GSVD

Ji Wang

2020-09-25

Contents

List of Figures

List of Tables

Listings

1 Definition

1.1 Our definition

The generalized singular value decomposition of an m-by-n matrix A and p-by-n matrix B is given as follows:

$$A = UCRQ^T, \quad B = VSRQ^T \tag{1}$$

- U is an m-by-m orthogonal matrix,
- V is a p-by-p orthogonal matrix,
- Q is an n-by-n orthogonal matrix,
- C is an m-by-$(k+l)$ real, non-negative diagonal matrix with 1s in the first k entries,
- S is a p-by-$(k+l)$ real, non-negative matrix whose top right l-by-l block is diagonal,
- R is a $(k+l)$-by-n matrix of structure $[0,\ R_0]$ where R_0 is $(k+l)$-by-$(k+l)$, upper triangular and nonsingular.

C and S also hold the following properties:

- $C^TC + S^TS = I$,
- $C^TC = \text{diag}(\alpha_1^2, ..., \alpha_{k+l}^2)$, $S^TS = \text{diag}(\beta_1^2, ..., \beta_{k+l}^2)$, where α_i, $\beta_i \in [0, 1]$ for $i = 1, ..., k+l$. The ratios α_i/β_i are called the **generalized singular values** of the pair A, B, and are in non-increasing order. The first k values are infinite, the remaining l values are finite,
- l is the rank of B and $k+l$ is the rank of $[A; B]$.

Structures of C and S depend on the row size of A and the rank of $[A; B]$. Two cases are detailed below:

(1) $m \geq k+l$

$$C = \begin{array}{r} \\ k \\ l \\ m-k-l \end{array}\!\!\begin{array}{c} \begin{array}{cc} k & l \end{array} \\ \begin{pmatrix} I & 0 \\ 0 & \Sigma_1 \\ 0 & 0 \end{pmatrix} \end{array}, \quad S = \begin{array}{r} \\ l \\ p-l \end{array}\!\!\begin{array}{c} \begin{array}{cc} k & l \end{array} \\ \begin{pmatrix} 0 & \Sigma_2 \\ 0 & 0 \end{pmatrix} \end{array}$$

Here, Σ_1 and Σ_2 are diagonal matrices and $\Sigma_1^2 + \Sigma_2^2 = I$, and Σ_2 is nonsingular. Also, $\alpha_1 = \cdots = \alpha_k = 1$, $\alpha_{k+i} = (\Sigma_1)_{ii}$ for $i = 1, \cdots, l$, $\beta_1 = \cdots = \beta_k = 0$, $\beta_{k+i} = (\Sigma_2)_{ii}$ for $i = 1, \cdots, l$.

(2) $m < k+l$

$$C = \begin{array}{r} \\ k \\ m-k \end{array}\!\!\begin{array}{c} \begin{array}{ccc} k & m-k & k+l-m \end{array} \\ \begin{pmatrix} I & 0 & 0 \\ 0 & \Sigma_1 & 0 \end{pmatrix} \end{array}, \quad S = \begin{array}{r} \\ m-k \\ k+l-m \\ p-l \end{array}\!\!\begin{array}{c} \begin{array}{ccc} k & m-k & k+l-m \end{array} \\ \begin{pmatrix} 0 & \Sigma_2 & 0 \\ 0 & 0 & I \\ 0 & 0 & 0 \end{pmatrix} \end{array}$$

Still, Σ_1 and Σ_2 are diagonal matrices and $\Sigma_1^2 + \Sigma_2^2 = I$, and Σ_2 is nonsingular. Also, $\alpha_1 = \cdots = \alpha_k = 1$, $\alpha_{k+i} = (\Sigma_1)_{ii}$ for $i = 1, \cdots, m-k$, $\alpha_{m+1} = \cdots \alpha_{k+l} = 0$, $\beta_1 = \cdots = \beta_k = 0$, $\beta_{k+i} = (\Sigma_2)_{ii}$ for $i = 1, \cdots, m-k$, $\beta_{m+1} = \cdots = \beta_{k+l} = 1$.

1.2 Essential properties

Property 1 Our formulation always reveals the rank of $[A; B]$.

From our decomposition, we can immediately know the rank of $[A; B]$ from the number of columns of C or S.

Property 2 We can get the common nullspace of A and B from our formulation.

If we rewrite our formulation of GSVD as:

$$A(Q_1,\ Q_2) = UC(0,\ R_0), \quad B(Q_1,\ Q_2) = VS(0,\ R_0) \tag{2}$$

where Q_1 is n-by-$(n-k-l)$, Q_2 is n-by-$(k+l)$ and R_0 is $(k+l)$-by-$(k+l)$. Then, we have null(A) $\cap$ null(B) = span$\{Q_1\}$. In other words, Q_1 is the orthonormal basis of the common nullspace of A and B.

Property 3 We can solve the generalized eigenvalue problem ($A^TAx = \lambda B^TBx$) from our formulation.

If we let $X = Q \begin{array}{c} \begin{array}{cc} n-k-l & k+l \end{array} \\ \begin{pmatrix} I & 0 \\ 0 & R_0^{-1} \end{pmatrix} \end{array}$, then

$$X^TA^TAX = \begin{array}{c} \\ n-k-l \\ k+l \end{array} \begin{array}{c} \begin{array}{cc} n-k-l & k+l \end{array} \\ \begin{pmatrix} 0 & 0 \\ 0 & C^TC \end{pmatrix} \end{array}, \quad X^TB^TBX = \begin{array}{c} \\ n-k-l \\ k+l \end{array} \begin{array}{c} \begin{array}{cc} n-k-l & k+l \end{array} \\ \begin{pmatrix} 0 & 0 \\ 0 & S^TS \end{pmatrix} \end{array}$$

Thus, we know the "non-trivial" eigenpairs of the generalized eigenvalue problem:

$$A^TAX_{i+n-k-l} = \lambda_i B^TBX_{i+n-k-l}, \quad i = 1, \cdots, k+l$$

$\lambda_i = (\alpha_i/\beta_i)^2$ are eigenvalues, where α_i/β_i is the generalized singular value of A and B. $X_{i+n-k-l}$ denotes the $(i+n-k-l)th$ column of X and are the corresponding eigenvectors.

Property 4 Two special cases of the generalized singular value decomposition.

- When B is square and nonsingular, the generalized singular value decomposition of A and B is equivalent to the singular value decomposition of AB^{-1}, regardless of how the GSVD is defined.
- No matter how we fomulate GSVD, if the columns of $(A^T,\ B^T)^T$ are orthonormal, then the generalized singular value decomposition of A and B is equivalent to the Cosine-Sine decomposition (CSD) of $(A^T,\ B^T)^T$, namely:

$$A = UCQ^T, \quad B = VSQ^T \tag{3}$$

where U is m-by-m, V is p-by-p and Q is n-by-n and all of them are orthogonal matrices.

1.3 Other notable definitions

We list four major definitions of GSVD for further discussion.

1.3.1 Definition(1): Van Loan (1976) [1]

Given an m-by-n matrix A and a p-by-n matrix B with $m \geq n$ and $r = \mathrm{rank}([A; B])$, the generalized singualr value decomposition of A and B is:

$$A = UCX^{-1}, \quad B = VSX^{-1} \tag{4}$$

where

$$C = \begin{array}{c c} & \begin{array}{ccc} q & r-q & n-r \end{array} \\ \begin{array}{c} q \\ r-q \\ m-r \end{array} & \begin{pmatrix} I & 0 & 0 \\ 0 & \Sigma_1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{array}, \quad S = \begin{array}{c c} & \begin{array}{ccc} q & r-q & n-r \end{array} \\ \begin{array}{c} q \\ r-q \\ p-r \end{array} & \begin{pmatrix} 0 & 0 & 0 \\ 0 & \Sigma_2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{array}$$

- U is an m-by-m orthogonal matrix.
- V is a p-by-p orthogonal matrix.
- X is an n-by-n nonsingular matrix.
- C and S are m-by-n and p-by-n, and $q = max\{r - p, 0\}$. $\alpha_1 = \cdots = \alpha_q = 1$, $\Sigma_1 = \mathrm{diag}(\alpha_{q+1}, \cdots, \alpha_r)$, $\beta_1 = \cdots = \beta_q = 0$, $\Sigma_2 = \mathrm{diag}(\beta_{q+1}, \cdots, \beta_r)$. $\Sigma_1^2 + \Sigma_2^2 = I$.

Remark This definition holds all properties in Section 1.2 except *Property* 2.

For *Property* 1, one can immediately know $\mathrm{rank}([A; B])$

For *Property* 3,

$$\begin{aligned} X^T A^T A X &= X^T (UCX^{-1})^T (UCX^{-1}) X \\ &= X^T (X^{-1})^T C^T U^T U C X^{-1} X \\ &= X^T (X^T)^{-1} C^T C \\ &= C^T C \end{aligned}$$

Similarly, $X^T B^T B X = S^T S$. Therefore, the first r quotients of the diagonal entries of $C^T C$ and $S^T S$ are the "non-trivial" eigenvalues of the generalized eigenvalue problem and the first r columns of X are the corresponding eigenvectors.

1.3.2 Definition(2): MATLAB 2019b

The generalized singular value decomposition of an m-by-n matrix A and a p-by-n matrix B is the following:

$$A = UCX^T, \quad B = VSX^T \tag{5}$$

- U is an m-by-m orthogonal matrix.
- V is a p-by-p orthogonal matrix.
- X is an n-by-q matrix where $q = min\{m + p, n\}$.

- C is an m-by-q block-diagonal matrix and S is a p-by-q diagonal matrix. Both are nonnegative and $C^TC + S^TS = I$. If $q > m$, the rightmost m-by-m block of C is diagonal. Otherwise, nonzero elements are on the main diagonal of C.

- $C^TC = \mathrm{diag}(\alpha_1^2, \cdots, \alpha_q^2)$, $S^TS = \mathrm{diag}(\beta_1^2, \cdots, \beta_q^2)$, where $\alpha_i,\ \beta_i \in [0,1]$ for $i = 1, \cdots, q$. The ratios α_i/β_i are called the generalized singular values of the pair A, B and are in non-decreasing order.

Specifically, C and S have the following structures:

1. $m + p \geq n$, thus $q = n$:

   (a) $n > m, n \leq p$:

$$C = \begin{array}{cc} & \begin{array}{cc} n-m & m \end{array} \\ m & \begin{pmatrix} 0 & \Sigma_1 \end{pmatrix} \end{array}, \quad S = \begin{array}{cc} & n \\ \begin{array}{c} n \\ p-n \end{array} & \begin{pmatrix} \Sigma_2 \\ 0 \end{pmatrix} \end{array}$$

   where $\Sigma_1 = \mathrm{diag}(\alpha_{n-m+1}, \cdots, \alpha_n)$ and $\Sigma_2 = \mathrm{diag}(\beta_1, \cdots, \beta_n)$.

   (b) $n \leq m, n > p$:

$$C = \begin{array}{cc} & n \\ \begin{array}{c} n \\ m-n \end{array} & \begin{pmatrix} \Sigma_1 \\ 0 \end{pmatrix} \end{array}, \quad S = \begin{array}{cc} & \begin{array}{cc} p & n-p \end{array} \\ p & \begin{pmatrix} \Sigma_2 & 0 \end{pmatrix} \end{array}$$

   where $\Sigma_1 = \mathrm{diag}(\alpha_1, \cdots, \alpha_n)$ and $\Sigma_2 = \mathrm{diag}(\beta_1, \cdots, \beta_p)$.

   (c) $n \leq m, n \leq p$:

$$C = \begin{array}{cc} & n \\ \begin{array}{c} n \\ m-n \end{array} & \begin{pmatrix} \Sigma_1 \\ 0 \end{pmatrix} \end{array}, \quad S = \begin{array}{cc} & n \\ \begin{array}{c} n \\ p-n \end{array} & \begin{pmatrix} \Sigma_2 \\ 0 \end{pmatrix} \end{array}$$

   where $\Sigma_1 = \mathrm{diag}(\alpha_1, \cdots, \alpha_n)$ and $\Sigma_2 = \mathrm{diag}(\beta_1, \cdots, \beta_n)$.

2. $m + p < n$, thus $q = m + p$:

$$C = \begin{array}{cc} & \begin{array}{cc} p & m \end{array} \\ m & \begin{pmatrix} 0 & \Sigma_1 \end{pmatrix} \end{array}, \quad S = \begin{array}{cc} & \begin{array}{cc} p & m \end{array} \\ p & \begin{pmatrix} \Sigma_2 & 0 \end{pmatrix} \end{array}$$

   where $\Sigma_1 = \mathrm{diag}(\alpha_{p+1}, \cdots, \alpha_{p+m})$ and $\Sigma_2 = \mathrm{diag}(\beta_1, \cdots, \beta_p)$.

Remark Only *Property* 4 is true given this definition.

1.3.3 Definition(3): Edelman (2019) [2]

The generalized singular value decomposition of an m-by-n matrix A and a p-by-n matrix B is the following:

$$A = UCH, \quad B = VSH \tag{6}$$

$$
C = \begin{array}{r} \\ k \\ s \\ m-k-s \end{array}
\begin{array}{c} \begin{array}{ccc} k & s & r-k-s \end{array} \\ \begin{pmatrix} I & 0 & 0 \\ 0 & \Sigma_1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{array}, \quad
S = \begin{array}{r} \\ p-r+k \\ s \\ r-k-s \end{array}
\begin{array}{c} \begin{array}{ccc} k & s & r-k-s \end{array} \\ \begin{pmatrix} 0 & 0 & 0 \\ 0 & \Sigma_2 & 0 \\ 0 & 0 & I \end{pmatrix} \end{array}
$$

- U is an m-by-m orthogonal matrix.
- V is a p-by-p orthogonal matrix.
- C is an m-by-r matrix and S is an n-by-r matrix where r = rank($[A;B]$). $C^TC + S^TS = I$. k = rank($[A;B]$) - rank(B), s = rank(A) + rank(B) - rank($[A;B]$). $\alpha_1 = \cdots = \alpha_k = 1$, $\Sigma_1 = \mathrm{diag}(\alpha_{k+1}, \cdots, \alpha_{k+s})$, $\alpha_{k+s+1} = \cdots = \alpha_r = 0$, $\beta_1 = \cdots = \beta_k = 0$, $\Sigma_2 = \mathrm{diag}(\beta_{k+1}, \cdots, \beta_{k+s})$, $\beta_{k+s+1} = \cdots = \beta_r = 1$. $\Sigma_1^2 + \Sigma_2^2 = I$.
- H is an r-by-n matrix and has full row rank.

Remark All properties in Section 1.2 hold true by this definition. Specifically,

Property 1 is true since rank($[A;B]$) is the number of columns in C and S.

Property 2 holds because null(A) $\cap$ null(B) = null(H). Alternatively, if we do RQ factorization on H, namely, $H = (0,\ R_0)Q^T$, where R_0 is an $(k+l)$-by-$(k+l)$ upper triangular matrix and Q is an n-by-n orthgonal matrix, then null(A) $\cap$ null(B) = span$\{Q(:, 1 : n-k-l)\}$.

Property 3 is verified as true if we do RQ factorization on H, namely, $H = (0,\ R_0)Q^T$, and let $X = Q \begin{array}{c} \begin{array}{cc} n-k-l & k+l \end{array} \\ \begin{pmatrix} I & 0 \\ 0 & R_0^{-1} \end{pmatrix} \end{array}$, then the “non-trivial" eigenvalues of the generalized eigenvalue problem are the square of the generalized singular values and and the last $k+l$ columns of X are the corresponding eigenvectors.

1.3.4 Link between Definition(1) and Definition(2)

Definition(1) imposes a constraint on the size of matrix A such that $m \geq n$. To discuss the connection between Definition(1) and Definition(2), we first narrow down Definition(2) to case 1(b) and 1(c) in Section 1.3.2.

In either case, one may verify that the size of C and S are the same as those in Definition(1). Namely, C is m-by-n and S is p-by-n. However, the ordering of diagonal entries in C and S are opposite from each other.

For Definition(1),

$$
\begin{pmatrix} A \\ B \end{pmatrix} = \begin{pmatrix} Q_1 \\ Q_2 \end{pmatrix} R = \begin{pmatrix} U & 0 \\ 0 & V \end{pmatrix} \begin{pmatrix} C \\ S \end{pmatrix} Z^T R
$$

Let $X = R^{-1}Z$, then

$$\begin{pmatrix} A \\ B \end{pmatrix} = \begin{pmatrix} U & 0 \\ 0 & V \end{pmatrix} \begin{pmatrix} C \\ S \end{pmatrix} X^{-1}$$

For Definition (2)

$$\begin{pmatrix} A \\ B \end{pmatrix} = \begin{pmatrix} Q_1 \\ Q_2 \end{pmatrix} R = \begin{pmatrix} U & 0 \\ 0 & V \end{pmatrix} \begin{pmatrix} \Sigma_1 \\ \Sigma_2 \end{pmatrix} Z^T R = \begin{pmatrix} U & 0 \\ 0 & V \end{pmatrix} \begin{pmatrix} C \\ S \end{pmatrix} P Z^T R$$

Let $X = R^T Z P^T$, then

$$\begin{pmatrix} A \\ B \end{pmatrix} = \begin{pmatrix} U & 0 \\ 0 & V \end{pmatrix} \begin{pmatrix} C \\ S \end{pmatrix} X^T$$

Note that if $[A; B]$ is rank deficient, we may use QR decomposition with column pivoting or SVD? [3] [4, pp. 310]

2 Algorithms

2.1 Proposed GSVD algorithm

The algorithm we propose consists of four steps. First is the pre-processing step when we reduce the input matrix pair to a triangular pair while revealing their ranks. [5] We further reduce two upper triangular matrices to one upper triangular matrix in the QR decomposition step. Next is the CS decomposition of a matrice with orthonormal columns that is partitioned into two blocks. [1] The last step is post-processing to get the final product of the decomposition.

Step 1 Pre-processing:

To reduce regular matrices to their triangular form and reveal rank, we employ URV decomposition (QR decomposition with column pivoting followed by RQ decomposition) [4] as well as QR decomposition. We detail this in nine steps below.

(1) QR decomposition with column pivoting of B:

$$BP = V \begin{array}{c c} & \begin{array}{cc} l & n-l \end{array} \\ \begin{array}{c} l \\ p-l \end{array} & \begin{pmatrix} B_{11} & B_{12} \\ 0 & 0 \end{pmatrix} \end{array}$$

(2) Update $A:\ A = AP$

(3) Set $Q:\ Q = I,\ \ Q = QP$

(4) If $p \geq l$ and $n \neq l$:

- RQ decomposition of $(B_{11}\ \ B_{12})$:

$$\begin{array}{c c} & \begin{array}{cc} l & n-l \end{array} \\ l & \begin{pmatrix} B_{11} & B_{12} \end{pmatrix} \end{array} = \begin{array}{c c} & \begin{array}{cc} n-l & l \end{array} \\ l & \begin{pmatrix} 0 & B_{13} \end{pmatrix} \end{array} Z$$

- Update $A:\ A = AZ^T$
- Update $Q:\ Q = QZ^T$

(5) Let

$$A = \begin{array}{c c} & \begin{array}{cc} n-l & l \end{array} \\ m & \begin{pmatrix} A_1 & A_2 \end{pmatrix} \end{array},$$

then QR decomposition with column pivoting of A_1 is:

$$A_1P_1 = U \begin{array}{c c} & \begin{array}{cc} k & n-l-k \end{array} \\ \begin{array}{c} k \\ m-k \end{array} & \begin{pmatrix} A_{11} & A_{12} \\ 0 & 0 \end{pmatrix} \end{array}$$

(6) Update $A_2:\ A_2 = U^TA_2$

(7) Update $Q:\ Q[1:n, 1:n-l] = Q[1:n, 1:n-l]P_1$

(8) If $n - l \geq k$:

- RQ decomposition of $(A_{11} \;\; A_{12})$:

$$\begin{array}{c c} & \begin{array}{cc} k & n-l-k \end{array} \\ k & \begin{pmatrix} A_{11} & A_{12} \end{pmatrix} \end{array} = \begin{array}{c c} & \begin{array}{cc} n-l-k & k \end{array} \\ k & \begin{pmatrix} 0 & A_{12} \end{pmatrix} \end{array} Z_1$$

- Update Q : $Q[1:n, 1:n-l] = Q[1:n, 1:n-l]Z_1^T$

(9) If $m \geq k$: Let

$$A_2 = \begin{array}{c c} & l \\ \begin{array}{c} k \\ m-k \end{array} & \begin{pmatrix} A_{13} \\ A_{23} \end{pmatrix} \end{array}$$

- QR decomposition of A_{23}:

$$A_{23} = U_1 \begin{array}{c c} & l \\ \begin{array}{c} l \\ m-k-l \end{array} & \begin{pmatrix} A_{23} \\ 0 \end{pmatrix} \end{array}$$

- Update U : $U[:, k+1:m] = U[:, k+1:m]U_1$

Putting it together, we have the following decomposition as pre-processing:

$$A = UR_A Q^T, \qquad B = VR_B Q^T \tag{7}$$

where

$$R_A = \begin{array}{c c} & \begin{array}{ccc} n-k-l & k & l \end{array} \\ \begin{array}{c} k \\ l \\ m-k-l \end{array} & \begin{pmatrix} 0 & A_{12} & A_{13} \\ 0 & 0 & A_{23} \\ 0 & 0 & 0 \end{pmatrix} \end{array}, \qquad R_B = \begin{array}{c c} & \begin{array}{ccc} n-k-l & k & l \end{array} \\ \begin{array}{c} l \\ p-l \end{array} & \begin{pmatrix} 0 & 0 & B_{13} \\ 0 & 0 & 0 \end{pmatrix} \end{array}$$

overwrite A and B, respectively, and A_{12} and B_{13} are non-singular upper triangular matrix. l is the rank of B, $k+l$ is the rank of $[A^T \; B^T]^T$. If $m-k-l \geq 0$, A_{23} is l-by-l upper triangular, otherwise, it's $(m-k)$-by-l upper trapezoidal.

Step 2 QR decomposition of $[A_{23}^T \; B_{13}^T]^T$:

$$\begin{array}{c c} & l \\ \begin{array}{c} l \\ l \end{array} & \begin{pmatrix} A_{23} \\ B_{23} \end{pmatrix} \end{array} = \begin{array}{c c} & l \\ \begin{array}{c} l \\ l \end{array} & \begin{pmatrix} Q_1 \\ Q_2 \end{pmatrix} \end{array} R_{23}$$

Thus, (7) can be rewritten as:

$$A = U(Q_A \hat{R})Q^T, \qquad B = V(Q_B \hat{R})Q^T \tag{8}$$

where

$$Q_A = \begin{array}{c c} & \begin{array}{cc} k & l \end{array} \\ \begin{array}{c} k \\ l \\ m-k-l \end{array} & \begin{pmatrix} I & 0 \\ 0 & Q_1 \\ 0 & 0 \end{pmatrix} \end{array}, \qquad Q_B = \begin{array}{c c} & \begin{array}{cc} k & l \end{array} \\ \begin{array}{c} l \\ p-l \end{array} & \begin{pmatrix} 0 & Q_2 \\ 0 & 0 \end{pmatrix} \end{array}, \qquad \hat{R} = \begin{array}{c c} & \begin{array}{ccc} n-k-l & k & l \end{array} \\ \begin{array}{c} k \\ l \end{array} & \begin{pmatrix} 0 & A_{12} & B_{13} \\ 0 & 0 & R_{23} \end{pmatrix} \end{array}$$

If $m-k-l \geq 0$, Q_1 is l-by-l, otherwise, Q_1 is $(m-k)$-by-l.

Step 3 CS decomposition of Q_1 and Q_2:

$$Q_1 = U_1 C_1 Z_1^T, \quad Q_2 = V_1 S_1 Z_1^T \tag{9}$$

We then can derive from *Step* 2 and the above CS decomposition that

$$A = U(\hat{U} C \hat{Q}^T)\hat{R}Q^T, \quad B = V(\hat{V} S \hat{Q}^T)\hat{R}Q^T \tag{10}$$

where

$$\hat{U} = \begin{array}{c} \\ k \\ l \\ m-k-l \end{array} \begin{array}{c} \begin{array}{ccc} k & l & m-k-l \end{array} \\ \begin{pmatrix} I & 0 & 0 \\ 0 & U_1 & 0 \\ 0 & 0 & I \end{pmatrix} \end{array}, \quad \hat{V} = \begin{array}{c} \\ l \\ p-l \end{array} \begin{array}{c} \begin{array}{cc} l & p-l \end{array} \\ \begin{pmatrix} V_1 & 0 \\ 0 & I \end{pmatrix} \end{array}, \quad \hat{Q}^T = \begin{array}{c} \\ l \\ p-l \end{array} \begin{array}{c} \begin{array}{cc} k & l \end{array} \\ \begin{pmatrix} I & 0 \\ 0 & Z_1^T \end{pmatrix} \end{array}$$

and

$$C = \begin{array}{c} \\ k \\ l \\ m-k-l \end{array} \begin{array}{c} \begin{array}{cc} k & l \end{array} \\ \begin{pmatrix} I & 0 \\ 0 & C_1 \\ 0 & 0 \end{pmatrix} \end{array}, \quad S = \begin{array}{c} \\ l \\ p-l \end{array} \begin{array}{c} \begin{array}{cc} k & l \end{array} \\ \begin{pmatrix} 0 & S_1 \\ 0 & 0 \end{pmatrix} \end{array}$$

Note that when $m-k-l<0$, U_1 and C_1 will only have $m-k$ rows.

More details regarding CS decomposition can be found in Section 2.1.1.

Step 4 Post-processing:

- $U = U\hat{U}$.
- $V = V\hat{V}$.
- Formulate R by RQ decomposition: $\hat{Q}^T\hat{R} = RQ_3$
- $Q = QQ_3^T$

To sum up, we can obtain:

$$A = UCRQ^T, \quad B = VSRQ^T \tag{11}$$

C and S have the following structures:

- if $m \geq k+l$

$$C = \begin{array}{c} \\ k \\ l \\ m-k-l \end{array} \begin{array}{c} \begin{array}{cc} k & l \end{array} \\ \begin{pmatrix} I & 0 \\ 0 & \Sigma_1 \\ 0 & 0 \end{pmatrix} \end{array}, \quad S = \begin{array}{c} \\ l \\ p-l \end{array} \begin{array}{c} \begin{array}{cc} k & l \end{array} \\ \begin{pmatrix} 0 & \Sigma_2 \\ 0 & 0 \end{pmatrix} \end{array}$$

- if $m < k+l$

$$C = \begin{array}{c} \\ k \\ m-k \end{array} \begin{array}{c} \begin{array}{ccc} k & m-k & k+l-m \end{array} \\ \begin{pmatrix} I & 0 & 0 \\ 0 & \Sigma_1 & 0 \end{pmatrix} \end{array}, \quad S = \begin{array}{c} \\ m-k \\ k+l-m \\ p-l \end{array} \begin{array}{c} \begin{array}{ccc} k & m-k & k+l-m \end{array} \\ \begin{pmatrix} 0 & \Sigma_2 & 0 \\ 0 & 0 & I \\ 0 & 0 & 0 \end{pmatrix} \end{array}$$

In either case, $\Sigma_1^2 + \Sigma_2^2 = I$.

Remark Michael Stewart in his paper [6] describes an alternate rank revealing mechanism of $[A; B]$ that more reliably determines the partitioning of a GSVD and shows improved numerical reliability.

2.1.1 CS Decomposition

We first define what is CS decomposition. Suppose we have an $(m+p)-by-l$ matrix Q such that $m+p \geq l$ and has orthnormal columns. If we partition Q into two block matrices as $[Q_1; Q_2]$, then the CS decomposition of Q_1 and Q_2 is the following:

$$Q_1 = UCZ^T, \quad Q_2 = VSZ^T \tag{12}$$

- U is an m-by-m orthogonal matrix,
- V is a p-by-p orthogonal matrix,
- Q is an l-by-l orthogonal matrix,
- C is an m-by-l real, non-negative diagonal matrix,
- S is a p-by-l real, non-negative matrix whose top right block is diagonal,
- $C^TC + S^TS = I$.

Specifically, C and S fall into four cases depending on their sizes.

1. $m \geq l$ and $p \geq l$:

$$C = \begin{array}{c c} & l \\ \begin{array}{c} l \\ m-l \end{array} & \begin{pmatrix} \Sigma_1 \\ 0 \end{pmatrix} \end{array}, \quad S = \begin{array}{c c} & l \\ \begin{array}{c} l \\ p-l \end{array} & \begin{pmatrix} \Sigma_2 \\ 0 \end{pmatrix} \end{array}$$

2. $m \geq l$ and $p < l$:

$$C = \begin{array}{c c} & \begin{array}{cc} l-p & p \end{array} \\ \begin{array}{c} l-p \\ p \\ m-l \end{array} & \begin{pmatrix} I & 0 \\ 0 & \Sigma_1 \\ 0 & 0 \end{pmatrix} \end{array}, \quad S = \begin{array}{c c} & \begin{array}{cc} l-p & p \end{array} \\ p & \begin{pmatrix} 0 & \Sigma_2 \end{pmatrix} \end{array}$$

3. $m \leq l$ and $p \geq l$:

$$C = \begin{array}{c c} & \begin{array}{cc} m & l-m \end{array} \\ m & \begin{pmatrix} \Sigma_1 & 0 \end{pmatrix} \end{array}, \quad S = \begin{array}{c c} & \begin{array}{cc} m & l-m \end{array} \\ \begin{array}{c} m \\ l-m \\ p-l \end{array} & \begin{pmatrix} \Sigma_2 & 0 \\ 0 & I \\ 0 & 0 \end{pmatrix} \end{array}$$

4. $m \leq l$ and $p < l$:

$$C = \begin{array}{c c} & \begin{array}{ccc} l-p & t & l-m \end{array} \\ \begin{array}{c} l-p \\ t \end{array} & \begin{pmatrix} I & 0 & 0 \\ 0 & \Sigma_1 & 0 \end{pmatrix} \end{array}, \quad S = \begin{array}{c c} & \begin{array}{ccc} l-p & t & l-m \end{array} \\ \begin{array}{c} t \\ l-m \end{array} & \begin{pmatrix} 0 & \Sigma_2 & 0 \\ 0 & 0 & I \end{pmatrix} \end{array}$$

   where $t = m + p - l$.

Note that Σ_1 and Σ_2 in all four cases are diagonal matrices and satisfy $\Sigma_1^2 + \Sigma_2^2 = I$.

Now, we explain the algorithm to compute the CS decomposition.

2.2 Other prominent algorithms

2.2.1 LAPACK algorithm

This algorithm [7, pp. 51–53] has two phases. First is a pre-processing step as described in Section 2.1. Next is a Jacobi-style method [8] [3] to directly compute the GSVD of two square upper trangular matrices, namely, A_{23} and B_{13} in (7) such that

$$A_{23} = U_1 C R Q_1^T, \qquad B_{13} = V_1 S R Q_1^T. \tag{13}$$

Here U_1, V_1 and Q_1 are orthogonal matrices, C and S are both real nonnegative matrices satisfying $C^T C + S^T S = I$, S is nonsingular, and R is upper triangular and nonsingular.

2.2.2 Van Loan's algorithm

Golub and Van Loan [4, pp. 502–503] introduced an algorithm to compute GSVD using CS decomposition for tall, full-rank matrix pairs.

Assume that A is m-by-n and B is p-by-n with $m \geq n$ and $p \geq n$, computes an m-by-m orthogonal matrix U, a p-by-p orthogonal matrix V, an n-by-n nonsingular matrix X and m-by-n diagonal matrice C, p-by-n diagonal matrice S such that $U^T A X = C$ and $V^T B X = S$.

Step 1 Compute the regular QR decomposition of $\begin{pmatrix} A \\ B \end{pmatrix}$:

$$\begin{pmatrix} A \\ B \end{pmatrix} = \begin{pmatrix} Q_1 \\ Q_2 \end{pmatrix} R$$

Step 2 Compute the CS decomposition of Q_1 and Q_2:

$U^T Q_1 Z = C = diag(\alpha_i, \cdots, \alpha_n)$, $V^T Q_2 Z = S = diag(\beta_i, \cdots, \beta_n)\cdot$

Step 3 Solve $RX = Z$ for X.

2.3 Justifications on the choice of CS decomposition over Jacobi method

3 Software

3.1 Interface design

The products of the GSVD are six matrices and two integers indicating the rank. To follow Julia's convention, we encapsulate all the products into a composite type named `GeneralizedSVD`. In this way, users do not need to explicitly enumerate every matrix or integer in the return statement. In addition, doing so will facilitate those who only want to access part of the products. Hence, we define the composite type as a struct.

```
struct GeneralizedSVD{T} <: Factorization{T}
    U::AbstractMatrix{T}
    V::AbstractMatrix{T}
    Q::AbstractMatrix{T}
    C::AbstractMatrix{T}
    S::AbstractMatrix{T}
    k::Int
    l::Int
    R::AbstractMatrix{T}
end
```

Interface 1 We adopt the practice of polymorphism when designing the interface of the GSVD. This enables SVD of one matrix and GSVD of a matrix pair to share a single interface with entities of different input parameters. Such polymorphism allows a function to be written generically and thus maintain the language's expressiveness. We now present the interface below.

```
svd(A, B) -> GeneralizedSVD
```

Compute the generalized SVD of `A` and `B`, returning a `GeneralizedSVD` factorization object `F`, such that `A = F.U*F.C*F.R*F.Q'` and `B = F.V*F.S*F.R*F.Q'`.

For an m-by-n matrix `A` and p-by-n matrix `B`,

- `U` is an m-by-m orthogonal matrix,
- `V` is a p-by-p orthogonal matrix,
- `Q` is an n-by-n orthogonal matrix,
- `C` is an m-by-(k+l) diagonal matrix with 1s in the first K entries,
- `S` is a p-by-(k+l) matrix whose top right L-by-L block is diagonal,

- `R` is a (k+l)-by-n matrix whose rightmost (k+l)-by-(k+l) block is nonsingular upper block triangular,
- `k+l` is the effective numerical rank of the matrix `[A; B]`.

Iterating the decomposition produces the components `U, V, Q, C, S, and R`.

Interface 2 As used elsewhere in Julia, we provide another interface that overrides input matrices.

```
svd!(A, B) -> GeneralizedSVD
```

`svd!` is the same as `svd`, but modifies the arguments `A` and `B` in-place, instead of making copies.

3.2 Architecture

We implement the GSVD algorithm described in the previous section in Julia 1.3 using `Float64` data. The structural unit called `Module` is native to Julia to group relevant functions and definitions. Considering that the CS decomposition not only serves as a building block for our GSVD algorithm, but is also a powerful tool in other applications, it is wise to separate CS decomposition as a standalone module called `CSD`. The main module is `GSVD`.

The algorithm starts from the main function `svd()` under module `GSVD`. It then calls `preproc()`. Once return, it calls `csd` intermodularly. Finally, the main function post processes to formulate the outputs.

$$\boxed{\texttt{svd()}} \rightarrow \boxed{\texttt{preproc()}} \rightarrow \boxed{\texttt{csd()}} \rightarrow \boxed{\texttt{svd():postproc}}$$

3.3 Implementation details

Step 1 Pre-processing:

This step is to reduce two input matrices A and B into two upper triangular forms. This is done via a call to `preproc()`. This function makes use of three fundamental orthogonal decompositions. First is QR decomposition with column pivoting to reveal the numerical rank of B and $[A; B]$ without forming the matrix explicitly. This is done by a call to `qr(A, pivot=Val(true)))`. Second is RQ decomposition via a call to `LAPACK.gerqf!()`. Last is QR decomposition by calling `qr()`. Upon return to `svd()`, two of the upper triangular matrices overwrites A and B, the orthogonal matrices are placed in U, V, and Q and rank information is stored in K and L.

Step 2 QR decomposition:

This step is to reduce two upper triangular matrices to one and is done by directly calling `qr()`. On exit, Q_1 and Q_2 overwrites A and B.

Step 3 CS decomposition:

This step calls `csd()` from module `CSD`. This function requires SVD, QR decomposition and QL decomposition and is done by calls to `svd()`, `qr()` and `LAPACK.geqlf!()` respectively. it return U_1, V_1, Z_1, C, S on exit.

Step 4 Post-processing: In this step, we update matrix U, V and Q by matrix-matrix multiply. To formulate R, we utilize RQ decomposition via a call to `LAPACK.gerqf!()`. Finally, we put matrices U, V, C, S, Q and K, L into the constructor of `GeneralizedSVD` as return.

3.4 GSVD in other languages: a comparison

We list several major languages that feature GSVD, shown in Table 1.

Language	GSVD Documentation
Native Julia (proposed)	`svd(A, B) -> GeneralizedSVD` Computes the generalized SVD of `A` and `B`, returning a `GSVD` factorization object `F`, such that `A = F.U*F.C*F.R*F.Q'` and `B = F.V*F.S*F.R*F.Q'`.
Julia 1.3 (LAPACK wrapper)	`svd(A, B) -> GeneralizedSVD` Computes the generalized SVD of `A` and `B`, returning a `GeneralizedSVD` factorization object `F`, such that `A = F.U*F.D1*F.R0*F.Q'` and `B = F.V*F.D2*F.R0*F.Q'`.

MATLAB (2019b)	`[U,V,X,C,S] = gsvd(A,B)`
	Returns unitary matrices `U` and `V`, a (usually) square matrix `X`, and nonnegative diagonal matrices `C` and `S` so that
	`A = U*C*X', B = V*S*X', C'*C + S'*S = I.`

Mathematica	`SingularValueDecomposition[m,a]` Gives a list of matrices {`u,ua,w,wa,v`} such that `m` can be written as `u.w.Conjugate[Transpose[v]]` and `a` can be written as `ua.wa.Conjugate[Transpose[v]]`.
R (geigen v2.3, LAPACK wrapper)	`z <- gsvd(A, B)` Computes The Generalized Singular Value Decomposition of matrices A and B such that $A = UD_1[0\ R]Q^T$ and $B = VD_2[0R]Q^T$. Note that the return value is the same as the output of LAPACK 3.6 and above.
Python (R. Luo's thesis)	Didn't disclose API design. The author defined GSVD as follows: Given two M_i-by-N column-matched but row-independent matrices D_i, each with full column rank and $N \leq Mi$, the GSVD is an exact simultaneous factorization $Di = Ui\Sigma_i V^T, i = 1, 2$. U_i is M_i-by-N and are column-wise orthonormal and V is N-by-N nonsingular matrix with normalized rows. $diag(\Sigma_i)$ returns two lists of N positive values and the ratios are called the generalized singular values.

Table 1: GSVD in different languages

4 Testing and Performance

4.1 Accuracy (backward stability)

Metric. We define the following metrics in order to test backward stability:

$$res_A = \frac{\|U^T AQ - CR\|_1}{max(m,n)\|A\|_1 \epsilon} \tag{14}$$

$$res_b = \frac{\|V^T BQ - SR\|_1}{max(p,n)\|B\|_1 \epsilon} \tag{15}$$

$$orth_U = \frac{\|I - U^T U\|_1}{m\epsilon} \tag{16}$$

$$orth_V = \frac{\|I - V^T V\|_1}{p\epsilon} \tag{17}$$

$$orth_Q = \frac{\|I - Q^T Q\|_1}{n\epsilon} \tag{18}$$

where ϵ is machine precision of input data type.

4.1.1 Numerical examples of small matrices

As documented in Section 1.1, we carry out numerical experiment on the two cases of the structures of C and S on small matrix pairs.

Example 1. Consider a 5-by-4 matrix A and a 3-by-4 matrix B:

$$A = \begin{bmatrix} 1 & 2 & 3 & 0 \\ 5 & 4 & 2 & 1 \\ 0 & 3 & 5 & 2 \\ 2 & 1 & 3 & 3 \\ 2 & 0 & 5 & 3 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 & 3 & -1 \\ -2 & 5 & 0 & 1 \\ 4 & 2 & -1 & 2 \end{bmatrix}$$

We obtain $k = 1$ and $l = 3$ from the computation of GSVD of A and B. Since $m = 5$ and $m \geq k + l$, C and S should fall into case (1) in Section 2.1. This is verified below.

```
C factor:
5×4 Array{Float64,2}:
 1.0  0.0       0.0       0.0
 0.0  0.894685  0.0       0.0
 0.0  0.0       0.600408  0.0
 0.0  0.0       0.0       0.27751
 0.0  0.0       0.0       0.0
```

```
S factor:
3×4 Array{Float64,2}:
 0.0  0.446698  0.0       0.0
 0.0  0.0       0.799694  0.0
 0.0  0.0       0.0       0.960723
```

Listing 1: C and S of Example 1 in proposed version

The computed orthogonal matrices U, V, Q, the R matrix and the generalized singular values are:

```
U factor:
5×5 Array{Float64,2}:
 -0.060976   -0.446679   -0.448921  -0.482187  -0.602266
  0.0904806  -0.867093    0.416172   0.115882   0.230944
 -0.481907   -0.212508   -0.636747   0.477322   0.298869
 -0.523214    0.0347528   0.410748   0.420777  -0.615851
 -0.69434     0.0475385   0.226075  -0.590913   0.339624
V factor:
3×3 Array{Float64,2}:
 -0.804633  -0.328486  -0.494634
 -0.288044  -0.512512   0.808927
 -0.519227   0.793365   0.317765
Q factor:
4×4 Array{Float64,2}:
  0.214542   0.484366    0.833941  -0.15461
  0.259709   0.413752   -0.147691   0.85997
 -0.361334   0.767117   -0.413972  -0.331052
 -0.86946   -0.0756949   0.333702   0.356304
R factor:
4×4 Array{Float64,2}:
 5.74065  -7.07986       0.125979     -0.316232
 0.0      -7.96103      -2.11852      -2.98601
 0.0      -4.44089e-16   5.72211      -0.43623
 0.0       1.33227e-15  -8.88178e-16   5.66474
Generalized singular values:
4-element Array{Float64,1}:
 Inf
   2.0028872436786482
   0.7507971450334572
   0.2888559753309598
```

Listing 2: Other products of Example 1 in proposed version

Likewise, we test GSVD in Julia 1.3 with the same inputs. For the numerical rank, $k = 1$ and $l = 3$. $D1$

and $D2$ (equivalent to C and S in the proposed version) are:

```
julia> Matrix(F.D1)
5×4 Array{Float64,2}:
 1.0  0.0       0.0       0.0
 0.0  0.894685  0.0       0.0
 0.0  0.0       0.600408  0.0
 0.0  0.0       0.0       0.27751
 0.0  0.0       0.0       0.0

julia> Matrix(F.D2)
3×4 Array{Float64,2}:
 0.0  0.446698  0.0       0.0
 0.0  0.0       0.799694  0.0
 0.0  0.0       0.0       0.960723
```

Listing 3: $D1$ and $D2$ of Example 1 in Julia 1.3

The computed orthogonal matrices U, V, Q, the $R0$ matrix (equivalent to R in the proposed version) are:

```
julia> F.U
5×5 Array{Float64,2}:
 -0.060976   -0.446679   -0.448921   0.482187  -0.602266
  0.0904806  -0.867093    0.416172  -0.115882   0.230944
 -0.481907   -0.212508   -0.636747  -0.477322   0.298869
 -0.523214    0.0347528   0.410748  -0.420777  -0.615851
 -0.69434     0.0475385   0.226075   0.590913   0.339624

julia> F.V
3×3 Array{Float64,2}:
 -0.804633  -0.328486   0.494634
 -0.288044  -0.512512  -0.808927
 -0.519227   0.793365  -0.317765

julia> F.Q
4×4 Array{Float64,2}:
  0.214542   0.484366   -0.833941   0.15461
  0.259709   0.413752    0.147691  -0.85997
 -0.361334   0.767117    0.413972   0.331052
 -0.86946   -0.0756949  -0.333702  -0.356304

julia> F.R0
```

```
4×4 Array{Float64,2}:
 5.74065  -7.07986  -0.125979  0.316232
 0.0      -7.96103   2.11852   2.98601
 0.0       0.0      -5.72211   0.43623
 0.0       0.0       0.0       5.66474
```

Listing 4: Other products of Example 1 in Julia 1.3

Results from MATLAB.

```
C =

    0.2775         0         0         0
         0    0.6004         0         0
         0         0    0.8947         0
         0         0         0    1.0000
         0         0         0         0


S =

    0.9607         0         0         0
         0    0.7997         0         0
         0         0    0.4467         0


U =

    0.4822   -0.4489   -0.4467   -0.0610   -0.6023
   -0.1159    0.4162   -0.8671    0.0905    0.2309
   -0.4773   -0.6367   -0.2125   -0.4819    0.2989
   -0.4208    0.4107    0.0348   -0.5232   -0.6159
    0.5909    0.2261    0.0475   -0.6943    0.3396


V =

    0.4946   -0.3285   -0.8046
   -0.8089   -0.5125   -0.2880
   -0.3178    0.7934   -0.5192


X =

    0.8758    4.8394   -5.1611   -2.0437
   -4.8715   -1.2203   -5.5489   -1.7290
```

```
    1.8753   -2.2244   -4.2415   -7.4528
   -2.0184    1.7541   -1.1683   -4.5260

sigma =

    0.2889
    0.7508
    2.0029
       Inf
```

Example 2. Consider a 3-by-4 matrix A and a 4-by-4 matrix B but with rank deficiency:

$$A = \begin{bmatrix} 1 & 2 & 1 & 0 \\ 2 & 3 & 1 & 1 \\ 3 & 4 & 1 & 2 \end{bmatrix}, \quad B = \begin{bmatrix} 4 & 5 & 1 & 3 \\ 5 & 6 & 1 & 4 \\ 6 & 7 & 1 & 5 \\ 7 & 1 & -6 & 13 \end{bmatrix}$$

Upon execution of GSVD of A and B, we get $k = 0$ and $l = 2$. This means that both B and $[A; B]$ are not in full rank. We shall check the structures of C and S and find it complies with that of case (1) in Section 1.1 when $m \geq k + l$.

```
C factor:
3×2 Array{Float64,2}:
 0.476231  0.0
 0.0       0.0697426
 0.0       0.0
S factor:
4×2 Array{Float64,2}:
 0.87932  0.0
 0.0      0.997565
 0.0      0.0
 0.0      0.0
```

Listing 5: C and S of Example 2 in proposed version

The computed orthogonal matrices U, V, Q, the R matrix and the generalized singular values are:

```
U factor:
3×3 Array{Float64,2}:
 -0.409031   0.816105  -0.408248
```

```
 -0.56342    0.126058   0.816497
 -0.71781   -0.563988  -0.408248
V factor:
4×4 Array{Float64,2}:
 -0.472375  -0.0876731  -0.390874   -0.785107
 -0.55599   -0.135916   -0.53894     0.618017
 -0.639606  -0.184159    0.745532    0.0342253
  0.242159  -0.969498   -0.0307137  -0.0221441
Q factor:
4×4 Array{Float64,2}:
 -0.436701  -0.689898   0.299328   0.493696
  0.563299   0.126599   0.793024   0.194368
 -0.689898   0.436701   0.493696  -0.299328
 -0.126599   0.563299  -0.194368   0.793024
R factor:
2×4 Array{Float64,2}:
 0.0  0.0  -12.2133        -8.28663
 0.0  0.0    3.55271e-15  -18.1154
Generalized singular values:
2-element Array{Float64,1}:
 0.5415903238738987
 0.06991284853891487
```

Listing 6: Other products of Example 2 in proposed version

It is easily verified that R has 2 zero columns in the leftmost since $k + l < n$.

Again, same inputs are tested in Julia 1.3. For the numerical rank, $k = 0$ and $l = 2$. $D1$ and $D2$ (equivalent to C and S in the proposed version) are:

```
julia> Matrix(F.D1)
3×2 Array{Float64,2}:
 0.476231  0.0
 0.0       0.0697426
 0.0       0.0

julia> Matrix(F.D2)
4×2 Array{Float64,2}:
 0.87932  0.0
 0.0      0.997565
 0.0      0.0
 0.0      0.0
```

Listing 7: $D1$ and $D2$ of Example 2 in Julia 1.3

The computed orthogonal matrices U, V, Q, the $R0$ matrix (equivalent to R in the proposed version) are:

```
julia> F.U
3×3 Array{Float64,2}:
 0.409031   0.816105  -0.408248
 0.56342    0.126058   0.816497
 0.71781   -0.563988  -0.408248

julia> F.V
4×4 Array{Float64,2}:
  0.472375  -0.0876731  -0.390874   -0.785107
  0.55599   -0.135916   -0.53894     0.618017
  0.639606  -0.184159    0.745532    0.0342253
 -0.242159  -0.969498   -0.0307137  -0.0221441

julia> F.Q
4×4 Array{Float64,2}:
 -0.436701  -0.689898  -0.299328   0.493696
  0.563299   0.126599  -0.793024   0.194368
 -0.689898   0.436701  -0.493696  -0.299328
 -0.126599   0.563299   0.194368   0.793024

julia> F.R0
2×4 Array{Float64,2}:
 0.0  0.0  -12.2133    8.28663
 0.0  0.0    0.0     -18.1154
```

Listing 8: Other products of Example 2 in Julia 1.3

It is clear that the leftmost 2 columns of $R0$ is all zeros.

Results from MATLAB.

```
C =

         0    0.0460         0         0
         0         0    0.6490         0
         0         0         0    0.9946


S =

    1.0000         0         0         0
```

```
         0    0.9989         0         0
         0         0    0.7608         0
         0         0         0    0.1039


U =


    0.0438    0.0710    0.9965
   -0.7618   -0.6430    0.0793
    0.6464   -0.7626    0.0259



V =


    0.0621    0.0228   -0.8563    0.5121
   -0.1574    0.3650   -0.4722   -0.7868
   -0.4326    0.8097    0.1962    0.3445
    0.8855    0.4589    0.0720   -0.0075



X =


    3.0643    9.9974   -5.3968    1.2397
   -2.7768    8.4399   -7.4530    2.3475
   -5.8412   -1.5575   -2.0562    1.1078
    8.9055   11.5549   -3.3406    0.1319

sigma =

         0
    0.0460
    0.8531
    9.5769
```

Example 3. Let A be a 3-by-4 matrix and B be a 4-by-4 matrix:

$$A = \begin{bmatrix} 1 & 4 & 1 & 0 \\ 5 & 3 & 1 & 1 \\ 3 & 0 & 1 & 2 \end{bmatrix}, \quad B = \begin{bmatrix} 4 & 5 & 1 & 3 \\ -2 & 0 & 1 & 4 \\ 3 & 2 & 1 & -5 \\ 1 & 1 & -6 & 3 \end{bmatrix}$$

We obtain $k = 0$ and $l = 4$ from the computation of GSVD of A and B. Since $m = 3$ and $m < k + l$, C and S should fall into case (2) in Section 2.1. This is verified below.

```
C factor:
3×4 Array{Float64,2}:
 0.99144  0.0       0.0       0.0
 0.0      0.681061  0.0       0.0
 0.0      0.0       0.167854  0.0
S factor:
4×4 Array{Float64,2}:
 0.130566  0.0       0.0       0.0
 0.0       0.732227  0.0       0.0
 0.0       0.0       0.985812  0.0
 0.0       0.0       0.0       1.0
```

Listing 9: C and S of Example 3 in proposed version

The computed orthogonal matrices U, V, Q, the R matrix and the generalized singular values are:

```
U factor:
3×3 Array{Float64,2}:
 -0.519777  0.747619   0.413398
  0.470025  0.654341  -0.592381
  0.713378  0.113599   0.691511
V factor:
4×4 Array{Float64,2}:
  0.259832   0.927018    0.177229  -0.20424
 -0.733955   0.0402919   0.652334  -0.184789
 -0.597084   0.369645   -0.576157   0.418206
 -0.1931    -0.0487437  -0.459449  -0.865588
Q factor:
4×4 Array{Float64,2}:
 -0.685431  -0.564405  -0.459976  -0.00724571
  0.681731  -0.704114  -0.149854  -0.130423
 -0.127188  -0.380896   0.646684   0.648491
 -0.221923  -0.201466   0.589716  -0.749931
R factor:
4×4 Array{Float64,2}:
 -3.71474      -2.42556      -0.179891    -0.941672
 -7.20246e-16  -9.84284      -1.8323      -0.522579
 -8.91076e-17   2.04711e-15   6.16149     -1.43582
  1.84152e-15   1.41087e-15   1.2978e-15   8.05363
Generalized singular values:
```

```
4-element Array{Float64,1}:
 7.593384394490093
 0.930122554989402
 0.17026951585960612
 0.0
```

Listing 10: Other products of Example 3 in proposed version

Similarly, we test GSVD in Julia 1.3 with the same inputs. For the numerical rank, $k = 0$ and $l = 4$. $D1$ and $D2$ (equivalent to C and S in the proposed version) are:

```
julia> Matrix(F.D1)
3×4 Array{Float64,2}:
 0.99144  0.0       0.0       0.0
 0.0      0.681061  0.0       0.0
 0.0      0.0       0.167854  0.0

julia> Matrix(F.D2)
4×4 Array{Float64,2}:
 0.130566  0.0       0.0       0.0
 0.0       0.732227  0.0       0.0
 0.0       0.0       0.985812  0.0
 0.0       0.0       0.0       1.0
```

Listing 11: $D1$ and $D2$ of Example 3 in Julia 1.3

The computed orthogonal matrices U, V, Q, the $R0$ matrix (equivalent to R in the proposed version) are:

```
julia> F.U
3×3 Array{Float64,2}:
  0.519777  0.747619   0.413398
 -0.470025  0.654341  -0.592381
 -0.713378  0.113599   0.691511

julia> F.V
4×4 Array{Float64,2}:
 -0.259832   0.927018    0.177229   0.20424
  0.733955   0.0402919   0.652334   0.184789
  0.597084   0.369645   -0.576157  -0.418206
  0.1931    -0.0487437  -0.459449   0.865588

julia> F.Q
```

```
4×4 Array{Float64,2}:
 -0.685431  0.564405   0.459976   0.00724571
  0.681731  0.704114   0.149854   0.130423
 -0.127188  0.380896  -0.646684  -0.648491
 -0.221923  0.201466  -0.589716   0.749931

julia> F.R0
4×4 Array{Float64,2}:
 3.71474  -2.42556  -0.179891  -0.941672
 0.0       9.84284   1.8323     0.522579
 0.0       0.0      -6.16149    1.43582
 0.0       0.0       0.0        8.05363
```

Listing 12: Other products of Example 3 in Julia 1.3

Results from MATLAB.

```
C =

         0    0.1679         0         0
         0         0    0.6811         0
         0         0         0    0.9914


S =

    1.0000         0         0         0
         0    0.9858         0         0
         0         0    0.7322         0
         0         0         0    0.1306


U =

    0.4134   -0.7476    0.5198
   -0.5924   -0.6543   -0.4700
    0.6915   -0.1136   -0.7134


V =

    0.2042    0.1772   -0.9270   -0.2598
    0.1848    0.6523   -0.0403    0.7340
   -0.4182   -0.5762   -0.3696    0.5971
    0.8656   -0.4594    0.0487    0.1931
```

```
X =

    0.0584   -2.8237   -6.4020   -4.0048
    1.0504   -0.7361   -7.2732    0.6748
   -5.2227    3.0534   -2.2253   -0.6694
    6.0397    4.7103   -1.2944   -1.9132

sigma =

         0
    0.1703
    0.9301
    7.5934
```

Example 4. Given a 3-by-5 matrix A and a 4-by-5 matrix B which are rank deficient:

$$A = \begin{bmatrix} 1 & 4 & 2 & 3 & 0 \\ 3 & 4 & 0 & -2 & 1 \\ 4 & 7 & 5 & 6 & 3 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 4 & 2 & 3 & 0 \\ 2 & 5 & 3 & 4 & 1 \\ 3 & 6 & 4 & 5 & 2 \\ 0 & 1 & -1 & 3 & 1 \end{bmatrix}$$

Upon execution of GSVD of A and B, we get $k = 1$ and $l = 3$. This means that both B and $[A; B]$ are not in full rank. We shall check the structures of C and S and find it complies with that of case (2) in Section 1.1 when $m < k + l$.

```
C factor:
3×4 Array{Float64,2}:
 1.0  0.0       0.0       0.0
 0.0  0.849235  0.0       0.0
 0.0  0.0       0.605834  0.0
S factor:
4×4 Array{Float64,2}:
 0.0  0.528015  0.0       0.0
 0.0  0.0       0.795591  0.0
 0.0  0.0       0.0       1.0
 0.0  0.0       0.0       0.0
```

Listing 13: C and S of Example 4 in proposed version

The computed orthogonal matrices U, V, Q, the R matrix and the generalized singular values are:

```
U factor:
3×3 Array{Float64,2}:
 -2.22045e-16   0.355381     -0.934722
  1.0          -1.74736e-16  -1.8521e-16
 -2.2915e-16   -0.934722     -0.355381
V factor:
4×4 Array{Float64,2}:
  0.571577     -0.711781      1.07608e-17  -0.408248
 -0.120069     -0.564727     -2.13123e-16   0.816497
 -0.811716     -0.417673     -1.59451e-16  -0.408248
  1.38917e-16   1.22399e-16  -1.0           3.46945e-17
Q factor:
5×5 Array{Float64,2}:
 -0.735494  -0.356936  -0.479812    0.318474    3.59984e-16
  0.29657   -0.540179   0.367864    0.633716    0.288675
  0.130491   0.610611  -0.189162    0.700722   -0.288675
 -0.237256   0.432143   0.0711454   0.0435931   0.866025
  0.545689  -0.145639  -0.770462   -0.0637737   0.288675
R factor:
4×5 Array{Float64,2}:
 0.0  -4.24145  -0.880735       3.33933      -0.288675
 0.0   0.0       2.7394        -8.38306      -5.97906
 0.0   0.0      -1.77636e-15  -12.2122       -8.79399
 0.0   0.0      -4.996e-16      2.22045e-16  -3.4641
Generalized singular values:
4-element Array{Float64,1}:
 Inf
   1.6083530545973714
   0.7614900645668164
   0.0
```

Listing 14: Other products of Example 4 in proposed version

We shall verify that R has a zero column in the leftmost since $k + l < n$.

Again, same inputs are tested in Julia 1.3. For the numerical rank determination, $k = 1$ and $l = 3$. $D1$ and $D2$ (equivalent to C and S in the proposed version) are:

```
julia> Matrix(F.D1)
3×4 Array{Float64,2}:
 1.0  0.0       0.0       0.0
 0.0  0.849235  0.0       0.0
 0.0  0.0       0.605834  0.0
```

```
julia> Matrix(F.D2)
4×4 Array{Float64,2}:
 0.0  0.528015  0.0       0.0
 0.0  0.0       0.795591  0.0
 0.0  0.0       0.0       1.0
 0.0  0.0       0.0       0.0
```

Listing 15: $D1$ and $D2$ of Example 4 in Julia 1.3

The computed orthogonal matrices U, V, Q, the $R0$ matrix (equivalent to R in the proposed version) are:

```
julia> F.U
3×3 Array{Float64,2}:
 -2.22045e-16  -0.355381     -0.934722
  1.0           1.74736e-16  -1.8521e-16
 -2.2915e-16    0.934722     -0.355381

julia> F.V
4×4 Array{Float64,2}:
 -0.571577     -0.711781      1.94289e-16  -0.408248
  0.120069     -0.564727      2.35922e-16   0.816497
  0.811716     -0.417673     -1.82146e-17  -0.408248
  7.69338e-17   2.44055e-16   1.0           3.46945e-17

julia> F.Q
5×5 Array{Float64,2}:
 -0.735494  -0.356936  -0.479812   -0.318474   -1.66533e-16
  0.29657   -0.540179   0.367864   -0.633716   -0.288675
  0.130491   0.610611  -0.189162   -0.700722    0.288675
 -0.237256   0.432143   0.0711454  -0.0435931  -0.866025
  0.545689  -0.145639  -0.770462    0.0637737  -0.288675

julia> F.R0
4×5 Array{Float64,2}:
 0.0  -4.24145  -0.880735  -3.33933   0.288675
 0.0   0.0      -2.7394    -8.38306  -5.97906
 0.0   0.0       0.0       12.2122    8.79399
 0.0   0.0       0.0        0.0      -3.4641
```

Listing 16: Other products of Example 4 in Julia 1.3

It is clear that the leftmost column of $R0$ is all zeros.

Results from MATLAB.

```
C =

         0         0    0.8178         0         0
         0         0         0    0.9995         0
         0         0         0         0    1.0000

S =

    1.0000         0         0         0         0
         0    1.0000         0         0         0
         0         0    0.5755         0         0
         0         0         0    0.0312         0

U =

   -0.1968    0.9805    0.0000
    0.0000   -0.0000    1.0000
   -0.9805   -0.1968   -0.0000

V =

   -0.8338         0    0.3365    0.4376
   -0.5289    0.0000   -0.2600   -0.8079
   -0.1581    0.0000   -0.9051    0.3947
   -0.0000   -1.0000   -0.0000   -0.0000

X =

   -2.3660    0.0000   -5.0363    0.1935    3.0000
   -6.9285   -1.0000   -9.3550    2.5457    4.0000
   -3.8868    1.0000   -6.4759    0.9776    0.0000
   -5.4077   -3.0000   -7.9154    1.7617   -2.0000
   -0.8451   -1.0000   -3.5968   -0.5906    1.0000

sigma =

         0
         0
    1.4209
```

```
  32.0780
      Inf
```

Conclusion. We find that in all cases, the computed C and S by our proposed version and Julia 1.3 are exactly the same. U, V, Q and R are mostly the same except for sign difference in certain columns.

However, the results computed by MATLAB bear less resemblance. On one hand, when the input matrices are of full rank (Case 1 and 3), the diagonal entries of C and S, and the generalized singular values produced by MATLAB are the same as those computed by proposed version and Julia 1.3, but in a reversed ordering. On the other hand, when the input matrices are rank deficient (Case 2 and 4), neither the diagonal entries of C and S nor the generalize singular values produced by MATLAB and Julia share anything in common, regardless of numerical values or length.

We also record the stability metrics computed by both versions in Julia in Table 2.

	Version	res_A	res_B	$orth_U$	$orth_V$	$orth_Q$
Example 1	proposed	0.2956	0.5646	0.5308	1.0417	1.1790
	Julia 1.3	0.3599	0.4571	0.9117	1.7083	1.3250
Example 2	proposed	0.6173	0.4098	1.5000	0.5613	1.3998
	Julia 1.3	0.5068	0.5689	1.4583	0.9245	1.2483
Example 3	proposed	0.4181	0.8941	0.7500	1.3940	1.3277
	Julia 1.3	0.3536	0.5938	1.4791	1.9540	1.1062
Example 4	proposed	0.3600	0.5900	0.6558	0.5385	1.4362
	Julia 1.3	0.4449	0.3056	1.3225	0.7205	1.1814

Table 2: Stability profiling for small matrices

4.1.2 Random dense matrices

Test matrix generation. As discussed in Section 1.1, we test stability on four cases depending on the row and column size of the input matrix pair. In this section, we test random dense matrices of `Float64`. For each case, we choose four subcases from low to high matrix size. We generate a total of 320 random matrix pairs, 20 for each subcase.

Results. As a demonstration, we list the results of five stability metrics for each subcase of a single test run in Table 3. All 320 test runs yield results no greater than two.

	m	p	n	$k+l$	res_A	res_B	$orth_U$	$orth_V$	$orth_Q$
$m \geq n$ $p \geq n$	60	50	40	40	0.1607	0.2710	0.7924	1.0079	0.4609
	300	250	200	200	0.0369	0.0484	0.5041	0.6408	0.3202
	900	750	600	600	0.0181	0.0193	0.3952	0.5157	0.2307
	1500	1250	1000	1000	0.0120	0.0142	0.3702	0.4129	0.1847
$m \geq n > p$	60	40	50	50	0.1529	0.2261	0.7653	1.1960	0.6074
	300	200	250	250	0.0412	0.0620	0.5559	0.7492	0.3150
	900	600	750	750	0.0169	0.0232	0.4174	0.5250	0.2411
	1500	1000	1250	1250	0.0122	0.0160	0.3726	0.4723	0.2080
$p \geq n > m$	40	60	50	50	0.1672	0.2028	1.1293	0.9373	0.4217
	200	300	250	250	0.0595	0.0530	0.7064	0.5855	0.3065
	600	900	750	750	0.0231	0.0231	0.5178	0.4186	0.2112
	1000	1500	1250	1250	0.0164	0.0153	0.4543	0.3673	0.1778
$n > m$ $n > p$	20	30	60	50	0.0483	0.0464	0.5472	0.5358	0.4547
	200	300	600	500	0.0120	0.0105	0.3036	0.3030	0.2374
	400	600	1200	1000	0.0081	0.0072	0.2888	0.2813	0.2315
	1000	1500	3000	2500	0.0053	0.0047	0.2700	0.2605	0.2410

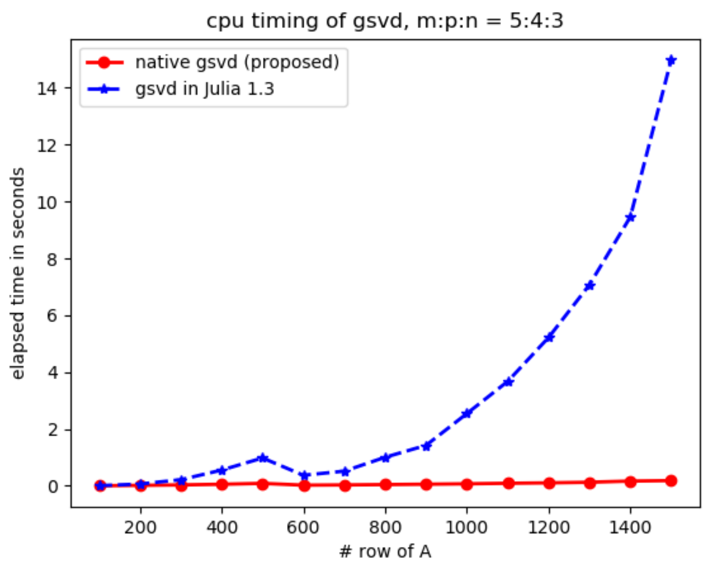
Table 3: Stability profiling for random dense matrices

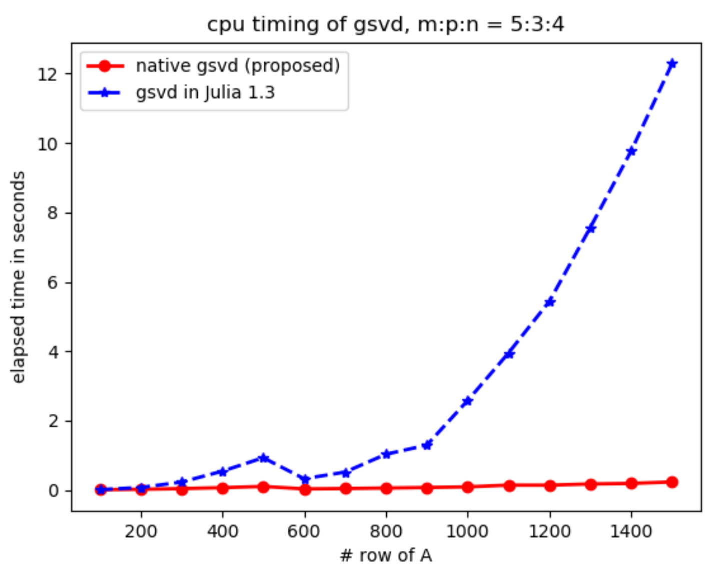
4.1.3 Special types of matrices

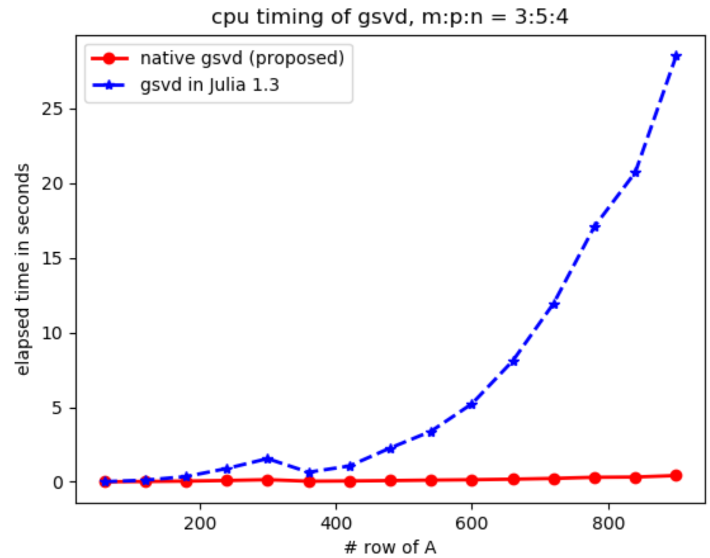
4.2 Timing

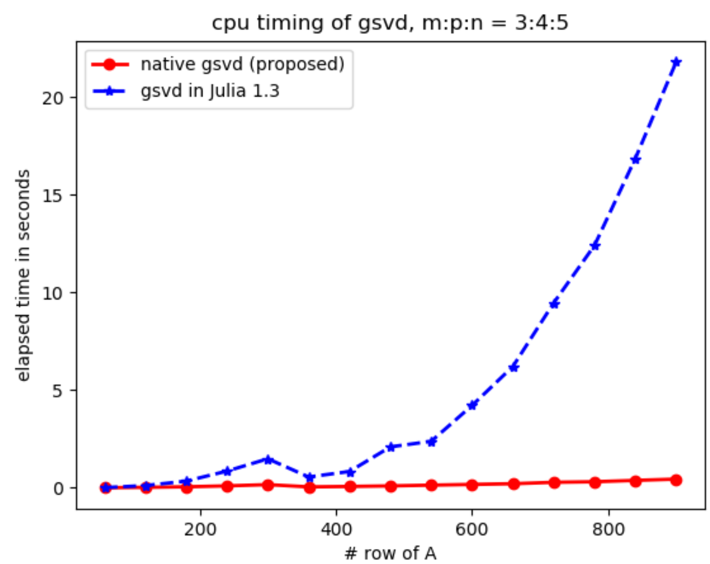
We want to evaluate the timing performance of our implementation between current version in Julia and MATLAB.

vs. Julia 1.3 For the comparison with Julia 1.3, we also spilt into four cases. Each case, we calculated the average CPU timing of 10 runs. In all cases, we can see that the speedup is exponential when input size is greater than a few hundreds.

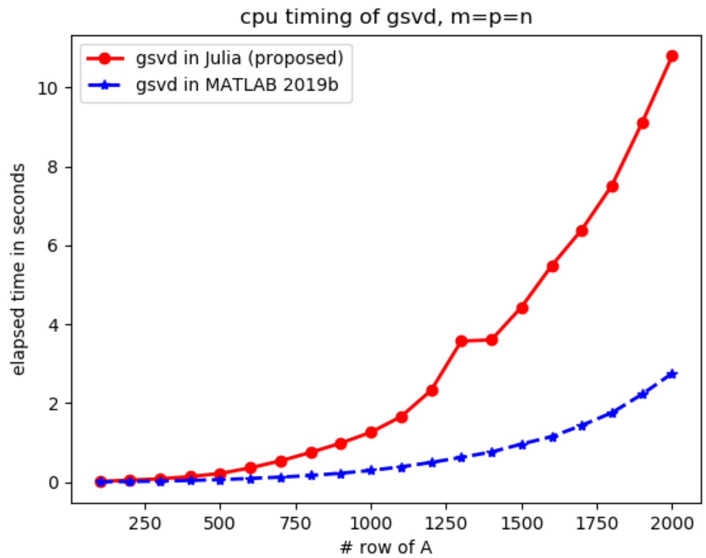








vs. MATLAB. For the comparison with MATLAB 2019b, we specify the input as square matrix. Our implementation is still slower than MATLAB. The major reason is due to the significant difference of decomposition discussed in 1.1 and 1.3.2.


cpu timing of gsvd, m=p=n
gsvd in Julia (proposed)
gsvd in MATLAB 2019b
elapsed time in seconds
row of A
0
2
4
6
8
10
250
500
750
1000
1250
1500
1750
2000

Profile. As detailed in 2.1, our algorithm insists of four parts: pre-processing, QR, CSD and post-processing. Here, we measure the CPU time spent in the first three parts and total time, denoted as t_{pre}, t_{qr}, t_{csd} and t_{all} and calculated the percentages that each part spent to total time, denoted as p_{pre}, p_{qr}, p_{csd}. Still, we separate our test into four cases and record the average of 10 test runs. In most cases, pre-processing dominates the computation effort. This motivates us to explore time profiling of pre-processing.

	m	p	n	t_{pre}	p_{pre}	t_{qr}	p_{qr}	t_{csd}	p_{csd}	t_{all}
$m \geq n$ $p \geq n$	1500	1200	1000	0.6242	41.13%	0.1683	11.09%	0.6011	39.61%	1.5175
	500	500	500	0.0651	26.78%	0.0347	14.29%	0.1191	48.94%	0.2433
	650	310	230	0.0418	54.63%	0.0084	11.08%	0.0195	25.47%	0.0766
	430	610	210	0.0345	47.65%	0.0067	9.25%	0.0247	34.11%	0.0725
$m \geq n > p$	1500	1000	1200	1.500	60.09%	0.1815	7.27%	0.6811	27.28%	2.4963
	720	220	540	0.1182	73.65%	0.0074	4.61%	0.0256	15.94%	0.1605
	440	180	440	0.0651	65.84%	0.0053	5.37%	0.0221	22.41%	0.0989
	370	290	350	0.0659	51.61%	0.0123	9.65%	0.0400	31.34%	0.1278
$p \geq n > m$	1000	1500	1200	0.5234	23.23%	0.2789	12.37%	1.2630	56.06%	2.2529
	250	300	300	0.0205	24.96%	0.0129	15.75%	0.0397	48.25%	0.0822
	360	660	600	0.0645	18.33%	0.0436	12.39%	0.2103	59.72%	0.3521
	130	520	480	0.0311	14.52%	0.0215	10.02%	0.1391	64.79%	0.2146
$n > m$ $n > p$	1000	1200	1500	1.7532	48.51%	0.2038	5.64%	1.4467	40.03%	3.6136
	260	600	770	0.2791	38.86%	0.0441	6.14%	0.3459	48.17%	0.7181
	370	250	700	0.1385	86.69%	0	0%	0	0%	0.1598
	120	120	400	0.0296	96.70%	0	0%	0	0%	0.0307

Table 4: Time profiling for GSVD

Pre-processing. To avoid skipping steps in pre-processing, we use rank-deficient matrix as input of B. Likewise the time profiling of GSVD, we record absolute time spent in each part and the relative percentage to total time. The meaning of subscript in Table 5 is explained below:

1. $qrpB$: QR decomposition with column pivoting of B.
2. $genV$: Generate V.
3. $updateA1st$: First time to update A.
4. $genQ$: Geneate Q.
5. rqB: RQ decomposition of B.
6. $updateA2nd$: Second time to update A.
7. $updateQ1st$: First time to update Q.
8. $qrpA$: QR decomposition with column pivoting of A.
9. $genU$: Generate U.
10. $updateA3rd$: Third time to update A.
11. $updateQ2nd$: Second time to update Q.
12. rqA: RQ decomposition of A.
13. $updateQ3rd$: Third time to update Q.
14. qrA: QR decomposition of A.
15. $updateU$: Update U.

	$m = 1200, p = 1000, n = 900$ $l = 800, k = 100$	$m = 500, p = 500, n = 600$ $l = 400, k = 200$	$m = 250, p = 200, n = 200$ $l = 150, k = 50$
t_{qrpB} (p-by-n)	0.036821	0.018432	0.002894
p_{qrpB}	15.29%	21.59%	11.19%
t_{genV} (p-by-p)	0.022350	0.006850	0.001578
p_{genV}	9.28%	8.02%	6.10%
$t_{updateA1st}$ (m-by-n)	0.012765	0.005162	0.000736
$p_{updateA1st}$	5.30%	6.05%	2.84%
t_{genQ} (n-by-n)	0.002553	0.001187	0.000195
p_{genQ}	1.06%	1.39%	0.75%
t_{rqB} (l-by-n)	0.024456	0.010305	0.001856
p_{rqB}	10.16%	12.07%	7.18%
$t_{updateA2nd}$ (m-by-n)	0.019261	0.005071	0.000781
$p_{updateA2nd}$	8.00%	5.94%	3.02%
$t_{updateQ1st}$ (n-by-n)	0.014279	0.005488	0.000732
$p_{updateQ1st}$	5.93%	6.43%	2.82%
t_{qrpA} (m-by-$n-l$)	0.002878	0.004063	0.000595
p_{qrpA}	1.20%	4.76%	2.30%
t_{genU} (m-by-m)	0.015431	0.007718	0.001051
p_{genU}	6.40%	9.04%	4.06%
$t_{updateA3rd}$ (m-by-l)	0.009105	0.002531	0.000412
$p_{updateA3rd}$	3.78%	2.96%	1.59%
$t_{updateQ2nd}$ (n-by-$n-l$)	0.000289	0.000871	0.000136
$p_{updateQ2nd}$	0.12%	1.02%	0.53%
t_{rqA} (k-by-$n-l$)	0	0	0
p_{rqA}	0%	0%	0%
$t_{updateQ3rd}$ (n-by-$n-l$)	0	0	0
$p_{updateQ3rd}$	0%	0%	0%
t_{qrA} ($m-k$-by-l)	0.022391	0.002823	0.001756
p_{qrA}	9.30%	4.76%	6.79%
$t_{updateU}$ (m-by-$m-k$)	0.022113	0.001799	0.000850
$p_{updateU}$	9.18%	2.11%	3.28%
t_{all}	0.240752	0.085373	0.025867

Table 5: Time profiling for Preprocessing

5 Applications

5.1 Linear discriminant analysis

Howland and Park [9] [10] applied the GSVD to discriminant analysis to overcome the limitation of nonsingular covariance matrices that are used to represent the scatter within and between clusted text data.

5.2 Genomic signal processing

The GSVD is applicable for comparative analysis of genome-scale expression datasets of two different organisms [11] and is further extended to tensor.

5.3 Tikhonov regularization

Tikhonov regularization in general form can be analyzed with the truncated GSVD when we are to solve the ill-posed linear least squares problem. [12] [13] [14] Computerized ionospheric tomography is one of the applications in this regard. [15]

5.4 Matrix pencil $A - \lambda B$

The GSVD is also used in the field of the canonical structure of matrix pencil $A - \lambda B$. [16] More specifically, the column and row nullities of A and B and common null space reveal the information about the Kronecker structure of $A - \lambda B$.

5.5 Generalized total least squares problem

By making use of the GSVD, one can solve the generalized TLS problem. TLS is also called error-in-variable regression in statistics domain. The great advantage of the GSVD is that it replaces these implicit transformation of data procedures by one, which is numerically reliable and can more easily handle (nearly) singular associated error covariance matrix. [17] [18]

References

[1] Charles F Van Loan. Generalizing the singular value decomposition. *SIAM Journal on Numerical Analysis*, 13(1):76–83, 1976.

[2] Alan Edelman and Yuyang Wang. The GSVD: Where are the ellipses?, matrix trigonometry, and more. *arXiv preprint arXiv:1901.00485*, 2019.

[3] Zhaojun Bai and James W Demmel. Computing the generalized singular value decomposition. *SIAM Journal on Scientific Computing*, 14(6):1464–1486, 1993.

[4] GH Golub and CF Van Loan. Matrix computations 4th edition the johns hopkins university press. *Baltimore, MD*, 2013.

[5] Zhaojun Bai and Hongyuan Zha. A new preprocessing algorithm for the computation of the generalized singular value decomposition. *SIAM Journal on Scientific Computing*, 14(4):1007–1012, 1993.

[6] Michael Stewart. Rank decisions in matrix quotient decompositions. *SIAM Journal on Matrix Analysis and Applications*, 37(4):1729–1746, 2016.

[7] Edward Anderson, Zhaojun Bai, Christian Bischof, L Susan Blackford, James Demmel, Jack Dongarra, Jeremy Du Croz, Anne Greenbaum, Sven Hammarling, Alan McKenney, et al. *LAPACK Users' guide*. SIAM, 1999.

[8] CC Paige. Computing the generalized singular value decomposition. *SIAM Journal on Scientific and Statistical Computing*, 7(4):1126–1146, 1986.

[9] Peg Howland, Moongu Jeon, and Haesun Park. Structure preserving dimension reduction for clustered text data based on the generalized singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 25(1):165–179, 2003.

[10] Hyunsoo Kim, Peg Howland, and Haesun Park. Dimension reduction in text classification with support vector machines. *Journal of machine learning research*, 6(Jan):37–53, 2005.

[11] Orly Alter, Patrick O Brown, and David Botstein. Generalized singular value decomposition for comparative analysis of genome-scale expression data sets of two different organisms. *Proceedings of the National Academy of Sciences*, 100(6):3351–3356, 2003.

[12] Per Christian Hansen. Regularization, gsvd and truncatedgsvd. *BIT numerical mathematics*, 29(3):491–504, 1989.

[13] Laura Dykes and Lothar Reichel. Simplified GSVD computations for the solution of linear discrete ill-posed problems. *Journal of Computational and Applied Mathematics*, 255:15–27, 2014.

[14] Yimin Wei, Pengpeng Xie, and Liping Zhang. Tikhonov regularization and randomized GSVD. *SIAM Journal on Matrix Analysis and Applications*, 37(2):649–675, 2016.

[15] K Bhuyan, SB Singh, and PK Bhuyan. Application of generalized singular value decomposition to ionospheric tomography. *Annales Geophysicae*, 22(10):3437–3444, 2004.

[16] Bo Kågström. The generalized singular value decomposition and the general ($\mathbf{A} - \lambda\mathbf{B}$)-problem. *BIT Numerical Mathematics*, 24(4):568–583, 1984.

[17] Sabine Van Huffel and Joos Vandewalle. Analysis and properties of the generalized total least squares problem $\mathbf{AX} \approx \mathbf{B}$ when some or all columns in $\mathbf{A}$ are subject to error. *SIAM Journal on Matrix Analysis and Applications*, 10(3):294, 1989.

[18] Zhaojun Bai. CSD, GSVD, their applications and computations. *IMA Preprint Series # 958*, 1992.