

xGESVDQ: A QR-preconditioned QR method for computing the singular value decomposition

Installation and Testing

User Guide

Zlatko Drmač*

Department of Mathematics, Faculty of Science, University of Zagreb

September 25, 2017

Abstract

xGESVDQ [6] is a LAPACK-based software for computing the singular value decomposition (SVD) by a preconditioned QR method. With respect to numerical accuracy, it is superior to the existing implementation of the QR method (**xGESVD**), and to the divide and conquer method (**xGESDD**) from LAPACK [1]. Although it is theoretically not yet entirely understood, based on overwhelming numerical evidence, the new method can be considered almost as accurate as the Jacobi SVD [4], [7, 8]. The software has a built in error estimate device and the result is accompanied with realistic error bounds for both the singular values and the singular vectors. Numerical experiments show close match between the measured and estimated errors, which behave as predicted by the perturbation theory: for a full column rank A , the relative errors in the computed singular values depend on the scaled condition number $\min_{D=\text{diag}} \kappa_2(AD)$, and the errors in the singular vectors, in addition, depend on the relative separations between the neighboring singular values. At the same time, the new implementation exploits highly optimized **xGESVD** in libraries such as the Intel's MKL and it can be considered as method of choice for fast and accurate SVD. Simple modular structure of the new software allows straightforward conversion to a ScaLAPACK parallel implementation.

This document describes the main properties, using and testing of the implementation of the **xGESVDQ** subroutines.

*The author's work is supported by the Croatian Science Foundation, under grant HRZZ-IP-11-2013-9345.

1 Introduction

The SVD (Singular Value Decomposition) is one of the most important decompositions in matrix computations, with a variety of applications in sciences and engineering. State of the art numerical software library LAPACK [1] contains three different methods for computing the SVD of general matrices:

- **xGESVD**, which reduces the matrix to bidiagonal form and then it deploys the zero shift bidiagonal QR SVD;
- **xGESDD**, which also uses the bidiagonal reduction, but the bidiagonal SVD is computed via the *divide and conquer* scheme;
- **xGEJSV**, **xGESVJ** which are one-sided Jacobi SVD methods.

Among them, **xGESDD** is the fastest and least accurate; **xGEJSV** and **xGESVJ** are the most accurate but the slowest, although **xGEJSV** in some implementations was able to outperform **xGESVD**.

xGESVDQ, described in this document, is an enhanced version of **xGESVD** – at a price of an extra pivoted QR decomposition (e.g. **xGEQP3**), the existing QR SVD **xGESVD** becomes nearly as accurate as the Jacobi SVD. The numerical details of the method are given in [6], and it is implemented in the SVD library **xGESVDQ** which provides LAPACK-style FORTRAN codes for computing the SVD decomposition of general matrices in the four data types REAL (**SGESVDQ**), DOUBLE PRECISION (**DGESVDQ**), COMPLEX (**CGESVDQ**) and DOUBLE COMPLEX (**ZGESVDQ**).

In §2 we give the necessary details for using the subroutines, and §3 describes installation and testing.

2 Calling xGESVDQ

The calling sequence of **xGESVDQ** follows the one of **XGESVD**, with few differences that reflect the specific new features of **xGESVDQ**. For the sake of brevity, we describe the arguments of **CGESVDQ** only. For the other three data types, the calling sequences are obtained *mutatis mutandis* and will be clear from the comments in the source codes.

2.1 Method and notation

For simpler and more precise description of the code, in Algorithm 1 we first describe the structure of the method. On input, the (real or complex) matrix A is assumed $m \times n$ with $m \geq n$. In applications with $m < n$, the usage of SVD should be reformulated in terms of A^* , in the obvious way.

Algorithm 1 $(U, \Sigma, V) = \text{xGESVDQ}(A, \text{method})$ ($A \in \mathbb{C}^{m \times n}$, $m \geq n$)

- 1: $(\Pi_r A) \Pi_c = Q \begin{pmatrix} R \\ 0 \end{pmatrix}$
 {Initial row sorting by the permutation matrix Π_r and QR factorization with dynamic column pivoting Π_c , e.g. **xGEQP3**, or simple pre-sorting. Full pivoting is also an option.}
 - 2: Determine the numerical rank ρ of R and set $R_\rho = R(1 : \rho, 1 : n)$.
 - 3: **if** $\rho = n$ and condition estimate needed **then**
 - 4: $\kappa \approx \|(R_\rho \text{diag}(1/\|R_\rho(:, i)\|_2))^{-1}\|_2$ {Use e.g. **xPOCON** and adjust to the norm $\|\cdot\|_2$.}
 - 5: **end if**
 - 6: **if** method = "upper" **then**
 - 7: Compute the SVD $R_\rho = \hat{U} \begin{pmatrix} \hat{\Sigma} & 0_{\rho, n-\rho} \end{pmatrix} \hat{V}^*$. {Use **xGESVD**}
 - 8: The SVD of A is $A = U \Sigma V^* \equiv \left[\Pi_r^T Q \begin{pmatrix} \hat{U} & 0 \\ 0 & I_{m-\rho} \end{pmatrix} \right] \begin{pmatrix} \hat{\Sigma} & 0_{\rho, n-\rho} \\ 0_{m-\rho, \rho} & 0_{m-\rho, n-\rho} \end{pmatrix} (\Pi_c \hat{V})^*$
 - 9: **else**
 - 10: Compute the SVD $R_\rho^* = \hat{U} \begin{pmatrix} \hat{\Sigma} \\ 0_{m-\rho, \rho} \end{pmatrix} \hat{V}^*$. {Use **xGESVD**}
 - 11: The SVD of A is $A = U \Sigma V^* \equiv \left[\Pi_r^T Q \begin{pmatrix} \hat{V} & 0 \\ 0 & I_{m-\rho} \end{pmatrix} \right] \begin{pmatrix} \hat{\Sigma} & 0_{\rho, n-\rho} \\ 0_{m-\rho, \rho} & 0_{m-\rho, n-\rho} \end{pmatrix} (\Pi_c \hat{U})^*$
 - 12: **end if**
-

2.2 Argument list (CGESVDQ)

SUBROUTINE CGESVDQ(JOBA, JOBP, JOBR, JOBU, JOBV, m, n, A, LDA, S, U, LDU, V, LDV, NUMRANK, IWORK, CWORK, LCWORK, RWORK, LRWORK, INFO)

1. **JOBA** [*input*][CHARACTER]

Provides information on the level of accuracy expected for the SVD of A . It specifies the cutoff threshold for truncating the smallest rows of the computed upper triangular factor R , thus truncating the smallest singular values. (The index k described below defines the value of ρ in Line 2. in Algorithm 1; ϵ is the roundoff unit.)

'A' This is the most aggressive option. After the initial QR factorization, the upper triangular factor will be truncated at the smallest index k such that the truncated part $R(k+1 : n, k+1 : n)$ of R is below $\epsilon \sqrt{n} \max_{i=1:n} \|A(:, i)\|_2$ ($\equiv \epsilon \sqrt{n} |R_{11}|$ in case of column pivoting that brings the largest column to the fore). In this case, the smallest $n - k$ singular values are flushed to zero, but the backward error is still within the standard backward stability parameters as in **xGESVD** and **xGESDD**.

- 'M' Similar as in the previous case, but the cutoff criterion is milder. The truncation occurs at the smallest index k for which $|R_{k+1,k+1}| \leq \epsilon |R_{kk}|$.
- 'H' High relative accuracy requested. A submatrix $R(k+1:n, k+1:n)$ will be truncated only if it is zero.
- 'E' This is the expert level. The level of accuracy is the same as with 'H', and, in addition, information needed for error estimate in the computed singular values and vectors is provided. The relevant (scaled) condition number is estimated and returned in **RWORK**. If R is computed as singular, the scaled condition number of its largest leading nonsingular submatrix will be estimated.

2. **JOBP** [*input*][CHARACTER]

Specifies the pivoting in the initial QR factorization, i.e. it determines whether to use row pivoting. (Permutation matrix Π_r in Line 1 of Algorithm 1.)

- 'P' Prior to the column pivoted QR factorization, the rows of A are permuted by a permutation π such that

$$\|A(\pi(1), :)\|_{\infty} \geq \|A(\pi(2), :)\|_{\infty} \geq \cdots \geq \|A(\pi(m), :)\|_{\infty}.$$

For high accuracy, this option is recommended if it is expected that A might have differently scaled rows, and high accuracy is desired.

- 'N' No row pivoting is used. The QR factorization in Line 1. of Algorithm 1 is computed only with a rank revealing column pivoting.

3. **JOB**R [*input*][CHARACTER] Specifies whether the SVD of R or R^* is computed. Both choices will work well numerically. This option is left for the implementor to decide which can be better optimized. Researchers can use this option for further study of the numerical properties of the bidiagonalization and the structure of the bidiagonal form.

- 'T' The SVD of R^* is computed and used in assembling the SVD of A .
- 'N' The SVD of R is computed and used in assembling the SVD of A . This may be preferred as it involves less data movement (matrix transpositions).

4. **JOB**U [*input*][CHARACTER]

- 'A' All m left singular vectors are computed and returned as columns in the array $[U]$. See the description of $[U]$.
- 'S' or 'U' The leading n left singular vectors are computed and returned as the leading n columns in the array $[U]$. See the description of $[U]$.

'R' The leading r left singular vectors (belonging to the r largest nonzero singular values) are computed and returned as columns in the array **[U]**. Here $r = \mathbf{NUMRANK}$ is the numerical rank that is determined by counting nonzero computed singular values. The value of r can be smaller than ρ , determined after the first QR factorization (depending on **JOBA**). (**CGESVDQ** actually computes ρ vectors, and $r < \rho$ may occur e.g. due to underflowed singular values.)

'F' The n left singular values are computed and returned in factored form as the product of the Q factor from the initial QR factorization and the n left singular vectors of $(R^*, 0)^*$. If row pivoting is used, then the necessary information on the row pivoting is stored in **IWORK**($N+1:N+M-1$). See the descriptions of **A**, **U**, **CWORK**, **IWORK**. See §2.3.2.

'N' The left singular vectors are not computed. The array **[U]** is not referenced.

5. **JOBV** [*input*][**CHARACTER**]

'A' or **'V'** All n right singular vectors are computed and returned as conjugate transposed in the rows of the array **[V]**.

'R' The leading r right singular vectors are computed and returned as conjugate transposed in the rows of the array **[V]**. Here $r = \mathbf{NUMRANK}$ is the numerical rank that is determined by counting nonzero computed singular values. The value of r can be smaller than ρ , determined after the first QR factorization (depending on **JOBA**). (**CGESVDQ** actually computes ρ vectors.) This option is allowed only if the same is set for **JOBV**, or if **JOBV**='N'. Otherwise, illegal value is declared.

'N' The right singular vectors are not computed. If **JOBA**='E', the array **[V]** might be used as a workspace; see the description of **[V]**.

6. **m** [*input*][**INTEGER**]

The number of rows of the matrix; $m \geq 0$.

7. **n** [*input*][**INTEGER**]

The number of columns of the matrix; $0 \leq n \leq m$.

8. **A** [*input/workspace/output*][**COMPLEX ARRAY**][**LDA** \times **n**]

On input, **[A]** contains the matrix A . After the QR factorization, the part of **[A]** below the main diagonal contains the Householder vectors used in the factorization. If **JOBV** \neq 'N', that part of **[A]** is left unchanged on the output. If **JOBV** = 'F', it will be used to restore or to apply the matrix of the left singular vectors.

9. **LDA** [*input*][INTEGER]

The leading dimension of the array **[A]**. $\mathbf{LDA} \geq \max(1, \mathbf{m})$.

10. **S** [*output*][REAL ARRAY][$\mathbf{n} \times 1$]

The computed singular values of A , ordered so that

$$\mathbf{S}(i) \geq \mathbf{S}(i+1), i = 1, \dots, \mathbf{n} - 1.$$

11. **U** [*output*] [COMPLEX ARRAY] The dimensions of **[U]** are determined as follows:

- (i) **LDU** $\times\mathbf{m}$, if **JOBV** = 'A'; see the description of **LDU**. In this case, on exit, **[U]** contains the \mathbf{m} left singular vectors.
- (ii) **LDU** $\times\mathbf{n}$, if **JOBV** $\in \{'S', 'U', 'R'\}$; see the description of **LDU**. In this case, **[U]** contains the leading \mathbf{n} or the leading r left singular vectors, where $r = \mathbf{NUMRANK}$ is the numerical rank. See the description of **JOBV**.
- (iii) **LDU** $\times\mathbf{n}$, if **JOBV** = 'F'; see the description of **LDU**. In this case **[U]** contains the $n \times n$ matrix of the left singular vectors of the upper triangular matrix R from Line 1. of Algorithm 1. This information can be used to form the left singular vectors of **A**.

If **JOBV** = 'N', then **[U]** is not referenced.

12. **LDU** [*input*][INTEGER]

The leading dimension of the array **[U]**.

- (i) If **JOBV** $\in \{'A', 'S', 'U', 'R'\}$, then $\mathbf{LDU} \geq \max(1, \mathbf{m})$.
- (ii) If **JOBV** = 'F', then $\mathbf{LDU} \geq \max(1, \mathbf{n})$.
- (iii) Otherwise, $\mathbf{LDU} \geq 1$.

13. **V** [*workspace/output*] [COMPLEX ARRAY] The dimensions of **[V]** are determined as follows:

- (i) **LDV** $\times\mathbf{n}$, if **JOBV** = 'A', or 'V'. In that case **[V]** contains the $\mathbf{n} \times \mathbf{n}$ unitary matrix V^* ; if **JOBA** = 'E', it is also used as a workspace.
- (ii) **LDV** $\times\mathbf{n}$, if **JOBV** = 'R'. In that case **[V]** contains the first $r = \mathbf{NUMRANK}$ rows of V^* (the right singular vectors, stored rowwise, of the $\mathbf{NUMRANK}$ largest singular values). If **JOBA** = 'E', it is also used as a workspace.
- (iii) If **JOBV** = 'N' and **JOBA** = 'E', **[V]** is used only as a workspace.

If **JOBV** = 'N', and **JOBA** ≠ 'E', [**V**] is not referenced.

14. **LDV** [*input*][INTEGER]

The leading dimension of the array [**V**].

- (i) If **JOBV** ∈ {'A', 'V', 'R'} or **JOBA** = 'E', then **LDV** ≥ max(**n**, 1).
- (ii) Otherwise, **LDV** ≥ 1.

15. **NUMRANK** [*output*][INTEGER]

NUMRANK is first set to the numerical rank determined after the rank revealing QR factorization, following the strategy determined by the value of **JOBA**. If **JOBV**='R', and if **JOBV**='R', only **NUMRANK** leading singular values and vectors are then requested in the call of **CGESVD**. The value of **NUMRANK** might be further reduced if some smallest singular values of the triangular factor are computed by **CGESVD** as exact zeros. (This is possible, e.g. due to underflows.)

16. **IWORK** [*workspace/output*][INTEGER ARRAY] The length of **IWORK** is:

- (i) $\mathbf{n} + \mathbf{m} - 1$, if **JOBP** = 'P',
- (ii) \mathbf{n} , if **JOBP** = 'N'

On exit, **IWORK**(1 : **n**) contains column pivoting permutation of the rank revealing QR factorization. If **JOBP** = 'P', **IWORK**(**n** + 1 : **n** + **m** - 1) contains the indices of the sequence of row swaps used in row pivoting. These can be used to restore the left singular vectors in the case **JOBV**='F'. See §2.3.2.

Note: In case of real data type (SGESVDQ, DGESVDQ) the length of **IWORK** increases by **n** in both cases above if **JOBA**='E'.

17. **CWORK** [*workspace/output*][COMPLEX ARRAY][**LCWORK**×1]

- (i) If, on entry, **LCWORK** ≠ -1, then on exit **CWORK**(1 : **n**) contains parameters needed to recover the *Q* factor from the QR factorization computed by CGEQP3.
- (ii) If, on entry, **LCWORK** = -1, then a workspace query is assumed and **CWORK** must be of length at least two. On exit **CWORK**(1) contains the optimal length of **CWORK** and **CWORK**(2) contains the minimal length.

18. **LCWORK** [*input/output*][INTEGER]

The dimension of the array **CWORK**.

- (i) If, on entry, **LCWORK** = -1, (workspace query) then the optimal and the minimal length of **CWORK** are computed and returned in the first two entries of **CWORK**.
- (ii) Otherwise, on entry, **LCWORK** must contain the length of **CWORK**. This ensures that the work space of proper length is allocated. The minimal and the maximal requirements depend on the workspaces required for **CGEQP3**, **CGESVD** and **CUNMQR** and are determined as follows: For the minimal length, define

$$\begin{aligned}
\text{LWQP3} &= \mathbf{n} + 1 \text{ (minimal workspace length for } \mathbf{m} \times \mathbf{n} \text{ CGEQP3),}^1 \\
\text{LWUNQ} &= \text{(minimal workspace length for 'left' } \mathbf{m} \times \mathbf{n} \text{ CUNMQR) =} \\
&\quad 1. \max(\mathbf{n}, 1) \text{ if } \mathbf{JOB} = \text{'R', 'S', or 'U' ;} \\
&\quad 2. \max(\mathbf{m}, 1) \text{ if } \mathbf{JOB} = \text{'A'} \\
\text{LWSVD} &= \max(1, 3\mathbf{n}) \text{ (minimal workspace length for } \mathbf{n} \times \mathbf{n} \text{ CGESVD)}^2 \\
\text{LWSVD2} &= \max(1, 3(\mathbf{n}/2)) \text{ (minimal workspace length for } \mathbf{n}/2 \times \mathbf{n}/2 \\
&\quad \text{CGESVD)} \\
\text{LWLQF} &= \max(1, \mathbf{n}/2) \text{ (minimal workspace length for } \mathbf{n}/2 \times \mathbf{n} \\
&\quad \text{CGELQF)} \\
\text{LWQRF} &= \max(1, \mathbf{n}/2) \text{ (minimal workspace length for } \mathbf{n} \times \mathbf{n}/2 \\
&\quad \text{CGEQRF)} \\
\text{LWUNLQ} &= \max(1, \mathbf{n}) \text{ (minimal workspace length for 'right' } \mathbf{n} \times \mathbf{n} \\
&\quad \text{CUNMLQ)} \\
\text{LWUNQ2} &= \max(1, \mathbf{n}) \text{ (minimal workspace length for 'right' } \mathbf{n} \times \mathbf{n} \\
&\quad \text{CUNMLQ)} \\
\text{LWCON} &= 2\mathbf{n} \text{ (minimal workspace length for } \mathbf{n} \times \mathbf{n} \text{ CPOCON)}^3
\end{aligned}$$

Then the minimal value of **LCWORK** is:

- Σ
 - $\max(\mathbf{n} + \text{LWQP3}, \text{LWSVD})$, if only the singular values are needed;
 - $\max(\mathbf{n} + \text{LWQP3}, \text{LWCON}, \text{LWSVD})$, if the singular values and a scaled condition estimate are requested;
- Σ, U
 - $\mathbf{n} + \max(\text{LWQP3}, \text{LWSVD}, \text{LWUNQ})$, if the singular values and the left singular vectors are requested;

¹In case of real data type, $\text{LWQP3} = 3\mathbf{n} + 1$.

²In case of real data type, $\text{LWSVD} = 5\mathbf{n}$.

³In case of real data type, $\text{LWCON} = 3\mathbf{n}$.

- $\mathbf{n} + \max(\text{LWQP3}, \text{LWCON}, \text{LWSVD}, \text{LWUNQ})$, if the singular values, the left singular vectors and a scaled condition estimate are requested;
- $\boxed{\Sigma, V}$
 - $\mathbf{n} + \max(\text{LWQP3}, \text{LWSVD})$, if the singular values and the right singular vectors are requested;
 - $\mathbf{n} + \max(\text{LWQP3}, \text{LWCON}, \text{LWSVD})$, if the singular values, the right singular vectors and a scaled condition estimate are requested;
- $\boxed{\Sigma, U, V}$
 - $\mathbf{n} + \max(\text{LWQP3}, \text{LWSVD}, \text{LWUNQ})$, if the full SVD is requested, **JOBV**='R', and independent of **JOBR**;
 - $\mathbf{n} + \max(\text{LWQP3}, \text{LWCON}, \text{LWSVD}, \text{LWUNQ})$, if the full SVD with scaled condition estimate is requested, **JOBV**='R', and independent of **JOBR**.

The above specified length of **CWORK** suffices also if **JOBV**='A' and the numerical rank of A (as seen after the rank revealing QR factorization) is at least $\mathbf{n}/2$. However, if **JOBV**='A' and A is estimated of low numerical rank, then a more efficient computation is possible and in that case the length of work is determined as follows:

- ◊ $\max(\mathbf{n} + \max(\text{LWQP3}, \text{LWSVD}, \text{LWUNQ}), \mathbf{n} + \max(\text{LWQP3}, \mathbf{n}/2 + \text{LWLQF}, \mathbf{n}/2 + \text{LWSVD2}, \mathbf{n}/2 + \text{LWUNLQ}, \text{LWUNQ}))$ if the full SVD is requested with **JOBV**='A', 'V', and **JOBR**='N' ;
- ◆ $\max(\mathbf{n} + \max(\text{LWQP3}, \text{LWCON}, \text{LWSVD}, \text{LWUNQ}), \mathbf{n} + \max(\text{LWQP3}, \text{LWCON}, \mathbf{n}/2 + \text{LWLQF}, \mathbf{n}/2 + \text{LWSVD2}, \mathbf{n}/2 + \text{LWUNLQ}, \text{LWUNQ}))$ if the full SVD with scaled condition estimate is requested, with **JOBV**='A', 'V', and **JOBR**='N' ;
- △ $\max(\mathbf{n} + \max(\text{LWQP3}, \text{LWSVD}, \text{LWUNQ}), \mathbf{n} + \max(\text{LWQP3}, \mathbf{n}/2 + \text{LWQRF}, \mathbf{n}/2 + \text{LWSVD2}, \mathbf{n}/2 + \text{LWUNQ2}, \text{LWUNQ}))$ if the full SVD is requested with **JOBV**='A', 'V', and **JOBR**='T' ;
- ▲ $\max(\mathbf{n} + \max(\text{LWQP3}, \text{LWCON}, \text{LWSVD}, \text{LWUNQ}), \mathbf{n} + \max(\text{LWQP3}, \text{LWCON}, \mathbf{n}/2 + \text{LWQRF}, \mathbf{n}/2 + \text{LWSVD2}, \mathbf{n}/2 + \text{LWUNQ2}, \text{LWUNQ}))$ if the full SVD with scaled condition estimate is requested with **JOBV**='A', 'V', and **JOBR**='T' ;

The optimal value of **LCWORK** is determined analogously, using the optimal workspace lengths for **CGEQP3**, **CGESVD** and **CUNMQR**.

19. $\boxed{\boxed{\mathbf{RWORK}}}$ [*workspace/output*][REAL ARRAY][**LRWORK**×1]

On exit, if **JOBA**='E', **RWORK**(1) contains an estimate of $\sqrt{\|(A_c^* A_c)^{-1}\|_1}$, where A_c is obtained from A by normalizing its columns to unit Euclidean length. Otherwise, **RWORK**(1)= -1. See §2.3.1.

RWORK(2) contains the number of singular values computed as exact zeros when **CGESVD** is applied to the (possibly truncated) matrix R . See the description of **NUMRANK**.

In case of early exit (no call to **CGESVD**, such as in the case of zero matrix) **RWORK**(2)=-1.

20. **LRWORK** [*input*][**INTEGER**]

The dimension of the array **RWORK**. If **JOBP**='P', then **LRWORK** $\geq \max(2, \mathbf{m}, 5\mathbf{n})$. Otherwise, **LRWORK** $\geq \max(2, 5\mathbf{n})$.

21. **INFO** [*output*][**INTEGER**]

= 0 : successful exit.

< 0 : if **INFO** = $-i$, the i -th argument had an illegal value.

> 0 : if **CBDSQR** did not converge, **INFO** specifies how many superdiagonals of an intermediate bidiagonal form B (computed in **CGESVD**) did not converge to zero.

2.3 Further details

2.3.1 Using the condition number estimate for error bounds

Since the method first computes the triangular factor R , the condition number that gives a realistic error bound in most cases can be efficiently estimated as follows: The columns of the upper triangular factor R are scaled to have unit Euclidean norms; the scaled matrix is denoted by R_c . If A_c is defined analogously, then $\Pi_c^T A_c^* A_c \Pi_c = R_c^* R_c$. Then, assuming full rank of A , **xPOCON** is used to estimate $\|(R_c^* R_c)^{-1}\|_1 (= \|(A_c^* A_c)^{-1}\|_1)$, and finally we deploy the relations

$$n^{-1/4} \sqrt{\|(A_c^* A_c)^{-1}\|_1} \leq \|A_c^\dagger\|_2 \leq n^{1/4} \sqrt{\|(A_c^* A_c)^{-1}\|_1},$$

that can be easily proved using the properties of the matrix norms. Since $1 \leq \|A_c\|_2 \leq \sqrt{n}$, $\kappa_2(A_c) \equiv \|A_c\|_2 \|A_c^\dagger\|_2 \leq n^{3/4} \sqrt{\|(A_c^* A_c)^{-1}\|_1}$. Then, the error bounds for the computed singular values and vectors are derived as explained in [6]. One should be careful when the estimated scaled condition number exceeds the inverse of the machine precision, because in such cases it might be a considerable underestimate of the true value.

2.3.2 Using left singular vectors in factored form

The option to leave the left singular vectors in factored form is useful in the case of tall and skinny A ($m \gg n$) that is e.g. the coefficient matrix in a least square problem $\|Ax - b\|_2 \rightarrow \min$, solved by SVD. The unique minimum norm optimal x is given as $x = A^\dagger b \equiv V \Sigma^\dagger U^* b$. Note that in that case the computation of $U^* b$ reads (see e.g. line 8 in Algorithm 1; similarly with the data given in line 11)

$$b \leftarrow \begin{pmatrix} \hat{U}^* & 0 \\ 0 & I_{m-\rho} \end{pmatrix} (Q^* (\Pi_r b))$$

and it is easily implemented using `CLASWP`, `CUNMQR` and `CGEMM`. Of course, if one wants to form U explicitly, the matrix \hat{U} should be placed in the leading block of the array U that is initialized as n columns of the identity I_m , and U should be pre-multiplied by Q using `CUNMQR`. For instance, the first n left singular vectors are recovered by the following code:

```
CALL CUNMQR( 'Left', 'No_Tr', m, n, n, A, LDA, CWORK, U, LDU, &
CWORK(N+1), LCWORK-N, IERR )
IF ( ROWPRM ) CALL CLASWP( N1, U, LDU, 1, M-1, IWORK(N+1), -1 )
```

The details can be seen in the source code of `xGESVDQ`.

2.3.3 Linker information

`xGESVDQ` subroutines are built on top of BLAS and LAPACK computational and driver subroutines.

2.3.4 Updates and maintenance

`xGESVDQ` subroutines are designed in the LAPACK environment both as ready to use numerically sound software and as a developer's framework for accurate SVD computations. The two main ingredients are `xGEQP3` and `xGESVD`, and any future improvements and updates of these two subroutines are automatically incorporated in `xGESVDQ`.

2.3.5 Modifications and further research and development

The Businger-Golub column pivoting can be replaced with simple pre-sorting of the columns of A with respect to their Euclidean lengths. This is implemented in the module `xGESVDQS` with simple replacement of the rank revealing QR factorization

```
CALL xGEQP3( M, N, A, LDA, IWORK, WORK, WORK(N+1), LWORK-N, IERR )
```

with initial column sorting (encoded in the permutation vector `IWORK`) followed by the ordinary QR factorization,

```
CALL xLAPMT( .TRUE., M, N, A, LDA, IWORK )
CALL xGEQRF( M, N, A, LDA, WORK, WORK(N+1), LWORK-N, IERR )
```

However since this static pre-sorting is not rank revealing, the options in **JOBA** are reduced to computation without ('H', without any intervention in the upper tringular factor R) or with scaled condition number estimation ('E') in the case of full rank R . For details see [6]. The module `xGESVDQS` is not included in this distribution, and it is available, upon request, from the author.

Another option is to use the windowed pivoting `xGEQPX`, `xGEQPY` from [3], [2]. We omit the details fo the sake of brevity, and refer to [6].

Further, one could replace e.g. `xGESVD` with some other SVD algorithm such as `xGESDD`, but in that case we do not expect as dramatic improvement in accuracy because the bidiagonal divide an conquer algorithm in `xGESDD` is not as accurate as the zero shift bidiagonal QR SVD. In other words, in this framework `xGESVD` can be turned into a highly accurate procedure that delivers both accurate output (when warranted bu the data) and a reliable error bound, and `xGESDD` cannot. It remains to be explored how this applies to the fast bidiagonal SVD [9].

Future research will include studying whether and to what extent the bidiagonalization can benefit (in the sense of enhanced accuracy) from the communication avoiding tournament pivoting [5].

3 xGESVDQ test drivers: user guide

3.1 Directories and files

In the **HOME** directory of the testing package, the program files are organized in the subdirectories as follows:

- **HOME**
 - **xGESVDQ**
 - **REFVAL**
 - **MATGEN**
 - **TESTING**
 - **AUXSBR**

3.1.1 xGESVDQ

xGESVDQ contains the source codes of the new SVD algorithm in 4 data types

- **SGESVDQ.f** (single precision)
- **DGESVDQ.f** (double precision)
- **CGESVDQ.f** (complex)
- **ZGESVDQ.f** (double complex)

These are the routine being tested. The routines are LAPACK-based and written in the LAPACK style. All four have been tested for compliance with FORTRAN standard, e.g. both passed, without errors or warnings, `g77 -c -fpedantic *.f`, `g77 -c -fpedantic -ff90 *.f`

For theoretical background and discussion of the numerical results, see [6].

3.1.2 REFVAL

REFVAL contains simple one-sided Jacobi SVD in double complex precision,

- **zgesvjx.f** (one sided double complex Jacobi SVD pure)

that is used for reference values in case of complex data.

For real test matrices (single and double precision), the reference values of the SVD are computed by double precision one sided Jacobi SVD `dgesvj()` from LAPACK.

3.1.3 MATGEN

MATGEN contains the generators of test matrices

- **dmgen.f** (double precision)
- **zmgen.f** (double complex)

See [6, §3.1] for more details on the kind of matrices generated by these subroutines.

3.1.4 TESTING

TESTING contains test drivers and makefiles

- **TestDGESVDQ.f** (interactive test for **SGESVDQ** and **DGESVDQ**; the user can choose different dimensions and options)

- `TestDGESVDQ_no_input.f` (automatic test for `SGESVDQ` and `DGESVDQ`; the user can review the hard-coded test parameters and it is only required to hit `enter` to start the test)
- `TestZGESVDQ.f` (interactive test for `CGESVDQ` and `ZGESVDQ`; the user can choose different dimensions and options)
- `TestZGESVDQ_no_input.f` (automatic test for `CGESVDQ` and `ZGESVDQ`; the user can review the hard-coded test parameters and it is only required to hit `enter` to start the test)
- `MakeTest` (make file for interactive test)
- `MakeTest_ni` (make file for automatic test with hard-coded parameters)

This directory also contains the Matlab script `TestSummary` that can be used to display the test results in several figures, see §3.4.1.

3.1.5 AUXSBR

AUXSBR contains auxiliary subroutines used by the test drivers:

- for real data tests:
 - `dchkor.f` (orthogonality check for singular vectors)
 - `dcmpsv.f` (compares singular vectors with reference values)
 - `dresvd.f` (computed the residual norm for the SVD)
 - `xges2d.f` (single to double precision matrix copy)
 - `xged2s.f` (double to single precision matrix copy)
- for complex data tests:
 - `zchkun.f` (orthogonality check for singular vectors)
 - `zcmpsv.f` (compares singular vectors with reference values)
 - `zresvd.f` (computed the residual norm for the SVD)
 - `xgec2z.f` (single complex to double complex precision matrix copy)
 - `xgez2c.f` (double complex to single complex precision matrix copy)

3.2 Making the test driver executables

3.2.1 Linker information

In addition to the files listed in §3.1, the linker needs the BLAS and LAPACK libraries. In the corresponding makefiles (`MakeTest`, `MakeTest_ni`) the user should adjust the home page for the directories described in §3.1 and linker information for BLAS and LAPACK. In the example files provided in this package one can use sequential or threaded Intel MKL libraries.

- interactive test drivers
 - `make -f MakeTest rtestt` (real data, threaded)
 - `make -f MakeTest rtests` (real data, sequential)
 - `make -f MakeTest ctestt` (complex data, threaded)
 - `make -f MakeTest ctests` (complex data, sequential)
- non-interactive test drivers; no input parameters
 - `make -f MakeTest_ni rtestt_ni` (real data, threaded)
 - `make -f MakeTest_ni rtests_ni` (real data, sequential)
 - `make -f MakeTest_ni ctestt_ni` (complex data, threaded)
 - `make -f MakeTest_ni ctests_ni` (complex data, sequential)

3.3 Using the test drivers

The test drivers can be used as interactive and automatic, with hard coded parameters. In either case, the run time can be considerable if the matrices are of larger dimensions. The total number of test cases in a single run is larger than 30000 and it can be larger than 60000. The purpose of the test is to determine to what extent the measured errors comply with the errors predicted by the perturbation theory.

3.3.1 Detailed interactive test program

In a simple interactive menu, the test starts with choosing one of the following jobs

1. SIGMA only
2. SIGMA, $U(M \times N)$
3. SIGMA, $V(N \times N)$
4. SIGMA, $U(M \times N)$, $V(N \times N)$
5. SIGMA, $U(M \times M)$, $V(N \times N)$

6. SIGMA, U(MxM)
7. SIGMA, U(MxNR), V(NxNR), (here NR denotes the numerical rank)
8. SIGMA, V(NxNR)
9. SIGMA, U(NxNR)

Next, accuracy level is set to one of the following:

- (1) The requested accuracy corresponds to the backward error bounded by

$$\|\delta A\|_F \leq f(m,n) * EPS * \|A\|_F,$$
 where $EPS = \{S,D\}LAMCH(Epsilon)$. This authorises {C,Z}GESVDQ to truncate the computed triangular factor in the rank revealing QR factorization whenever the truncated part is below the threshold $EPS * \|A\|_F$.
- (2) Similarly as with (1), but the truncation is more gentle: it is allowed only when there is a drop on the diagonal of the triangular factor.
 This is a medium level of accuracy.
- (3) High accuracy requested. No numerical rank determination based on the rank revealing QR factorization is attempted.
- (4) Accuracy level same as in (3), and in addition an estimate of the scaled cond. number is computed.

Next, one can choose between only column or complete row and column pivoting.

Select the pivoting in the initial QR factorization:

- (1) Column-pivoted QR factorization as preconditioner.
- (2) Row sorting prior to column pivoted QRF.
 (This may give more accurate results if the rows of A vary in length due to both very large and very small weighting factors, causing large $s_cond(A)$).

The computation of the SVD of A proceeds with the SVD of its computed triangular factor R. xGESVDQ can work with both R and R^* .

After the QR factorization, proceed with computing:

- (1) The SVD of the upper triangular factor R
 (This is recommended when using xGESVDQ.)
- (2) The SVD of the conjugate-transposed R^*H
 (This involves more data movement and it is left as optional for experimenting in the R&D.)

Finally, the row dimension m and the column dimension n are entered, and the number of test cases (1 or 2) per class of test matrices is set.

3.3.2 Example driver, no input required

In this case, the options described in §3.3.1 are hard-coded; the user can review them and hit `enter` to start the test.

3.4 Visualization of the test results - one graph is worth thousands of numbers

For the selected options, the test driver runs 32448 or 64896 test cases. The results of testing `xGESVDQ` are stored in the files

- `SGESVDQ.table` and `DGESVDQ.table` in the case of real data
- `CGESVDQ.table` and `ZGESVDQ.table` in the case of complex data.

Each row in those files corresponds to 14 measurements related to the computed SVD of one test matrix. The collected test data are:

1. The scaled condition number, computed by scaling the columns to unit norm and computing the singular values in double (complex). precision.
2. Estimated condition number, if that options is selected (otherwise, set to -1).
3. The residual $\|A - U\Sigma V^*\|_F / \|A\|_F$ computed in double (complex) precision, if singular values and all singular vectors are computed (otherwise set to -1).
4. $\max_i \frac{|\sigma_i - \hat{\sigma}_i|}{\hat{\sigma}_i}$, where σ_i is the i th computed singular value, and $\hat{\sigma}_i$ is the reference value.
5. $\max_{i \neq j} |u_i^* u_j|$, where u_i denotes the computed i th left singular vector. If the left singular vectors are not computed, this value is set to -1 .
6. $\min_i \|u_i\|_2$. If the left singular vectors are not computed, this value is set to -1 .
7. $\max_i \|u_i\|_2$. If the left singular vectors are not computed, this value is set to -1 .
8. $\max_{i \neq j} |v_i^* v_j|$, where v_i denotes the computed i th right singular vector. If the right singular vectors are not computed, this value is set to -1 .
9. $\min_i \|v_i\|_2$. If the right singular vectors are not computed, this value is set to -1 .

10. $\max_i \|v_i\|_2$. If the right singular vectors are not computed, this value is set to -1 .
11. $\max_i \{\|v_i - \hat{v}_i(\hat{v}_i^* v_i)\|_2 \cdot r_{gap_i}\}$, where \hat{v}_i is the reference value for v_i . If the right singular vectors are not computed, this value is set to -1 .
12. $\max_i \{\|u_i - \hat{u}_i(\hat{u}_i^* u_i)\|_2 \cdot r_{gap_i}\}$, where \hat{u}_i is the reference value for u_i . If the left singular vectors are not computed, this value is set to -1 .
13. $\max_i \frac{|\sigma_i - \hat{\sigma}_i|}{\hat{\sigma}_1}$, where σ_i is the i th computed singular value, and $\hat{\sigma}_i$ is the reference value.
14. The computed numerical rank.

The test results stored in the files are best analyzed visually. Upon completion of the test, the user can start the Matlab script **TestSummary**. The "genes" of the test matrices (parameters to restore each particular test matrix in the test) are stored in the file **SJEME.all**. This allows additional separate study of examples found interesting during the analysis of the results.

3.4.1 An example

The following eight figures illustrate how **TestSummary** displays the test results. The test is performed using *Intel® Fortran Composer 2015.4.221* with multithreaded *Intel® Math Kernel Library 10.3* on an Intel® Core (TM) 2 Duo CPU T6670 @ 2.20GHz 2.20 GHz based Laptop with 8GB RAM, running under MS Windows 7.

References

- [1] E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, Jack J. Dongarra, J. Du Croz, S. Hammarling, A. Greenbaum, A. McKenney, and D. Sorensen. *LAPACK Users' Guide (Third Ed.)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1999.
- [2] C. H. Bischof and G. Quintana-Orti. Algorithm 782: codes for rank-revealing QR factorizations of dense matrices. *ACM Transactions on Mathematical Software*, 24(2):254–257, 1998.
- [3] C. H. Bischof and G. Quintana-Orti. Computing rank-revealing QR factorizations of dense matrices. *ACM Transactions on Mathematical Software*, 24(2):226–253, 1998.
- [4] J. Demmel and K. Veselić. Jacobi's method is more accurate than QR. *SIAM J. Matrix Anal. Appl.*, 13(4):1204–1245, 1992.

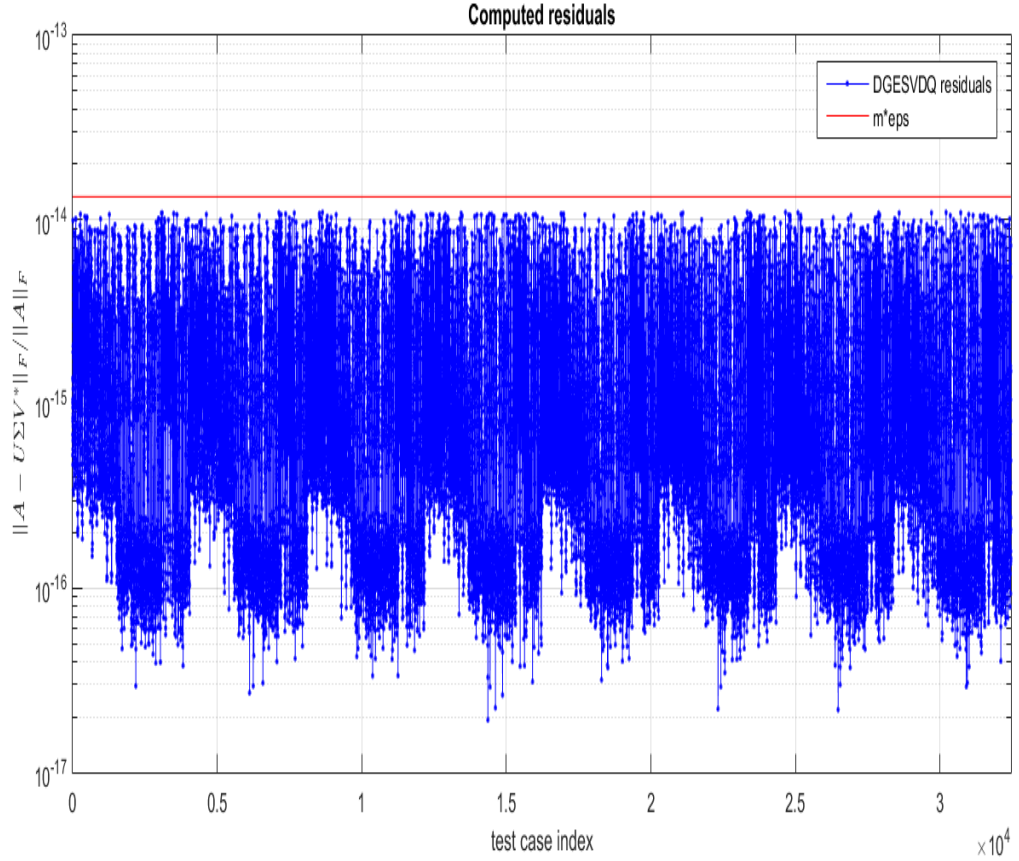


Figure 1: The residuals $\|A - U\Sigma V^*\|_F / \|A\|_F$.

- [5] James Demmel, Laura Grigori, Ming Gu, and Hua Xiang. Communication avoiding rank revealing QR factorization with column pivoting. *SIAM J. Matrix Analysis Applications*, 36(1):55–89, 2015.
- [6] Z. Drmač. A QR-preconditioned QR SVD method for computing the SVD with high accuracy. *ACM Trans. Math. Softw.*, pages ??–??, 2016.
- [7] Z. Drmač and K. Veselić. New fast and accurate Jacobi SVD algorithm: I. *SIAM J. Matrix Anal. Appl.*, 29(4):1322–1342, 2008.
- [8] Z. Drmač and K. Veselić. New fast and accurate Jacobi SVD algorithm: II. *SIAM J. Matrix Anal. Appl.*, 29(4):1343–1362, 2008.
- [9] Benedikt Groß and Bruno Lang. An $O(n^2)$ algorithm for the bidiagonal SVD. *Linear Algebra and its Applications*, 358(13):45 – 70, 2003.

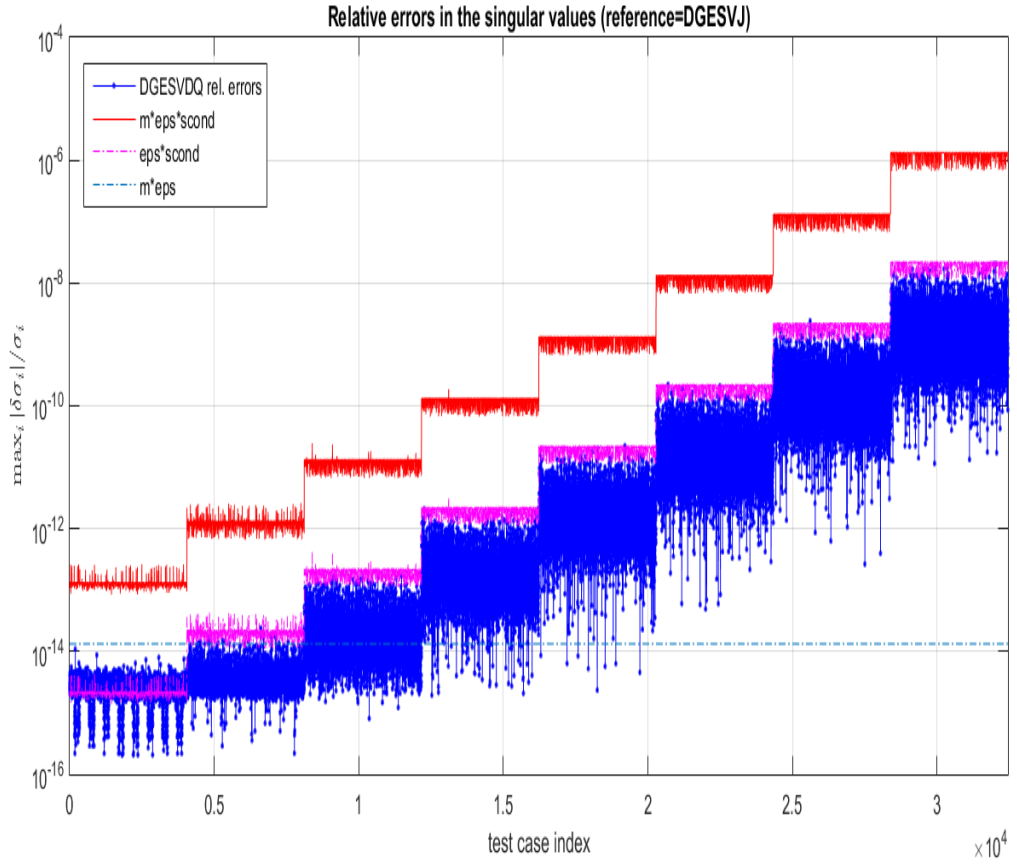


Figure 2: Relative errors $\max_i |\delta\sigma_i|/\sigma_i$.

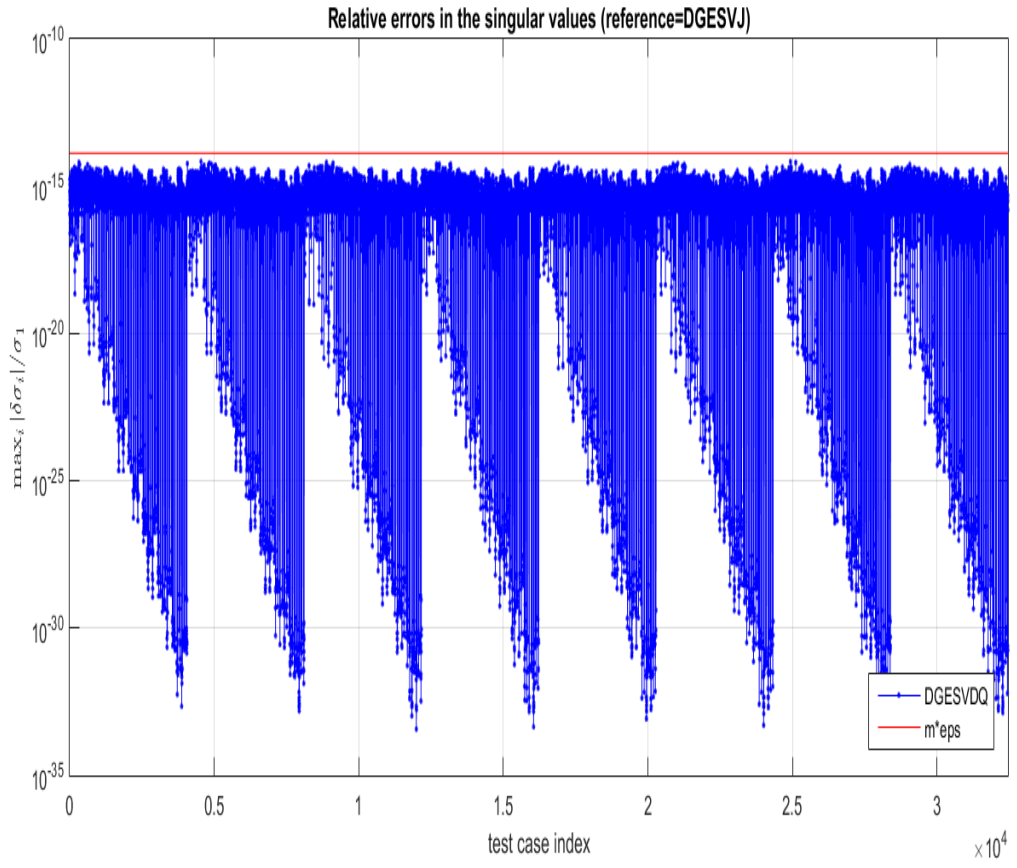


Figure 3: Relative errors $\max_i |\delta \sigma_i| / \|A\|_2$.

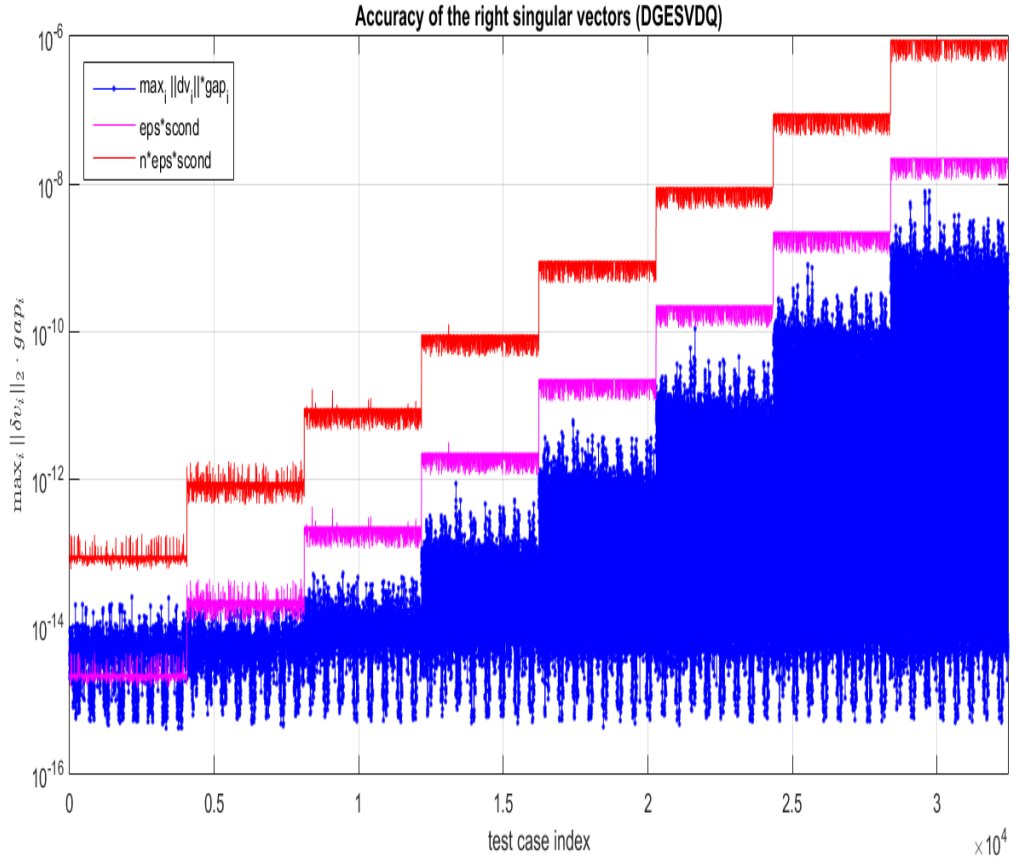


Figure 4: Accuracy of the computed right singular vectors, compared with the bounds obtained through the perturbation theory.

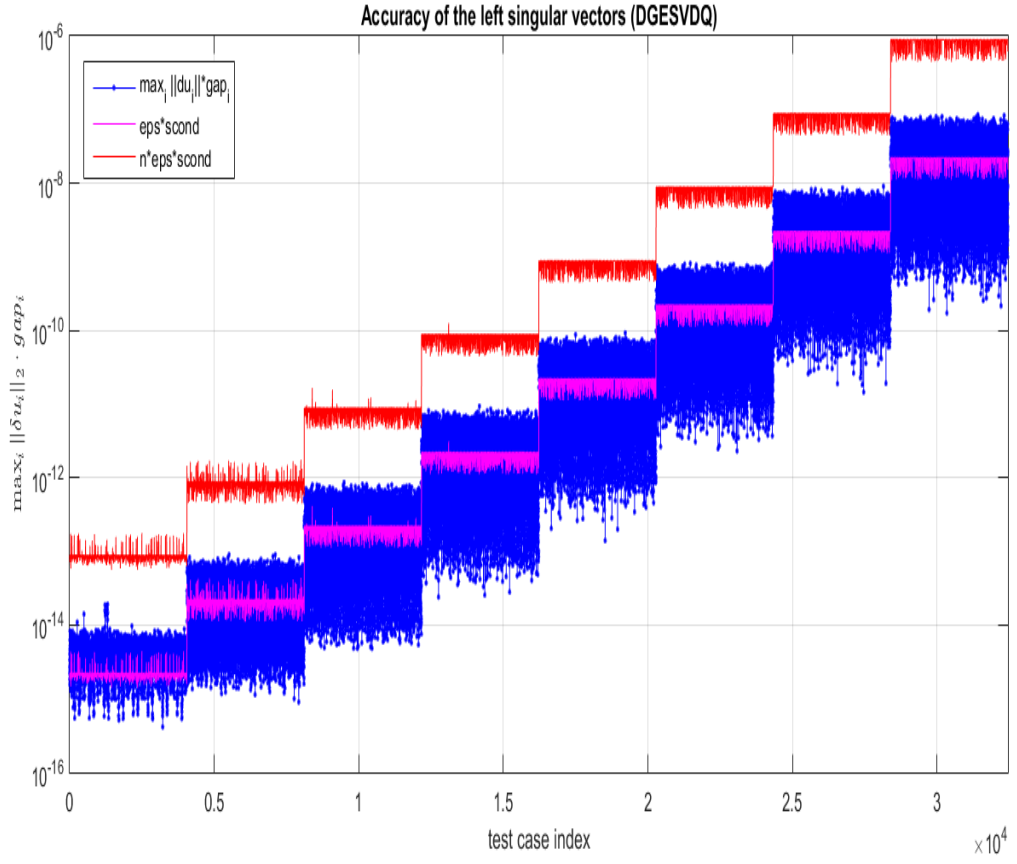


Figure 5: Accuracy of the computed left singular vectors, compared with the bounds obtained through the perturbation theory.

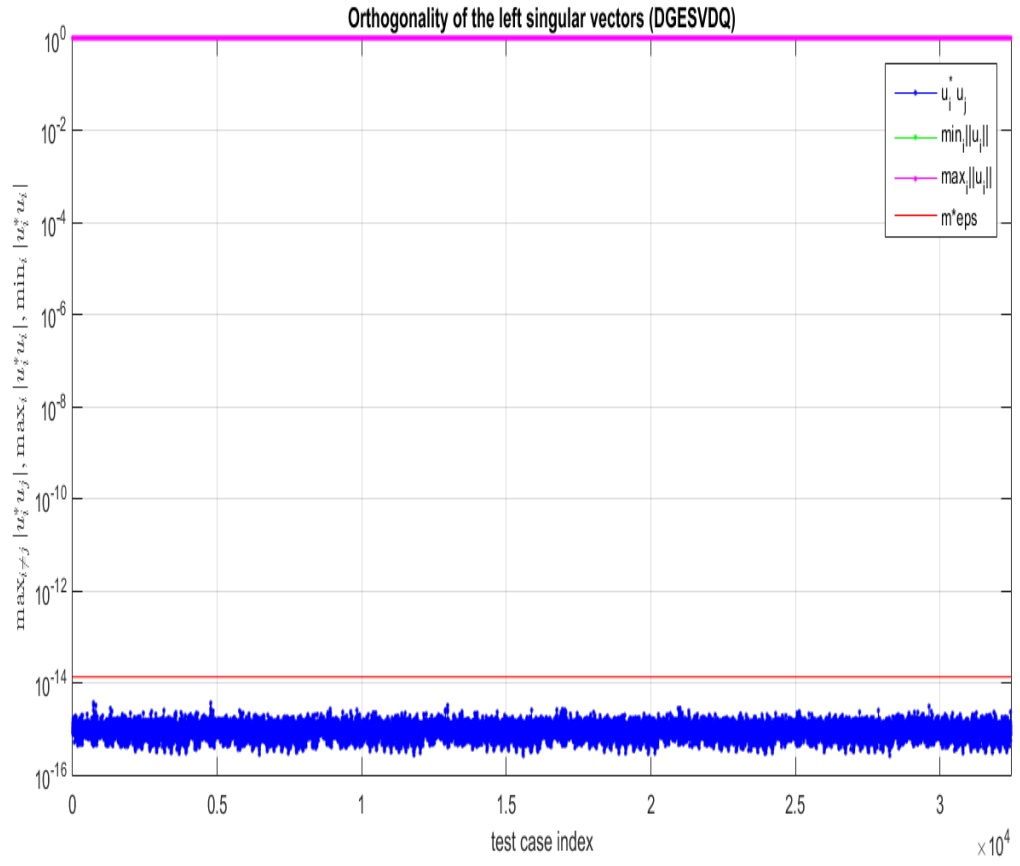


Figure 6: Orthogonality of the left singular vectors.

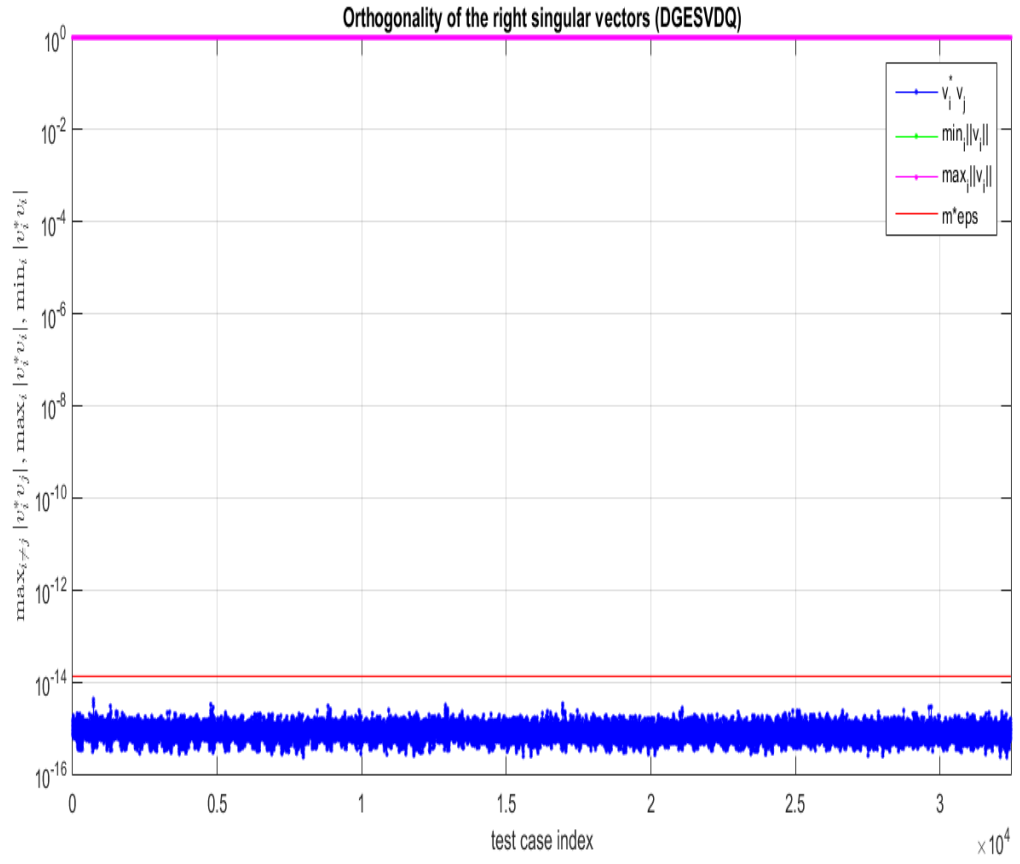


Figure 7: Orthogonality of the right singular vectors.

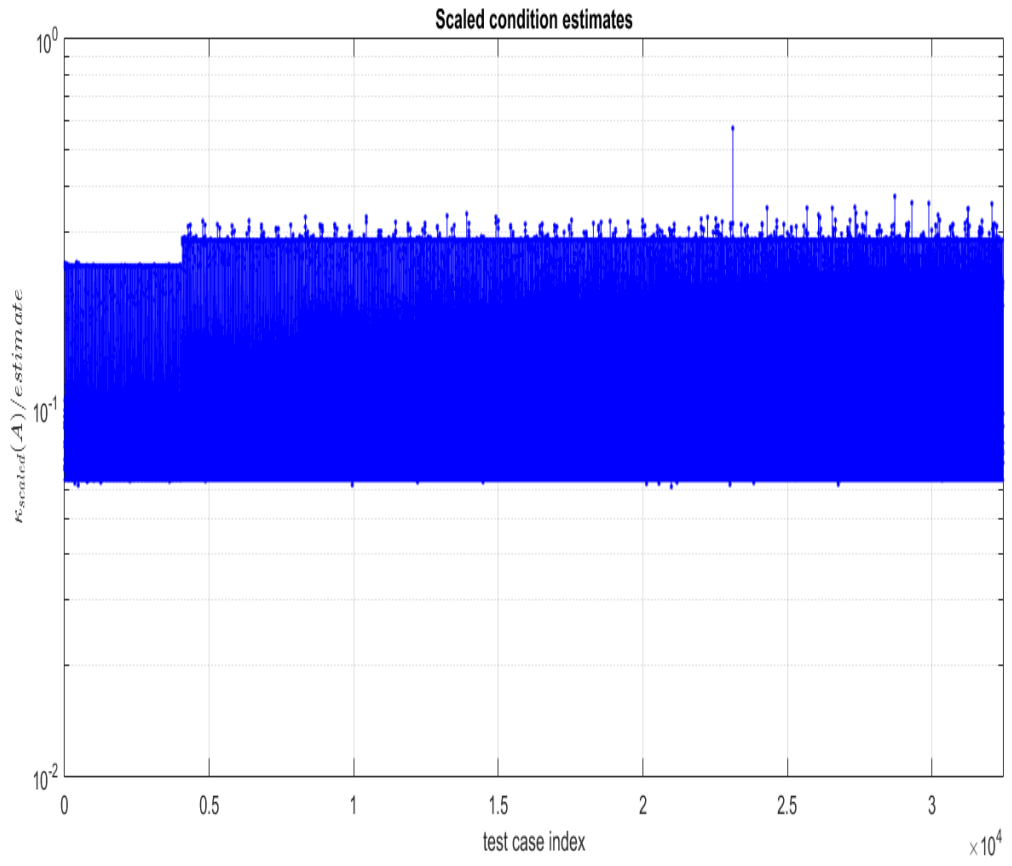


Figure 8: The quotient of the true (i.e. computed using SVD) and the estimated scaled condition number.