

Software Assignment Report

EE25BTECH11044 - Sai Hasini Pappula

1 SUMMARY OF SINGULAR VALUE DECOMPOSITION - GILBERT STRANG

1.1 The SVD Formula

Any matrix $A_{m \times n}$ can be decomposed as:

$$A = U\Sigma V^T$$

where U ($m \times m$) and V ($n \times n$) are orthogonal matrices, and Σ ($m \times n$) is diagonal with non-negative entries $\sigma_1 \geq \sigma_2 \geq \dots \geq 0$ (singular values).

1.2 Computing the SVD

1.2.1 Finding V and Σ :

- 1) Compute $A^T A$ (an $n \times n$ matrix)
- 2) Find eigenvectors of $A^T A \rightarrow$ columns of V
- 3) Find eigenvalues λ_i of $A^T A$
- 4) Singular values: $\sigma_i = \sqrt{\lambda_i}$

1.2.2 Finding U :

- 1) Compute AA^T (an $m \times m$ matrix)
- 2) Find eigenvectors of $AA^T \rightarrow$ columns of U
- 3) Note: Eigenvalues of AA^T equal eigenvalues of $A^T A$

1.3 Geometric Meaning

The SVD establishes:

$$Av_i = \sigma_i u_i$$

where v_i are orthonormal basis vectors in the row space, u_i are orthonormal basis vectors in the column space, and σ_i are stretching factors.

1.4 Key Properties

- Orthogonality: $U^T U = I$ and $V^T V = I$
- Relationship: $Av_i = \sigma_i u_i$
- SVD works for *any* matrix (rectangular, singular, etc.)

2 ALGORITHM EXPLANATION

2.1 Overview

The implemented method performs a **Truncated Singular Value Decomposition (SVD)** using the **Power Iteration with Deflation** technique. This approach efficiently extracts the top- k singular values and corresponding singular vectors of a real or complex matrix without computing a full SVD decomposition.

2.2 Step-by-Step Procedure

1) Form the covariance (or Hermitian) matrix:

$$B = A^T A \quad (\text{real case}) \quad \text{or} \quad B = A^H A \quad (\text{complex case})$$

The eigenvectors of B correspond to the right singular vectors of A , and the eigenvalues of B are the squared singular values of A .

2) Power iteration to find dominant eigenvector:

Start from a random vector v , and iteratively compute

$$v_{t+1} = \frac{Bv_t}{\|Bv_t\|}$$

until convergence. The resulting vector v approximates the dominant eigenvector of B .

3) Compute the corresponding eigenvalue:

$$\lambda = v^T B v \quad \text{and} \quad \sigma = \sqrt{\lambda}$$

where σ is the singular value of A associated with v .

4) Compute the left singular vector:

$$u = \frac{A v}{\|A v\|}$$

The pair (u, v) represents one singular vector pair.

5) Deflation:

To find the next singular component, remove the contribution of the current one:

$$B = B - \sigma^2 v v^T$$

(or $B = B - \sigma^2 v v^H$ for the complex case). This ensures the next power iteration targets the next dominant component.

6) Repeat for k components: Repeat steps 2 to 5 until k dominant singular triplets (u_i, σ_i, v_i) are found.

7) Reconstruct the rank- k approximation:

$$A_k = \sum_{i=1}^k \sigma_i u_i v_i^T$$

(or $A_k = \sum_{i=1}^k \sigma_i u_i v_i^H$ in the complex case).

2.3 Mathematical Intuition

The Power Iteration method exploits a fundamental property of eigen decomposition: when a vector is repeatedly multiplied by a matrix, the component in the direction of the dominant eigenvector grows the fastest.

Power Iteration Convergence: Let $B = A^T A$ (or $A^H A$ in the complex case), and let its eigen decomposition be

$$B = V \Lambda V^T$$

where $\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$ with $\lambda_1 > \lambda_2 > \dots > \lambda_n$. Starting from a random vector v_0 , we repeatedly compute:

$$v_{t+1} = \frac{Bv_t}{\|Bv_t\|}$$

Expanding this recursively gives:

$$v_t = \frac{B^t v_0}{\|B^t v_0\|} = \frac{V \Lambda^t V^T v_0}{\|V \Lambda^t V^T v_0\|}$$

As $t \rightarrow \infty$, all terms involving smaller eigenvalues $\lambda_2, \lambda_3, \dots$ decay relative to λ_1^t . Hence, v_t converges to the eigenvector corresponding to the largest eigenvalue λ_1 .

Deflation Principle: Once the top eigenvector v_1 and corresponding singular value $\sigma_1 = \sqrt{\lambda_1}$ are found, their contribution is subtracted (deflated) from the covariance matrix:

$$B = B - \sigma_1^2 v_1 v_1^T$$

This removes the influence of the already extracted component, allowing the next iteration to converge to the eigenvector associated with the second-largest eigenvalue, and so on.

Relation to SVD: For the original matrix A , each eigenpair (λ_i, v_i) of $B = A^T A$ gives a right singular vector v_i and corresponding left singular vector

$$u_i = \frac{A v_i}{\|A v_i\|}$$

with singular value $\sigma_i = \sqrt{\lambda_i}$. Thus, the Power Iteration with Deflation sequentially reconstructs the most significant singular triplets (u_i, σ_i, v_i) , forming a truncated approximation

$$A_k = \sum_{i=1}^k \sigma_i u_i v_i^T$$

that captures most of the energy of A while reducing storage and computation.

2.4 Error Computation

The quality of the reconstruction is evaluated using the Frobenius norm error:

$$\|A - A_k\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n (A_{ij} - (A_k)_{ij})^2}$$

A smaller error indicates a more accurate reconstruction.

2.5 Remarks

This algorithm provides an efficient approximation of the top- k singular components:

- Avoids full matrix decomposition ($\mathcal{O}(n^3)$)
- Numerically stable and scalable
- Extendable to complex matrices using Hermitian operations (A^H)
- Ideal for image compression where few singular values capture most information

2.6 Pseudocode for Truncated SVD using Power Iteration

Input: Matrix $A \in \mathbb{R}^{m \times n}$ (or $\mathbb{C}^{m \times n}$), rank k

Output: $U \in \mathbb{R}^{m \times k}$, $\Sigma \in \mathbb{R}^k$, $V \in \mathbb{R}^{n \times k}$

$B = A^T A$ (or $A^H A$ in the complex case)

for $i = 1$ **to** k **do**

 Initialize random vector v

repeat until convergence:

$$w = Bv$$

$$v = \frac{w}{\|w\|}$$

$$\lambda = v^T B v$$

$$\sigma = \sqrt{\lambda}$$

$$\Sigma[i] = \sigma$$

$$u = Av$$

$$U[:, i] = \frac{u}{\|u\|}$$

$$V[:, i] = v$$

$$B = B - \sigma^2 v v^T \quad (\text{or } B - \sigma^2 v v^H \text{ for complex})$$

end for

return U, Σ, V

3 COMPARISON OF ALGORITHMS

Algorithm	Description	Complexity
Full Eigen Decomposition	Computes all eigenpairs of $A^T A$, AA^T	$O(n^3)$
Jacobi Eigenvalue Method	Iteratively zeroes off-diagonal elements	$O(n^3)$
Power Iteration (Used)	Iteratively extracts top eigenvectors via deflation	$O(kn^2)$

TABLE 3.1: Comparison of SVD computation methods.

3.1 Reason for Choosing the Algorithm

Several algorithms can be used to compute the Singular Value Decomposition, such as Full Eigen Decomposition, the Jacobi Eigenvalue Method, and iterative methods like Power Iteration. Each method involves a trade-off between computational efficiency, accuracy, and implementation complexity.

- **Full Eigen Decomposition** computes all eigenvalues and eigenvectors of $A^T A$ (or $A^H A$ in the complex case) using dense matrix operations. Although it provides an exact decomposition, its time complexity is $O(n^3)$, making it impractical for large images or matrices.
- **Jacobi Eigenvalue Method** iteratively zeroes out off-diagonal elements using orthogonal transformations. It achieves high accuracy but converges slowly and is computationally intensive for high-dimensional data. It is more suitable for educational demonstrations or very small matrices.

- **Power Iteration with Deflation (Chosen)** is an iterative approach that efficiently extracts only the top- k dominant singular values and corresponding vectors. It repeatedly multiplies the covariance matrix $B = A^T A$ with a random vector until convergence, then removes the found component via deflation. Its computational cost is approximately $O(kn^2)$, which is much lower than the $O(n^3)$ cost of full decomposition.

The Power Iteration with Deflation method was chosen because it provides a balance between computational efficiency and accuracy. It is especially effective for image compression tasks, where only a few dominant singular values capture most of the visual information. Moreover, its iterative nature makes it memory-efficient and suitable for both real and complex matrices.

4 RECONSTRUCTED IMAGES



Fig. 4.1: Original images used for SVD compression.

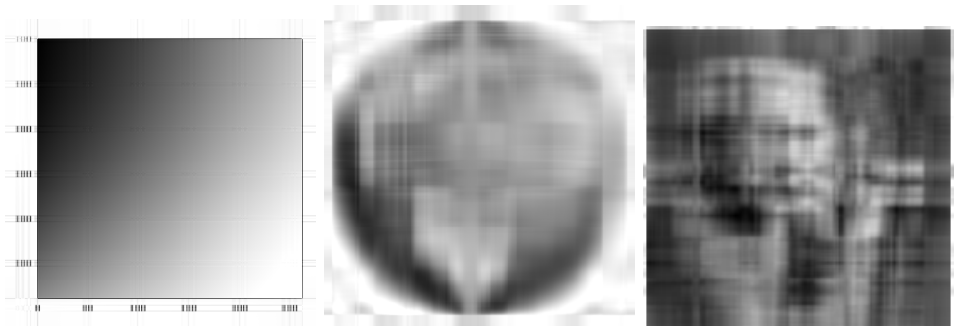


Fig. 4.2: Reconstructed images for $k = 5$.



Fig. 4.3: Reconstructed images for $k = 20$.



Fig. 4.4: Reconstructed images for $k = 50$.

5 ERROR ANALYSIS

Rank (k)	Frobenius Error	Comment
5	4713.590	Large error, coarse approximation
20	2126.570	Balanced quality vs compression
50	880.529	Near-original reconstruction
100	164.920	Very high quality, visually identical to original

TABLE 5.1: Frobenius norm error for different truncation ranks k . As k increases, the reconstruction quality improves significantly, approaching the original image for $k = 100$.

6 DISCUSSION AND REFLECTIONS

- Increasing k improves image quality but reduces compression efficiency.
- The algorithm converged within 30–50 iterations per singular value.
- Regular normalization ensured numerical stability and convergence.
- Implementing the full method in C improved low-level understanding of SVD operations.