - \frac{1}{\sqrt{2}}| \geq tol$, similarly for $\hat{v}$, where $tol = \sqrt{\varepsilon}$, 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 $B_{8\times 8}$ ($n = 8$) with diagonal and superdiagonal entries defined as[5] $a_i = 10^{-(2i-1)}$, $i = 1, 2, \ldots n$, $b_i = 10^{-(2i-2)}$, $i = 1, 2, \ldots n-1$, whose two-norm is ≈ 1.005, two-norm 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, Figure 1(a)–(b) 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 (which returns U and V orthogonal to machine precision), we observe that the largest differences in those entries are in the 7th entry of u_8 and in the 8th entry of v_8—the output from BDSVDX agrees with the output from BDSQR to 10 and 9 digits, respectively, in those particular entries, which are $O(10^{-3})$ and $O(10^{-6})$. Figure 1(c)–(d) shows the differences in the entries of u_8 and v_8 computed by BDSQR and BDSVDX.

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 [Anderson et al. 1999, 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.

5 NUMERICAL EXPERIMENTS

In this section, we discuss our experiments with BDSVDX using a large set of bidiagonal matrices. We assess the accuracy of BDSVDX , in single and double precision, and compare its performance to that of BDSQR and BDSDC. We also compare the performance of GESVDX against GESVD and GESDD. Insights on the refinement of vectors are also provided.

[5]This matrix is not included in the tests in Section 5.



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

(c) diff($|u_8^{\text{QR}}|$, $|u_8^{\text{VDX}}|$) (d) diff($|v_8^{\text{QR}}|$, $|v_8^{\text{VDX}}|$)

Fig. 1. Top: orthogonality level (surface mesh, log scale) of U and V for $B_{8\times8}$ computed by BDSVDX. Bottom: difference in the j-th entry of u_8 and v_8 computed by BDSQR and BDSVDX (the difference is scaled by ε, plotted in log scale for values larger than 1).

Computational environment

We tested 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 precision, using the Intel and GNU Fortran compilers. The results shown here were obtained with the Intel compiler [Intel 2017a], with flags `-O2 -fp-model strict`, using Intel's MKL (BLAS) [Intel 2017b], and LAPACK 3.7.0.

Test matrices

Most of the bidiagonal matrices in our testbed were derived from symmetric tridiagonal matrices described in Marques et al. [2008] (also used in Demmel et al. [2008]). In this case, we factored $T - \nu I = LL^T$ (Cholesky) for a proper value of ν (obtained from the Gerschgorin bounds of T), then set $B = L^T$. The testbed includes a variety of tridiagonal matrices, for example:

- Matrices obtained by running the Lanczos algorithm ([Parlett 1998, Chapter 13], [Demmel 1997, Chapter 7], [Demmel et al. 2000, Chapter 4]) without reorthogonalization. This procedure can lead to matrices with very close eigenvalues (see one case below).
- Matrices used in Dhillon [1997] and Dhillon et al. [1997] (the matrices in the latter reference inspired the development of the MRRR algorithm).
- Matrices used in Matsekh [2005] to test Godunov's two-sided Sturm sequence.

- Glued matrices defined as

$$\text{tridiag}\begin{pmatrix} & \gamma_1 & & \gamma_2 & & \gamma_{k-1} & \\ M_1 & & M_2 & & \dots & & M_k \\ & \gamma_1 & & \gamma_2 & & \gamma_{k-1} & \end{pmatrix}, \quad M_i = \begin{pmatrix} & e_1^{(i)} & & e_2^{(i)} & & e_{n_i-1}^{(i)} & \\ d_1^{(i)} & & d_2^{(i)} & & \dots & & d_{n_i}^{(i)} \\ & e_1^{(i)} & & e_2^{(i)} & & e_{n_i-1}^{(i)} & \end{pmatrix}$$

  for different types of matrices M_i, dimensions n_i, and glue factors γ_i, $i = 1, 2, \dots k$. This allows us to combine and experiment with a variety of eigenvalue distributions; see Parlett and Vömel [2009] (and one case below).
- Classical orthogonal polynomials, see Abramowitz [1974, Chapter 22] and Olver et al. [2010, Chapter 2018].
- Matrices that exposed bugs in LAPACK eigensolvers and provided by Intel, Mathworks, and developers of the R package.

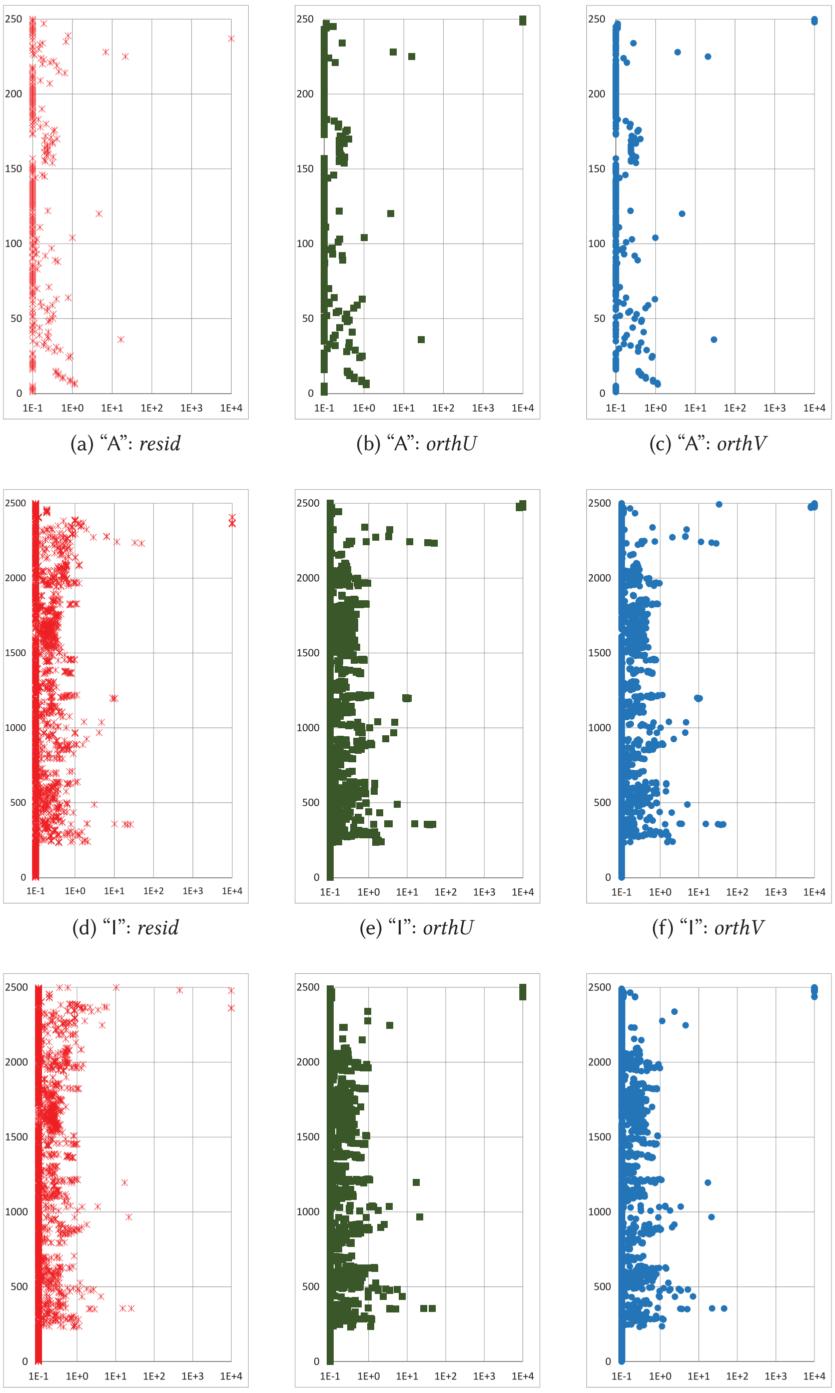
The testbed also includes random bidiagonal matrices, with entries obtained from a standard normal distribution and a uniform distribution, generated by LAPACK's functions LARND and LARNV. A notoriously difficult matrix (borrowed from the LAPACK SVD tester, CHKBD) has entries defined as e^x, where x is chosen uniformly on $[2\log\varepsilon, -2\log\varepsilon]$, therefore ranging from ε^{-2} to ε^2 (see one case below). Our testing infrastructure as well as all matrices used in our experiments are available upon request. The matrices will be included in a future release of LAPACK.

Accuracy tests

To test the accuracy of BDSVDX, we compute $resid = \|U^T BV - S\|/(\|B\|n\varepsilon)$, $orthU = \|I - U^T U\|/(n\varepsilon)$, and $orthV = \|I - V^T V\|/(n\varepsilon)$, 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 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, 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. 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.

Accuracy in double precision. Figure 2 shows the accuracy of BDSVDX in double precision. Figure 2(a)–(c) correspond to the computation of all singular values and vectors (RANGE="A"), for 250 bidiagonal matrices with dimensions ranging from 9 to 4,006. Figure 2(d)–(i) 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, Figure 3 shows a histogram of the symbols displayed in Figure 2.

For RANGE="A", as can be seen in Figure 2(a)–(c) and Figure 3(a), 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 Figure 2(a):

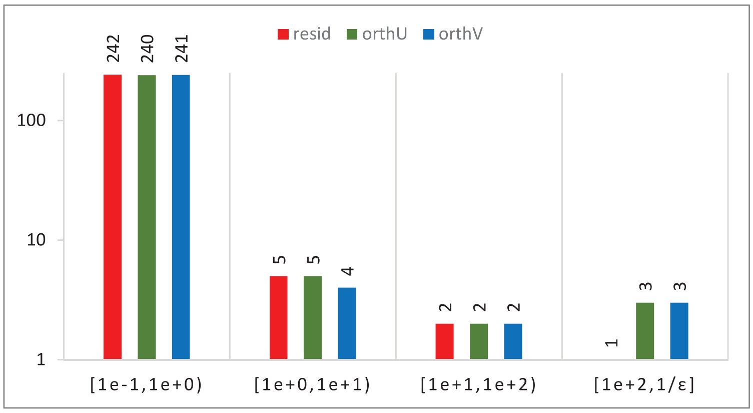


(a) "A": *resid* (b) "A": *orthU* (c) "A": *orthV*

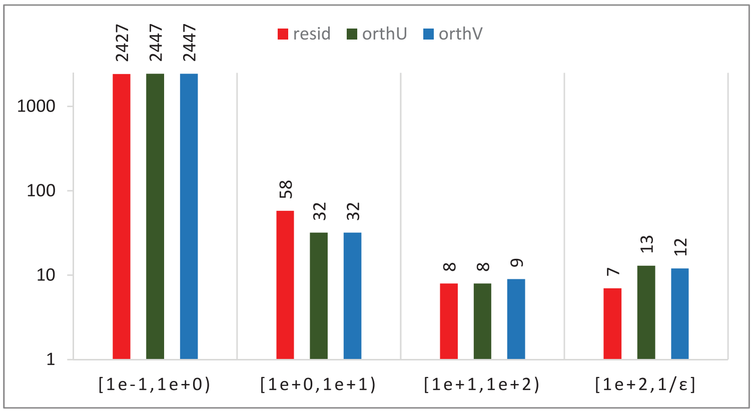
(d) "I": *resid* (e) "I": *orthU* (f) "I": *orthV*

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

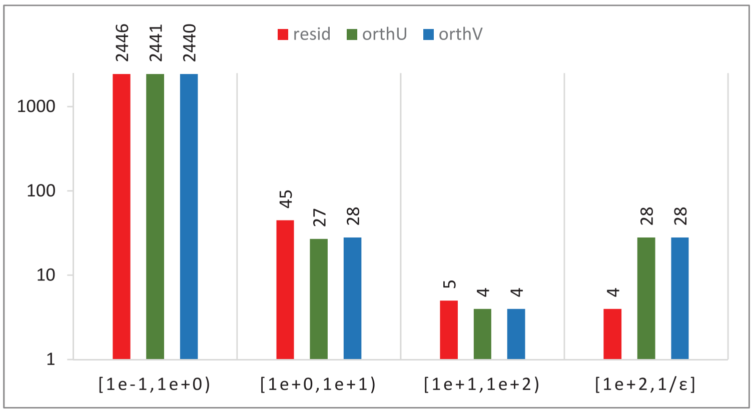
Fig. 2. *resid*, *orthU*, *orthV* (x-axis, log scale) for RANGE="A", "I", and "V", double precision. (2(a)–(c)) 250 matrices (y-axis), increasing condition numbers; (2(d)–(f)) n_I = 10 for each matrix of RANGE="A"; (2(g)–(i)) n_V = 10 for each matrix of RANGE="A".



(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) Matrix 36 is a bidiagonal matrix of dimension 1,260 obtained from a tridiagonal matrix computed by running the Lanczos algorithm without reorthogonalization (as discussed above) $3 \times n_0$ steps, where n_0 is the dimension of the original (sparse) matrix (BCSSTK07, from the BCSSTRUC1 set in MatrixMarket [2018]). The largest 138 eigenvalues of the tridiagonal matrix are of order $O(10^{-3})$ and agree to 12 digits.

(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, are of order $O(10^{-2})$ and agree to 13 digits.

(c) Matrices 248–250 (outliers) are defined as e^x, as discussed above. 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 Figure 2(a)–(c), are propagated to Figure 2(d)–(i). However, Figure 2(g)–(i) contains additional results above 10.0. In these figures, case 966 (matrix 97 in Figure 2(a)–(c)) is a bidiagonal matrix of dimension 2,100 obtained from a tridiagonal matrix formed by gluing 100 copies of the Wilkinson matrix W21+ (its diagonal entries are $10, 9, 8, \ldots, 0, \ldots 8, 9, 10$ and its off-diagonal entries are all 1) with glue factor $\gamma = 10^{-11}$, 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 1,657 singular values and vectors.

Accuracy in single precision

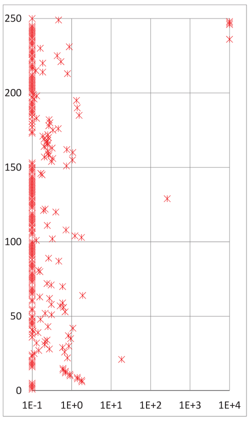
Figure 4 shows the accuracy of BDSVDX in single precision. To assist in the interpretation of the results, Figure 5 shows a histogram of the symbols displayed in Figure 4. Most of the 250 matrices used in Figure 2 are read from files and are also used for the experiments in single precision. The matrices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

Figure 4(a)–c) correspond to the computation of all singular values and vectors (RANGE="A"). Figure 4(d)–(i) show the accuracy of BDSVDX for the same matrices of Figure 4(a), 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.

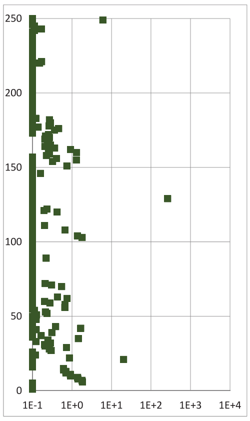
The matrices that lead to results larger than 1,000 in single precision are similar to the ones in double precision. Matrix 129 in Figure 4(a)–(c) has dimension 1,083, and it is another case of a bidiagonal obtained from a tridiagonal matrix computed by running the Lanczos algorithm without reorthogonalization. In double precision, its 139 largest eigenvalues are of order $O(10^{-8})$ and agree to 12 digits. There are more results larger than 100 in single precision than in double precision for RANGE="I" and RANGE="V". For example, cases 1,030 and 1,501 in Figure 4(d)–(f) (matrices 103 and 151 in Figure 4(a)–(c)) are instances of bidiagonal matrices of dimension 2,100 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".

Refinement of vectors

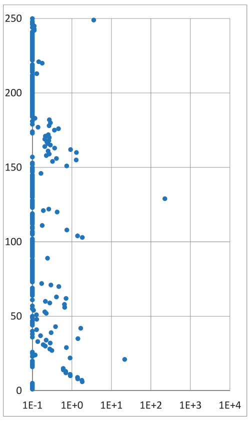
The strategy for refinement of vectors discussed in Section 4 was set off for matrices 248–250 in Figure 2(a), but it was not sufficient to produce orthogonal vectors. As mentioned before, the entries



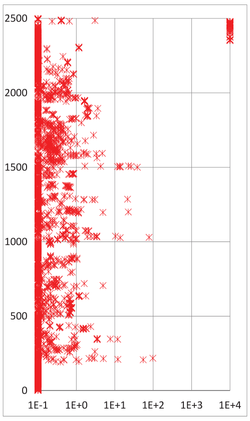
(a) “A”: *resid*



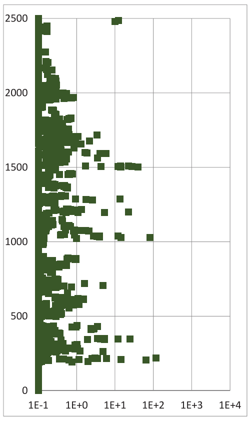
(b) “A”: *orthU*



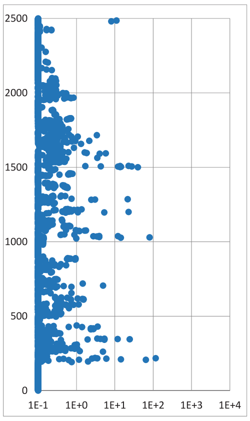
(c) “A”: *orthV*



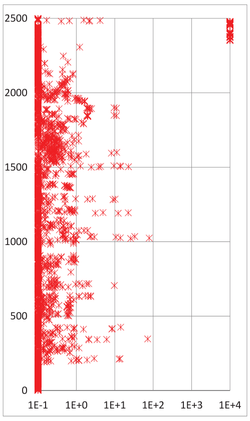
(d) “I”: *resid*



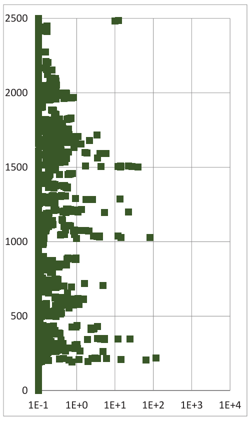
(e) “I”: *orthU*



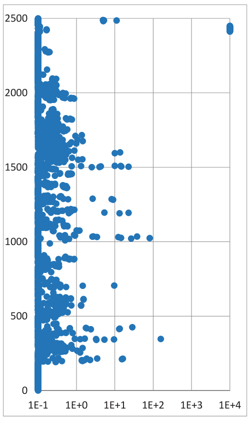
(f) “I”: *orthV*



(g) “V”: *resid*

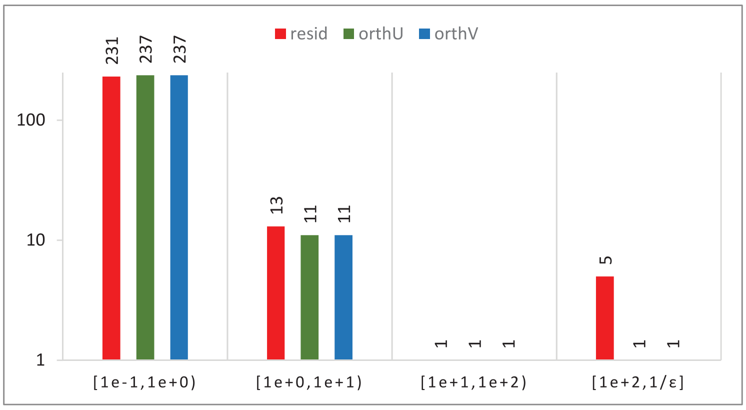


(h) “V”: *orthU*

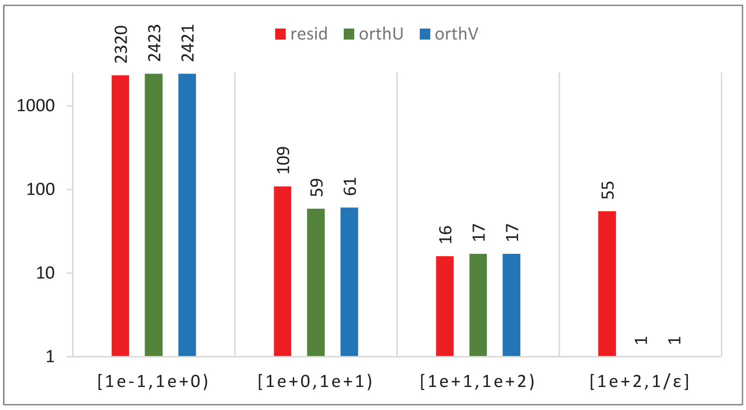


(i) “V”: *orthV*

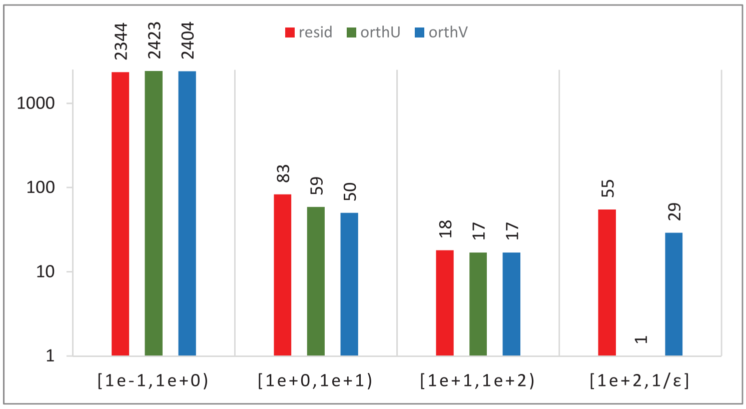
Fig. 4. *resid*, *orthU*, *orthV* (x-axis, log scale) for RANGE=“A”,“I”, and“V”, single precision. (4(a)–(c)) 250 matrices (y-axis), increasing condition numbers; (4(d)–(f)) $n_I = 10$ for each matrix of RANGE=“A”; (4(g)–(i)) $n_V = 10$ for each matrix of RANGE=“A”.



(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).

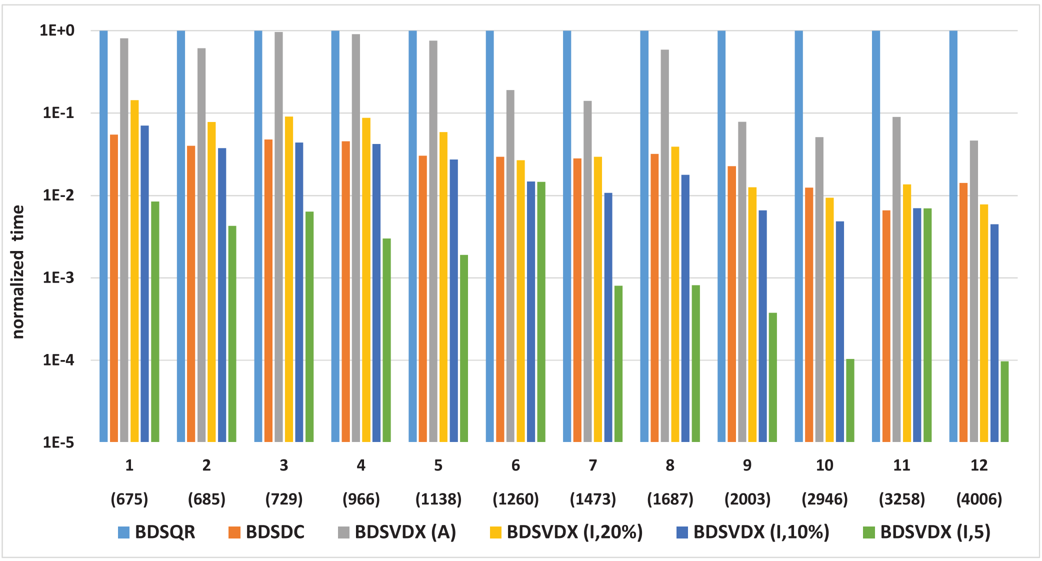


Fig. 6. Timings for BDSQR, BDSDC, and BDSVDX on 12 bidiagonal matrices with dimensions ranging from 675 to 4,006 (x-axis, dimensions in parentheses), in double precision, average time over 10 runs per matrix. BDSVDX: all singular values/vectors, the largest 20%, the largest 10%, and the largest 5 singular values/vectors. For each matrix, the timings (y-axis) are normalized with respect to the largest time and are plotted in log scale.

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 Figure 4(a). For all these matrices, the resulting orthogonality level is smaller than 10. We note that the characteristics of the jth matrix in Figure 4(a) may differ significantly from the characteristics of the jth matrix in Figure 2(a) due to way the matrices are generated, and also differences in the condition numbers in single and double precision.

Performance

We consider 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.

Figure 6 shows the times taken by BDSQR, BDSDC, and BDSVDX on 12 bidiagonal matrices (a sample from Figure 2(a)) with dimensions ranging from 675 to 4,006, 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 MatrixMarket [2018]) and computational chemistry provided by George Fann [Dhillon et al. 1997]. For BDSVDX, we ask

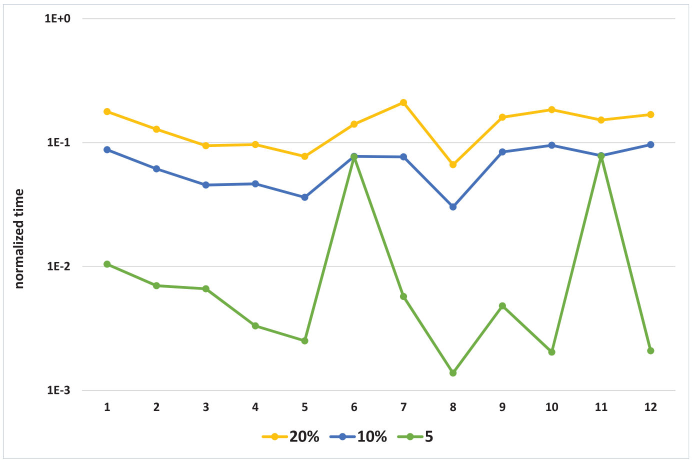


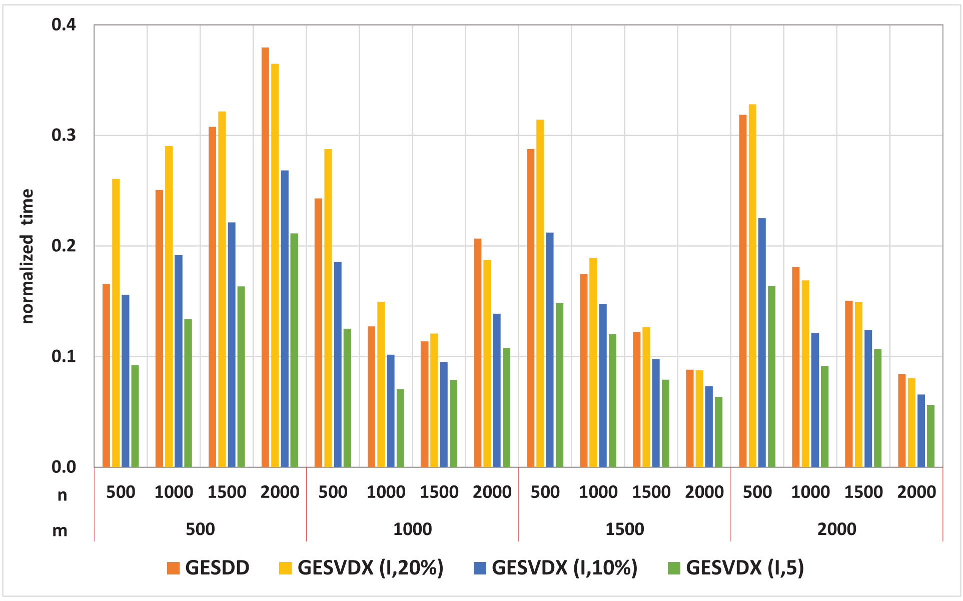
Fig. 7. Performance of BDSVDX for the matrices in Figure 6. The data points (y-axis, log scale) correspond to the computing times for the three subset scenarios (20%, 10%, largest 5) normalized with respect to t_A, where t_A is the time required for the computation of all singular values/vectors.

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 three orders of magnitude faster than BDSQR and two orders of magnitude faster than BDSDC for the computation of the largest five singular values and vectors of the largest matrix.

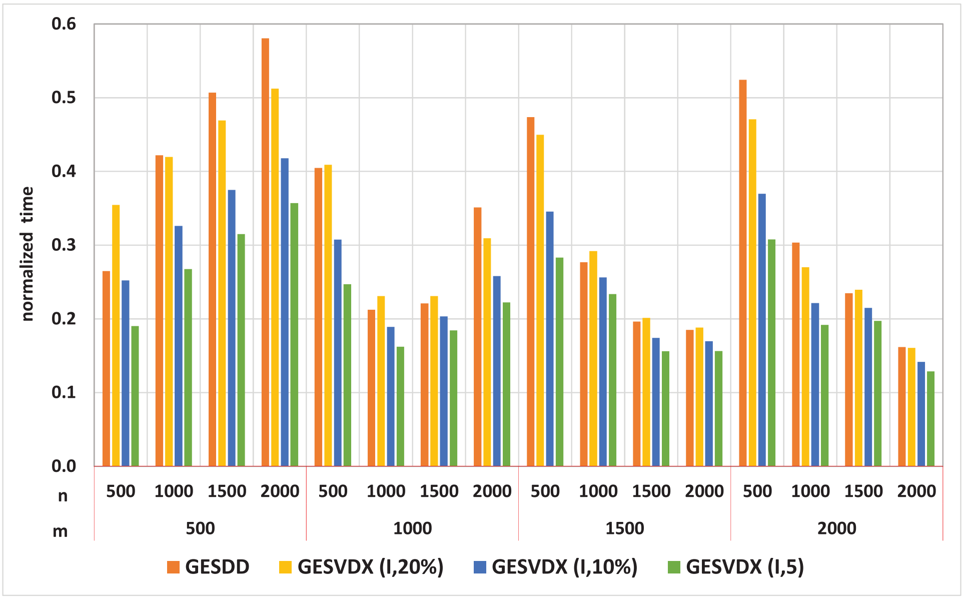
Note that the computation of the largest 10% singular values/vectors and the largest five singular values/vectors for matrix 6 takes about the same time; similarly for matrix 11. Those bidiagonal matrices are obtained from tridiagonal matrices computed by running the Lanczos algorithm without reorthogonalization, as mentioned before. Matrix 6 in Figure 6 is matrix 36 in Figure 2(a) (its largest 138 eigenvalues are of order $O(10^{-3})$ and agree to 12 digits) and matrix 11 in Figure 6 is matrix 129 in Figure 2(a) (its largest 325 eigenvalues are of order $O(10^{7})$ and agree to 13 digits). 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 when compared to BDSQR and BDSVDX, 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.

Figure 7 shows the performance of BDSVDX for the matrices in Figure 6, for the computation of the largest 20%, the largest 10%, and the largest 5 singular values/vectors. The data points in Figure 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

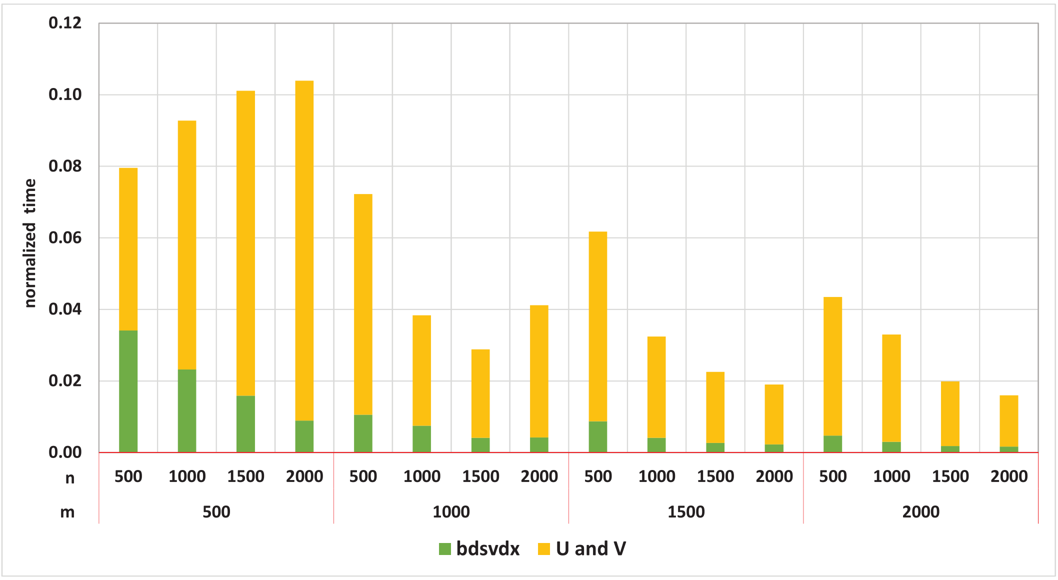


(a) double precision

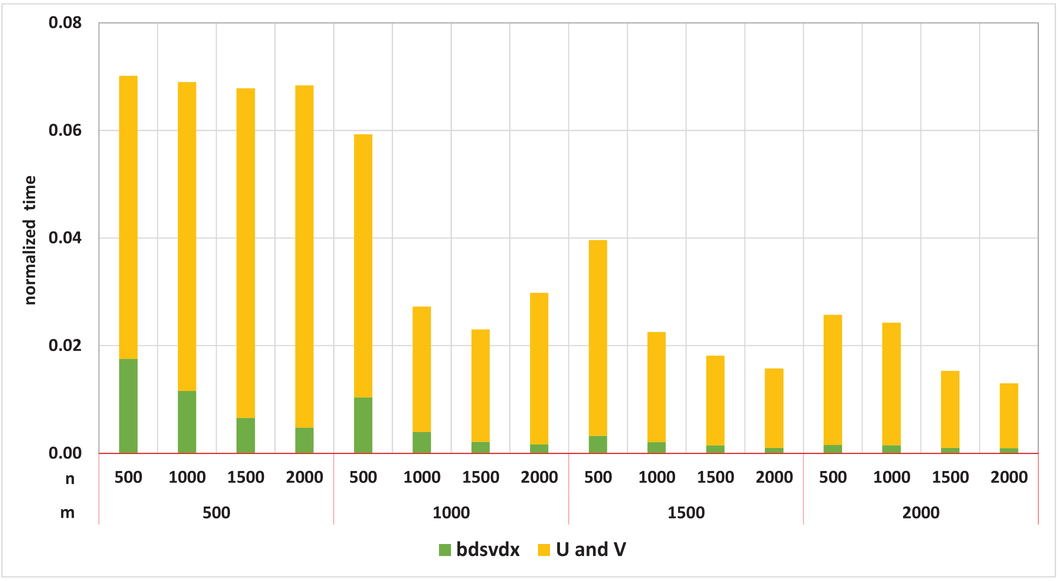


(b) double precision complex

Fig. 8. Timings for GESVD, GESDD, and GESVDX on 16 random dense $m \times n$ matrices with $m, n =$ 500, 1,000, 1,500, 2,000. 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 Figure 8(a) and (b); 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.

clearly shows how the performance of BDSVDX may be affected by the distribution of the singular values, as discussed in the previous paragraph.

We have used TAU [2018] to profile BDSVDX while computing the largest five 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) dominate.

We compare now the performance of the implementations for the SVD of general matrices. The time taken by GESVD, GESDD, and GESVDX in double precision, on random dense $m \times n$ matrices with $m, n = 500, 1{,}000, 1{,}500, 2{,}000$, is shown in Figure 8. GESVDX is also used to compute the largest 20%, the largest 10%, and the largest 5 singular values/vectors. GESVDX 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, Figure 8(a), GESVDX is up to 14 times faster than GESVD and 2 times faster than GESDD. We observe that BDSVDX is faster than BDSDC for all matrices, since the singular value of those matrices are relatively well separated, in contrast to some of the matrices in Figure 6. In the double precision complex case, Figure 8(b), 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.

Finally, Figure 9 shows how the time is spent by GESVDX for the matrices of Figure 8, for the five largest singular values/vectors scenario. The relevant calculations are a 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.

6 CONCLUSIONS

In this work, we presented an algorithm for computing 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 large problems where the full set is not required. 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 (GESVDX).

We carried out extensive experiments using a large set of test matrices to evaluate the effectiveness of the algorithm. Numerical results 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.

Potential future work includes the parallel implementation of the algorithm presented in this article, building upon the workflow of the parallel subroutines PDSYEVX or PDSYEVR implemented in ScaLAPACK [Blackford et al. 1997; ScaLAPACK 2012]. 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 [Voemel 2010]. Specific tests will be required (e.g., with cases similar to the difficult ones in Figure 2) to assess the best alternative. We observe that once a matrix similar to Equation (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.

Finally, we are aware that a modified version of BDSVDX invoking MRRR has been used in Dongarra et al. [2018] (with the assistance of the first author of this article). However, the focus of Dongarra et al. [2018] was on performance and not accuracy.

APPENDICES

A A HIGH-LEVEL DESCRIPTION OF BDSVDX

Table A.1 summarizes the main steps performed by BDSVDX. The steps are matched to the sections where they are discussed, in particular:

(a) This step determines the singular values/vectors to be computed according to the variable RANGE, using bisection, as discussed in Section 2. We note that if RANGE="I", then il,iu are mapped into values.
(b) This test is for splittings in b. These splittings are trivial in the sense that the associated bidiagonal matrices are square, as discussed in Section 3.

Table A.1. Pseudo-code Summarizing the Workflow of BDSVDX

```
BDSVDX: input(range,n,a,b,vl,vu,il,iu), output(s,z)
% range: ‘‘A’’, ‘‘V’’ or ‘‘I’’
% n: dimension of the bidiagonal matrix $B$
% a: diagonal entries of $B$
% b: superdiagonal entries of $B$
% vl,vu: bounds of the interval to be searched for singular values
% il,iu: indexes of the smallest/largest singular values to be returned
% s: singular values
% z: array containing the singular vectors
form matrix (4): t(1:2:2n)=a, t(2:2:2n-1)=b, $j_l$ = 1
determine singular values/vectors to be computed                      (a)
for i = 2:2:2n
   if t(i) is negligible or i = 2n, $j_r$ = i, then                   (b)
       for j = $j_l$:2:$j_r$
          if t(j) is negligible or j = $j_r$, then                    (c)
             invokeSTEVX to process active matrix                     (d)
             normalize singular vectors                               (e)
             apply reorthogonalization                                (f)
          end
       end
   end
end
sort singular values/vectors
discard singular values/vectors depending on range                    (g)
reorder array z                                                       (h)
```

(c) This test checks for splittings in a. These splittings may be nontrivial in the sense that the resulting bidiagonal matrices may be rectangular, as discussed in Section 3. At this level, a bookkeeping is needed for the subsequent composition of the singular vectors.
(d) If a submatrix is found or the bottom of the matrix is reached, we have obtained an "active matrix" and STEVX is invoked to compute eigenvalues/vectors of that matrix according to the mapping indicated by RANGE. In this step, STEVX could be replaced by MRRR, as discussed in Section 1. This step also offers opportunities for computing eigenvectors in parallel.
(e) After the singular vectors are extracted from the eigenvectors, this step performs an explicit normalization of the singular vectors, as discussed in Section 4.
(f) Reorthogonalization of the singular vectors may be needed in pathological cases, as also discussed in Section 4.
(g) This step is for postprocessing, to discard singular values and vectors that may have been computed in excess. This situation can happen when il,iu are mapped into values in large clusters, as we have seen in Figure 7.
(h) This reordering is to conform to Equation (5) in Section 2.2.

B A CASE OF FAILURE IN STEXR

We show here a case of misbehavior of STEXR (double precision) introduced in Willems and Lang [2013], 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 B.1 lists the entries of T used in the test. Here, we have used the GNU Fortran compiler because the distribution in Willems and Lang [2013] does not provide a configuration for the Intel Fortran compiler. Although not shown, the eigenvalues of T computed with the eigensolvers listed in Table 1 and also STEXR are in very good agreement. Specifically, $||T - Z_{\mathsf{XR}}\Lambda_{\mathsf{XR}}Z_{\mathsf{XR}}^T||/(||T||n\varepsilon) \approx 0.6$, where Λ_{XR} contains the eigenvalues returned by STEXR on its main diagonal, and Z_{XR} is the matrix of eigenvectors returned by STEXR. However, $||I - Z_{\mathsf{XR}}^T Z_{\mathsf{XR}}||/(n\varepsilon) \approx 3.95 \times 10^4$; see Figure B.1. In contrast, $||I - Z_{\mathsf{VX}}^T Z_{\mathsf{VX}}||/(n\varepsilon) \approx 0.9$ and $||I - Z_{\mathsf{MR}}^T Z_{\mathsf{MR}}||/(n\varepsilon) \approx 0.1$, where Z_{VX} and Z_{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

Table B.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	-

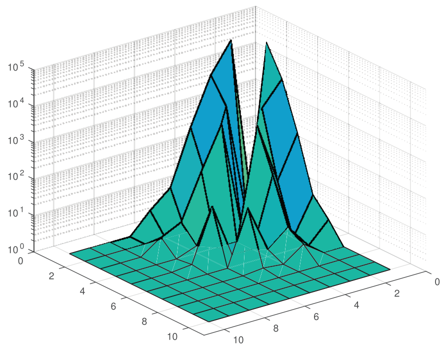


Fig. B.1. Surface plot of $|I - Z_{\mathsf{XR}}^T Z_{\mathsf{XR}}|/(n\varepsilon)$ in log scale, where Z_{XR} contains the eigenvectors returned by STEXR for the tridiagonal matrix given in Table B.1. The first four columns of Z_{XR} are linearly dependent: those columns correspond to $\lambda_1 \approx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

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.

ACKNOWLEDGMENTS

The authors thank Beresford Parlett for his comments and suggestions on earlier versions of the manuscript.

REFERENCES

M. Abramowitz. 1974. *Handbook of Mathematical Functions, With Formulas, Graphs, and Mathematical Tables*. Dover Publications, Inc., New York, NY.

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. 1999. *LAPACK Users' Guide* (3rd ed.). Society for Industrial and Applied Mathematics, Philadelphia, PA.

T. Auckenthaler, V. Blum, H. J. Bungartz, T. Huckle, R. Johanni, L. Krämer, B. Lang, H. Lederer, and P. R. Willems. 2011. Parallel solution of partial symmetric eigenvalue problems from electronic structure calculations. *Parallel Comput.* 37 (2011), 783–794.

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

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. 1997. *ScaLAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA.

J. Demmel. 1997. *Applied Numerical Linear Algebra*. Vol. 56. Society for Industrial and Applied Mathematics, Philadelphia, PA.

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

J. Demmel and W. Kahan. 1990. Accurate singular values of bidiagonal matrices. *SIAM J. Sci. and Stat. Comput.* 11 (1990), 873–912.

J. Demmel, O. Marques, C. Voemel, and B. Parlett. 2008. Performance and accuracy of LAPACK's symmetric tridiagonal eigensolvers. *SIAM J. Sci. Comput.* 30 (2008), 1508–1526.

I. Dhillon. 1997. *A New $O(N^2)$ Algorithm for the Symmetric Tridiagonal Eigenvalue/Eigenvector Problem*. Ph.D. Dissertation. University of California, Berkeley.

I. Dhillon, G. Fann, and B. Parlett. 1997. Application of a new algorithm for the symmetric eigenproblem to computational quantum chemistry. In *Proceedings of the 8th SIAM Conference on Parallel Processing for Scientific Computing*. SIAM, Philadelphia. http://www.cs.utexas.edu/ inderjit/public_papers/fannchem1.ps.

I. Dhillon and B. Parlett. 2004. Multiple representations to compute orthogonal eigenvectors of symmetric tridiagonal matrices. *Linear Algebra Appl.* 387 (2004), 1–28.

I. Dhillon, B. Parlett, and C. Voemel. 2006. The design and implementation of the MRRR algorithm. *ACM Trans. Math. Softw.* 32 (2006), 533–560.

J. Dongarra, M. Gates, A. Haidar, J. Kurzak, P. Luszczek, S. Tomov, and I. Yamazaki. 2018. The singular value decomposition: Anatomy of optimizing an algorithm for extreme scale. *SIAM Rev.* 60, 4 (2018), 808–865.

K. Fan and A. Hoffman. 1955. Some metric inequalities in the space of matrices. *Proc. Amer. Math. Soc.* 6, 1 (1955), 111–116.

G. Golub and W. Kahan. 1965. Calculating the singular values and pseudo-inverse of a matrix. *J. Society for Ind. Appl. Math. Ser. B Numer. Anal.* 2, 2 (1965), 205–224.

B. Grosser and B. Lang. 2003. An $O(n^2)$ algorithm for the bidiagonal SVD. *Linear Algebra Appl.* 358, 1 (2003), 45–70.

B. Grosser and B. Lang. 2005. On symmetric eigenproblems induced by the bidiagonal SVD. *SIAM. J. Matrix Anal. and Appl.* 26, 3 (2005), 599–620.

Nicholas J. Higham and Pythagoras Papadimitriou. 1993. *Parallel Singular Value Decomposition via the Polar Decomposition.* University of Manchester, Department of Mathematics.

Intel. 2017a. Intel(R) Fortran Compiler 2018 Update 1 for Linux.

Intel. 2017b. Intel(R) Math Kernel Library 2018 Update 1 for Linux.

LAPACK. 2018. LAPACK—Linear Algebra PACKage. https://github.com/Reference-LAPACK.

S. Li, M. Gu, L. Cheng, X. Chi, and M. Sun. 2014. An accelerated divide-and-conquer algorithm for the bidiagonal SVD problem. *SIAM. J. Matrix Anal. and Appl.* 35 (2014), 1038–1057.

O. Marques, J. Demmel, C. Voemel, and B. Parlett. 2008. A testing infrastructure for symmetric tridiagonal eigensolvers. *ACM Trans. Math. Softw.* 35 (2008), 8:1–8:13.

O. Marques and P. B. Vasconcelos. 2016. Computing the bidiagonal SVD through an associated tridiagonal eigenproblem. In *12th International Conference on High Performance Computing for Computational Science (VECPAR 2016) Porto, Portugal, June 28-30, 2016, Revised Selected Papers.* Springer, Heildeberg, Germany, 64–74.

MatrixMarket. 2018. Matrix Market. Retrieved from http://math.nist.gov/MatrixMarket.

A. M. Matsekh. 2005. The Godunov-inverse iteration: A fast and accurate solution to the symmetric tridiagonal eigenvalue problem. *Appl. Numer. Math.* 54, 2 (2005), 208–221.

Yuji Nakatsukasa, Zhaojun Bai, and François Gygi. 2010. Optimizing Halley's iteration for computing the matrix polar decomposition. *SIAM J. Matrix Anal. Appl.* 31, 5 (2010), 2700–2720.

Yuji Nakatsukasa and Nicholas J. Higham. 2013. Stable and efficient spectral divide and conquer algorithms for the symmetric eigenvalue decomposition and the SVD. *SIAM J. Sci. Comput.* 35, 3 (2013), A1325–A1349.

F. Olver, D. Lozier, R. Boisvert, and C. Clark. 2010. *NIST Handbook of Mathematical Functions Hardback and CD-ROM.* Cambridge University Press, New York.

B. Parlett. 1995. The new QD algorithms. *Acta Numer.* 4 (1995), 459–491.

B. Parlett. 1998. *The Symmetric Eigenvalue Problem.* Vol. 20. Society for Industrial and Applied Mathematics, Philadelphia, PA.

B. Parlett and C. Vömel. 2009. The spectrum of a glued matrix. *SIAM J. Matrix Anal. Appl.* 31, 1 (2009), 114–132.

ScaLAPACK. 2012. ScaLAPACK, version 2.0.2. Retrieved from http://www.netlib.org/scalapack.

TAU. 2018. TAU Performance System. Retrieved from https://www.cs.uoregon.edu/research/tau.

C. Voemel. 2010. ScaLAPACK's MRRR algorithm. *ACM Trans. Math. Softw.* 37 (2010), 1–35.

J. Vogel, J. Xia, S. Cauley, and V. Balakrishnan. 2016. Superfast divide-and-conquer method and perturbation analysis for structured eigenvalue solutions. *SIAM J. Sci. Comput.* 38, 3 (2016), A1358–A1382.

P. Willems. 2010. *On MR^3-type Algorithms for the Tridiagonal Symmetric Eigenproblem and the Bidiagonal SVD.* Ph.D. Dissertation. University of Wuppertal.

P. Willems and B. Lang. 2013. A framework for the MR^3 algorithm: Theory and implementation. *SIAM J. Sci. Comput.* 35 (2013), 740–766.

P. Willems, B. Lang, and C. Voemel. 2006. Computing the bidiagonal SVD using multiple relatively robust representations. *SIAM. J. Matrix Anal. and Appl.* 28 (2006), 907–926.

Received April 2018; revised January 2019; accepted September 2019