Project Report

24300720157 Ruijie Li

December 21, 2025

Contents

1 Introduction

The Singular Value Decomposition (SVD) is a cornerstone of numerical linear algebra, with applications spanning data compression, signal processing, and machine learning. While the standard Golub–Kahan algorithm is numerically stable, its sequential nature limits performance on modern parrallel architectures. This led to the development of Divide-and-Conquer SVD algorithms (Cuppen, 1981), which recursively break the problem into subproblems, allowing for parallel execution and improved efficiency.

However, the naive implementation of the Divide-and-Conquer strategy suffers from a critical numerical instability. When singular values are closely spaced, the standard formulas for updating singular vectors are extremely sensitive to round-off errors, leading to significant inaccuracies in the computed SVD.

Fortunately, Gu and Eisenstat (1995) proposed a bidiagonal Divide-and-Conquer algorithm (BDC) that addresses these stability issues. Instead of directly computing the singular vectors from the original rank-1 perturbed system, their algorithm modifies the diagonal elements slightly to ensure that the secular equation's roots are well-separated. This adjustment significantly enhances numerical stability, allowing for accurate computation of singular values and vectors even in challenging scenarios. The BDC algorithm computes all the eigenvalues in $O(n^2)$ time and the singular vectors in $O(n^3)$ time, making it efficient for large-scale problems.

This report presents a complete implementation of the Gu–Eisenstat Divide-and-Conquer SVD algorithm. In section 2, we provide a detailed description of the algorithm, highlighting ... (to be filled in). Some important implementation details are discussed in section 3, including ... (to be filled in). Finally, section 4 showcases numerical experiments that validate the accuracy and efficiency of our implementation compared to Numpy's built-in SVD function.

2 Algorithm Description

Given a matrix $A \in \mathbb{C}^{m\times n}$, the goal of the Divide-and-Conquer SVD algorithm is to compute its singular value decomposition $A = U\Sigma V^\star$, where $U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$ are unitary matrices, and $\Sigma \in \mathbb{C}^{m\times n}$ is a diagonal matrix containing the singular values of A.

2.1 Bidiagonalization

The first step is to reduce A to a bidiagonal form using Householder transformations. This involves applying a series of unitary transformations from the left and right to zero out the appropriate elements, resulting in a matrix B such that $A = Q_1 B Q_2^\star$, where Q_1 and Q_2 are unitary matrices and B is lower bidiagonal if $m \geq n$ or upper bidiagonal if $m < n$. For simplicity, we will assume $m \geq n$ in the following discussion and experiments. Since bidiagonalization is a well-studied process, we will not delve into its details here.

Since bidiagonalization is a standard, well-studied process with a complexity of $O(mn^2)$, it typically dominates the computational cost of the entire SVD. **To isolate and rigorously evaluate the performance of the Divide-and-Conquer algorithm (which specifi-**

cally targets the bidiagonal phase), we will bypass this reduction step. In our implementation and experiments, we will generate random bidiagonal matrices B directly as inputs.

2.2 Divide-and-Conquer Strategy

The core idea of the BDC algorithm is to recursively divide the lower bidiagonal matrix $B \in \mathbb{R}^{(n+1)\times n}$ into smaller submatrices, compute their SVDs, and then merge the results to obtain the SVD of the original matrix. Given B, we divide B into two subproblems as follows:

$$B = \begin{bmatrix} B_1 & \alpha_k e_k & 0 \\ 0 & \beta_k e_1 & B_2 \end{bmatrix},$$

where $1 < k < n$, $B_1 \in \mathbb{R}^{k\times(k-1)}$ and $B_2 \in \mathbb{R}^{(n-k+1)\times(n-k)}$ are lower bidiagonal submatrices, and e_j is the j-th unit vector of appropriate dimension. The scalars α_k and β_k are the coupling elements that connect the two submatrices. Usually, we choose $k = \lfloor n/2 \rfloor$ to balance the sizes of the subproblems. We then recursively compute the SVDs of B_1 and B_2 and obtain:

$$B_i = [Q_i, q_i] \begin{bmatrix} D_i \\ 0 \end{bmatrix} W_i^{\mathrm{T}}, \quad i = 1, 2.$$

Notice that

$$\begin{aligned} \alpha_k e_k &= [Q_1, q_1] \begin{bmatrix} Q_1^{\mathrm{T}} \\ q_1^{\mathrm{T}} \end{bmatrix} \alpha_k e_k = [Q_1, q_1] \begin{bmatrix} Q_1^{\mathrm{T}} e_k \alpha_k \\ q_1^{\mathrm{T}} e_k \alpha_k \end{bmatrix} \\ \beta_k e_1 &= [Q_2, q_2] \begin{bmatrix} Q_2^{\mathrm{T}} \\ q_2^{\mathrm{T}} \end{bmatrix} \beta_k e_1 = [Q_2, q_2] \begin{bmatrix} Q_2^{\mathrm{T}} e_1 \beta_k \\ q_2^{\mathrm{T}} e_1 \beta_k \end{bmatrix}. \end{aligned}$$

Let l_1^{T} and λ_1 be the last row and last element of Q_1 and q_1, and similarly define f_2^{T} and φ_2 be the first row and first element of Q_2 and q_2. We can rewrite B as

$$B = \begin{bmatrix} q_1 & Q_1 & & \\ & & Q_2 & q_2 \end{bmatrix} \begin{bmatrix} \boxed{\alpha_k \lambda_1} & & \\ \alpha_k l_1 & D_1 & \\ \beta_k f_2 & & D_2 \\ \boxed{\beta_k \varphi_2} & & \end{bmatrix} \begin{bmatrix} & W_1 & \\ 1 & & \\ & & W_2 \end{bmatrix}^{\mathrm{T}}$$

There is only one nonzero element in the first and last columns of the middle matrix. Applying a Givens rotation to zero out $\beta_k \varphi_2$, we have

$$\begin{aligned} B &= \begin{bmatrix} q_1 & Q_1 & & \\ & & Q_2 & q_2 \end{bmatrix} \begin{bmatrix} c & & & -s \\ & I_{k-1} & & \\ & & I_{n-k} & \\ s & & & c \end{bmatrix} \begin{bmatrix} r_0 & & \\ \alpha_k l_1 & D_1 & \\ \beta_k f_2 & & D_2 \\ 0 & & \end{bmatrix} \begin{bmatrix} & W_1 & \\ 1 & & \\ & & W_2 \end{bmatrix}^{\mathrm{T}} \\ &= \left[\begin{pmatrix} c q_1 & Q_1 & \\ s q_2 & & Q_2 \end{pmatrix} \ \begin{pmatrix} -s q_1 \\ c q_2 \end{pmatrix} \right] \begin{bmatrix} r_0 & & \\ \alpha_k l_1 & D_1 & \\ \beta_k f_2 & & D_2 \\ 0 & & \end{bmatrix} \begin{bmatrix} & W_1 & \\ 1 & & \\ & & W_2 \end{bmatrix}^{\mathrm{T}} \\ &:= \begin{bmatrix} Q & q \end{bmatrix} \begin{bmatrix} M \\ 0 \end{bmatrix} W^{\mathrm{T}}, \end{aligned}$$

where c and s are the cosine and sine of the Givens rotation angle, and

$$r_0 = \sqrt{(\alpha_k \lambda_1)^2 + (\beta_k \varphi_2)^2}.$$

Thus, the problem reduces $\begin{bmatrix} M \\ 0 \end{bmatrix}$ by orthogonal transformations. Note that M has nonzero elements only on the diagonal and the first column. In next subsection, we will discuss how to compute the SVD of this matrix efficiently and stably. Now, suppose we have computed the SVD of M as $M = U\Omega V^{\mathrm{T}}$, then the SVD of B can be constructed as

$$B = \begin{bmatrix} Q & q \end{bmatrix} \begin{bmatrix} U\Omega V^{\mathrm{T}} \\ 0 \end{bmatrix} W^{\mathrm{T}} := X \begin{bmatrix} \Omega \\ 0 \end{bmatrix} Y^{\mathrm{T}}.$$

To compute the SVD of B_1 and B_2, this process can be recursively applied until the submatrices are so small that their SVDs can be computed directly using Golub–Kahan algorithm or other stable methods. There can be at most $O(\log n)$ levels of recursion.

3 Implementation Details

Unimplemented.

4 Numerical Experiments

Unimplemented.

5 Conclusion

Unimplemented.