TD_Solution_With_Code

January 12, 2026

1 Exercise 1- Determinant, EigenValues, EigenVectors and Diagonalization

Consider the matrix:

$$A = \begin{pmatrix} 3/2 & 1/2 & -1/2 \\ -1/2 & 5/2 & 1/2 \\ -1 & 1 & 2 \end{pmatrix}$$

1. Compute the **determinant** of A and verify if it is **invertible**.
2. Find all **eigenvalues** of A by solving the characteristic polynomial.
3. For each **eigenvalue**, determine a corresponding **eigenvector**.
4. Check whether A is **diagonalizable**, and if yes, find matrices P and D such that $A = PDP^{-1}$.

1.1 Solution

1. To compute the determinant of A, we use the formula for the determinant of a 3x3 matrix:

   $$\det(A) = a(ei - fh) - b(di - fg) + c(dh - eg)$$

   where the matrix elements are:

   $$A = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

   Plugging in the values from matrix A:

   $$\det(A) = \frac{3}{2}\left(\frac{5}{2}\cdot 2 - \frac{1}{2}\cdot 1\right) - \frac{1}{2}\left(-\frac{1}{2}\cdot 2 - \frac{1}{2}\cdot(-1)\right) + \left(-\frac{1}{2}\right)\left(-\frac{1}{2}\cdot 1 - \frac{5}{2}\cdot(-1)\right)$$

   Calculating step by step:

   - First term: $\frac{3}{2}(5 - \frac{1}{2}) = \frac{3}{2}\cdot\frac{9}{2} = \frac{27}{4}$
   - Second term: $-\frac{1}{2}(-1 + \frac{1}{2}) = -\frac{1}{2}\cdot(-\frac{1}{2}) = \frac{1}{4}$
   - Third term: $-\frac{1}{2}(-\frac{1}{2} + \frac{5}{2}) = -\frac{1}{2}\cdot 2 = -1$

$$\det(A) = \frac{27}{4} + \frac{1}{4} - 1 = \frac{28}{4} - 1 = 7 - 1 = 6$$

Since $\det(A) = 6 \neq 0$, matrix A is invertible.

2. To find the eigenvalues, we solve the characteristic polynomial given by:

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

where I is the identity matrix and λ is the eigenvalue. Thus:

$$A - \lambda I = \begin{pmatrix} \frac{3}{2} - \lambda & \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{5}{2} - \lambda & \frac{1}{2} \\ -1 & 1 & 2 - \lambda \end{pmatrix}$$

Computing the determinant:

$$\det(A{-}\lambda I) = \left(\frac{3}{2} - \lambda\right)\left[\left(\frac{5}{2} - \lambda\right)(2 - \lambda) - \frac{1}{2} \cdot 1\right] - \frac{1}{2}\left[-\frac{1}{2}(2 - \lambda) - \frac{1}{2} \cdot (-1)\right] + \left(-\frac{1}{2}\right)\left[-\frac{1}{2} \cdot 1 - \left(\frac{5}{2} - \lambda\right)(-1$$

Expanding the first bracket:

$$\left(\frac{5}{2} - \lambda\right)(2 - \lambda) - \frac{1}{2} = 5 - \frac{5\lambda}{2} - 2\lambda + \lambda^2 - \frac{1}{2} = \lambda^2 - \frac{9\lambda}{2} + \frac{9}{2}$$

Expanding the second bracket:

$$-\frac{1}{2}(2 - \lambda) - \frac{1}{2}(-1) = -1 + \frac{\lambda}{2} + \frac{1}{2} = \frac{\lambda}{2} - \frac{1}{2}$$

Expanding the third bracket:

$$-\frac{1}{2} - (-1)\left(\frac{5}{2} - \lambda\right) = -\frac{1}{2} + \frac{5}{2} - \lambda = 2 - \lambda$$

Combining all terms and simplifying yields the characteristic polynomial:

$$-\lambda^3 + 6\lambda^2 - 11\lambda + 6 = 0$$

Or equivalently:

$$\lambda^3 - 6\lambda^2 + 11\lambda - 6 = 0$$

We can factor this as:

$$(\lambda - 1)(\lambda - 2)(\lambda - 3) = 0$$

Therefore, the eigenvalues are:

- $\lambda_1 = 1$
- $\lambda_2 = 2$

- $\lambda_3 = 3$

3. For each eigenvalue, we find the corresponding eigenvector by solving $(A - \lambda I)\vec{v} = \vec{0}$ using **Gauss-Jordan elimination**:

   For $\lambda_1 = 1$:

   We need to solve:

$$(A - I)\vec{v} = \begin{pmatrix} 1/2 & 1/2 & -1/2 \\ -1/2 & 3/2 & 1/2 \\ -1 & 1 & 1 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

   Apply row reduction to the augmented matrix:

   Starting matrix:

$$\left[\begin{array}{ccc|c} 1/2 & 1/2 & -1/2 & 0 \\ -1/2 & 3/2 & 1/2 & 0 \\ -1 & 1 & 1 & 0 \end{array}\right]$$

   $R_1 \times 2$:

$$\left[\begin{array}{ccc|c} 1 & 1 & -1 & 0 \\ -1/2 & 3/2 & 1/2 & 0 \\ -1 & 1 & 1 & 0 \end{array}\right]$$

   $R_2 + \frac{1}{2}R_1$ and $R_3 + R_1$:

$$\left[\begin{array}{ccc|c} 1 & 1 & -1 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 2 & 0 & 0 \end{array}\right]$$

   $R_2 \times \frac{1}{2}$:

$$\left[\begin{array}{ccc|c} 1 & 1 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 \end{array}\right]$$

   $R_3 - 2R_2$:

$$\left[\begin{array}{ccc|c} 1 & 1 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right]$$

   $R_1 - R_2$:

$$\left[\begin{array}{ccc|c} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right]$$

From the reduced form:

- $v_1 - v_3 = 0 \Rightarrow v_1 = v_3$
- $v_2 = 0$
- v_3 is free

Choosing $v_3 = 1$, we get: $\vec{v}_1 = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$

For $\lambda_2 = 2$:

We need to solve:

$$(A - 2I)\vec{v} = \begin{pmatrix} -1/2 & 1/2 & -1/2 \\ -1/2 & 1/2 & 1/2 \\ -1 & 1 & 0 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Apply row reduction:

Starting matrix:

$$\left[\begin{array}{ccc|c} -1/2 & 1/2 & -1/2 & 0 \\ -1/2 & 1/2 & 1/2 & 0 \\ -1 & 1 & 0 & 0 \end{array}\right]$$

$R_1 \times (-2)$:

$$\left[\begin{array}{ccc|c} 1 & -1 & 1 & 0 \\ -1/2 & 1/2 & 1/2 & 0 \\ -1 & 1 & 0 & 0 \end{array}\right]$$

$R_2 + \frac{1}{2}R_1$ and $R_3 + R_1$:

$$\left[\begin{array}{ccc|c} 1 & -1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right]$$

$R_3 - R_2$:

$$\left[\begin{array}{ccc|c} 1 & -1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right]$$

$R_1 - R_2$:

$$\left[\begin{array}{ccc|c} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right]$$

From the reduced form:

- $v_1 - v_2 = 0 \Rightarrow v_1 = v_2$
- $v_3 = 0$
- v_2 is free

Choosing $v_2 = 1$, we get: $\vec{v}_2 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$

For $\lambda_3 = 3$:

We need to solve:

$$(A - 3I)\vec{v} = \begin{pmatrix} -3/2 & 1/2 & -1/2 \\ -1/2 & -1/2 & 1/2 \\ -1 & 1 & -1 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Apply row reduction:

Starting matrix:

$$\left[\begin{array}{ccc|c} -3/2 & 1/2 & -1/2 & 0 \\ -1/2 & -1/2 & 1/2 & 0 \\ -1 & 1 & -1 & 0 \end{array}\right]$$

$R_1 \times (-2/3)$:

$$\left[\begin{array}{ccc|c} 1 & -1/3 & 1/3 & 0 \\ -1/2 & -1/2 & 1/2 & 0 \\ -1 & 1 & -1 & 0 \end{array}\right]$$

$R_2 + \frac{1}{2}R_1$ and $R_3 + R_1$:

$$\left[\begin{array}{ccc|c} 1 & -1/3 & 1/3 & 0 \\ 0 & -2/3 & 2/3 & 0 \\ 0 & 2/3 & -2/3 & 0 \end{array}\right]$$

$R_2 \times (-3/2)$:

$$\left[\begin{array}{ccc|c} 1 & -1/3 & 1/3 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 2/3 & -2/3 & 0 \end{array}\right]$$

$R_3 - \frac{2}{3}R_2$:

$$\left[\begin{array}{ccc|c} 1 & -1/3 & 1/3 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right]$$

$R_1 + \frac{1}{3}R_2$:

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right]$$

From the reduced form:

- $v_1 = 0$
- $v_2 - v_3 = 0 \Rightarrow v_2 = v_3$
- v_3 is free

Choosing $v_3 = 1$, we get: $\vec{v}_3 = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$

4. **Diagonalizability Check:**

A matrix A is diagonalizable if it has n linearly independent eigenvectors, where n is the dimension.

We have found three eigenvectors:

$$\vec{v}_1 = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \quad \vec{v}_2 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \quad \vec{v}_3 = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$$

To check linear independence, we form the matrix with these vectors as columns and check its determinant:

$$P = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$$

$$\det(P) = 1(1 \cdot 1 - 1 \cdot 0) - 1(0 \cdot 1 - 1 \cdot 1) + 0(0 \cdot 0 - 1 \cdot 1) = 1(1) - 1(-1) + 0 = 1 + 1 = 2 \neq 0$$

Since $\det(P) \neq 0$, the eigenvectors are linearly independent, and **A is diagonalizable**.

The diagonal matrix D contains the eigenvalues:

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix}$$

And the decomposition is:

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

where:

$$P = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}, \quad D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix}$$

[ ]:

```python
import numpy as np

# Define matrix A
A = np.array([[3 / 2, 1 / 2, -1 / 2], [-1 / 2, 5 / 2, 1 / 2], [-1, 1, 2]])

print("Matrix A:")
print(A)
```

[ ]:

```python
# 1. Compute the determinant
det_A = np.linalg.det(A)
print(f"Determinant of A: {det_A:.4f}")
print(f"Is A invertible? {det_A != 0}")
print(f"Since det(A)   0, matrix A is invertible.")
```

[ ]:

```python
# 2. Find eigenvalues
eigenvalues, eigenvectors = np.linalg.eig(A)

print("Eigenvalues of A:")
for i, eigenvalue in enumerate(eigenvalues):
    print(f"   {i+1} = {eigenvalue:.6f}")
```

[ ]:

```python
# 3. Find eigenvectors for each eigenvalue
print("\nEigenvectors of A:")
print("Matrix P (columns are eigenvectors):")
print(eigenvectors)
print("\nEach column corresponds to an eigenvalue:")
for i in range(len(eigenvalues)):
    print(f"\nEigenvector for  {i+1} = {eigenvalues[i]:.6f}:")
    print(eigenvectors[:, i])
```

[ ]:

```python
# 4. Check if A is diagonalizable
# A matrix is diagonalizable if it has n linearly independent eigenvectors
# where n is the dimension of the matrix

# Check if eigenvectors are linearly independent by checking rank of P
P = eigenvectors
rank_P = np.linalg.matrix_rank(P)
```

```
print(f"Rank of eigenvector matrix P: {rank_P}")
print(f"Dimension of A: {A.shape[0]}")

if rank_P == A.shape[0]:
    print(
        "\nMatrix A is DIAGONALIZABLE because it has 3 linearly independent eigenvectors."
    )
else:
    print("\nMatrix A is NOT diagonalizable.")
```

[ ]:

```
# Construct diagonal matrix D with eigenvalues on the diagonal
D = np.diag(eigenvalues)

print("\nMatrix D (diagonal matrix of eigenvalues):")
print(D)

print("\nMatrix P (eigenvectors as columns):")
print(P)
```

[ ]:

```
# Verify that A = P * D * P^(-1)
P_inv = np.linalg.inv(P)
A_reconstructed = P @ D @ P_inv

print("\nVerification: A = P * D * P^(-1)")
print("\nOriginal matrix A:")
print(A)
print("\nReconstructed A from P * D * P^(-1):")
print(A_reconstructed)
print("\nDifference (should be close to zero):")
print(np.abs(A - A_reconstructed))
print(f"\nMaximum difference: {np.max(np.abs(A - A_reconstructed)):.2e}")

if np.allclose(A, A_reconstructed):
    print("\n  Verification successful! A = P D P^(-1)")
```

---

2 Exercise 2 - Singular Value Decomposition (SVD)

Consider the matrix:

$$S = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

1. Show that S has a single eigenvalue $\lambda = 1$ with only one independent eigenvector. Conclude that S is not diagonalizable.

2. Compute S^TS and its eigenvalues $\{\lambda_1, \lambda_2\}$; set the singular values $\sigma_i = \sqrt{\lambda_i}$.
3. Find the right singular vectors v_i as eigenvectors of S^TS and the left singular vectors $u_i = \frac{Sv_i}{\sigma_i}$.
4. Conclude the **SVD** $S = U\Sigma V^T$ with $U = [u_1\ u_2]$, $V = [v_1\ v_2]$ and $\Sigma = \operatorname{diag}(\sigma_1, \sigma_2)$.

2.1 Solution

2.1.1 Part 1: Eigenvalues and Diagonalizability

To find the eigenvalues, we solve the characteristic equation:

$$\det(S - \lambda I) = 0$$

$$S - \lambda I = \begin{pmatrix} 1-\lambda & 1 \\ 0 & 1-\lambda \end{pmatrix}$$

$$\det(S - \lambda I) = (1-\lambda)(1-\lambda) - 0 \cdot 1 = (1-\lambda)^2 = 0$$

Therefore, $\lambda = 1$ is the only eigenvalue with **algebraic multiplicity 2**.

Now, find the eigenvectors by solving $(S - I)\vec{v} = \vec{0}$:

$$(S - I) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

Apply row reduction:

$$\left[\begin{array}{cc|c} 0 & 1 & 0 \\ 0 & 0 & 0 \end{array}\right]$$

From this: $v_2 = 0$ and v_1 is free.

Choosing $v_1 = 1$, we get: $\vec{v} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

Conclusion: There is only **one independent eigenvector** for eigenvalue $\lambda = 1$. The geometric multiplicity (1) is less than the algebraic multiplicity (2), so S **is NOT diagonalizable**.

2.1.2 Part 2: Compute S^TS and Singular Values

First, compute S^T:

$$S^T = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$

Now compute S^TS:

$$S^TS = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}$$

Find eigenvalues of S^TS by solving $\det(S^TS - \lambda I) = 0$:

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

Using the quadratic formula:

$$\lambda = \frac{3 \pm \sqrt{9-4}}{2} = \frac{3 \pm \sqrt{5}}{2}$$

Therefore:

- $\lambda_1 = \frac{3+\sqrt{5}}{2} \approx 2.618$
- $\lambda_2 = \frac{3-\sqrt{5}}{2} \approx 0.382$

The singular values are:

$$\sigma_1 = \sqrt{\lambda_1} = \sqrt{\frac{3+\sqrt{5}}{2}} \approx 1.618$$

$$\sigma_2 = \sqrt{\lambda_2} = \sqrt{\frac{3-\sqrt{5}}{2}} \approx 0.618$$

2.1.3 Part 3: Right and Left Singular Vectors

Right singular vectors are eigenvectors of S^TS.

For $\lambda_1 = \frac{3+\sqrt{5}}{2}$:

Solve $(S^TS - \lambda_1 I)\vec{v}_1 = \vec{0}$:

$$\begin{pmatrix} 1 - \frac{3+\sqrt{5}}{2} & 1 \\ 1 & 2 - \frac{3+\sqrt{5}}{2} \end{pmatrix} = \begin{pmatrix} \frac{-1-\sqrt{5}}{2} & 1 \\ 1 & \frac{1-\sqrt{5}}{2} \end{pmatrix}$$

From the first row: $\frac{-1-\sqrt{5}}{2} v_1 + v_2 = 0$

$$v_2 = \frac{1+\sqrt{5}}{2} v_1$$

Normalizing with $v_1 = 1$: $\vec{v}_1 = \begin{pmatrix} 1 \\ \frac{1+\sqrt{5}}{2} \end{pmatrix}$

After normalization: $\vec{v}_1 = \frac{1}{\sqrt{1+\left(\frac{1+\sqrt{5}}{2}\right)^2}} \begin{pmatrix} 1 \\ \frac{1+\sqrt{5}}{2} \end{pmatrix}$

For $\lambda_2 = \frac{3-\sqrt{5}}{2}$:

Similarly, solving $(S^TS - \lambda_2 I)\vec{v}_2 = \vec{0}$ gives:

$$v_2 = \frac{1-\sqrt{5}}{2} v_1$$

After normalization: $\vec{v}_2 = \frac{1}{\sqrt{1+\left(\frac{1-\sqrt{5}}{2}\right)^2}} \begin{pmatrix} 1 \\ \frac{1-\sqrt{5}}{2} \end{pmatrix}$

Left singular vectors are computed as $u_i = \frac{Sv_i}{\sigma_i}$:

$$u_1 = \frac{Sv_1}{\sigma_1}, \quad u_2 = \frac{Sv_2}{\sigma_2}$$

2.1.4 Part 4: SVD Decomposition

The Singular Value Decomposition is:

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

where:

- $U = [u_1 \; u_2]$ is a 2×2 orthogonal matrix of left singular vectors
- $\Sigma = \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix} = \begin{pmatrix} \sqrt{\frac{3+\sqrt{5}}{2}} & 0 \\ 0 & \sqrt{\frac{3-\sqrt{5}}{2}} \end{pmatrix}$
- $V = [v_1 \; v_2]$ is a 2×2 orthogonal matrix of right singular vectors

This decomposition always exists even when S is not diagonalizable, making SVD more general than eigendecomposition.

[ ]:

```python
# Define matrix S
S = np.array([[1, 1], [0, 1]])

print("Matrix S:")
print(S)
```

[ ]:

```python
# Part 1: Find eigenvalues and eigenvectors of S
eigenvalues_S, eigenvectors_S = np.linalg.eig(S)

print("Part 1: Eigenvalues and Eigenvectors of S")
print("=" * 50)
print(f"\nEigenvalues of S: {eigenvalues_S}")
print(f"\nEigenvectors of S:")
print(eigenvectors_S)

# Check rank of eigenvector matrix
rank_eig = np.linalg.matrix_rank(eigenvectors_S)
print(f"\nRank of eigenvector matrix: {rank_eig}")
print(f"Dimension of S: {S.shape[0]}")
```

```
if rank_eig < S.shape[0]:
    print("\n  S is NOT diagonalizable (rank < dimension)")
else:
    print("\n  S is diagonalizable")
```

```
[ ]: # Part 2: Compute S^T S and its eigenvalues, then singular values
S_T = S.T
S_T_S = S_T @ S

print("\nPart 2: S^T S and Singular Values")
print("=" * 50)
print("\nS^T (transpose of S):")
print(S_T)

print("\nS^T S:")
print(S_T_S)

# Find eigenvalues of S^T S
eigenvalues_STS, eigenvectors_STS = np.linalg.eig(S_T_S)

# Sort eigenvalues in descending order
idx = eigenvalues_STS.argsort()[::-1]
eigenvalues_STS = eigenvalues_STS[idx]
eigenvectors_STS = eigenvectors_STS[:, idx]

print(f"\nEigenvalues of S^T S:")
print(f"   1 = {eigenvalues_STS[0]:.6f}")
print(f"   2 = {eigenvalues_STS[1]:.6f}")

# Compute singular values
singular_values = np.sqrt(eigenvalues_STS)

print(f"\nSingular values:")
print(f"   1 = √ 1 = {singular_values[0]:.6f}")
print(f"   2 = √ 2 = {singular_values[1]:.6f}")
```

```
[ ]: # Part 3: Compute right and left singular vectors
print("\nPart 3: Right and Left Singular Vectors")
print("=" * 50)

# Right singular vectors V (eigenvectors of S^T S)
V = eigenvectors_STS

print("\nRight singular vectors V (columns are v1, v2):")
print(V)
print(f"\nv1 = {V[:, 0]}")
print(f"v2 = {V[:, 1]}")
```

```
# Left singular vectors U: u_i = S*v_i / _i
U = np.zeros((2, 2))
for i in range(2):
    U[:, i] = (S @ V[:, i]) / singular_values[i]

print("\nLeft singular vectors U (columns are u1, u2):")
print(U)
print(f"\nu1 = {U[:, 0]}")
print(f"u2 = {U[:, 1]}")

# Verify orthonormality
print("\nVerification of orthonormality:")
print(f"U^T U = \n{U.T @ U}")
print(f"V^T V = \n{V.T @ V}")
```

[ ]:

```
# Part 4: Construct and verify SVD
print("\nPart 4: Singular Value Decomposition (SVD)")
print("=" * 50)

# Construct diagonal matrix Σ
Sigma = np.diag(singular_values)

print("\nMatrix Σ (diagonal matrix of singular values):")
print(Sigma)

print("\nMatrix U (left singular vectors):")
print(U)

print("\nMatrix V (right singular vectors):")
print(V)

# Reconstruct S using SVD: S = U Σ V^T
S_reconstructed = U @ Sigma @ V.T

print("\n\nVerification: S = U Σ V^T")
print("\nOriginal matrix S:")
print(S)

print("\nReconstructed S from U Σ V^T:")
print(S_reconstructed)

print("\nDifference (should be close to zero):")
print(np.abs(S - S_reconstructed))

print(f"\nMaximum difference: {np.max(np.abs(S - S_reconstructed)):.2e}")
```

```python
if np.allclose(S, S_reconstructed):
    print("\n  SVD verification successful! S = U Σ V^T")
```

[ ]:

```python
# Bonus: Compare with NumPy's built-in SVD
print("\n" + "=" * 50)
print("Bonus: Comparison with NumPy's SVD function")
print("=" * 50)

U_numpy, Sigma_numpy, VT_numpy = np.linalg.svd(S)

print("\nNumPy's U:")
print(U_numpy)

print("\nNumPy's singular values:")
print(Sigma_numpy)

print("\nNumPy's V^T:")
print(VT_numpy)

print("\nNumPy's V:")
print(VT_numpy.T)

print("\n\nComparison:")
print(f"Our  1 = {singular_values[0]:.6f}, NumPy's  1 = {Sigma_numpy[0]:.6f}")
print(f"Our  2 = {singular_values[1]:.6f}, NumPy's  2 = {Sigma_numpy[1]:.6f}")
print("\n  Results match (note: signs of vectors may differ, which is normal)")
```

---

3 Exercise 3- PCA on a tiny 2D dataset

Given the four points (rows are samples):

$$X = \begin{pmatrix} 2 & 0 \\ 0 & 2 \\ 3 & 1 \\ 1 & 3 \end{pmatrix} \in \mathbb{R}^{4\times 2}$$

1. Center the data: compute the column mean μ and $Z = X - \mathbf{1}\,\mu^T$.
2. Compute the sample covariance matrix $C = \frac{1}{n-1} Z^T Z$.
3. Compute the eigenvalues and eigenvectors of C (principal components).
4. Order components by decreasing eigenvalues, and give the explained variance ratio for $k = 1$ and $k = 2$.
5. Project the centered data on the first principal component $T_1 = Zv_1$.

3.1 Solution

1. Centering the data

- Column mean:

$$\mu = \frac{1}{4}\sum_{i=1}^{4} X_{i,:} = \begin{pmatrix} \frac{2+0+3+1}{4} \\ \frac{0+2+1+3}{4} \end{pmatrix} = \begin{pmatrix} 1.5 \\ 1.5 \end{pmatrix}$$

- Centered matrix $Z = X - \mathbf{1}\,\mu^T$:

$$Z = \begin{pmatrix} 2-1.5 & 0-1.5 \\ 0-1.5 & 2-1.5 \\ 3-1.5 & 1-1.5 \\ 1-1.5 & 3-1.5 \end{pmatrix} = \begin{pmatrix} 0.5 & -1.5 \\ -1.5 & 0.5 \\ 1.5 & -0.5 \\ -0.5 & 1.5 \end{pmatrix}$$

2. Sample covariance matrix

- Compute $Z^T Z$:

$$Z^T Z = \begin{pmatrix} 5 & -3 \\ -3 & 5 \end{pmatrix}$$

- With $n = 4$, the unbiased sample covariance is:

$$C = \frac{1}{n-1} Z^T Z = \frac{1}{3}\begin{pmatrix} 5 & -3 \\ -3 & 5 \end{pmatrix} = \begin{pmatrix} \frac{5}{3} & -1 \\ -1 & \frac{5}{3} \end{pmatrix}$$

3. Eigen-decomposition of C

- For a matrix of the form $\begin{pmatrix} a & b \\ b & a \end{pmatrix}$, eigenvalues are $a \pm b$ with eigenvectors proportional to $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ and $\begin{pmatrix} 1 \\ -1 \end{pmatrix}$.
- Here $a = \frac{5}{3}$, $b = -1$:

$$\lambda_1 = a - b = \tfrac{5}{3} - (-1) = \tfrac{8}{3} \approx 2.6667, \quad v_1 \propto \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

$$\lambda_2 = a + b = \tfrac{5}{3} + (-1) = \tfrac{2}{3} \approx 0.6667, \quad v_2 \propto \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

- Unit-norm eigenvectors:

$$v_1 = \tfrac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -1 \end{pmatrix}, \quad v_2 = \tfrac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

4. Explained variance ratio

- Total variance $= \lambda_1 + \lambda_2 = \frac{8}{3} + \frac{2}{3} = \frac{10}{3}$.
- For $k = 1$ (first PC only):

$$\mathrm{EVR}_{k=1} = \frac{\lambda_1}{\lambda_1 + \lambda_2} = \frac{\frac{8}{3}}{\frac{10}{3}} = \frac{8}{10} = 0.8$$

- For $k = 2$ (both PCs): EVR $= 1.0$.

5. Projection onto the first principal component

- Using $v_1 = \frac{1}{\sqrt{2}}(1, -1)^T$, the scores $T_1 = Zv_1$ are:

$$T_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 0.5 - (-1.5) \\ -1.5 - 0.5 \\ 1.5 - (-0.5) \\ -0.5 - 1.5 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 2 \\ -2 \\ 2 \\ -2 \end{pmatrix} = \begin{pmatrix} \sqrt{2} \\ -\sqrt{2} \\ \sqrt{2} \\ -\sqrt{2} \end{pmatrix}$$

Therefore, the principal directions are along $(1, -1)$ and $(1, 1)$, with the first PC explaining 80% of the variance, and the projections alternate $\pm\sqrt{2}$ across the four centered points.

[ ]:

```python
# Exercise 3 - PCA (code verification)
import numpy as np

# Data matrix X (4x2)
X = np.array([[2, 0], [0, 2], [3, 1], [1, 3]], dtype=float)

print("X:\n", X)

# 1) Center the data
mu = X.mean(axis=0)
Z = X - mu
print("\nMean mu:", mu)
print("\nCentered Z:\n", Z)

# 2) Sample covariance matrix (unbiased)
C = (Z.T @ Z) / (len(X) - 1)
print("\nCovariance C = (1/(n-1)) Z^T Z:\n", C)

# 3) Eigen-decomposition (symmetric)
vals, vecs = np.linalg.eigh(C)  # eigh for symmetric matrices
# Sort in descending order
idx = np.argsort(vals)[::-1]
vals = vals[idx]
vecs = vecs[:, idx]
print("\nEigenvalues (desc):", vals)
print("Eigenvectors (columns, aligned with eigenvalues):\n", vecs)

# 4) Explained variance ratios
explained = vals / vals.sum()
print("\nExplained variance ratios:", explained)
print(f"EVR k=1: {explained[0]:.4f}, EVR k=2: {explained.sum():.4f}")

# 5) Projection on first principal component
v1 = vecs[:, 0]
T1 = Z @ v1
print("\nFirst principal direction v1:", v1)
print("Projection T1 = Z v1:\n", T1)
```

[ ]:

```
# Bonus: Compare with scikit-learn PCA (optional)
from sklearn.decomposition import PCA

pca = PCA(n_components=2, svd_solver="full")
Z_sklearn = X - X.mean(axis=0)
pca.fit(Z_sklearn)

print("\n[sklearn] Components (rows):\n", pca.components_)
print("[sklearn] Explained variance:", pca.explained_variance_)
print("[sklearn] Explained variance ratio:", pca.explained_variance_ratio_)

# Ensure direction consistency (signs may differ)
print("\nDirection alignment check (abs dot with our v1, v2):")
print("|v1·comp1|:", abs(vecs[:, 0] @ pca.components_[0]))
print("|v2·comp2|:", abs(vecs[:, 1] @ pca.components_[1]))
```