Project Report

24300720157 Ruijie Li

December 25, 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 $m < n$.

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 specifically 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

$$\begin{aligned} B &= \begin{bmatrix} q_1 & Q_1 & & \\ & & Q_2 & q_2 \end{bmatrix} \begin{bmatrix} c & & & -s \\ & I_{k-1} & & \\ & & I_{n-k} & \\ s & & & c \end{bmatrix} \begin{bmatrix} r_0 & & \\ \alpha_k l_1 & D_1 & \\ \beta_k f_2 & & D_2 \\ 0 & & \end{bmatrix} \begin{bmatrix} & W_1 & \\ 1 & & \\ & & W_2 \end{bmatrix}^{\mathrm{T}} \\ &= \begin{bmatrix} \begin{pmatrix} cq_1 & Q_1 & \\ sq_2 & & Q_2 \end{pmatrix} & \begin{pmatrix} -sq_1 \\ cq_2 \end{pmatrix} \end{bmatrix} \begin{bmatrix} r_0 & & \\ \alpha_k l_1 & D_1 & \\ \beta_k f_2 & & D_2 \\ 0 & & \end{bmatrix} \begin{bmatrix} & W_1 & \\ 1 & & \\ & & W_2 \end{bmatrix}^{\mathrm{T}} \\ &:= \begin{bmatrix} Q & q \end{bmatrix} \begin{bmatrix} M \\ 0 \end{bmatrix} W^{\mathrm{T}}, \end{aligned}$$

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.

2.3 Computing the SVD of M

Now we focus on computing the SVD of the matrix

$$M = \begin{bmatrix} z_1 & & & \\ z_2 & d_2 & & \\ \vdots & & \ddots & \\ z_n & & & d_n \end{bmatrix}$$

where $D = \operatorname{diag}(d_1, \dots, d_n)$, with $0 =: d_1 \le d_2 \le \cdots \le d_n$ and $z = [z_1, \dots, z_n]^{\mathrm{T}}$. We further assume that

$$d_{j+1} - d_j \ge \tau \|M\|_2, \quad |z_j| \ge \tau \|M\|_2, \quad j = 1, \dots, n-1,$$

where τ is a small positive constant to be specified later. Any matrix of this form can be reduced to one that satisfies these conditions by using the deflation procedure described in the next subsection and a simple permutation.

Lemma 2.1. *Let $M = U\Omega V^{\mathrm{T}}$ be the SVD of M with*

$$U = [u_1, \dots, u_n], \quad V = [v_1, \dots, v_n], \quad \Omega = \operatorname{diag}(\omega_1, \dots, \omega_n),$$

Then the singular values ω_i satisfy the interlacing property

$$0 < \omega_1 < d_2 < \omega_2 < d_3 < \cdots < d_n < \omega_n < d_n + \|z\|_2,$$

and the secular equation

$$f(\omega) := 1 + \sum_{k=1}^{n} \frac{z_k^2}{d_k^2 - \omega^2} = 0. \tag{1}$$

The singular vectors are given by

$$u_i = \begin{bmatrix} \frac{z_1}{d_1^2-\omega_i^2} \\ \frac{z_2}{d_2^2-\omega_i^2} \\ \vdots \\ \frac{z_n}{d_n^2-\omega_i^2} \end{bmatrix} \Bigg/ \sqrt{\sum_{j=1}^{n} \left(\frac{z_j}{d_j^2 - \omega_i^2} \right)^2} \tag{2}$$

and

$$v_i = \begin{bmatrix} -1 \\ \frac{d_2 z_2}{d_2^2-\omega_i^2} \\ \vdots \\ \frac{d_n z_n}{d_n^2-\omega_i^2} \end{bmatrix} \Bigg/ \sqrt{1 + \sum_{j=2}^{n} \left(\frac{d_j z_j}{d_j^2 - \omega_i^2} \right)^2} \tag{3}$$

Given D and all singular values, we can reconstruct M using the lemma below.

Lemma 2.2. *Given a diagonal matrix* $D = \operatorname{diag}(d_1, \ldots, d_n)$ *and a set of numbers* $\{\hat{\omega}_i\}_{i=1}^n$ *satisfying the interlacing property*

$$0 = d_1 < \hat{\omega}_1 < d_2 < \hat{\omega}_2 < \cdots < d_n < \hat{\omega}_n < d_n + \|z\|_2,$$

there exists a matrix

$$\hat{M} = \begin{bmatrix} \hat{z}_1 & & & \\ \hat{z}_2 & d_2 & & \\ \vdots & & \ddots & \\ \hat{z}_n & & & d_n \end{bmatrix}$$

whose singular values are exactly $\{\hat{\omega}_i\}_{i=1}^n$. *The elements* $\hat{z}_i$ *can be computed by*

$$|\hat{z}_i| = \sqrt{(\hat{\omega}_n^2 - d_i^2) \prod_{k=1}^{i-1} \frac{(\hat{\omega}_k^2 - d_i^2)}{(d_k^2 - d_i^2)} \prod_{k=i}^{n} \frac{(\hat{\omega}_k^2 - d_i^2)}{(d_{k+1}^2 - d_i^2)}} \tag{4}$$

You can refer to Gu and Eisenstat (1995) for the proofs of Lemma 2.1 and 2.2.

2.3.1 Computing the Singular Vectors

In practice, we can compute the approximate singular values $\{\hat{\omega}\}_{i=1}^n$ by solving the secular equation (1). Once we have the approximate singular values, we can compute the singular vectors using equations (2) and (3). However, directly using these formulas can lead to numerical instability, especially when the singular values are closely spaced — small perturbations can cause great loss of orthogonality among the computed singular vectors. To address this issue, Gu and Eisenstat (1995) proposed a more stable method for computing the singular vectors.

Lemma 2.2 allows us to reconstruct a matrix $\hat{M}$ whose singular values are exactly the computed approximate singular values $\{\hat{\omega}_i\}_{i=1}^n$. We can then compute the singular vectors of $\hat{M}$ using equations (2) and (3), which are more stable since the singular values are now exact. This approach significantly improves the numerical stability of the singular vector computation. Thus, the orthogonality of the computed singular vectors can be guaranteed to high precision. We can use $\hat{M}$'s SVD as an approximation to M's SVD.

2.3.2 Computing the Singular Values

Now we discuss how to compute the singular values by solving the secular equation (1). However, if we directly compute $d_k^2 - \omega^2$, it may lead to severe cancellation errors when ω is close to d_k. To avoid this, we can compute the offsets. Specifically, for any $i \le n-1$, we first check whether $f\left(\frac{d_i+d_{i+1}}{2}\right)$ is positive to determine whether the root ω_i lies in the left or right half of the interval (d_i, d_{i+1}).

First assume that $\omega_i \in \left(d_i, \frac{d_i+d_{i+1}}{2}\right)$. Now assume the offset $\mu = \omega_i - d_i$ and the length of the interval $\delta_j = d_j - d_i$. Then the denominator can be computed as

$$d_j^2 - \omega_i^2 = (d_j - d_i - \mu)(d_j + d_i + \mu) = (\delta_j - \mu)(d_i + d_j + \mu).$$

Note that this method is numerically stable since $\delta_j - \mu \in (\delta_j/2, \delta_j)$. We will then seek the root $\mu_i = \omega_i - d_i \in (0, \delta_{i+1}/2)$ in the reformulated secular equation

$$f(\mu_i) := 1 + \sum_{k=1}^{n} \frac{z_k^2}{(\delta_k - \mu_i)(d_k + d_i + \mu_i)} = 0.$$

For the convienience of subsequent discussion, we denote

$$\varphi_i(\mu) = \sum_{k=1}^{i} \frac{z_k^2}{(\delta_k - \mu)(d_k + d_i + \mu)},$$

$$\psi_i(\mu) = \sum_{k=i+1}^{n} \frac{z_k^2}{(\delta_k - \mu)(d_k + d_i + \mu)}.$$

Now assume that $\omega_i \in \left(\frac{d_i+d_{i+1}}{2}, d_{i+1}\right)$. We set the offset $\mu_i = d_{i+1} - \omega_i$ and the length of the interval $\delta_j = d_j - d_{i+1}$. We can similarly give the reformulated secular equation

$$f(\mu_i) := 1 + \sum_{k=1}^{n} \frac{z_k^2}{(\delta_k + \mu_i)(d_k + d_{i+1} - \mu_i)} = 0.$$

We will seek the root $\mu_i = d_{i+1} - \omega_i \in (0, \delta_i/2)$. Similarly, we denote

$$\varphi_i(\mu) = \sum_{k=1}^{i} \frac{z_k^2}{(\delta_k + \mu)(d_k + d_{i+1} - \mu)},$$

$$\psi_i(\mu) = \sum_{k=i+1}^{n} \frac{z_k^2}{(\delta_k + \mu)(d_k + d_{i+1} - \mu)}.$$

Finally we consider the last singular value $\omega_n > d_n$. Let $\mu_n = \omega_n - d_n$ and $\delta_j = d_j - d_n$, and we have the secular equation

$$f(\mu_n) := 1 + \sum_{k=1}^{n} \frac{z_k^2}{(\delta_k - \mu_n)(d_k + d_n + \mu_n)} = 0.$$

And we will seek the root $\mu_n \in (0, \|z\|_2)$.

We can use bisection or the rational interpolation method to solve these secular equations efficiently. The stopping criterion is set to be

$$|f(\mu)| \le n\varepsilon(1 + |\varphi_i(\mu)| + |\psi_i(\mu)|),$$

3 Implementation Details

Unimplemented.

4 Numerical Experiments

Unimplemented.

5 Conclusion

Unimplemented.