Low-Rank Approx. Technique for Neural Signal Denoising via SVD

Abhinav Raj, 2311006
School Of Physical Sciences, National Institute Of Science Education and Research, HBNI, Bhubaneswar-752050
(Dated: nov 17 , 2025)

This document presents theory, a worked numeric SVD example, and a clear, line-by-line explanation of an SVD-based neural signal denoising implementation in Python. It also instructs how to save and embed the output figures (variance curve, time-domain plots, and PSD comparison) into this LaTeX report.

I. INTRODUCTION

Neural recordings (EEG, LFP, multi-electrode arrays) often contain structured signals mixed with unstructured noise. When channel-wise signals share correlated structure (common oscillations or evoked responses), the data matrix tends to have a few dominant modes. SVD decomposes data into orthogonal modes ranked by energy; a low-rank reconstruction keeps dominant modes and suppresses noise.

This report explains the mathematics of SVD, demonstrates a hand-worked 2×2 SVD, embeds your Python code, explains each step, and shows how to include resulting plots into the PDF.

II. MATHEMATICAL BACKGROUND: SVD AND ENERGY

For any real matrix $A \in \mathbb{R}^{m\times n}$ of rank r, the Singular Value Decomposition (SVD) is

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

where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal matrices and Σ is $m\times n$ with non-negative diagonal singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$ (rest zeros).

Key facts:

- $\|A\|_F^2 = \sum_{i=1}^r \sigma_i^2$ (Frobenius norm equals sum of squared singular values).
- Truncated (rank-k) approximation $A_k = U_k\Sigma_k V_k^T = \sum_{i=1}^k \sigma_i u_i v_i^T$ minimizes $\|A - A_k\|_F$ among all rank-k matrices (Eckart–Young theorem).
- Fraction of energy explained by top k modes:

$$\mathrm{Var}(k) = \frac{\sum_{i=1}^k \sigma_i^2}{\sum_{i=1}^r \sigma_i^2}.$$

III. MANUAL WORKED EXAMPLE (2×2) WITH ARITHMETIC

We show a full manual SVD for a simple symmetric matrix:

$$A = \begin{pmatrix} 3 & 1 \\ 1 & 3 \end{pmatrix}.$$

A. Step 1: eigenvalues of A

Solve $\det(A - \lambda I) = 0$:

$$\det\begin{pmatrix} 3-\lambda & 1 \\ 1 & 3-\lambda \end{pmatrix} = (3-\lambda)^2 - 1 = 0.$$

So:

$$(3-\lambda)^2 = 1 \quad \Rightarrow \quad 3 - \lambda = \pm 1.$$

Thus

$$\lambda_1 = 3 - (-1) = 4, \qquad \lambda_2 = 3 - 1 = 2.$$

B. Step 2: eigenvectors (normalized)

For $\lambda_1 = 4$, solve $(A - 4I)v = 0$:

$$\begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix} v = 0 \Rightarrow v \propto \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Normalize:

$$u_1 = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

For $\lambda_2 = 2$, eigenvector $\propto (1, -1)^T$. Normalize:

$$u_2 = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -1 \end{pmatrix}.$$

Set $U = \begin{pmatrix} u_1 & u_2 \end{pmatrix}$ (orthonormal).

C. Step 3: singular values

Because A is symmetric, $A = U\Lambda U^T$ with $\Lambda = \mathrm{diag}(4, 2)$. For a symmetric matrix A, singular values are |eigenvalues| and here they are positive:

$$\sigma_1 = 4, \qquad \sigma_2 = 2.$$

Thus $\Sigma = \mathrm{diag}(4, 2)$.

D. Step 4: SVD assembly

Let $V = U$ (since A is symmetric and positive-definite in this example). Then

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

which reconstructs A. Check Frobenius identity:

$$\sum_{i,j} A_{ij}^2 = 3^2 + 1^2 + 1^2 + 3^2 = 9 + 1 + 1 + 9 = 20,$$

and $\sigma_1^2 + \sigma_2^2 = 4^2 + 2^2 = 16 + 4 = 20$ — they match.

E. Low-rank (rank-1) approximation

Keep only σ_1:

$$A_1 = \sigma_1 u_1 v_1^T = 4 \cdot \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \cdot \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \end{pmatrix} = 4 \cdot \frac{1}{2} \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 2 \\ 2 & 2 \end{pmatrix}.$$

Error (Frobenius squared) is:

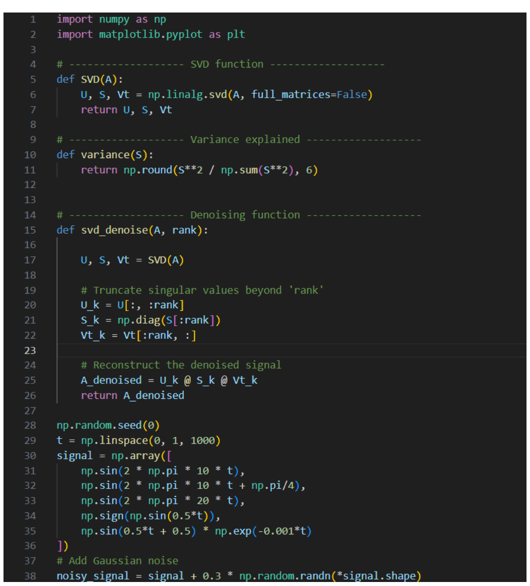
$$\|A - A_1\|_F^2 = \sigma_2^2 = 2^2 = 4.$$

Relative error:

$$\frac{\|A - A_1\|_F^2}{\|A\|_F^2} = \frac{4}{20} = 0.20 = 20\%.$$

This small example illustrates: top singular modes capture most energy; truncation error equals sum of squared discarded singular values.

IV. PYTHON CODE (EMBEDDED) AND LINE-BY-LINE EXPLANATION



```python
import numpy as np
import matplotlib.pyplot as plt

# ------------------ SVD function ------------------
def SVD(A):
    U, S, Vt = np.linalg.svd(A, full_matrices=False)
    return U, S, Vt

# ------------------ Variance explained ------------------
def variance(S):
    return np.round(S**2 / np.sum(S**2), 6)


# ------------------ Denoising function ------------------
def svd_denoise(A, rank):

    U, S, Vt = SVD(A)

    # Truncate singular values beyond 'rank'
    U_k = U[:, :rank]
    S_k = np.diag(S[:rank])
    Vt_k = Vt[:rank, :]

    # Reconstruct the denoised signal
    A_denoised = U_k @ S_k @ Vt_k
    return A_denoised

np.random.seed(0)
t = np.linspace(0, 1, 1000)
signal = np.array([
    np.sin(2 * np.pi * 10 * t),
    np.sin(2 * np.pi * 10 * t + np.pi/4),
    np.sin(2 * np.pi * 20 * t),
    np.sign(np.sin(0.5*t)),
    np.sin(0.5*t + 0.5) * np.exp(-0.001*t)
])
# Add Gaussian noise
noisy_signal = signal + 0.3 * np.random.randn(*signal.shape)
```

FIG. 1. code snapshot

```python
# ------------------ Plotting ------------------

U, S, Vt = SVD(noisy_signal)
y = np.cumsum(variance(S))*100


plt.plot(np.cumsum(variance(S))*100)
plt.title("Cumulative variance explained")
plt.xlabel("Number of singular values")
plt.ylabel("Variance (%)")
plt.show()

rank = 3
denoised_signal = svd_denoise(noisy_signal, rank)

plt.figure(figsize=(12, 6))
plt.subplot(2, 1, 1)
plt.title("Noisy Neural Signals")
plt.plot(t, noisy_signal.T)
plt.subplot(2, 1, 2)
plt.title(f"Denoised Neural Signals (Rank={rank})")
plt.plot(t, denoised_signal.T)
plt.xlabel("Time (s)")
plt.tight_layout()
plt.show()

from scipy.signal import welch
f, Pxx_noisy = welch(noisy_signal[0], fs=1000)
f, Pxx_denoised = welch(denoised_signal[0], fs=1000)
plt.figure()
plt.semilogy(f, Pxx_noisy, label='Noisy')
plt.semilogy(f, Pxx_denoised, label='Denoised')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Power spectral density')
plt.legend()
plt.title('Power spectrum before vs after SVD denoising')
plt.show()
```

FIG. 2. code snapshot

V. PYTHON IMPLEMENTATION

A. SVD of a Neural Signal Matrix

Using NumPy:

```
U, S, Vt = np.linalg.svd(A, full_matrices=False)
```

B. Low-Rank Reconstruction

```
U_k = U[:, :k]
S_k = np.diag(S[:k])
V_k = Vt[:k, :]
A_denoised = U_k @ S_k @ V_k
```

C. Variance Explained

```
variance = S**2 / np.sum(S**2)
```

This quantity determines how much energy each singular component contributes.

VI. LOW-RANK APPROXIMATION AND NEURAL SIGNAL DENOISING

A. Motivation

Neural signals exhibit:

- strong correlated oscillations (theta, alpha, gamma bands),
- low intrinsic dimensionality,

- noise that spreads across many weak singular directions.

Thus SVD can effectively separate structured neural activity from noise.

B. What is Low-Rank Approximation?

A rank-k approximation of A:

$$A_k = U_k D_k V_k^T$$

minimizes

$$\|A - A_k\|_F$$

among all rank-k matrices. This is the Eckart–Young theorem.

C. How SVD Helps?

If the singular spectrum satisfies

$$\sigma_1 \gg \sigma_2 \gg \cdots \gg \sigma_k \gg \sigma_{k+1} \approx \cdots \approx \sigma_r,$$

then:

- large σ_i correspond to neural dynamics,
- small σ_i correspond to noise.

Thus discarding the small singular components produces a robust denoised signal.

D. Error in Low-Rank Approximation

Relative retained energy:

$$\frac{\|A_k\|_F^2}{\|A\|_F^2} = \frac{\sigma_1^2 + \cdots + \sigma_k^2}{\sigma_1^2 + \cdots + \sigma_r^2}.$$

Error:

$$\frac{\|A - A_k\|_F^2}{\|A\|_F^2} = \frac{\sigma_{k+1}^2 + \cdots + \sigma_r^2}{\sum_{i=1}^r \sigma_i^2}.$$

E. output plots

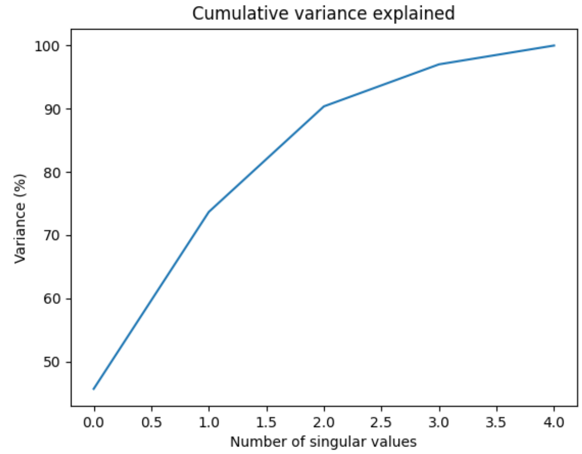


FIG. 3. plot for variance

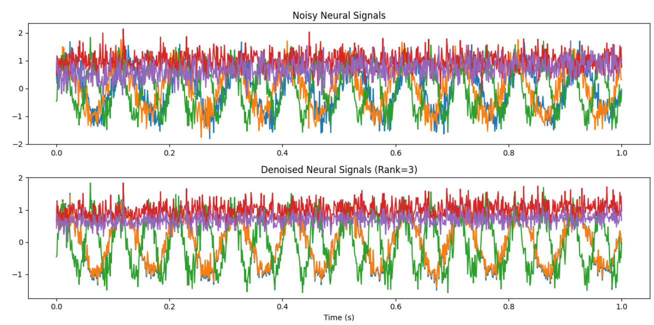


FIG. 4. Noised and denoised signal comparision

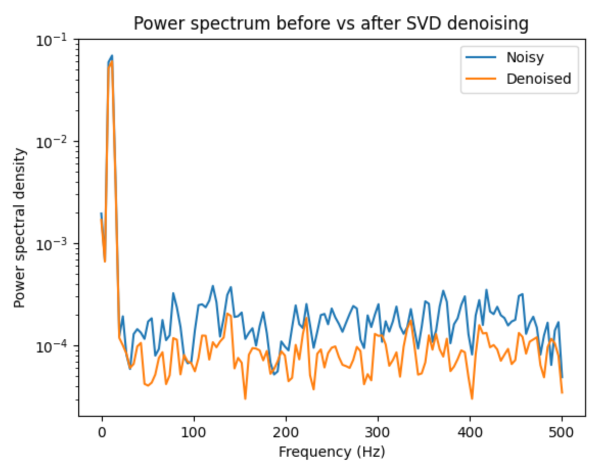


FIG. 5. power density spectrum plot

VII. CONCLUSION

In this project, we applied the Singular Value Decomposition to denoise multi-channel neural time series. SVD provides an interpretable and robust way to separate strongly correlated neural activity from uncorrelated noise. Low-rank approximation significantly reduces noise while preserving the structure of neural oscillations. The variance spectrum gives a natural criterion for rank selection. This makes SVD a powerful tool for neural preprocessing, dimensionality reduction, and feature extraction.

VIII. ACKNOWLEDGEMENT

I thank the instructor and course for introducing the mathematical foundations of SVD. I am grateful to the authors of texts on linear algebra, signal processing, and machine learning for the insights that guided this work.

IX. REFERENCES

1. Gilbert Strang, *Introduction to Linear Algebra*.
2. S. Mallat, *A Wavelet Tour of Signal Processing*.
3. Brunton & Kutz, *Data-Driven Science and Engineering*.
4. EEG and LFP Signal Processing papers and resources.