Computing the CSD and GSVD

Jenny Wang

September 29, 2004

1 Introduction

The notion of generalized singular value decomposition (GSVD) was first introduced in [15] by Von Loan and has become one of the essential matrix decompositions in the recent decades [2, 8, 9, 16]. The primary objective of this paper is to describe in detail a reliable method of computing the GSVD. The back bone of this method employs Von Loan's algorithm [14] for computing the cosine-sine decomposition (CSD). This method is rather stable because each modular decomposition is numerically stable, and most matrices that subject to multiplications are orthogonal.

Notations in this paper follows from the convention used in standard linear algebra literatures such as [11, 5]. Whenever necessary, Matlab-style notation is also used for indexing, such as $A(i : j \ , \ i : j)$ denotes the submatrix of A taken from its i-th to j-th rows and columns. All norms used in the text are 2-norms unless explicitly specified. The paper can be expanded as the project progresses, and is thusfar laid out as follows: section 2 establishes the formal definitions of CSD and GSVD; section 3 presents the mathematical and practical details about the modular decompositions which the GSVD algorithm relies on; section 4 explains Von Loan's CSD algorithm; section 5 describes the algorithm of the GSVD; section 6 discusses some implementation details about the GSVD and other associated algorithms.

2 Formal Definitions of CSD and GSVD

Definition 1.1 (CSD): Given two matrices $Q_1 \in \mathbb{R}^{m\times n}$ and $Q_2 \in \mathbb{R}^{p\times n}$ such that

$$\begin{array}{cc} n & \\ \left[\begin{array}{c} Q_1 \\ Q_2 \end{array}\right] & \begin{array}{c} m \\ p \end{array} \end{array} \tag{1}$$

has orthonormal columns. a *cosine-sine decomposition (CSD)* of Q_1 and Q_2 is the joint factorization

$$Q_1 = UCZ^T \text{ and } Q_2 = VSZ^T, \tag{2}$$

where

$U \in \mathbb{R}^{m\times m}$, $V \in \mathbb{R}^{p\times p}$ and $Z \in \mathbb{R}^{n\times n}$ are orthogonal matrices;

$C \in \mathbb{R}^{m\times n}$ and $S \in \mathbb{R}^{p\times n}$ are blocked diagonal (not necessarily square) matrices satisfying

$$C^TC + S^TS = I. \tag{3}$$

Specifically, C and S are of one of the following structures.

1. if $m \geq n$ and $p \geq n$,

$$C = \begin{array}{c} \scriptstyle n \\ \left[\begin{array}{c} \Sigma_1 \\ O \end{array}\right] \end{array} \begin{array}{l} \scriptstyle n \\ \scriptstyle m-n \end{array}, \quad S = \begin{array}{c} \scriptstyle n \\ \left[\begin{array}{c} \Sigma_2 \\ O \end{array}\right] \end{array} \begin{array}{l} \scriptstyle n \\ \scriptstyle p-n \end{array}. \tag{4}$$

2. if $m \geq n$ and $p < n$,

$$C = \begin{array}{c} \begin{array}{cc} \scriptstyle n-p & \scriptstyle p \end{array} \\ \left[\begin{array}{cc} I & O \\ O & \Sigma_1 \\ O & O \end{array}\right] \end{array} \begin{array}{l} \scriptstyle n-p \\ \scriptstyle p \\ \scriptstyle m-n \end{array}, \quad S = \begin{array}{c} \begin{array}{cc} \scriptstyle n-p & \scriptstyle p \end{array} \\ \left[\begin{array}{cc} O & \Sigma_2 \end{array}\right] \end{array} \scriptstyle p . \tag{5}$$

3. if $m < n$ and $p \geq n$,

$$C = \begin{array}{c} \begin{array}{cc} \scriptstyle m & \scriptstyle m-n \end{array} \\ \left[\begin{array}{cc} \Sigma_1 & 0 \end{array}\right] \end{array} \scriptstyle m , \quad S = \begin{array}{c} \begin{array}{cc} \scriptstyle m & \scriptstyle m-n \end{array} \\ \left[\begin{array}{cc} \Sigma_2 & O \\ O & I \\ O & O \end{array}\right] \end{array} \begin{array}{l} \scriptstyle m \\ \scriptstyle n-m \\ \scriptstyle p-n \end{array}. \tag{6}$$

4. if $m < n$ and $p < n$, and let $t = m + p - n$ and

$$C = \begin{array}{c} \begin{array}{ccc} \scriptstyle n-p & \scriptstyle t & \scriptstyle n-m \end{array} \\ \left[\begin{array}{ccc} I & O & O \\ O & \Sigma_1 & O \end{array}\right] \end{array} \begin{array}{l} \scriptstyle n-p \\ \scriptstyle t \end{array}, \quad S = \begin{array}{c} \begin{array}{ccc} \scriptstyle n-p & \scriptstyle t & \scriptstyle n-m \end{array} \\ \left[\begin{array}{ccc} O & \Sigma_2 & O \\ O & O & I \end{array}\right] \end{array} \begin{array}{l} \scriptstyle t \\ \scriptstyle n-m \end{array}. \tag{7}$$

Σ_1 and Σ_2 are k-by-k diagonal matrices whose entries are nonnegative, and are called the *cosine and sine values pairs* (hence the name CS decomposition). Σ_1 and Σ_2 are in opposite sequence so that constraint (3) is equivalent to

$$\Sigma_1^2 + \Sigma_2^2 = 1 \tag{8}$$

It should be noted that permuting the blocks of C and S by rows or columns does not change the meaning of the decomposition. Take the structure in (7) for example, exchanging the first two columns of both C and S then switching the rows in S yields the same formulation as the last form in in [2] (pp. 4); namely (again, $t = m + p - n$):

$$C = \begin{array}{cc} \begin{array}{ccc} t & n-p & n-m \end{array} & \\ \left[\begin{array}{ccc} \Sigma_1 & O & O \\ O & I & O \end{array}\right] & \begin{array}{c} t \\ n-p \end{array} \end{array}, \quad S = \begin{array}{cc} \begin{array}{ccc} t & n-p & n-m \end{array} & \\ \left[\begin{array}{ccc} \Sigma_2 & O & O \\ O & O & I \end{array}\right] & \begin{array}{c} t \\ n-m \end{array} \end{array}.$$

Definition 1.2 (GSVD): Given two matrices $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{p\times n}$. Let $r = \text{rank}([A^T, B^T]^T) \leq n$. A *generalized singular value decomposition (GSVD)* of A and B is the joint factorization

$$A = UCRZ^T \text{ and } B = VCRZ^T \tag{9}$$

where

$U \in \mathbb{R}^{m\times m}$, $V \in \mathbb{R}^{p\times p}$ and $Z \in \mathbb{R}^{n\times n}$ are orthogonal matrices just like they are in the CSD,

$C \in \mathbb{R}^{m\times r}, S \in \mathbb{R}^{p\times r}$. They satisfy (3) and other properties as they are in the CSD. Of course, they also follow structures (4)-(7).

$R \in \mathbb{R}^{r\times n}$ consists of a block of non-singular triangular matrix. That is, $R = [0, R_{11}]$ or $R = [R_{11}, 0]$ where R_{11} is an r-by-r non-singular triangular matrix.

The r diagonal entries in Σ_1 and Σ_2 of C and S are called the *generalized singular value pairs*, or sometimes it is preferable to define *the generalized singular values*, σ_i, of the pair A, B as follows:

$$\sigma_i = \begin{cases} \alpha_i/\beta_i & \text{for } \alpha_i \in diag(C^TC),\ \beta_i \in diag(S^TS) \text{ and } \beta_i \neq 0 \\ \text{inifinite} & \text{for } \beta_i = 0. \end{cases}$$

C and S are ordered the same way as they are in the cosine-sine decomposition. In fact, if $[A^T, B^T]^T$ has orthonormal columns, then its GSVD and CSD are identical (i.e. $r = n$ and $R = I$). Thereby the CSD can be viewed as a special case of the GSVD.

If B is square and non-singular, then $m = n = r$ and the GSVD of A and B gives the full SVD of AB^{-1}: $U^T(AB^{-1})V = CS^{-1}$, and the generalized singular values of the pair A, B are equal to the singular values of AB^{-1}. This further implies that if $B = I$, then GSVD of A and B is just the full SVD of A.

The connection between the GSVD and *generalized eigen value decomposition* (GEVD) is analogous to that of the SVD and *eigen value decomposition* (EVD): The SVD of A solves the EVD problem while the GSVD of A and B gives the solution to the GEVD problem.

3 Preliminaries

Von Loan in [14] proposed an efficient method of computing the CSD, which can be employed to compute the GSVD and is presented in detail in the next section. Besides it, a QR, RQ decompositions and a rank-revealing algorithm are necessary in order to obtain the GSVD of the form in (9). This section is devoted to describe these decompositions. For the rest of this section, let $A \in \mathbb{R}^{m\times n}$ for any arbitrary m and n unless explicitly noted otherwise.

1. QR decomposition

A can be decomposed in the form $A = QR$ where $Q \in \mathbb{R}^{m\times m}$ has orthogonal columns ($Q^TQ = I$), and $R \in \mathbb{R}^{m\times n}$ is upper triangular. Since there is no assumption on the sparsity of the input matrix A, the decomposition is computed by applying a sequence of Householder Reflectors. Note that we do not rely on this decomposition to reveal the ranking information, so no pivoting is needed. This decomposition is handedly obtainable from Matlab's built-in routine [7].

2. RQ decomposition

A can be decomposed in the form $A = RQ$ where $R \in \mathbb{R}^{m\times n}$, and $Q \in \mathbb{R}^{n\times n}$. Like QR decomposition, RQ decomposition is also computed via the way of Householder's orthogonal triangularization; unlike QR, this routine is generally not available in today's Matlab package. Fortunately, its implementation is rather similar with the QR decomposition; the only difference is that householder reflector is now constructed to annihilate a row a time, working from last row up. The implementation of RQ is thus straightforward and can be summarized as follows:

ALGORITHM 1. (Computing the RQ Decomposition)
Input: A
Output: R, Q

```
0. initialize $r = \min(m,n)$; $s = r; R = A$
1. for $k = m : -1 : (t+2)$      % where $t = m - r$
2.     $x = R(k, 1:s)$
```

```
3.     $v = x$
4.     $v_s = sign(x_s)\|x\|$
5.     $v_s = v_s + x_s$
6.     $v = v/\|v\|$
7.     $R(1:k-1, 1:s) = R(1:k-1, 1:s) - 2*(R(1:k-1, 1:s)*v^T)*v$
8.     $W(s-1, 1:s) = v$      % store the Householder Reflectors
9.     for $j = k-t:r$            % update Q one row at a time.
10.        $v = W(j-1, 1:j)$
11.        $Q(s, 1:j) = Q(s, 1:j) - 2*Q(s, 1:j)*v^T*v$
12.    end
13.    $s = s-1$
14. end
15. for $j = k-t:r$            % update the last row of Q
16.     $v = W(j-1, 1:j)$
17.     $Q(s, 1:j) = Q(s, 1:j) - 2*Q(s, 1:j)*v^T*v$
18. end
```

3. Rank-Revealing Decomposition

The rank-revealing feature is critical to our algorithm for the GSVD because it determines the size of the partitioned orthonormal matrix Q passed into CSD, which subsequently effects the accuracy of the entire GSVD computation. Of course, a standard QR decomposition (without pivoting) would be sufficient if G has full rank. Whereas a more robust rank-revealing algorithm may be necessary otherwise. Unfortunately, determining the numerical rank of a matrix requires a rank-revealing algorithm per se. Hence there is no need to use different algorithms whether or not G has full rank.

Although there no formal proof yet exists, it has been conjectured that the rank-revealing feature can only be depicted through two-sided decompositions, with examples includes the pivoting QR, SVD and UTV decompositions. There are two favors of UTV decomposition: URV and ULV. The former is defined as follows.

Definition 1.3 (URV): Given a matrix $A \in \mathbb{R}^{m\times n}$. Without lost of generality, Let $m \geq n$, $k = \text{rank}(A) \leq n$ and σ_i denote the i-th singular value of A. The *URV decomposition* of A is the factorization

$$A = URV^T \tag{10}$$

where

$U \in \mathbb{R}^{m\times n}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal,

R is upper triangular and $\text{rank}(R) = \text{rank}(A) = k$. Moreover, R can be partitioned into

$$R = \begin{array}{cc} & \begin{array}{cc} k & n-k \end{array} \\ & \left[\begin{array}{cc} R_{11} & R_{12} \\ O & R_{22} \end{array}\right] \begin{array}{l} k \\ n-k \end{array} \end{array} , \tag{11}$$

where

R_{11} is non-singular,

$\|R_{12}\| = \mathcal{O}(\epsilon)$ (ϵ is machine epsilon),

$\|R_{22}\| \simeq \sigma(k+1)$.

The other type of UTV decomposition, *ULV decomposition*, differs from the one defined above in that the middle matrix is lower triangular (hence ULV) whose (2,1)-block is $\mathcal{O}(\epsilon)$.

Theoretical and experimental results in [4] and [11] showed that URV decomposition gives an efficient and reliable way of detecting the gap in the singular value spectrum at the predetermined threshold (could be the order of machine epsilon or user-specified). UTV decomposition is cheaper to compute than the SVD and the resulted middle matrix is most resemble to a diagonal form.

Some numerical methods for UTV decomposition are proposed. The first such numerical algorithm was presented by Stewart in 1982. Next, Hansen and Fierro modified Stewart's method in order to optimize for sparse or low-rank matrices and their method determines the ranking information more speedily [4]. Most recently, Li and Zheng devised another method which eases the task of updating and downdating [6]. Rank-revealing algorithms is important and practical in its own right, and the topic remains an active research.

For our purpose, Stewart's method for computing the URV decomposition is chosen since it works well with input matrices that are generally dense and of moderate sized. Both the implementation for URV and ULV decompositions are rather similar, thus the explanation for the latter is omitted here. For extensive detail about Stewart's URV and ULV algorithm, readers are referred to [13] and [12].

The URV algorithm about to describe works on rectangular input matrix, A ($m \geq n$). If $m < n$, then the ULV algorithm is invoked on A^T and taking the transpose of each resulted matrix would yield the desired form for the URV of A.

Besides the m-by-n matrix A, the input of the algorithm may optionally include a rank decision tolerance τ and an upper bound value δ on the

2-norm of the R_{12} block relative to the Frobenius-norm of R in (11).

Stewart's Method starts out by computing the reduced orthogonal triangular factorization of A, obtaining a m-by-n orthogonal matrix U and a n-by-n upper triangular matrix R. Letting V be the n-by-n identity matrix, URV^T can be considered as the initial URV factorization. It follows by initializing $k = n$ and estimating the smallest singular value of A, $\hat{\sigma}_k$, and the corresponding left singular value $\hat{w}_k$. If $\hat{\sigma}_k > \tau$, U, R, V are returned and the algorithm exits. Otherwise, the algorithm proceeds with the revealment, refinement and deflation steps described below.

In the revealment step, a Given's rotation is used to determine an orthogonal matrix P_k, which pre-multiplying it would annihilate the k-th column of $R(1:k, 1:k)$. However, doing so would introduce a non-zero element in the (2, 1)-position of $R(1:k, 1:k)$, so a second Given's rotation is again used to determine an orthogonal matrix Q_k, which post-multiplying it would restore the upper-triangularity of $R(1:k, 1:k)$. The intermediate orthogonal matrices P_k and Q_k are accumulated into U and V respectively if the final U and V are desired.

After the revealment step, one may reduce the norm of the upper off-diagonal block of R. As the proof presented in [4], doing so can improve the accuracy of the estimated singular subspaces, which stores in U and V. One way to achieve it, as in Stewart's algorithm, is by applying a sequence of right rotations to reduce the first $(k-1)$ elements in the last columns of $R(1:k, 1:k)$ to zero. It follows by applying a sequence of left rotations to restore the structures of $R(1:k, 1:k)$. Meanwhile, many other refinement techniques are possible and discussed further in [4].

Lastly, by updating $k = k - 1$, new $\hat{\sigma}_k$ and $\hat{w}_k$ can be estimated and thus the problem can been deflated. The process loops until $\hat{\sigma}_k$ is at least the tolerance τ, thereby the final k represents the numerical rank of A with respect to τ.

The algorithm for URV can be summarized as follows:

```
ALGORITHM 2. (Computing the URV)
Input: $A$, $[\,\tau,\, \delta\,]$
Output: $U, R, V$
1.    If $A$ has more columns than rows (underdetermined)
          Use ULV algorithm on $A^T$, obtaining $\hat{U}L\hat{V}^T = A$
          Set $U = \hat{V}$, $V = \hat{U}$ $R = L^T$. Done.
2.    Initialize $k = n$, $V = I$.
3.    Compute the reduced QR of $A$: $UR = A$.
4.    Condition estimation: let $\hat{\sigma}_k$ estimate $\sigma_{min}(R(1:k, 1:k))$,
            and let $\hat{w}_k$ estimate the corresponding left singular vector.
5.      If $\sigma_k > \tau$
            Done.
6.      Revealing: determine an orthogonal $P_k$ such that $P_k\hat{w}_k = [0, ..., 0, 1]^T$
7.          Update $R(1:k, 1:k) = P_k^T R(1:k, 1:k)$
8.          Update $R(1:k, 1:k) = R(1:k, 1:k)Q_k$, where $Q_k$ is an orthogonal
            matrix chosen to restore the upper-triangularity of the updated
            R(1:k, 1:k).
9.      (Optional) Refinement: while $\|R(1:k-1, k)\|_2 > \delta\|L\|_F$
10.         apply a sequence of right and left rotations to reduce $R(k, 1:k-1)$.
11.     Deflation: update $k = k - 1$
12.     Repeat step 4.
```

4 CSD Algorithm

This section describes a stable algorithm that computes the CSD of two matrices who can be combined to form a matrix with orthonormal columns. Output of the algorithm follows the structures in (4)-(7), depending on the size of the input matrices. It can be proven that CSD exists for any partitioned matrices with orthnormal columns. Although the resulted orthogonal matrices are not necessarily unique, neither do the ordering of the cosine sine value pairs, the values of the pairs are uniquely determined. Detail proof of the theorem can be found in [2, 14].

Before we proceed, let's define the notion of stability for the CSD algorithm. Recall that ϵ denote the machine precision and suppose the CSD algorithm outputs $\tilde{U}$, $\tilde{V}$,$\tilde{Z}$, $\tilde{C}$, $\tilde{S}$, which are the computed values of U, V, Z, C, S, respectively, as they are in (2). Furthermore, assume that "stacking up" the input Q_1, Q_2 forms a matrix that is orthnormal within ϵ, that is:

$$\|Q_1^T Q_1 + Q_2^T Q_2 - I_n\| \simeq \epsilon. \tag{12}$$

Then the algorithm for computing the CSD is *stable* if all of the following conditions are satisfied:

$$\|\tilde{U}^T\tilde{U} - I_m\| \simeq \epsilon, \qquad \|\tilde{V}^T\tilde{V} - I_p\| \simeq \epsilon, \qquad \|\tilde{Z}^T\tilde{Z} - I_n\| \simeq \epsilon, \tag{13}$$

$$\|\tilde{U}Q_1\tilde{Z} - \tilde{C}\| \simeq \epsilon\|Q_1\|, \qquad \|\tilde{V}Q_2\tilde{Z} - \tilde{S}\| \simeq \epsilon\|Q_2\|. \tag{14}$$

The method of computing the CSD exploits the property that a matrix can be safely diagonalized by QR decomposition if it is well-conditioned and its columns are nearly orthogonal [14]. Also, the method requires the setting of a threshold to separate the large and small singular values. The ad-hoc value of $1/\sqrt{2}$ has been chosen for this purpose because [14] claims that it can minimize the backward error bounds in (12)-(14). More theoretical and empirical results on choices of this threshold value is yet to be investigated.

Without lost of generality, assume $m \leq p$. Rationale for this assumption and way to handle the case when $m > p$ is given toward the end of this section. To compute the CSD of Q_1 and Q_2 in (1) we first compute the full SVD of Q_2 and arrange its singular value in non-decreasing order. That is, let $U^T Q_2 Z = S$, where $U \in \mathbb{R}^{p\times p}$, $Z \in \mathbb{R}^{n\times n}$, and

$$S = \left[\begin{array}{cc|c} \multicolumn{2}{c|}{\overbrace{\begin{matrix} s_1 & & \\ & \ddots & \\ & & s_{q2} \end{matrix}}^{q2}} & \overset{n-q2}{O} \\ \hline \multicolumn{2}{c|}{O} & \end{array}\right] \begin{array}{l} q2 \\ p-q2 \end{array}$$

where $q2 = \min(p, n)$, and $1 \geq s_1 \geq ... \geq s_k \geq 1/\sqrt{2} > s_{k+1} ... \geq s_{q2}$.

Next we reverse the order of the singular values in Q_2, and flip the columns of U and Z.

$$S = \left[\begin{array}{c|c} \overset{n-q2}{O} & \overbrace{\begin{matrix} s_1 & & \\ & \ddots & \\ & & s_{q2} \end{matrix}}^{q2} \\ \hline & O \end{array}\right] \begin{array}{l} q2 \\ p-q2 \end{array}$$

where $0 \leq s_1 \leq ... \leq s_k \leq 1/\sqrt{2} < s_{k+1} ... \leq s_{q2}$.

Reversing the singular value of Q_2 makes the columns norm of $T = Q_1 Z$ to range from large to small so that more entries in T becomes negligible.

Next, compute the full QR decomposition of $Q_1 Z$: $R = V^T(Q_1 Z)$, where

$$R = \begin{array}{c} \begin{array}{cccc} n-q2 & k & t & n-q1 \end{array} \\ \left[\begin{array}{ccc|c} I & O & O & \\ O & R_{22} & O & O \\ O & O & R_{33} & \\ \hline \multicolumn{3}{c|}{O} & \end{array}\right] \end{array} \begin{array}{l} q1 \\ m-q1 \end{array} \qquad t = q2 + q1 - n - k.$$

Since we started out with $Q_1^T Q_1 + Q_2^T Q_2 = I$, so $S^T S + R^T R = I$, implying columns of R is orthogonal and the first (n-q2) diagonal entries must be ones since it corresponds to the zeroes in singular values of Q_2. Furthermore, since values in S are sorting from small to large, the middle k blocks in T is well-conditioned, so it can be safely diagonalized into R_{22}. That is $R_{22} = diag(c_{n-q2+1}, ..., c_k)$ where $1 > c_{n-q2+1} \geq c_{n-q2+2} \geq ... \geq c_k$.

The last $n - q1$ columns (if $m < n$) or $m - q1$ rows (if $m > n$) of R are zeros because $V^T Q_1 Z = R$ can be theoretically viewed as the SVD of Q_1. Note that also implies that Q_2 has $(n - q1)$ singular values being ones if $m < n$.

Due to inevitable roundoff error, however, we cannot neglect the upper-diagonal in R_{33} because diagonal entries in R_{33} are small, so we need to compute the SVD of R_{33}: $U_r^T R_{33} Z_r = diag(c_{n-q2+k+1}, ..., c_{q1}) = C_r$.

Let $\hat{C} = \text{diag}(R_{22}, C_r)$ and $\hat{S}$ be the first $k + t$ singular values of Q_2, the sine and cosine value pairs of Q_1 and Q_2 are computed and can be written as (again, $t = q2 + q1 - n - k$):

$$C = \begin{array}{c} \begin{array}{ccc} \scriptstyle n-q2 & \scriptstyle k+t & \scriptstyle n-q1 \end{array} \\ \left[\begin{array}{cc|c} I & O & \multirow{2}{*}{O} \\ O & \hat{C} & \\ \hline \multicolumn{2}{c|}{O} & \end{array}\right] \end{array} \begin{array}{l} \\ \scriptstyle n-q2 \\ \scriptstyle k+t \\ \scriptstyle m-q1 \end{array} \tag{15}$$

$$S = \begin{array}{c} \begin{array}{ccc} \scriptstyle n-q2 & \scriptstyle k+t & \scriptstyle n-q1 \end{array} \\ \left[\begin{array}{c|cc} \multirow{2}{*}{O} & \hat{S} & O \\ & O & I \\ \hline & \multicolumn{2}{c}{O} \end{array}\right] \end{array} \begin{array}{l} \\ \scriptstyle k+t \\ \scriptstyle n-q1 \\ \scriptstyle p-q2 \end{array} \tag{16}$$

The dimensions of the matrices in the description of the algorithm above may look cumbersome at the fist glance because it is generalized to allow any arbitrary sizes of Q_1 and Q_2 as long as they can be "stacked up" to form a matrix with orthonormal columns. If one substitutes for the cases where (1) $q1 = q2 = n$, (2) $q1 = n$, $q2 = p$, (3) $q1 = m$, $q2 = n$, and (4) $q1 = m$, $q2 = p$, it is relatively easy to verify (neglect the submatrices whose dimension are small than one) that those the structures in (15)-(16) follow to the four cases in (4)-(7), respectively.

Now forming U and Z can be easily done by combing U with U_r and Z with Z_r.

$$U = U \begin{array}{c} \begin{array}{ccc} \scriptstyle n-q2+k & \scriptstyle t & \scriptstyle m-q1 \end{array} \\ \left[\begin{array}{ccc} I & & \\ & U_r & \\ & & I \end{array}\right] \end{array} \begin{array}{l} \\ \scriptstyle n-q2+k \\ \scriptstyle t \\ \scriptstyle m-q1 \end{array}, \quad Z = Z \begin{array}{c} \begin{array}{cc} \scriptstyle n-t & \scriptstyle t \end{array} \\ \left[\begin{array}{cc} I & \\ & Z_r \end{array}\right] \end{array} \begin{array}{l} \\ \scriptstyle n-t \\ \scriptstyle t \end{array}.$$

In order to obtain V correctly, a last QR decomposition must be performed. Right-multiply Z_r with the last t columns and rows of the non-zero blocks of S, and let the resulted matrix be W, then

$$D = \begin{array}{c} \\ \\ \\ \end{array}\begin{array}{ccc|c} & {\scriptstyle n-q2} & {\scriptstyle q2-t} & {\scriptstyle t} & \\ \left[\begin{array}{c} O \\ \\ \hline \\ \end{array}\right. & & & & \end{array}$$

$$D = \begin{array}{c@{}c} \begin{array}{ccc} {\scriptstyle n-q2} & {\scriptstyle q2-t} & {\scriptstyle t} \end{array} & \\ \left[\begin{array}{c|cc} O & diag(s_1, ..., s_{q2-t}) & \\ & & W \\ \hline & O & \end{array}\right] & \begin{array}{l} {\scriptstyle q2-t} \\ {\scriptstyle t} \\ {\scriptstyle p-q2} \end{array} \end{array}$$

Since $W^T W = I_t - diag(S_{q2-t+1}, ..., S_{q2})$, its smallest singular value is at least $1/\sqrt{2}$, thus it can be safely diagonalized by QR. Furthermore, the upper-triangular matrix of the resulted QR should just be the last t singular values of Q_2. Namely, the QR result of W is $V_w^T W =$diag$(s_{q2-t+1}, ..., s_{q2})$. Hence

$$V = V \begin{array}{c@{}c} \begin{array}{ccc} {\scriptstyle n-q1+k} & {\scriptstyle t} & {\scriptstyle p-q2} \end{array} & \\ \left[\begin{array}{ccc} I & & \\ & V_w & \\ & & I \end{array}\right] & \begin{array}{l} {\scriptstyle n-q1+k} \\ {\scriptstyle t} \\ {\scriptstyle p-q2} \end{array} \end{array}$$

Although the structures and ordering of C and S may vary depending on implementation, the cosine and sine values pairs should remain the same. On the other hand, the orthogonal matrices U, V and Z are not necessarily uniquely determined.

Assumption that $m \leq p$ is made in order to ease the index arithmetics and condition checks throughout the rest of the computation. To circumvent the case where $m > p$, one may simply swap Q_1 and Q_2, and the exact algorithm can be invoked to compute the CSD of $[Q_2^T, Q_1^T]^T$. Upon archiving the results U, V, Z, C, and S, be sure to flip each of these matrices by columns to obtain the correct results for the CSD of $[Q_1^T, Q_2^T]^T$.

The CSD algorithm is stable since the QR decomposition routine is invoked on well-condition matrices (whose smallest singular value is at least the prescribed threshold) so that the upper diagonal of the resulted upper triangular matrix is safely neglected. As the algorithm suggested, the value of this threshold determines the size of the subproblems. Reducing it may speed up the algorithm but may hinder the accuracy of the computed results.

The above algorithm can be summarized in the following:

ALGORITHM 3. (Computing the CSD)
Input: Q_1, Q_2
Output: U, V, Z, C, S

```
1.    Initialize index variables: $q1 = min(m, n), q2 = min(p, n)$.
```

```
2.    If $m > p$
          Swap $Q_1$ and $Q_2$, computed the CSD of $[Q_2^T, Q_1^T]^T$.
          Swap the the columns of all the resulted matrices
          and the values in $C$ and $S$. Done.
3.    Compute the SVD of $Q_2$: $U^T Q_2 Z = S$,
          let the singular values of $Q_2$ be $s_1, ..., s_{q2}$.
4.    Arrange the values of $S$ in non-decreasing order.
5.    Find $k$ such that $0 \leq s_1 \leq ... \leq s_k \leq 1/\sqrt{2} < s_{k+1} \leq ... \leq s_{q2} \leq 1$.
6.    Compute the QR decomposition of $Q_1 Z : R = V^T(Q_1 Z)$.
7.    Extract $t$ columns in $R$ starting from the $(n - q2 + k + 1)^{th}$ column,
          label it $R_{33}$, where $t = q2 + q1 - n - k$.
8.    Compute the SVD of $R_{33}$: $U_r^T R_{33} Z_r = diag(c_{n-q2+k+1}, ..., c_{q1}) = C_r$.
9.    $S = O_{n-q2} \cup$ diag$(s_1, ..., s_{q2})$
      $C = R(n - q2 + k : n - q2 + k) \cup C_r \cup O_{n-q1}$
10.   $U(:, q1 + 1 - t : q1) = U(:, q1 + 1 - t : q1) * U_r$
11.   $Z(: n + 1 - t : n) = Z(:, n + 1 - t : n) * Z_r$
12.   Let $\tilde{S}$ be the square submatrix taken from the last $t$ columns and rows of
          diag$(s_1, ..., s_{q2})$. Compute $W = \tilde{S} Z_r$
13.   Compute the QR of $W$: $V_w \tilde{S} = W$.
14.   $V(:, q2 + 1 - t : q2) = V(:, q2 + 1 - t : q2) * V_w$
```

5 GSVD

Having introduced all the modular decompositions, we are now ready to outline a complete algorithm for computing the GSVD.

Recall that the input of GSVD are matrices $A \in \mathbb{R}^{m \times n}$, and $B \in \mathbb{R}^{p \times n}$ for any arbitrary natural numbers m, p, and n. To start off, we first need to compute the URV decomposition of $M = [A^T, B^T]^T$. Suppose rank$(M) = r$, then the URV decomposition can be partitioned into the following:

$$\begin{array}{cc} n & \\ \left[\begin{array}{c} A \\ B \end{array}\right] & \begin{array}{c} m \\ p \end{array} \end{array} = \begin{array}{cc} \begin{array}{cc} r & n-r \end{array} & \\ \left[\begin{array}{cc} Q_{11} & Q_{21} \\ Q_{12} & Q_{22} \end{array}\right] & \begin{array}{c} m \\ p \end{array} \end{array} \begin{array}{cc} \begin{array}{cc} r & n-r \end{array} & \\ \left[\begin{array}{cc} \hat{R}_{11} & \hat{R}_{12} \\ 0 & \hat{R}_{22} \end{array}\right] & \begin{array}{c} r \\ n-r \end{array} \end{array} \begin{array}{cc} n & \\ [\, \hat{X} \,]^T & n \end{array},$$

where

$$\sigma(k+1) \geq \left\| \left[\begin{array}{c} \hat{R}_{12} \\ \hat{R}_{22} \end{array} \right] \right\|_2 .$$

Since the $[Q_{11}^T, Q_{21}^T]^T$ has orthonormal columns, we can compute their CSD: $UQ_{11}\hat{Z}^T = C$, and $VQ_{12}\hat{Z}^T = S$. Since in the resulted CSD can be written into matrix form:

$$\begin{array}{cc} r & \\ \left[\begin{array}{c} Q_{11} \\ Q_{21} \end{array}\right] & \begin{array}{c} m \\ p \end{array} \end{array} = \begin{array}{cc} \begin{array}{cc} m & p \end{array} & \\ \left[\begin{array}{cc} U & O \\ O & V \end{array}\right] & \begin{array}{c} m \\ p \end{array} \end{array} \begin{array}{cc} r & \\ \left[\begin{array}{c} C \\ S \end{array}\right] & \begin{array}{c} m \\ p \end{array} \end{array} \begin{array}{cc} r & \\ [\,\hat{Z}\,]^T & r \end{array},$$

it follows that

$$\left[\begin{array}{c} A \\ B \end{array}\right] = \left[\begin{array}{cc} U & 0 \\ 0 & V \end{array}\right] \left[\begin{array}{cc} C & U^T Q_{12} \\ S & V^T Q_{22} \end{array}\right] \underbrace{\left[\begin{array}{cc} \hat{Z}^T & 0 \\ 0 & I \end{array}\right] \left[\begin{array}{cc} \hat{R}_{11} & \hat{R}_{12} \\ 0 & \hat{R}_{22} \end{array}\right]}_{T} \hat{X}^T.$$

Now we compute the RQ decomposition of T:

$$\begin{array}{cc} \begin{array}{cc} r & n-r \end{array} & \\ \left[\begin{array}{cc} Z^T & 0 \\ 0 & I \end{array}\right] & \begin{array}{c} r \\ n-r \end{array} \end{array} \begin{array}{cc} \begin{array}{cc} r & n-r \end{array} & \\ \left[\begin{array}{cc} \hat{R}_{11} & \hat{R}_{12} \\ 0 & \hat{R}_{22} \end{array}\right] & \begin{array}{c} r \\ n-r \end{array} \end{array} = \begin{array}{cc} \begin{array}{cc} r & n-r \end{array} & \\ \left[\begin{array}{cc} R_{11} & R_{12} \\ 0 & R_{22} \end{array}\right] & \begin{array}{c} r \\ n-r \end{array} \end{array} \begin{array}{cc} n & \\ [\,Q\,] & n \end{array}.$$

Since

$$\left[\begin{array}{cc} C & U^T Q_{12} \\ S & V^T Q_{22} \end{array}\right] \left[\begin{array}{cc} R_{11} & R_{12} \\ 0 & R_{22} \end{array}\right] = \left[\begin{array}{c} C \\ S \end{array}\right] \left[\begin{array}{cc} R_{11} & R_{12} \end{array}\right] + \left[\begin{array}{c} U^T Q_{12} \\ V^T Q_{22} \end{array}\right] \left[\begin{array}{cc} 0 & R_{22} \end{array}\right]$$

and $\|[R_{12}^T, R_{22}^T]^T\|_2 \leq \sigma(k+1)$, the second term above and R_{12} can be neglected with error bounded by $\sigma(k+1)$. Hence

$$\left[\begin{array}{c} A \\ B \end{array}\right] = \left[\begin{array}{cc} U & 0 \\ 0 & V \end{array}\right] \left[\begin{array}{c} C \\ S \end{array}\right] \left[\begin{array}{cc} R_{11} & 0 \end{array}\right] Z^T, \text{ where } Z^T = Q\hat{X}^T. \tag{17}$$

Clearly, (17) yields the GSVD of $[A^T, B^T]^T$ with R in (9) with R being $[R_{11}, 0]$.

Note that by ordering the singular values of Q_2 non-decreasingly in CSD, the generalize singular value pairs of A and B has the same ordering as the SVD of AB^{-1}, which may provide a quick comparison on the generalize singular value of A and B with the singular value of AB^{-1}. This is the reason for choosing to reversing the singular values of Q_2 rather than Q_1.

Like the prescribed threshold in CSD, the user may specify the tolerance value that controls the rank determination at the URV decomposition step. The error of the GSVD of $[A^T, B^T]^T$ then should be bounded in terms of this user-controlled tolerance.

Overall, the algorithm can be summarized in the following:

```
ALGORITHM 4. (Compute the GSVD)
**Input:** $A, B, [\,\tau, \delta\,]$
**Output:** $U, V, Z, X, C, S$
1.    Compute the URV decomposition of $[A^T, B^T]^T$,
      obtaining $\hat{Q}$ and $\hat{X}$ (orthogonal), $\hat{R}$ (upper triangular) , and $r$ (rank).
2.    Compute the CSD of first $r$ columns of $\hat{Q}$,
      obtaining $U$, $V$, and $\hat{Z}$ (orthogonal), $C$ and $S$ ($C^TC$ and $S^TS$ are diagonal).
3.    Compute the RQ decomposition of $D = \hat{Z}^T\hat{R}(1:r,:)$,
      yielding $R$ (upper triangular) and $Q$ (orthogonal).
4.    Set $Z^T = Q\hat{X}^T$.
```

6 Discussion and Implementation Details

Algorithm 1-4 has been coded up in Matlab and their prototypes are the following.

$$\begin{array}{ll}
[\,k, U, R, V,\ ] & = \text{urv}(A,\ \text{tol\_rank},\ \text{tol\_ref}) \\
[\,R, Q\,] & = \text{rq}(A) \\
[\,U, V, Z, C, S\,] & = \text{csdj}(Q1,\ Q2) \\
[\,U, V, Z, R, C, S\,] & = \text{gsvdj}(A,\ B,\ \text{tol\_rank},\ \text{tol\_ref})
\end{array}$$

The csdj routine supers from the corresponding current Matlab subroutine named csd in a sense that the former code is shorter and more self-explanatory. Moreover, if the input matrices, Q_1 and Q_2, subject to some perturbation, the errors of all the resulted matrices would be bounded by the same amount, whereas the latter Q_1 and Q_2 fails instantly when (12) does not hold. This is important in the practice where the orthogonality of $[Q_1^T,\ Q_2^T]^T$ cannot be guaranteed to the machine's precision.

Up to date, the Matlab's built-in gsvd routine works under no assumption that $M = [A^T, B^T]^T$ has full rank, thus it does not invoke rank-revealing algorithm. In gsvdj, however, by employing URV decomposition as the first major modular factorization, the subsequently csdj is performed only on the first r linearly independent columns of M where $r = \text{rank}(M)$. Hence in the case where $[A^T,\ B^T]^T$ is rank-deficient, gsvdj provides a more reliable results than the built-in gsvd.

The current implementation of urv, csdj and gsvdj are carefully designed so that they will be easily transferred into LAPACK codes. These three functions will primarily rely on LAPACK's subroutine SGERQF, SORMRQ

(Housholder RQ), SGEQRF, SORMQR (Housholder QR) and SGESVD (SVD), along with different levels of BLAS (Basic Linear Algebra Subprograms). Some highlights of the LAPACK implementations are follow.

In LAPACK's implementation of csdj, SGESVD is used to compute the SVD of $Q2$. Since SGESVD returns the singular values in non-increasing order, csdj must reverse the ordering. Assuming the input matrices can be overwritten, space for $Q2$ can be used to store the product of $(Q1Z)$ at Step 6 before performing QR (by calling SGEQRF) because the algorithm guarantees that $m > p$ and $Q2$ is no longer needed in the rest of the computation. The orthogonal matrix V is then explicitly form by calling SORMQR subsequently. At Step 13, SGEQRF can be call again to perform the QR factorization of W. Note that since this is a well-conditioned matrix (smallest singular value is at least $1/\sqrt{2}$), it would be less costly to simply normalize the its columns and the resulted upper triangular matrix would be safely diagonal. This time, the orthogonal matrix is no longer needed to form explicitly because the Householder reflectors are right-multiplied with and stored in V. Unlike the current Matlab implementation, csdj only stores the sine and cosine values whose values lay in (0, 1). That is, it only returns the diagonal entries in Σ_1 and Σ_2 in (4)-(7). This gives considerable saving in memory space and is sufficient to construct the their exact matrix form if it is desirable.

In LAPACK's implementation of gsvdj, input matrices A and B first need to be "stacked up" in order to compute their URV in Step 1. In Step 3, SGERQF is called to compute the RQ of D. The resulted Householder reflectors are then multiplied with and stored in $\hat{X}$. The generalized singular value pairs are stored in the same way as in the csdj.

The amount of memory space required for these three functions depends on the dimensions of the input matrices. For csdj, the index k also plays a role in the overall quantification of the work space and it is not determined until runtime. As a result, it is difficult to determine the exact amount of working space at this point of progress. However, it can be roughly analyzed that the overall work space needed for csdj is no more than the square of $\max(m, n, p)$ and gsvdj needs additional $(m + p)$-by-n primarily due to the URV decomposition at the first step.

7 Numerical Experiments

In this section, we test the accuracies of the routines RQ, URV and GSVD that are implemented in Matlab functions rq, urv, and gsvdj, respectively as mentioned in the previous section. All experiments are performed in Matlab 7.0 on Dell Inspiron 1100 Notebook with a Celeron CPU of 2 Ghz, 512 Mb of RAM and machine precision $\epsilon \simeq 10^{-16}$.

All the matrices are generated in three main steps as coded in Matlab file named genMat1.m. First, this function generated two random square orthogonal matrices U and V with user-input size, whose columns resemble the left and right singular vectors of the desired matrix. Second, it uses the optional specified information such as gaps between singular values, tolerance, fixed rank to construct a proper diagonal matrix S. Finally, the desired matrix A is generated by setting $A = USV^T$. Each test for each matrix are repeated three times and the results listed are the average of the three trials. All the matrices generated in this section are chosen to have decreasing gap between their non-increasing singular values. Without further specification, all matrices labelled A and B are of dimension m-by-n and p-by-n, respectively, for any positive integer m, n, and p.

1. In the RQ routine, we measure the relative residual errors and the orthogonality of Q. That is, let the RQ decomposition of A be $A = RQ$, then we measure the following:

$$\begin{aligned} res &= \frac{\|RQ - A\|}{max(m,n)\|A\|} \\ orthog &= \frac{Q^TQ - I_k}{k} \text{ where } k = min(m,n) \ . \end{aligned}$$

The tolerance for all the tested matrices in this case is of machine epsilon, and all the matrices have condition number around 1000. The tested results depicted in Table 1 illustrate that the RQ decomposition is rather stable and accurate as all of their residual errors remain around machine precision and do not lose any digit of orthogonality.

size of A	**100 x 100**	**100 x 200**	**200 x 400**	**400 x 800**	**800 x 1600**
res	1.4139E-17	5.3315E-18	3.3106E-18	5.3941E-17	1.3522E-17
orthog	2.8063E-17	1.3498E-17	8.4976E-18	5.8075E-18	3.9826E-18

Table 1: Test RQ routine, $\tau \simeq 10^{-16}$, decreasing gap, full rank, cond$(A) \simeq$ 1000

2. In the URV routine, besides measuring the relative residual errors and testing orthogonality of the resulted orthogonal matrices similar to the previous RQ case, we also measure the norm (2-norm) of R_{12}, the minimum singular value of R_{11}, and the maximum singular value (i.e. 2-norm) of R_{22}. That is, let the URV decomposition of A be $A = URV^T$ with tolerance τ where R is of the form in (11), then we measure the following quantities:

$$\begin{aligned}
k_{exact} &= \text{rank of } A \text{ approximated by SVD with respect to } \tau \\
k_{approx} &= \text{rank of } A \text{ approximated by URV decomposition with respect to } \tau \\
res &= \frac{\|URV^T - A\|}{max(m,n)\|A\|} \\
orthog_U &= \frac{U^TU - I_j}{j} \quad \text{where } j = min(m,n) \\
orthog_V &= \frac{V^TV - I_j}{j} \quad \text{where } j = min(m,n) \\
nrm_{R_{12}} &= \frac{\|R_{12}\|}{j - exact_k} \\
\sigma_{min}(R_{11}) &= \text{minimum singular value of } R_{11} \\
\sigma_{max}(R_{22}) &= \text{maximum singular value of } R_{22}\ .
\end{aligned}$$

The difference between Table 2 and Table 3 is that each of the tested matrices in the former case has rank around to half of its smaller dimension (i.e. rank = min(m,n)/2) whereas in the latter case, each tested matrix is close to be full rank (each rank $\simeq$ min(m,n)-3).

Test results in Table 2 and Table 3 illustrate that the resulted matrices in the URV decomposition are rather accurate as they keep all digits of accuracy. However, in the lower rank case, the rank approximated by URV is further away from the "exact" rank (approximated by SVD) whereas the URV reveals a rank fairly close to the "exact" rank in the high rank case. Furthermore, in Table 3, all the results in $\sigma_{min}(R_{11})$ are above the tolerance while results in same row of Table 2 are slightly below the tolerance. This shows that in the lower-rank case, the resulted R_{11} may be close to be rank-deficient.

It can be concluded from this experiment that although the URV decomposition we chose to employed is able to keep all digits of accuracy in each of the resulted matrices, it has no guarantee on large sparse matrix. But it does provide a good estimation on numerical rank for dense matrix, which is our primary concern.

size of A	**25 x 45**	**50 x 50**	**100 x 70**	**110 x 150**	**400 x 75**
r_{exact}	13	25	35	55	38
r_{approx}	14	30	43	64	47
res	2.300E-16	5.355E-17	3.198E-17	1.229E-15	9.783E-18
$orthog_U$	6.525E-16	1.175E-16	1.054E-16	2.500E-15	1.167E-16
$orthog_V$	6.410E-16	6.240E-17	6.022E-17	2.865E-15	5.912E-17
$nrm_{R_{12}}$	6.410E-16	6.240E-17	6.022E-17	2.865E-15	5.912E-17
$\sigma_{min}(R_{11})$	1.400E-13	7.706E-14	6.398E-14	8.703E-14	6.374E-14
$\sigma_{max}(R_{22})$	7.246E-14	5.836E-14	5.867E-14	6.416E-14	6.086E-14

Table 2: Test URV routine, $\tau \simeq 10^{-13}$, decreasing gap, middle rank, cond(A) $\simeq 1000$

size of A	**25 x 45**	**50 x 50**	**100 x 70**	**110 x 150**	**400 x 75**
k_{exact}	22	47	67	107	72
k_{approx}	23	48	68	108	73
res	1.348E-17	1.359E-17	3.719E-17	6.695E-18	2.750E-18
$orthog_U$	3.787E-17	4.893E-17	2.899E-17	2.049E-17	6.213E-17
$orthog_V$	8.779E-17	8.706E-17	5.945E-17	1.006E-16	7.391E-17
$nrm_{R_{12}}$	1.262E-17	3.060E-20	1.931E-17	7.399E-19	5.678E-19
$minSV_{R_{11}}$	2.300E-13	4.800E-13	6.800E-13	1.080E-12	7.300E-13
$maxSV_{R_{22}}$	4.796E-15	6.928E-15	7.807E-15	1.039E-14	8.544E-15

Table 3: Test URV routine, $\tau \simeq 10^{-13}$, decreasing gap, high rank, cond(A) $\simeq 1000$

3. In the GSVD routine, we again measure the relative residual errors for each of the input matrices and test the orthogonality for all three resulted orthogonal matrices. Suppose A, B are the two input matrices, denote $G = (A^T, B^T)^T$, and their GSVD are $U^T AZ = CR$ and $V^T BZ = SR$,

then the following quantities are recorded:

$$\begin{aligned}
k_{exact} &= \text{rank of } G \text{ approximated by SVD with respect to } \tau \\
k_{approx} &= \text{rank of } G \text{ approximated by URV module with respect to } \tau \\
res_A &= \frac{\|U^T AZ - CR\|}{max(m,n)\|A\|} \\
res_B &= \frac{\|V^T BZ - SR\|}{max(p,n)\|B\|} \\
orthog_U &= \frac{\|I_m - U^T U\|}{m} \\
orthog_V &= \frac{\|I_p - V^T V\|}{p} \\
orthog_Z &= \frac{\|I_n - Z^T Z\|}{n}.
\end{aligned}$$

The sizes of A and B are chosen so that all possible combinations of m, n and p are tested. In this experiment, all test matrices are high-rank (each rank $\simeq$ min(m,n)-3) and their tolerance is around 10^{-10}. Since the rank of G in GSVDJ relies on the internal URV module, which is shown to work well with high-rank matrices, the rank returned from GSVDJ should be very close to the rank approximated by SVD, which is confirm by results in Table 4. While none of the resulted orthogonal matrices loss any digits of accuracy, the residual errors of A and B loss the last 2 digits of accuracy due to the tolerance being 10^{-10}.

size of A	**20x20**	**50x10**	**35x70**	**70x50**	**25x50**
B	**20x20**	**50x10**	**35x70**	**25x50**	**70x50**
k_{exact}	18	7	68	48	48
k_{approx}	19	8	68	48	48
res_A	2.590E-14	1.414E-13	1.686E-14	1.373E-14	1.905E-14
res_B	2.590E-14	1.414E-13	1.686E-14	1.373E-14	1.905E-14
$orthog_U$	9.817E-17	2.291E-17	4.601E-17	4.233E-17	6.180E-17
$orthog_V$	1.082E-16	3.866E-17	4.631E-17	5.439E-17	4.073E-17
$orthog_Z$	1.003E-16	1.962E-16	1.249E-16	1.434E-16	1.071E-16

Table 4: Test GSVDJ routine, $\tau \simeq 10^{-10}$, decreasing gap, high rank, cond $(G) \simeq 10^{10}$

References

[1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide, Third Edition,* SIAM Press, Philadelphia, 3rd Edition, 1999.

[2] Z. Bai, "The CSD, GSVD, Their Applications and Computations," *Preprint Series 958, Institute for Mathematics and Its Applications,* University of Minnesota, Minneapolis, April 1992.
Available at http://www.cs.ucdavis.edu/ bai.

[3] R. D. Fierro, J.R. Bunch, "Bounding the Subspaces From Rank Revealing Two-sided Orthogonal Decomposition," *SIAM Matrix Analysis and Applications,* **16**, (1995): 943-759.

[4] R. D. Fierro, P. C. Hansen, P. S. K. Hansen, "UTV Tools: Matlab Templates for Rank-Revealing UTV Decompositions," *Numiercal Algorithms,* **22**, (1999): 165-194.

[5] G. H. Golub and C. F. Van Loan, *Matrix Computations,* Johns Hopkins University Press, 3rd Edition, 1996.

[6] T. Y. Li and Z. Zheng, "A Rank-Revealing Method with Updating, Downdating and Applications," Research Report (DMS-0104009), Department of Mathematics, Michigan State University, East Lansing, MI and Northeastern Illinois University, Chicago, IL, 2004.
Available at http://www.neiu.edu/ zzeng/papers.htm.

[7] "Documentation for MathWorks Products, Release 14,"
The MathWorks
Available at http://www.mathworks.com/access/helpdesk/help/techdoc/

[8] H. Park, M. Jeon, and P.J. Howland, "Dimension Reduction for Text Data Representation based on Cluster Structure Preserving Projection," Research Report (TR01-013), Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN, 2001.
Available at https://wwws.cs.umn.edu/tech_reports/

[9] J. Speiser, "Linear Algebra Problems Arising from Signal Processing," *Invited Presentation 3, SIAM Annual Meeting,* (1989).

[10] G. W. Stewart, "Computing the CS Decomposition of a Partitioned Orthonormal Matrix," *Numerische Mathematik,* **40**, (1982): 297-306.

[11] G. W. Stewart, *Matrix Algorithms Volumne I: Basic Decompositions* SIAM Press, Philadelphia, 1998

[12] G. W. Stewart, “Updating a Rank-Revealing ULV Decomposition,” *SIAM J. Matrix Analysis and Applications,* **14**, (1993): 494-499.

[13] G. W. Stewart, “An Updating Algorithm for Subspace Tracking,” *IEEE Transaction Signal Processing,* **40**, (1982): 1535-1541.

[14] C. F. Van Loan, “Computing the CS and the Generalized Singular Value Decomposition,” *Numerische Mathematik,* **46**(number), (1985): 479-491.

[15] C. F. Von Loan, “Generalizing the Singular Value Decomposition,” *SIAM Journal on Numerical Analysis,* **13**(No. 1), (Mar. 1976):76-83.

[16] H. Zha, Z. Zhang, “Modifying the generalized Singular Value Decomposition with Application in Direction-of-Arrival Finding,” *BIT,* **38**(1), (1998): 200-216.