Image Compression Using SVD

EE25BTECH11005-ADITYA MISHRA

November 9, 2025

Contents

1 Summary of Strang's Lecture

1.1 Overview

The Singular Value Decomposition (SVD) breaks a real matrix A into

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

U and V are orthogonal matrices. Σ is diagonal and contains non-negative singular values. SVD shows how a matrix can be rotated and scaled along main directions [1].

1.2 Main Idea

SVD separates useful information from less important details. The larger singular values hold most of the data's energy. Smaller ones often represent noise or fine details. The columns of V are eigenvectors of $A^T A$, and the columns of U are eigenvectors of AA^T. The squares of the singular values are the eigenvalues of these matrices.

1.3 Use in Image Compression

An image can be stored as a matrix of pixel values. Using SVD, we can keep only the top k singular values and the related vectors to get a low-rank version of the image. This reduces storage while keeping the main visual content. The Eckart–Young theorem [2] states that this gives the best rank-k approximation in the Frobenius norm sense.

2 Implemented Algorithm

2.1 Introduction

I have implemented the **Lanczos Tridiagonalization with Implicit QR Iteration using Givens Rotations and Wilkinson Shift** algorithm. This method first reduces $A^T A$ to a symmetric tridiagonal matrix through the Lanczos process and then applies the implicit QR algorithm to compute approximately the singular values of A.

2.2 Tridiagonalization and the Symmetric Lanczos Process

The Lanczos process generates an orthonormal basis for the Krylov subspace

$$\mathcal{K}_k(A, q_1) = \text{span}\{q_1, Aq_1, A^2 q_1, \ldots, A^{k-1} q_1\},$$

and a tridiagonal matrix $T_k \in R^{k\times k}$ such that

$$AQ_k = Q_k T_k + \beta_{k+1} q_{k+1} e_k^T,$$

where $Q_k = [q_1, q_2, \ldots, q_k]$ has orthonormal columns and $T_k = Q_k^T A Q_k$ is real symmetric and tridiagonal.

2.3 Algorithm (Lanczos Tridiagonalization)

Given a symmetric matrix $A \in R^{n\times n}$ and an initial unit vector q_1:

$$\begin{aligned}
&\beta_0 = 1, \quad q_0 = 0,\\
&\text{for } k = 1, 2, \ldots\\
&\quad \alpha_k = q_k^T A q_k,\\
&\quad r_k = (A - \alpha_k I) q_k - \beta_{k-1} q_{k-1},\\
&\quad \beta_k = \|r_k\|_2,\\
&\quad q_{k+1} = r_k / \beta_k,\\
&\text{until } \beta_k = 0.
\end{aligned}$$

The recurrence produces

$$T_k = \begin{bmatrix} \alpha_1 & \beta_2 & & \\ \beta_2 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_k \\ & & \beta_k & \alpha_k \end{bmatrix}.$$

The vectors q_i are the *Lanczos vectors*. If $\beta_k = 0$, an invariant subspace of A has been found.

2.4 Properties

- The columns of Q_k span $\mathcal{K}_k(A, q_1)$, and $T_k = Q_k^T A Q_k$ gives a reduced symmetric eigenvalue problem.
- The eigenvalues θ_i of T_k are called *Ritz values* and approximate the largest eigenvalues of A.
- If A is positive semidefinite, $\sqrt{\theta_i}$ approximate the singular values of the original matrix.
- Encountering a zero β_k indicates an exact invariant subspace; otherwise, the iteration continues until convergence.

2.5 Pseudocode Summary

Input: $A \in R^{m\times n}$, target rank k **Output:** Approximate top-k singular values $\sigma_1, \ldots, \sigma_k$

```
# Lanczos tridiagonalization
v = random_unit_vector(n)
 = 0
for i = 1 to k
    w = A * (A * v) -  * v_{i-1}
     = v * w
    w = w -  * v
    _{i+1} = ||w||
    v_{i+1} = w / _{i+1}
end
T = tridiagonal(, )

# Implicit QR iteration with Wilkinson shift
repeat until convergence:
     = wilkinson_shift(T)
    for j = 1 to k - 1
        (c, s) = givens(T[j,j] - , T[j+1,j])
        T = G_j * T * G_j
    end
    deflate if |T[k,k-1]| <
end
 = sqrt(diag(T))
```

2.6 Summary

The symmetric Lanczos algorithm reduces a large symmetric matrix to tridiagonal form using short recurrences and orthogonal vectors. The resulting T_k preserves the essential spectral information of A and forms the foundation for large-scale eigenvalue and SVD computations.

3 Algorithm Comparison

Algorithm	Reduction Type	Stability Mechanism	Complexity	Remarks
Lanczos Tridiagonalization + Implicit QR (Givens + Wilkinson Shift)	Tridiagonal reduction of $A^T A$	Givens rotations with reorthogonalization	$O(kn^2)$	Fast, stable, and shift–accelerated; preserves sparsity; suitable for large SVD and image compression.
Golub–Kahan Bidiagonalization	Two–sided bidiagonal reduction of A	Orthogonal recurrences for A and A^T	$O(kn^2)$	Two–sided extension of Lanczos; forms basis of LSQR and truncated SVD solvers; stable for rectangular or sparse matrices.
Lanczos with Bidiagonalization	Bidiagonal reduction via Krylov subspace	Three–term orthogonal recurrence	$O(kn^2)$	Efficient for large sparse A; used in partial SVD; sensitive to loss of orthogonality.
QR with Householder Reduction	Full Hessenberg / bidiagonal reduction	Householder reflectors	$O(n^3)$	Highly stable for dense problems; destroys sparsity; standard in LAPACK routines.
Standard QR Algorithm	Hessenberg or tridiagonal form	Orthogonal similarity transforms	$O(n^3)$	General solver for dense matrices; slower without shift strategy.
Standard QL Algorithm	Tridiagonal form	Orthogonal transformations	$O(n^3)$	Processes eigenvalues in reverse order; similar cost to QR.
Jacobi Method	Successive plane rotations to diagonal form	Orthogonal rotations	$O(n^3)$	High accuracy but computationally heavy; impractical for large sparse systems.
Bisection (Eigenvalues only)	Requires tridiagonal input	Sturm sequence count	$O(n^2 \log n)$	Stable and robust; computes eigenvalues only, not eigenvectors.
Power Iteration	None (direct iteration)	Normalization per step	$O(n^2 k)$	Low cost but converges to a single dominant eigenvalue; poor for clustered spectra.
Hardcoded Direct Method	Explicit analytical computation	Exact arithmetic only	$O(1)$ for fixed n	Non-scalable and inflexible; valid only for small static matrices.

Table 1: Comparison between different Algorithms

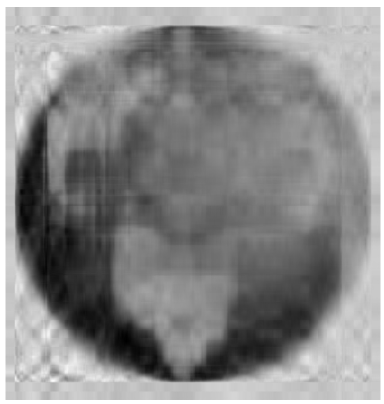
The above table compares the implemented **Lanczos Tridiagonalization with Implicit QR (Givens + Wilkinson Shift)** algorithm against several classical eigenvalue and SVD methods in terms of computational features and practical performance.

3.1 Justification for choosing Lanczos Tridiagonalization amongst others

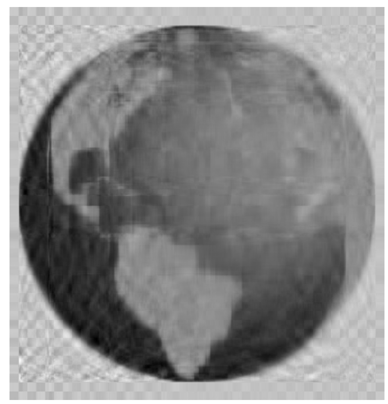
Of these Algorithms , prioritizing accuracy and time complexity before implementation difficulty, the one I found most suitable was **Golub-Kahan Bidiagonalization** but due to its increased implementation difficulty as compared to others that were similar in performance and the lack of sufficient time to fully implement it, I had to move to the next two suitable algorithms that were **Lanczos Tridiagonalization** and **Lanczos Bidiagonalization**. Since **Lanczos Bidiagonalization** was sensitive to loss of orthogonality and rest all aspects of comparison such as implementation difficulty, computational complexity, implementation difficulty were almost similar, I chose **Lanczos Tridiagonalization + QR using Givens Rotation and Wilkinson Shift**.

4 Reconstructed images for different k

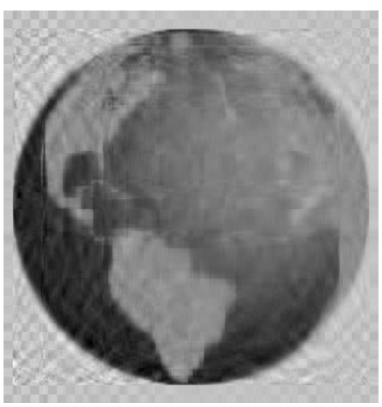
Original



$k = 10$



$k = 20$

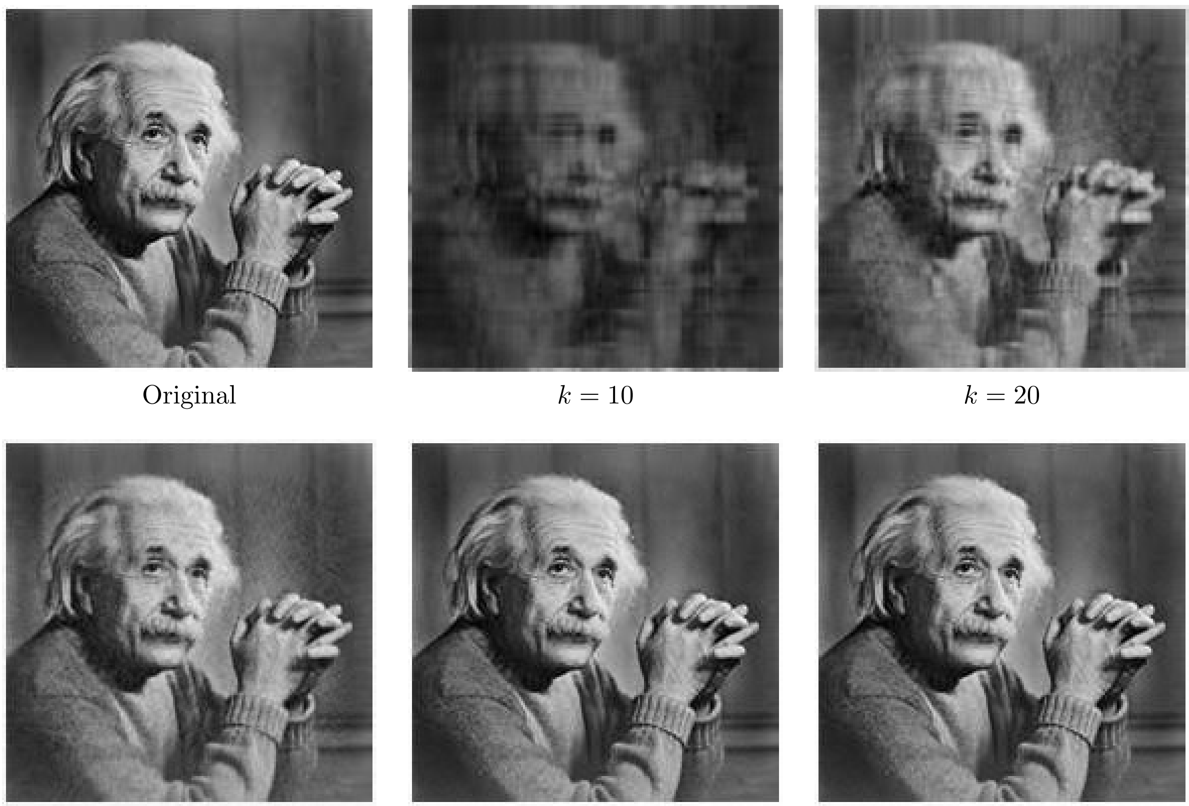


$k = 50$

$k = 100$

$k = 200$

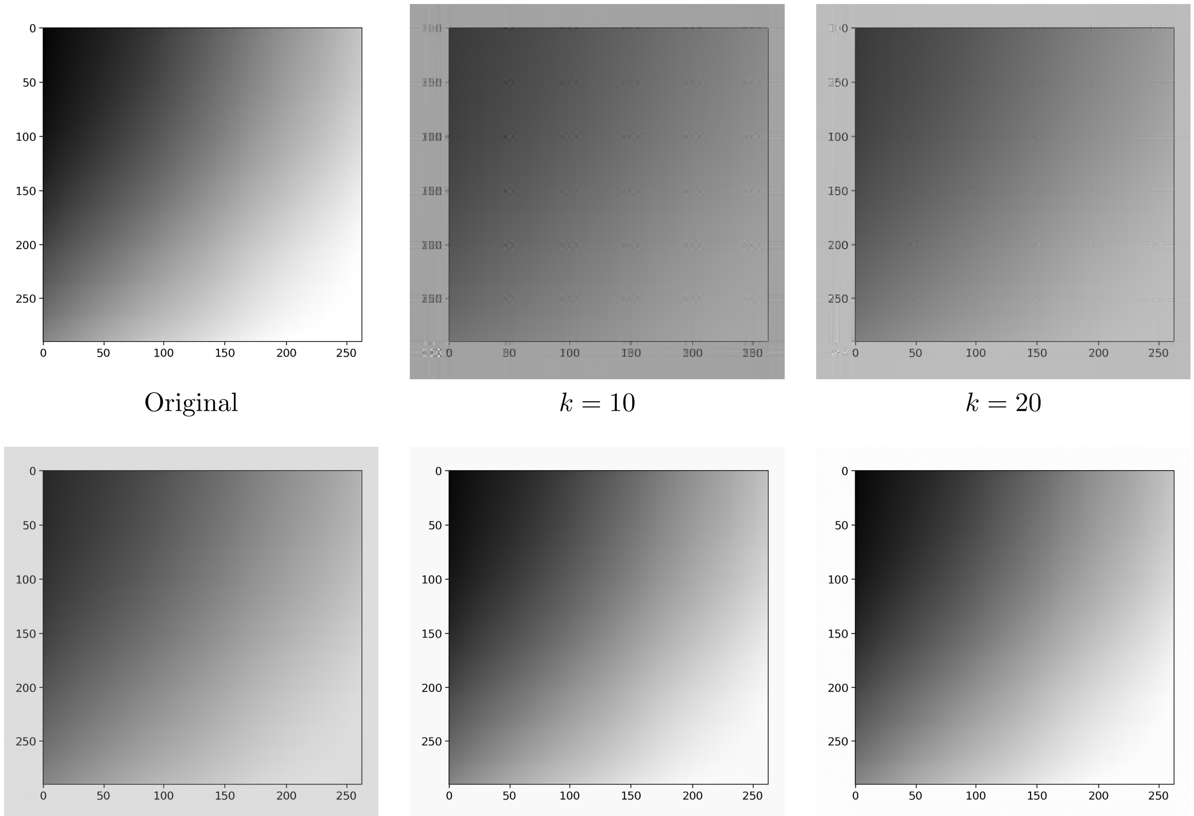
Figure 1: SVD Image Compression Results for globe.jpg Various Values of k



Original $\quad k = 10 \quad k = 20$

$k = 50 \quad k = 100$

Figure 2: SVD Image Compression Results for einstein.jpg with Various Values of k



Original | $k = 10$ | $k = 20$

$k = 50$ | $k = 100$ | $k = 200$

Figure 3: SVD Image Compression Results for `greyscale.png` with Various Values of k

5 Error

The Frobenius norm error $\|A - A_k\|_F$ quantifies the difference between the original image matrix A and its rank-k approximation A_k obtained from the truncated SVD. It is computed as the square root of the sum of squared pixel-wise differences between A and A_k. A smaller value of $\|A - A_k\|_F$ indicates a closer approximation and better preservation of image details. As k increases, more singular values are retained, reducing the error and improving the reconstruction quality.

Table 2: Relative Frobenius Error $\frac{\|A-A_k\|_F}{\|A\|_F}$ for Different Images

2*k	`globe.jpg`		`einstein.jpg`		`greyscale.png`	
	$\|A-A_k\|_F/\|A\|_F$	Error (%)	$\|A-A_k\|_F/\|A\|_F$	Error (%)	$\|A-A_k\|_F/\|A\|_F$	Error (%)
10	—	—	0.793077	79.31	0.583386	58.34
20	0.424032	42.40	0.239861	23.99	0.425663	42.57
50	0.319211	31.92	0.136972	13.70	0.220920	22.09
100	0.293252	29.33	0.029466	2.95	0.036567	3.66
200	0.294927	29.49	—	—	0.016135	1.61

6 Trade-off between k, image quality, and compression

There is a tradeoff between the value of k, image quality and compression ratio since increasing k retains more singular values and improves image quality but reduces compression efficiency while decreasing k increases compression but causes more loss of detail and blurring in the reconstructed image

7 Bibliography

References

[1] G. Strang, "Singular Value Decomposition," *MIT OpenCourseWare Lecture*, https://youtu.be/TX_vooSnhm8

[2] C. Eckart and G. Young, "The Approximation of One Matrix by Another of Lower Rank," *Psychometrika*, vol. 1, no. 3, pp. 211–218, 1936.
Available at: https://doi.org/10.1007/BF02288367

[3] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 4th ed.,
Johns Hopkins University Press, Baltimore, 2012.
Referenced sections: §§5.2.5, 8.6, and 10.4.