GSVD

Ji Wang

Contents

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 $\alpha_i,\ \beta_i \in [0,1]$ 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) $m \geq k + l$

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

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) $m < k + l$

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

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

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.

   We can also gain rank($[A; B]$) from Definition(1), (2) and (4). However, we cannot obtain such information from Definition(3).

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

   If we rewrite our formulation of GSVD as:

$$A(Q_1,\ Q_2) = UC(0,\ R_0), \quad B(Q_1,\ Q_2) = VS(0,\ R_0)$$

   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.

   We can also get the common nullspace of A and B from Definition(2) and (4).

   - If we rewrite the GSVD of Definition(2) as:

$$A(Q_1,\ Q_2) = UC(W^T R,\ 0), \quad B(Q_1,\ Q_2) = VS(W^T R,\ 0)$$

     where Q_1 is n-by-$(k+l)$, Q_2 is n-by-$(n-k-l)$. Then, we have null$(A) \cap$ null$(B) =$ span$\{Q_2\}$.
   - In Definition(4), 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$\{Q(:, 1 : n-k-l)\}$.

3. We can solve the generalized eigenvalue problem ($A^T Ax = \lambda B^T Bx$) 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

$$X^T A^T A X = \begin{array}{c|cc} & n-k-l & k+l \\ n-k-l & 0 & 0 \\ k+l & 0 & C^T C \end{array}, \quad X^T B^T B X = \begin{array}{c|cc} & n-k-l & k+l \\ n-k-l & 0 & 0 \\ k+l & 0 & S^T S \end{array}$$

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

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

   $\lambda_i = (\alpha_i/\beta_i)^2$ 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.

   We can solve the generalized eigenvalue problem from Definition(1), (2) and (4).

- In Definition(1),

$$\begin{aligned} X^TA^TAX &= X^T(UCX^{-1})^T(UCX^{-1})X \\ &= X^T(X^{-1})^TC^TU^TUCX^{-1}X \\ &= X^T(X^T)^{-1}C^TC \\ &= C^TC \end{aligned}$$

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

- In Definition(2),

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

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

  Thus, we know that the "non-trivial" eigenvalues of the generalized eigenvalue problem are the square of the generalized singular values and and the first $k+l$ columns of X are the corresponding eigenvectors.

- In Definition(4),

  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.

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:

$$A = UCQ^T, \quad B = VSQ^T$$

  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, and they are ordered below:

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 $r = \text{rank}([A;B])$, the generalized singualr value decomposition of A and B is:

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

where

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

- 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 $q = max\{r-p, 0\}$. $\alpha_1 = \cdots = \alpha_q = 1$, $\Sigma_1 = \text{diag}(\alpha_{q+1}, \cdots, \alpha_r)$, $\beta_1 = \cdots = \beta_q = 0$, $\Sigma_2 = \text{diag}(\beta_{q+1}, \cdots, \beta_r)$. $\Sigma_1^2 + \Sigma_2^2 = I$.

1.3.2 Definition(2): Paige (1981) [2]

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

$$A = UC(W^TR,\ 0)Q^T, \quad B = VS(W^TR,\ 0)Q^T$$

where

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

- U is an m-by-m orthogonal matrix.
- V is a p-by-p orthogonal matrix.
- W is an r-by-r orthogonal matrix.
- R is an r-by-r nonsingular matrix, its singular values are equal to the nonzero singular values of $[A;B]$. $\text{rank}(R) = \text{rank}([A;B])$.
- Q is an n-by-n orthogonal matrix.
- C and S are m-by-r and p-by-r. $k = \text{rank}([A;B])$ - $\text{rank}(B)$, $s = \text{rank}(A) + \text{rank}(B)$ - $\text{rank}([A;B])$. $\alpha_1 = \cdots = \alpha_k = 1$, $\Sigma_1 = \text{diag}(\alpha_{k+1}, \cdots, \alpha_{k+s})$, $\alpha_{k+s+1} = \cdots = \alpha_r = 0$, $\beta_1 = \cdots = \beta_k = 0$, $\Sigma_2 = \text{diag}(\beta_{k+1}, \cdots, \beta_{k+s})$, $\beta_{k+s+1} = \cdots = \beta_r = 1$. α_i/β_i are called the "non-trivial" generalized singular values of matrix pair A, B. $\Sigma_1^2 + \Sigma_2^2 = I$.

1.3.3 Definition(3): MATLAB 2019b

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

$$A = UCX^T, \quad B = VSX^T$$

- 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 matrix and S is a p-by-q. Both are nonnegative and $C^TC + S^TS = I$. The nonzero elements of S are always on its main diagonal. If $q > m$, the nonzero elements of C are on the $(q-m)$-th diagonal. Otherwise, they are on the main diagonal of C.
- $C^TC = \text{diag}(\alpha_1^2, \cdots, \alpha_q^2)$, $S^TS = \text{diag}(\beta_1^2, \cdots, \beta_q^2)$, where α_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.

1.3.4 Definition(4): Edelman (2019) [3]

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

$$A = UCH, \quad B = VSH$$

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

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

1.3.5 Link between Definition(1) and Definition(3)

MATLAB documents the algorithm as follows:

"The generalized singular value decomposition uses the CS decomposition described in [4], as well as the built-in `svd` and `qr` functions. The CS decomposition is implemented in a local function in the `gsvd` program file."

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:

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

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

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

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

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

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

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

(5) Let

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

Then QR decomposition with column pivoting of $A11$:

$$A_1 P_1 = U \begin{array}{c|cc} & k & n-l-k \\ k & A_{11} & A_{12} \\ m-k & 0 & 0 \end{array}$$

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

(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 $(A_{11} \quad A_{12})$:

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

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

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

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

- QR decomposition of A_{23}:

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

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

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

$$A = UR_AQ^T, \quad B = VR_BQ^T \tag{2}$$

where

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

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 $[A_{23}^T \ B_{13}^T]^T$:

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

Thus, (2) can be rewritten as:

$$A = U(Q_A\hat{R})Q^T, \quad B = V(Q_B\hat{R})Q^T \tag{3}$$

where

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

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:

$$Q_1 = U_1 C_1 Z_1^T, \quad Q_2 = V_1 S_1 Z_1^T$$

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

$$A = U(\hat{U} C \hat{Q}^T)\hat{R} Q^T, \quad B = V(\hat{V} S \hat{Q}^T)\hat{R} Q^T \tag{4}$$

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 2.3.

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:

$$A = UCRQ^T, \quad B = VSRQ^T \tag{5}$$

2.2 Other prominent algorithms

2.2.1 LAPACK algorithm

This algorithm has two phases. First is a pre-processing step described in 2.1. Next is a Jacobi-style method to directly compute the GSVD of two square upper trangular matrices. [6] [7]

2.2.2 Van Loan's algorithm

Golub and Van Loan [4] 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 $U^T AX = C$ and $V^T BX = S$.

Step 1 Compute the 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 CS Decomposition algorithm

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:

$$Q_1 = UCZ^T, \quad Q_2 = VSZ^T \tag{6}$$

- 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,
- $C^T C + S^T S = I$.

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

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

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

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

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

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

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

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

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

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.

2.4 Justification 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
```

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.D1*F.R0*F.Q’` and `B = F.V*F.D2*F.R0*F.Q’`.
For an M-by-N matrix `A` and P-by-N matrix `B`,

- `U` is a M-by-M orthogonal matrix,
- `V` is a P-by-P orthogonal matrix,
- `Q` is a N-by-N orthogonal matrix,
- `C` is a 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`.

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.

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

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.D1*F.R0*F.Q'` and `B = F.V*F.D2*F.R0*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. The following metrics are computed to test stability:

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

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

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

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

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

where ϵ is machine precision of input data type.

Test matrix generation. As discussed in 1.1, we test stability on four cases depending on the row and column dimension of the input matrix pair. For the time being, 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 2. All 320 test runs yield results no greater than two.

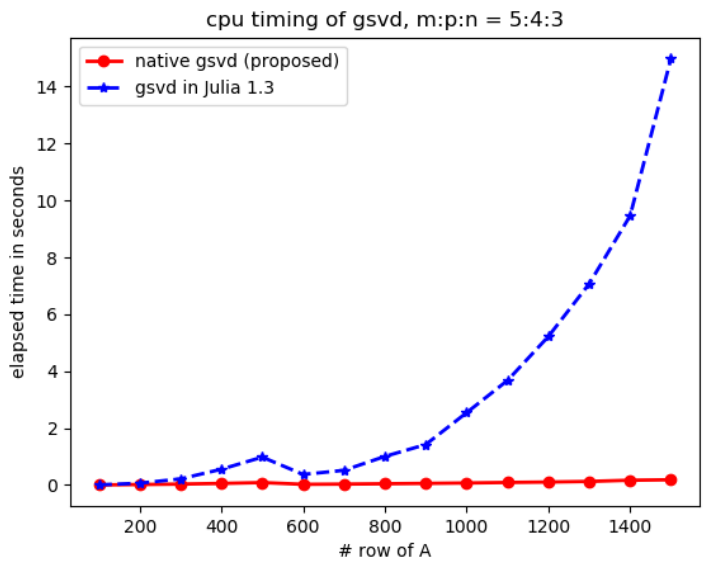
	m	p	n	r	res_A	res_B	$orth_U$	$orth_V$	$orth_Q$
$m \geq n$ $p \geq n$	60	50	40	40	0.1607	0.2710	0.7924	1.0079	0.4609
	300	250	200	200	0.0369	0.0484	0.5041	0.6408	0.3202
	900	750	600	600	0.0181	0.0193	0.3952	0.5157	0.2307
	1500	1250	1000	1000	0.0120	0.0142	0.3702	0.4129	0.1847
$m \geq n > p$	60	40	50	50	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

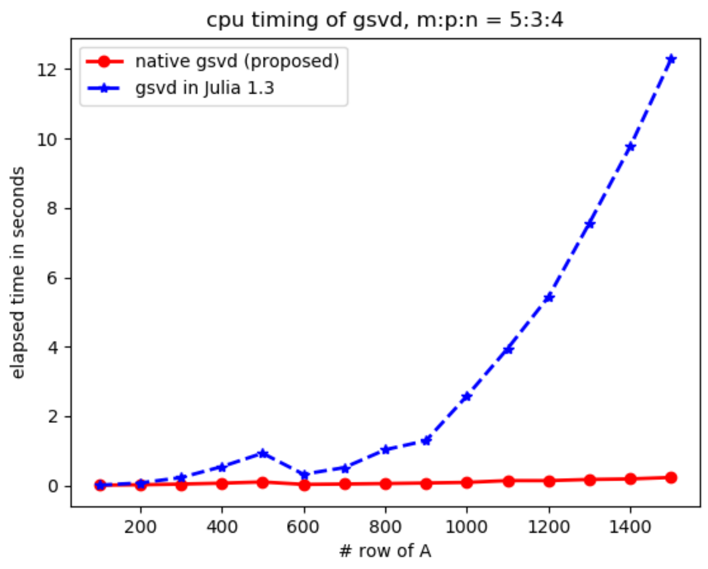
Table 2: Stability profiling for GSVD

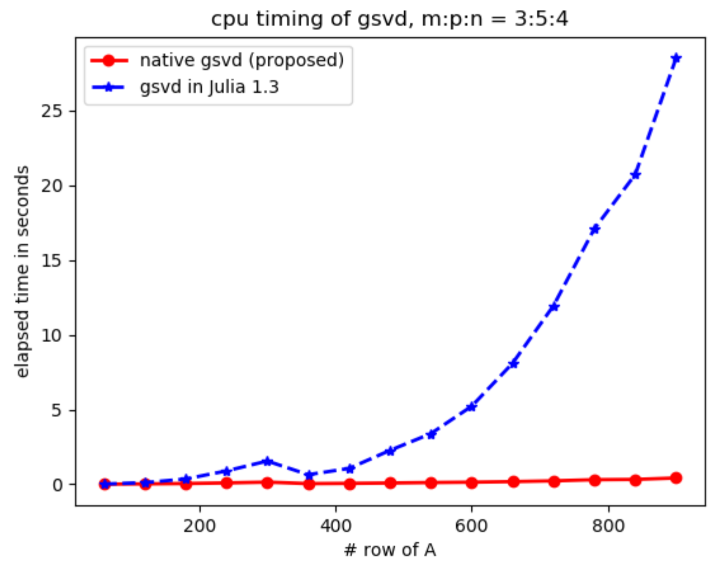
4.2 Timing

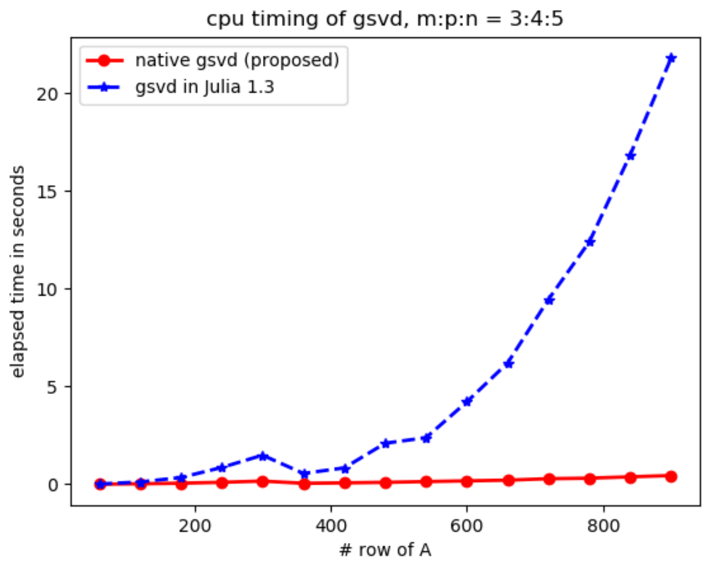
We want to evaluate the timing performance of our implementation between current version in Julia and MATLAB.

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.

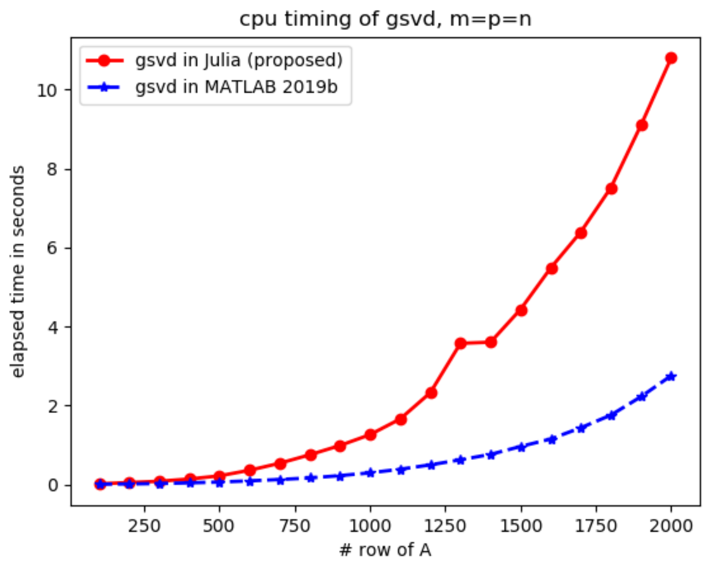








vs. MATLAB. For the comparison with MATLAB 2019b, we specify the input as square matrix. Our implementation is still slower than MATLAB. The major reason is due to the significant difference of decomposition discussed in 1.1 and 1.3.3.


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 3: 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 4 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 4: Time profiling for Preprocessing

5 Application

References

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

[2] Christopher C Paige and Michael A Saunders. Towards a generalized singular value decomposition. *SIAM Journal on Numerical Analysis*, 18(3):398–405, 1981.

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

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

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