ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

Comparative Analysis of Singular Value Decomposition and Eigenvalue Decomposition Methods for Solving Large Scale Linear Systems

Nwokolo, Peter C., Unaegbu, Ebenezer N., Ugwueze, Precious N. and Okeke Mmesoma L.

Nnamdi Azikiwe University, Awka, Anambra State.

DOI: https://doi.org/10.51583/IJLTEMAS.2026.150500046

Received: 02 May 2026; Accepted: 07 May 2026; Published: 26 May 2026

ABSTRACT

This study compares the effectiveness of Singular Value Decomposition (SVD) and Eigenvalue Decomposition in solving large-scale linear systems, focusing on computational efficiency, numerical stability, and versatility across different problem domains. At its core, matrix decomposition is a crucial tool in numerical linear algebra, allowing us to break down complex systems into more manageable forms.

The study delves into both the theoretical underpinnings and practical performance of these methods, highlighting their respective strengths and weaknesses. SVD stands out for its ability to handle ill-conditioned matrices and is widely used in dimensionality reduction and data analysis, whereas Eigenvalue Decomposition is commonly applied to structured problems and spectral analysis.

To validate the performance claims, computational experiments were conducted using randomly generated matrices of sizes 100×100, 1000×1000, and 5000×5000. The results show that both methods perform efficiently for small matrices, but as matrix size increases, SVD demonstrates better computational performance on the tested system. Moreover, for large and ill-conditioned matrices, SVD exhibits superior numerical stability, making it a more reliable choice than Eigenvalue Decomposition.

The study also explores applications in machine learning, optimization, artificial intelligence, recommender systems, and structural engineering to illustrate the practical relevance of these findings. Ultimately, the choice of method depends on the specific problem at hand, requiring a trade-off between computational efficiency, numerical stability, and scalability.

INTRODUCTION

Matrix decomposition, also known as matrix factorization, is a fundamental tool in linear algebra and numerical analysis that simplifies complex matrix operations by breaking them into the product of simpler, structured matrices. Historically, the motivation for decomposition arose from the need to solve large systems of linear equations, which form the backbone of scientific computation, engineering models, and data-driven research. Early techniques such as Gaussian elimination laid the foundation for systematic approaches to matrix reduction, but it was not until the mid-20th century with the advent of digital computers that decomposition methods gained prominence (Trefethen, L.N and Bau, D. 1997) . Householder's Principles of Numerical Analysis (1954) emphasized the role of partitioned matrices and block-based LU decomposition as an efficient strategy for high speed computation, marking a significant departure from purely manual methods (Householder 1954); (Lu, J. 2021) & (Chandra, R. and Guha, R. 2007). These advancements were not only theoretical but also practical, enabling large-scale problems in geology, physics, and engineering to be solved more reliably.

In contemporary applications, matrix decomposition methods are essential for tackling large-scale linear systems that appear in diverse domains. Several matrix decomposition techniques exist, including LU, QR and Cholesky decompositions. However, this study focuses specifically on Singular Value Decomposition (SVD) and Eigenvalue Decomposition due to their wide applicability in date analysis, engineering, and scientific computing. For instance, single value decomposition and Eigenvalue decomposition, are widely employed in solving systems of equations, least squares problems, eigenvalue computations, and optimization challenges (Higham,

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

2002 & Ejikeme et al, 2015). In computer science and machine learning, matrix decomposition plays a critical role in the development of algorithms for dimensionality reduction, collaborative filtering, and feature extraction (Meyer 2000). For example, SVD is foundational in Principal Component Analysis (PCA), which is used for pattern recognition and data compression. Similarly, factorization techniques underpin neural network training via back propagation, reinforcing the link between numerical linear algebra and artificial intelligence (Lu, 2021; Duan, Jiang, and Jain, 2022).

Despite their ubiquity, decomposition methods encounter limitations when applied to modern large-scale datasets and computational systems. Sparse and structured matrices, common in graph theory and scientific computing, pose unique challenges (Trefethen, 2014).

Beyond computational science, matrix decomposition has expanded into interdisciplinary applications. In recommender systems, for instance, matrix factorization is employed to address the "sparsity problem" in user-item rating datasets. By integrating review-based collaborative filtering with matrix factorization. Duan et al. (2022), demonstrated that latent factor models can enhance predictive accuracy in cases where data is incomplete. This reflects the versatility of decomposition approaches, as methods originally developed for solving linear systems in mathematics now power modern innovations in artificial intelligence, business analytic, and decision support systems. Such applications reaffirm that matrix decomposition is not merely a mathematical tool but a foundational bridge between theory and practice (Benzi 2002 and Demnel 2007).

Matrix decomposition methods, although powerful, face significant challenges when applied to large-scale linear systems(Golu, G.H., 2009) and (Peng, R.and Vempala, S. 2021).
While several classical decomposition methods such as LU and QR exist in the literature, they are not the focus of this study. Instead, this research concentrates on evaluating the performance of Singular Value Decomposition (SVD) and Eigenvalues Decomposition in large scale settings.
These methods are particularly suitable for modern applications involving high dimension and structured data, where numerical stability and robustness are critical.
The aim of the study is to analyze and compare Singular Value Decomposition (SVD) and Eigenvalue Decomposition in order to evaluate their effectiveness and efficiency in solving large scale linear system.
This study is restricted to two matrix decomposition methods: Singular Value Decomposition (SVD) and Eigenvalue Decomposition. Rather than providing detailed derivations, the study focuses on comparing their computational efficiency, numerical stability, and performance on real world data sets and randomly generated matrices. and complexity in solving large-scale linear systems, as well as their performance on real-world datasets and randomly generated matrices. Consequently, the scope of this research does not extend to implementations in specialized software packages such as LAPACK, ARPACK, or Suite-sparse.

Theory of Methods

Singular Value Decomposition

In linear algebra, few tools are as powerful and versatile as the Singular Value Decomposition (SVD). While *LU* decomposition targets square matrices for solving linear systems, SVD extends its utility far beyond, encompassing even non-square matrices, whether tall or wide (Ryan 2008). From numerical stability to dimensionality reduction, SVD is a cornerstone in applied mathematics, machine learning, signal processing, and statistics.

Given any $m \times n$ real matrix A, SVD enables us to express A as the product of three matrices:

$$A = U\Sigma V^{\mathrm{T}}$$

Here, U is an $m \times m$ orthogonal matrix, Σ is an $m \times n$ diagonal matrix with non-negative real numbers (called singular values) on the diagonal, and V is an $n \times n$ orthogonal matrix.

SVD is more than just a factorization; it unveils the geometry of linear transformations. The columns of V represent the directions in the input space, the diagonal entries of Σ scale them, and the columns of U reorient

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

the result. These insights are critical when analyzing or compressing data, computing pseudoinverses, or solving ill-posed problems.

Definition 1. (Singular Value Decomposition)**.** Let $A \in R^{m\times n}$. Then there exist matrices

$U \in R^{m\times m}$, $V \in R^{n\times n}$, and $\Sigma \in R^{m\times n}$ such that:

$$A = U\Sigma \mathrm{V}^{\mathrm{T}} ,$$

where:

- $U^T U = I_m$ (orthogonal),
- $V^T V = I_n$ (orthogonal),
- Σ is diagonal with non-negative real numbers $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\mathrm{r}} > 0$, called **singular values**.

Here, r = rank(A) is the number of non-zero singular values.

Theorem 1. (Existence of SVD)**.** Every real matrix $A \in R^{m\times n}$ admits a singular value decomposition.

Proof. The proof is rooted in spectral theory. Consider the symmetric matrices $A^T A \in R^{n\times n}$ and $AA^T \in R^{m\times m}$. Both are positive semi-definite and hence have orthonormal eigenvectors with non-negative eigenvalues. Let:

$$A^T A = V \Lambda V^T ,$$

where Λ is diagonal with eigenvalues $\lambda_i \geq 0$, and V is orthogonal. Define singular values as $\sigma_{\mathrm{i}} = \sqrt{\lambda_i}$. Set:

$$\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_{\mathrm{r}}, 0, \ldots), \qquad U = AV\Sigma^{\dagger},$$

where $\Sigma\dagger$ is the pseudoinverse of Σ. It can be shown that U is orthogonal, and thus:

$$A = U\Sigma \mathrm{V}^{\mathrm{T}}$$

Remark:

(i). The singular values of A are the square roots of the eigenvalues of ATA.

(ii). The columns of V are the eigenvectors of ATA; those of U are the eigenvectors of AAT .

(iii). The number of non-zero singular values equals the rank of A.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

Applications of Singular Value Decomposition Method:

1. **Dimensionality reduction:** Principal Component Analysis (PCA) is based on SVD.

2. **Solving least squares problems:** For over determined systems *Ax* = *b*, SVD yields the

minimum norm solution.

3. **Data compression:** Low-rank approximations use the largest singular values to represent the matrix efficiently.

4. **Image processing:** SVD compresses grayscale or RGB images by truncating smaller singular values

Singular Value Decomposition: Step-by-Step Method

Let *A* be an $m \times n$ real matrix.

1. Compute $A^T A$ and find its eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$.

2. The singular values are $\sigma_i = \sqrt{\lambda_i}$.

3. Find the eigenvectors $v_1, \ldots, v_n$ of $A^T A$. Form $V = [v_1, \ldots, v_n]$.

4. Compute $u_i = \frac{1}{\sigma_i} A v_i$ for $\sigma_i \neq 0$, and form $U = [u_1, \ldots, u_m]$.

5. Form the diagonal matrix Σ with $\sigma_1, \ldots, \sigma_r$ on the diagonal.

6. Then, $A = U\Sigma V^T$

Let

$$A = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}$$

Step 1: Compute $A^T A$

$$A^T A = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}^T \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} 10 & 6 \\ 6 & 10 \end{bmatrix}$$

Step 2: Compute eigen value of $A^T A$

$$\lambda^2 - 20\lambda + 64 = 0 \qquad \Longrightarrow \lambda_1 = 16, \qquad \lambda_2 = 4.$$

So, singular values are: $\sigma_1 = \sqrt{16} = 4 \qquad \sigma_2 = \sqrt{4} = 2$

Step 3: Compute V

Eigenvectors of $A^T A$ are:

$$v_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \qquad v_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

Step 4: Compute *U*

$$u_1 = \frac{1}{4}Av_1 = \frac{1}{4}\begin{bmatrix}3 & 1\\ 1 & 3\end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix}1\\ 1\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\ 1\end{bmatrix}$$

$$u_2 = \frac{1}{2}Av_2 = \frac{1}{2}\begin{bmatrix}3 & 1\\ 1 & 3\end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix}-1\\ 1\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}-1\\ 1\end{bmatrix}$$

So:

$$U = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & -1\\ 1 & 1\end{bmatrix}, \quad \Sigma = \begin{bmatrix}4 & 0\\ 0 & 2\end{bmatrix}, \quad V^T = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ -1 & 1\end{bmatrix}$$

□

Example 2: Tall Matrix

Let

$$\mathrm{A} = \begin{bmatrix}1 & 0\\ 0 & 1\\ 0 & 0\end{bmatrix}$$

Then

$$A^TA = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} \Rightarrow \ \sigma_1 = \sigma_2 = 1$$

$$V = I_2, \quad \Sigma = \begin{bmatrix}1 & 0\\ 0 & 1\\ 0 & 0\end{bmatrix}, \ \mathrm{U} = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$$

So,

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

Example 3: Low-Rank Matrix

Let

$$A = \begin{bmatrix}2 & 4\\ 1 & 2\end{bmatrix}$$

Then

$$A^TA = \begin{bmatrix}5 & 10\\ 10 & 20\end{bmatrix}$$

Eigenvalues: $\lambda_1 = 25$, $\lambda_2 = 0$, so $\sigma_1 = 5$, $\sigma_2 = 0$.

Eigenvectors of λ_1 : $v_1 = \frac{1}{\sqrt{5}}\begin{bmatrix}1\\ 2\end{bmatrix}$

$$u_1 = \frac{1}{5}Av_1 = \frac{1}{\sqrt{5}}\begin{bmatrix}1\\ 0.5\end{bmatrix}$$

So,

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

$$U = \begin{bmatrix} \frac{1}{\sqrt{1.25}} & * \\ \frac{0.5}{\sqrt{1.25}} & * \end{bmatrix} \quad \Sigma = \begin{bmatrix} 5 & 0 \\ 0 & 0 \end{bmatrix} \quad V = \begin{bmatrix} \frac{1}{\sqrt{5}} & * \\ \frac{2}{\sqrt{5}} & * \end{bmatrix}$$

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

SVD is a universal matrix decomposition method applicable to any real matrix. It reveals the intrinsic structure of the matrix, identifies its rank, and provides the optimal low-rank approximation. Its numerical stability and theoretical depth make it indispensable in applied linear algebra.

Eigenvalue Decomposition:

In linear algebra, understanding the internal structure of matrices plays a central role in both

theoretical and applied mathematics. One such structural insight is offered by **eigenvalue**

decomposition, also known as **spectral decomposition**. This decomposition provides a

way of analyzing a square matrix by breaking it down into a set of eigenvectors and eigenvalues quantities that capture the matrix's most essential properties.

Eigenvalue decomposition applies to a special class of matrices specifically, square matrices

that are diagonalizable. For such matrices, the decomposition expresses the matrix as the

product of three matrices: one composed of its eigenvectors, one diagonal matrix of eigenvalues, and the inverse of the eigenvector matrix. In its simplest form, if $A \epsilon R^{n \times n}$ i is diagonalizable,

then

$$A = PDP^{-1},$$

where *D* is a diagonal matrix whose entries are the eigenvalues of *A*, and *P* is a matrix whose columns are the corresponding eigenvectors.

The utility of eigenvalue decomposition is vast. It underlies the solution of differential equations, stability analysis, quantum mechanics, and machine learning algorithms such as Principal Component Analysis (PCA). Diagonal matrices are computationally simple to manipulate raising them to powers, computing exponentials, and applying functions become straight forward. Thus, diagonalizing a matrix simplifies these operations substantially.

However, not all matrices are diagonalizable. A matrix may fail to have enough linearly independent eigenvectors, making *P* non-invertible. In such cases, the Jordan canonical form or Schur decomposition is employed instead. Nevertheless, for symmetric (or Hermitian in the complex case) matrices, diagonalizability is always guaranteed, and the decomposition gains additional structure—orthogonal eigenvectors and real eigenvalues.

Eigenvalue decomposition is rooted in the fundamental equation:

$$Av = \lambda v,$$

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

where v is a non-zero vector (the **eigenvector**) and λ is a scalar (the **eigenvalue**). This equation states that the action of matrix A on vector v merely stretches (or compresses or flips) the vector without changing its direction.

Definition 3. (Eigenvalue Decomposition)**.** Let □ □ □^□×□ be a square matrix. If A has n linearly independent eigenvectors, then A is diagonalizable, and there exists an invertible matrix P and a diagonal matrix D such that:

$$A = PDP^{-1},$$

where:

- The columns of P are the eigenvectors of A,
- The diagonal entries of D are the corresponding eigenvalues.

Theorem 3. (Diagonalizability Criterion)**.** A matrix $A \in R^{n\times n}$ is diagonalizable if and only if it has n linearly independent eigenvectors.

Proof. ($\Rightarrow$) Suppose $A = PDP^{-1}$ for some invertible P. Then the columns of P are linearly independent and satisfy $AP = P\,D$, which implies each column of P is an eigenvector of A. ($\Leftarrow$) Conversely, suppose A has n linearly independent eigenvectors $v_1, v_2 \ldots, v_n$ with eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$. Construct matrix $P = [v_1, v_2 \ldots, v_n]$ and diagonal matrix D = diag ($\lambda_1, \lambda_2, \ldots, \lambda_n$).

Then

$$AP = [Av_1, Av_2 \ldots, Av_n] = [\lambda_1 v_1, \lambda_2 v_2, \ldots, \lambda_n v_n] = P\,D.$$

Thus, $A = PDP^{-1}$.

Theorem 4. (Symmetric Matrices)**.** If $A \in R^{n\times n}$ is a symmetric matrix, i.e., $A^T = A$,

then:

1. All eigenvalues of A are real.

2. A is diagonalizable.

3. There exists an orthogonal matrix Q such that:

$$A = Q\Lambda Q^T,$$

where Λ is a diagonal matrix of eigenvalues and Q contains orthonormal eigenvectors.

This is known as the **spectral theorem** for real symmetric matrices.

2.2.1 Step-by-Step Method for Eigenvalue Decomposition

Let $A \in R^{n\times n}$ be a matrix. We aim to find $A = PDP^{-1}$ or $A = Q\Lambda Q^T$ (if A is symmetric), Follow the step:

Step 1: Compute the Characteristic Polynomial

Solve the equation:

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

$$\det(A - \lambda I) = 0,$$

to find the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$.of A.

Step 2: Find the Eigenvectors

For each eigenvalue λ_i, solve the system:

$$(A - \lambda_i I)v = 0,$$

to find the corresponding eigenvector(s) v_i.

Step 3: Form Matrices *P* and *D*

- Form *P* using the eigenvectors as columns.
- Form *D* as a diagonal matrix of eigenvalues.

Step 4: Verify Decomposition

Check that:

$$A = PDP^{-1} \text{ or } A = \square\square Q^T$$

Example 1

Let:

$$A = \begin{bmatrix} 4 & 1 \\ 2 & 3 \end{bmatrix}\square$$

Step 1: Find the eigenvalues

$$det(A - \lambda I) = det\begin{bmatrix} 4-\lambda & 1 \\ 2 & 3-\lambda \end{bmatrix} = (4-\lambda)(3-\lambda) - 2 = \lambda^2 - 7\lambda + 10 = 0$$

$$\lambda = 5, \quad \lambda = 2$$

 Step 2: Find the eigenvector

 For $\lambda = 5$,

$$(A - 5I)v = 0 \Rightarrow \begin{bmatrix} -1 & 1 \\ 2 & -2 \end{bmatrix} v = 0 \Rightarrow v_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

For $\lambda = 2$,

$$(A - 2I)v = 0 \Rightarrow \begin{bmatrix} 2 & 1 \\ 2 & 1 \end{bmatrix} v = 0 \Rightarrow v_2 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$$

Step 3: Construct *P* and *D*

$$P = \begin{bmatrix} 1 & -1 \\ 1 & 2 \end{bmatrix}, \qquad D = \begin{bmatrix} 5 & 0 \\ 0 & 2 \end{bmatrix}$$

Then:

$$A = PDP^{-1}$$

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

Example 2: (Symmetric Matrix)

Let

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

Step 1: Characteristic polynomial

$$det(A - \lambda I) = (2 - \lambda)^2 - 1 = \lambda^2 - 4\lambda + 3 = 0 \Rightarrow \lambda = 3,1$$

Step 2: Eigen vector

For $\lambda = 3$,

$$(A - 3I)v = 0 \Rightarrow v_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

For $\lambda = 1$,

$$(A - I)v = 0 \Rightarrow v_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Normalize eigenvectors:

$$q_1 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad q_2 = \frac{1}{\sqrt{2}}\begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Then:

$$Q = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \qquad \Lambda = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix}$$

$$A = Q\Lambda Q^T$$

Summary

Eigenvalue decomposition expresses a matrix as $A = PDP^{-1}$ if it has *n* linearly independent eigenvectors.

Symmetric matrices always admit an orthogonal eigenvalue decomposition: $A = Q\Lambda Q^T$3. The decomposition simplifies many matrix operations and is foundational in applications like systems of differential equations, quantum mechanics, and dimensionality reduction.

Applications of Eigenvalue Decomposition Method:

Eigenvalue decomposition expresses a matrix *A* as $A=PDP^{-1}$, where *D* is a diagonal matrix of eigenvalues and *P* contains the corresponding eigenvectors.

Principal Component Analysis (PCA)

Eigenvalue decomposition is at the core of PCA, which is used for dimensionality reduction in statistics and machine learning.

Stability Analysis

In dynamical systems, the eigenvalues of the system matrix determine stability. The system is stable if all eigenvalues have negative real parts.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

Quantum Mechanics

In quantum mechanics, eigenvalue problems are fundamental. Observable correspond to eigenvalues of linear operators.

Solving Differential Equations

Systems of linear differential equations such as $\frac{dx}{dt} = AX$ can be solved using eigenvalue decomposition, especially when computing matrix exponential.

Matrix Powers and Exponentials

For a diagonalizable matrix:

$$A^n = PD^nP^{-1} \text{ and } e^{At} = Pe^{Dt}P^{-1}$$

This simplifies computation in systems theory and control.

PageRank Algorithm

Google's Page Rank algorithm uses eigenvalue computations on the web's link matrix to determine the importance of pages.

Vibration and Modal Analysis

In mechanical engineering, eigenvalue analysis identifies natural frequencies and mode shapes in vibrating systems.

RESULTS AND ANALYSIS

Theoretical Investigation into the Numerical Stability and Accuracy of Matrix Decomposition Methods

Matrix decomposition techniques play a pivotal role in numerical linear algebra, underpinning a wide range of computational methods in science and engineering. We presents a theoretical examination of the numerical stability and accuracy of two matrix decomposition methods: SVD decomposition and eigenvalue decomposition. By exploring each method's mathematical structure, sensitivity to perturbations, and inherent stability characteristics, we aim to elucidate the conditions under which each technique excels or deteriorates in practical computation.

Singular Value Decomposition (SVD)

SVD represent any $m \times n$ matrix A as $A = U\Sigma V^T$, where U and V are orthogonal matrices and Σ is a diagonal matrix containing the singular values. It is a powerful and general decomposition, used in least squares problems, pseudo inverse computation, and data compression.

Numerical Stability of SVD

SVD is among the most stable of all matrix decomposition. The orthogonality of U and V ensures that norm-preserving transformations are used throughout the process, thereby minimizing the propagation of errors.

SVD is robust even for ill-conditioned or rand-deficient matrices, making it particularly useful when other decomposition fail.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

Numerical Accuracy of SVD

SVD provides excellent numerical accuracy, especially in problems involving low-rank approximations or noisy data. The singular values indicate the intrinsic dimensionality of the data, and small singular values can be safely truncated to reduce noise.

Backward and forward errors are typically minimal, and the method produces the best low-rank approximation of a matrix in terms of the Frobenius and 2-norms.

Eigenvalue Decomposition

Eigenvalue decomposition expresses a matrix A as $A = PDP^{-1}$, where D is a diagonal matrix containing the eigenvalues of A, and P contain the corresponding eigenvectors. This decomposition is fundamental in analyzing systems of differential equations, quantum mechanics, and principal component analysis.

Numerical Stability of Eigenvalue Decomposition

The numerical stability of eigenvalue decomposition is highly dependent on whether the matrix is normal (i.e., it commutes with its conjugate transpose: $AA^* = A^*A$). For normal matrices, the eigenvectors are orthogonal, and the decomposition is stable.

For non-normal or defective matrices (those with repeated eigenvalues and fewer linearly independent eigenvectors), small perturbations in the matrix can lead to large changes in eigenvalues or eigenvectors, significantly compromising numerical stability.

Numerical Accuracy of Eigenvalue Decomposition

Eigenvalue computations can be inaccurate for matrices with clustered or nearly equal eigenvalues. Perturbation theory indicates that the sensitivity of an eigenvalue to perturbations in the matrix is inversely proportional to the angle between left and right eigenvectors. Therefore, poorly conditioned eigenvectors result in large errors in both the eigenvalues and eigenvectors.

3.4. Comparative Analysis

The table below summarizes key aspect of stability and accuracy for each method:

Method	Pivoting required	Stability	Accuracy
Singular Value Decomposition	No	Very High	Very High
Eigen Value Decomposition	No	Variable	Variable

The theoretical investigation underscores that no single decomposition method is universally optimal. SVD stands out for its robustness and accuracy, particularly in ill-posed or data-sensitive problems while Eigenvalue decomposition can be sensitive and unstable especially for non-normal matrices. In choosing a decomposition method, one must weigh stability, accuracy, computational cost, and the structural properties of the matrix. Understanding the numerical behavior of these algorithm ensures more reliable and informed application in mathematical modeling and scientific computing.

Computational Experiments

To assess the performance of Singular Value Decomposition and Eigenvalue Decomposition, a series of computational experiments were carried out on randomly generated square matrices of growing size. These matrices, measuring 100x100, 1000x1000, and 5000x5000, were created using a uniform random distribution across the interval [0,1], ensuring that each entry conformed to this rule. The computations were performed on

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

a system boasting a core i3 processor, 16 GB of RAM, and an SSD drive. Standard numerical linear algebra libraries were employed to calculate both SVD and Eigenvalue Decomposition, with execution time and memory usage meticulously recorded for the purpose of performance comparison.

Generating Random Matrices (Uniform Distribution)

We generate a matrix $A \epsilon R^{n \times n}$ where each entry is drawn from a uniform distribution on [0,1].

(i) Generating the time

```
In [2]: import numpy as np
   ...: import time
   ...:
   ...: np.random.seed(0)
   ...:
   ...: def measure_time(func, A, runs=3):
   ...:     times = []
   ...:     for _ in range(runs):
   ...:         start = time.time()
   ...:         func(A)
   ...:         times.append(time.time() - start)
   ...:     return sum(times) / len(times)
   ...:
   ...: A_100 = np.random.rand(100,100)
   ...: A_1000 = np.random.rand(1000,1000)
   ...:
   ...: svd_100 = measure_time(np.linalg.svd, A_100)
   ...: eig_100 = measure_time(np.linalg.eig, A_100)
   ...:
   ...: svd_1000 = measure_time(np.linalg.svd, A_1000)
   ...: eig_1000 = measure_time(np.linalg.eig, A_1000)
   ...:
   ...: print(svd_100, eig_100, svd_1000, eig_1000)
0.0060490767161051435 0.0 0.23761812845865884 0.5907041231791178

In [3]:
```

```
In [11]: import numpy as np
    ...: import time
    ...:
    ...: np.random.seed(0)
    ...:
    ...: def measure_time(func, A, runs=3):
    ...:     times = []
    ...:     for _ in range(runs):
    ...:         start = time.time()
    ...:         func(A)
    ...:         times.append(time.time() - start)
    ...:     return sum(times) / len(times)
    ...:
    ...: A_5000 = np.random.rand(5000,5000)
    ...:
    ...:
    ...: svd_5000 = measure_time(np.linalg.svd, A_5000)
    ...: eig_5000 = measure_time(np.linalg.eig, A_5000)
    ...:
    ...: print(svd_5000, eig_5000)
54.042051235834755 63.84279171625773
```

(ii) Generating the memory usage

```
In [21]: # Memory in MB

In [22]: A_100.nbytes / (1024**2)
Out[22]: 0.0762939453125

In [23]: A_1000.nbytes / (1024**2)
Out[23]: 7.62939453125

In [24]: A_5000.nbytes / (1024**2)
Out[24]: 190.73486328125

In [25]:
```

(iii) Generating condition number

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

```
In [28]: # Condition Number

In [29]: cond_100 = np.linalg.cond(A_100)
    ...: print("Condition Number:", cond_100)
Condition Number: 2651.2010650061616

In [30]: cond_1000 = np.linalg.cond(A_1000)
    ...: print("Condition Number:", cond_1000)
Condition Number: 157686.8212437287

In [31]: cond_5000 = np.linalg.cond(A_5000)
    ...: print("Condition Number:", cond_5000)
Condition Number: 346430.299261837

In [32]:
```

xperimental Results

Matrix Size	SVD time (s)	Eigen time (s)	Memory usage (MB)	Condition number	Observation
100x100	0.0061	0.0000	0.07	Low	Both methods very fast
1000x1000	0.2376	0.5907	7.63	Moderate	SVD faster
5000x5000	54.0421	63.8427	190.73	High	SVD more stable and slightly faster

DISCUSSION OF RESULTS

The results clearly show that as matrix size increases, both computational time and memory usage skyrocket. For smaller matrices, like those with (100x100), both Singular Value Decomposition (SVD) and Eigenvalue Decomposition run smoothly with virtually no computation time.

However, as we move to medium-sized matrices with (1000x1000), SVD starts to outperform Eigenvalue Decomposition in terms of execution speed, suggesting it's more efficient on our tested system.

When we get to large matrices with (5000x5000), both methods become computationally expensive, but SVD still manages to edge out Eigenvalue Decomposition. Furthermore, the high condition number indicates that the matrix is ill-conditioned, which is a problem. In such cases, SVD offers superior numerical stability compared to Eigenvalue Decomposition, making it a more reliable choice for large-scale problems.

Overall, our results indicate that while Eigenvalue Decomposition might be efficient for smaller problems, SVD becomes the more advantageous option for larger and ill-conditioned matrices due to its stability and consistent performance.

Theoretical illustration of numerical Stability of SVD and Eigenvalue Decomposition Methods.

Here, we present solved examples for Singular Value Decomposition (SVD) and Eigenvalue Decomposition. Each example includes forward and backward error analysis to assess numerical stability and accuracy. A concluding section compares computational speed and robustness across methods.

Singular Value Decomposition (SVD)

Example: Low-Rank Approximation

Let

$$A = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}$$

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

$$A^T A = \begin{bmatrix} 10 & 6 \\ 6 & 10 \end{bmatrix} \Rightarrow \lambda = 16{,}4 \Rightarrow \sigma = 4{,}2$$

SVD:

$$A = \square \Sigma V^T, \Sigma = \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix}$$

Low-rank approximation using only $\sigma_1 = 4$

$$A_1 = \sigma_1 u_1 v_1^T$$

Conclusion: SVD is numerically robust, even under noise or Ill-conditioning.

3.6.2. Eigenvalue Decomposition

Example: Perturbation Sensitivity

Let

$$A = \begin{bmatrix} 4 & 1 \\ 2 & 3 \end{bmatrix}$$

Characteristic polynomial:

$$\lambda^2 - 7\lambda + 10 = 0 \Rightarrow \lambda = 5, 2$$

Eigenvectors:

$$x_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad x_2 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$$

Perturb $A \rightarrow A^\star = \begin{bmatrix} 4 & 1.01 \\ 2 & 3 \end{bmatrix}$

Characteristic polynomial:

$$\lambda^{\star 2} - 7\lambda^\star + 9.98 = 0 \Rightarrow \lambda^\star = 5.0067, \qquad 1.9933$$

Eigenvectors:

$$x_1^\star = \begin{bmatrix} -1.9933 \\ 1 \end{bmatrix}, \qquad x_2^\star = \begin{bmatrix} -5.0067 \\ 1 \end{bmatrix}$$

A small changes in matrix entries leads to noticeably changes in eigenvalues and eigenvectors, demonstrating sensitivity to perturbation.

Comparative Applications of SVD and Eigenvalue decomposition methods in solving various cases of linear equations.

Here, we compare the SVD and Eigenvalue matrix decomposition techniques in solving various cases of linear systems. For each method, we apply it to two representative systems it is best suited for and provide a complete, worked example with explicit formulation of the system, augmented matrix (if applicable), and step-by-step solution. Accuracy and stability are assessed to determine the approach for different system structures.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

SVD: Rank-Deficient System

Example: Low-Rank Approximation

Let

$$A = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}$$

$$A^T A = \begin{bmatrix} 10 & 6 \\ 6 & 10 \end{bmatrix} \Rightarrow \lambda = 16{,}4 \Rightarrow \sigma = 4{,}2$$

SVD:

$$A = U\Sigma V^T, \Sigma = \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix}$$

Low-rank approximation using only $\sigma_1 = 4$

$$A_1 = \sigma_1 u_1 v_1^T$$

Conclusion: SVD is numerically robust, even under noise or Ill-conditioning.

3.6.2. Eigenvalue Decomposition

Example: Perturbation Sensitivity

Let

$$A = \begin{bmatrix} 4 & 1 \\ 2 & 3 \end{bmatrix}$$

Characteristic polynomial:

$$\lambda^2 - 7\lambda + 10 = 0 \Rightarrow \lambda = 5, 2$$

Eigenvectors:

$$x_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad x_2 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$$

Perturb $A \rightarrow A^\star = \begin{bmatrix} 4 & 1.01 \\ 2 & 3 \end{bmatrix}$

Characteristic polynomial:

$$\lambda^{\star 2} - 7\lambda^\star + 9.98 = 0 \Rightarrow \lambda^\star = 5.0067, \qquad 1.9933$$

Eigenvectors:

$$x_1^\star = \begin{bmatrix} -1.9933 \\ 1 \end{bmatrix}, \qquad x_2^\star = \begin{bmatrix} -5.0067 \\ 1 \end{bmatrix}$$

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

SUMMARY, RECOMMENDATION AND CONCLUSION

Summary

This study compared two key matrix decomposition methods - Eigenvalue Decomposition and Singular Value Decomposition (SVD) - to assess their effectiveness in solving large-scale linear systems. The analysis focused on their numerical stability, computational efficiency, and suitability for different types of problems.

Eigenvalue Decomposition proved particularly useful for theoretical analysis, offering insights into matrix structure, stability, and diagonalization. It's also a crucial component in spectral analysis and applications like Principal Component Analysis (PCA). However, our experiments showed that its computational performance was outpaced by SVD in solving large-scale systems.

SVD, on the other hand, demonstrated strong performance across the board, showcasing better computational efficiency and superior numerical stability - especially when dealing with large, ill-conditioned matrices. This makes SVD a reliable choice for practical applications, including machine learning, recommender systems, and dimensionality reduction. Although SVD is theoretically computationally intensive, our results suggest that it can outperform Eigenvalue Decomposition in practice, depending on the system configuration and implementation.

Ultimately, the study highlights that no single decomposition method is universally optimal. The choice of method depends on the problem context, matrix properties, and computational environment. For large-scale problems, especially those involving ill-conditioned matrices, SVD provides a more stable and effective solution, while Eigenvalue Decomposition remains valuable for theoretical insights and structured analyses.

Recommendation

1. Use SVD for solving ill-conditioned or rank-deficient problems where accuracy is critical, such as in signal processing, machine learning, or inverse problems.

2. Employ Eigenvalue decomposition mainly for spectral analysis, stability studies, or system where matrix diagonalization is required.

Conclusion

There is no one size solution when it comes to solving large scale systems of equations using matrix decomposition. Each method has its advantages depending on the nature of the matrix and the requirements of the problem. Eigenvalue decomposition serves well in theoretical and analytical contexts while SVD offer greater stability for ill-conditioned or over determined systems. The best approach is to first analyze the structure and condition of the matrix. Based on this, select the decomposition method that offers the best trade off between speed, stability, and accuracy. In practice, hybrid approaches such as combining decomposition with iterative refinement or regularization can provide even better performance, particularly for large scale problems encountered in scientific computing, engineering, and data intensive applications.

REFERENCES

1. Householder, A.S. (1954). Principle of numerical analysis. McGraw-Hill
2. Trefethen, L.N., and Bau, D. (1997). Numerical linear algebra. SIAM.
3. Meyer, C.D. (2000). Matrix analysis and applied linear algebra. SIAM.
4. Benzi, M. (2002). Preconditioning techniques for large linear systems: A survey. Journal of Computational Physics, 182(2), 148-477.
5. Higham, N.J. (2002). Accuracy and stability of numerical algorithm (2nd ed.). SIAM.
6. Saad, Y. (2003). Iterative method for sparse linear syatem (2nd ed.). SIAM.
7. Chandra, R. and Guha R. (2007). Advanced matrix computations. Academic Press.
8. Demmel, J.W. (2007). Applied numerical linear algebra. SIAM.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue V, May 2026

9. Ryan, L. (2008). Applied numerical linear algebra: The diverse toolkit. Cambridge University Press.
10. Golub, G.H. and Ortega, J.M. (2009). Scientific computing: An introduction with parallel computing (2nd ed.) Academic Press.
11. Trefethen, L.N. (2014). Approximation theory and approximation practice. SIAM.
12. C. L. Ejikeme, C.L., Unaegbu, E.N., and Okofu, M.B. (2015). The Existence of a Periodic Solution of a Boundary Value Problem for a Linear System, IOSR Journal of Mathematics (IOSR-JM) e-ISSN: 2278-5728, p-ISSN: 2319-765X. Volume 11, Issue 4 Ver. I. PP 73-82
13. Gentle, J.E. (2017). Numerical linear algebra for applications in statistics (2nd ed.) Spriger.
14. Lu, J. (2021). Numerical matrix decomposition : A review of LU, QR, SVD, and Cholesky methods.
15. Peng, R., and Vempala, S. (2021). Solvimg sparse linear systems faster than matrix multiplication.
16. Duan, R., Jiang, C. and Jain, H.K. (2022). Combining review based collaborative filtering and matrix factorization: A solution to rating's sparsity problem. Decision Support Systems, 156, 113748.