Understanding POD
Proper Orthogonal Decomposition

Ning An (安宁)
School of Aeronautics and Astronautics,
Sichuan University
2025.02.03

Proper Orthogonal Decomposition

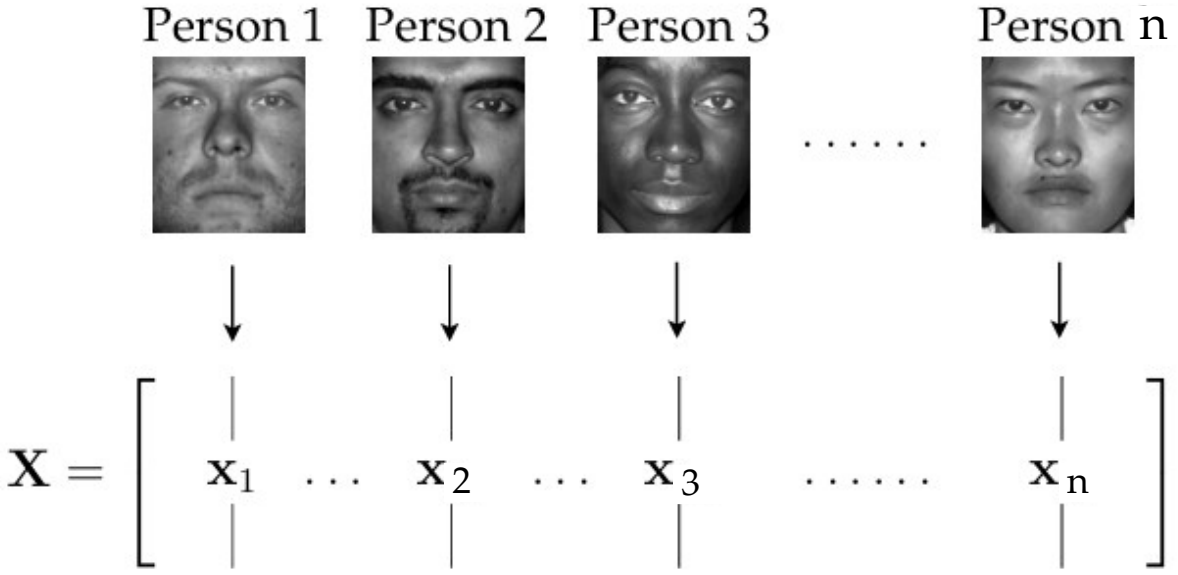
SICHUAN UNIVERSITY

- **Principal Component Analysis**
- **Purpose**: Reduces the dimensionality of complex systems by extracting dominant modes or features.
- **Key Concept**: Decomposes data into orthogonal basis functions that capture the most significant patterns.
- **Applications**: Used in fluid dynamics, structural mechanics, and data-driven modeling to simplify complex systems and reduce computational cost.
- **Benefit**: Preserves essential physics while enabling efficient analysis of large-scale problems.

Face Recognition Example

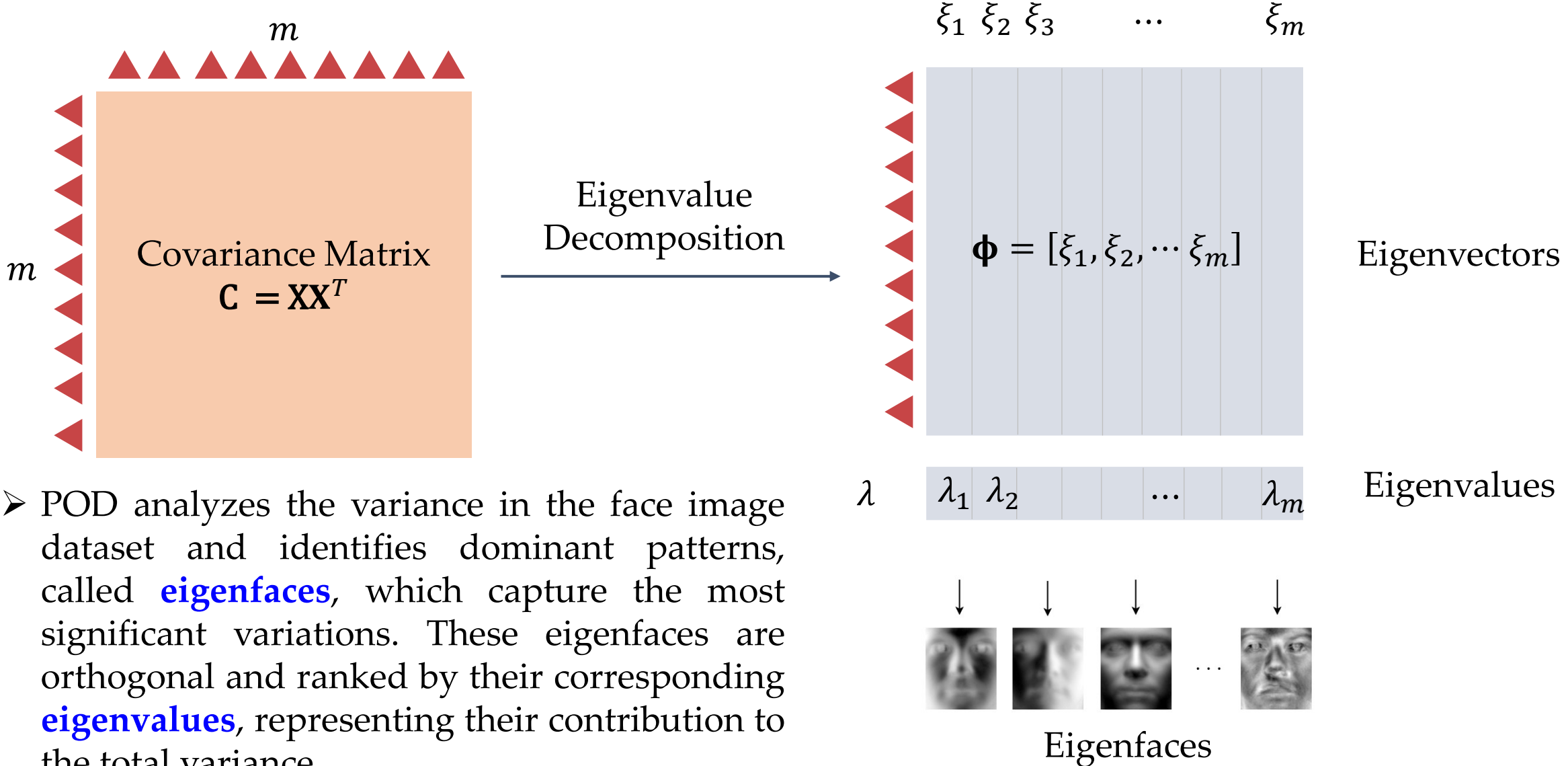
➢ Face Image Representation

- Each face image is treated as a m-dimensional vector (flattening the 2D image into a 1D array of pixel values).
- A dataset of n face images forms **a data matrix of** $\mathbf{X}_{m \times n}$ where each column represents an individual person.



Face Recognition Example

- **POD Process**



- POD analyzes the variance in the face image dataset and identifies dominant patterns, called **eigenfaces**, which capture the most significant variations. These eigenfaces are orthogonal and ranked by their corresponding **eigenvalues**, representing their contribution to the total variance.

Face Recognition Example

➢ Dimensionality Reduction

The full data matrix $\mathbf{X}$ is $m \times n$, where m is the number of features and n is the number of data points (or people).

$$\mathbf{X} = \boldsymbol{\Phi}\boldsymbol{\Phi}^{T}\mathbf{X} \approx \bar{\boldsymbol{\Phi}}\bar{\boldsymbol{\Phi}}^{T}\mathbf{X} = \bar{\boldsymbol{\Phi}}\hat{\mathbf{A}}$$

where $\boldsymbol{\Phi}_{m\times m} = [\xi_1, \xi_2, \cdots \xi_m]$ is the **POD basis** consisting of eigenvectors (eigenfaces). By truncating the POD basis to the first l modes, we get the truncated POD basis $\bar{\boldsymbol{\Phi}}_{m\times l} = [\xi_1, \xi_2, \cdots \xi_l]$.

The reduced data matrix $\hat{\mathbf{A}}_{l\times \boldsymbol{n}}$ is obtained by projecting $\mathbf{X}$ onto the truncated POD basis:

$$\hat{\mathbf{A}} = \bar{\boldsymbol{\Phi}}^{T}\mathbf{X}$$

The **reconstructed data matrix** $\hat{\mathbf{X}}_{m\times n}$ is obtained by multiplying the truncated POD basis with the reduced data matrix:

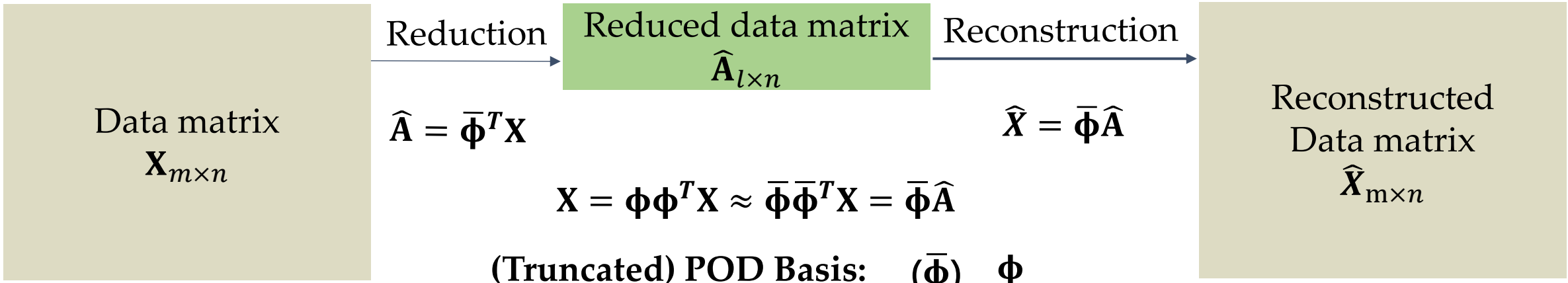
$$\hat{\mathbf{X}} = \bar{\boldsymbol{\Phi}}\hat{\mathbf{A}}$$

The **truncated error** $e(l)$ quantifies the loss of information when using only the first l eigenfaces. It is given by:

$$e(l) = 1 - \frac{\Sigma_{j=1}^{l}\lambda_j}{\Sigma_{j=1}^{m}\lambda_j} < 0.01$$

Face Recognition Example

➢ Dimensionality Reduction

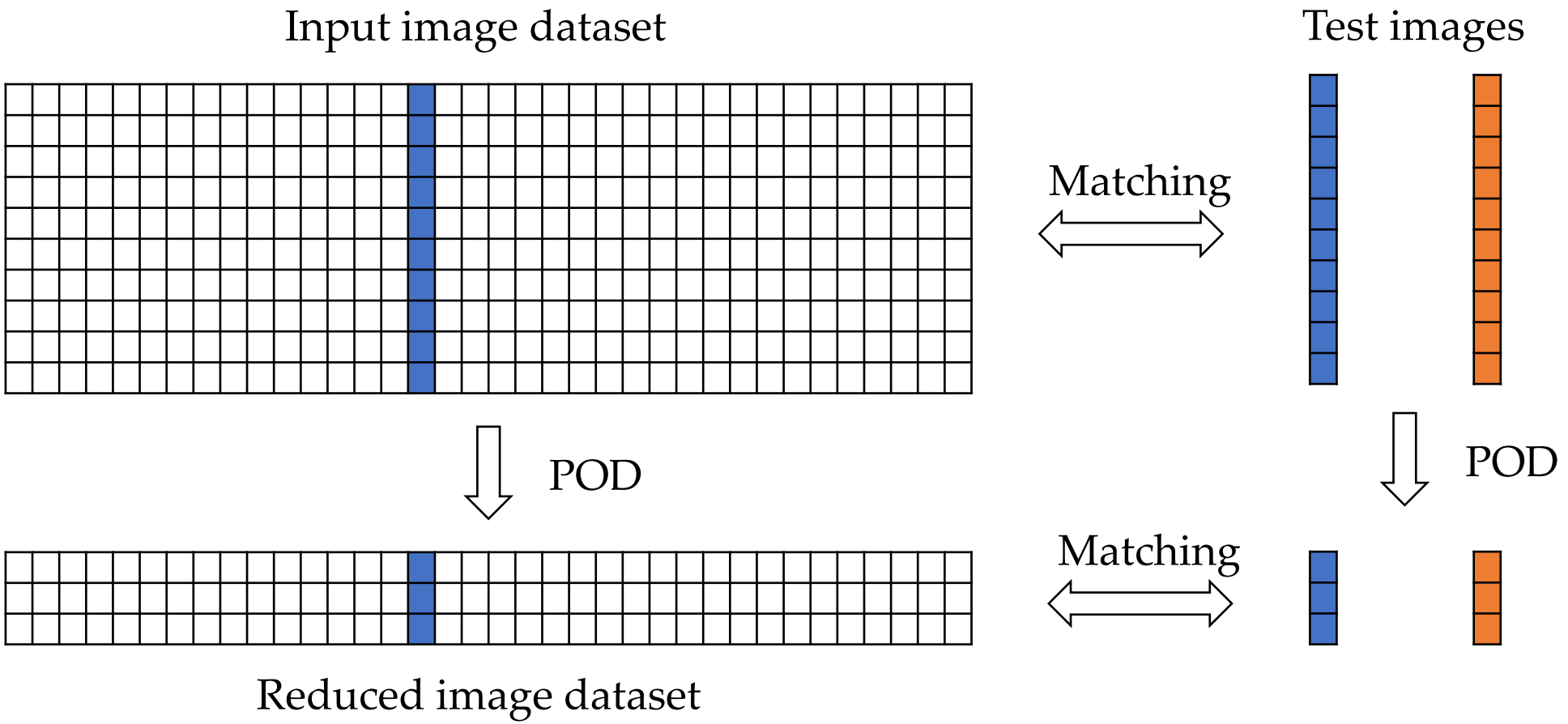


- By applying POD, the dimensionality of an individual's feature vector is reduced from m (original pixel values) to l (the number of retained eigenfaces).
- The face is no longer represented by pixel values but as a linear combination of the first l eigenfaces, which capture the most significant variations in the dataset.
- This reduced representation retains the key features of the face while discarding less important details, simplifying both the computational complexity and the face recognition process.

Face Recognition Example

- **Face Recognition**

The test image is converted into a small-dimensional coefficient vector by projecting it onto a set of orthogonal basis vectors derived from the dataset. This reduced representation, is then compared to the stored reduced coefficient vectors in the dataset for recognition.



POD in fluid dynamics

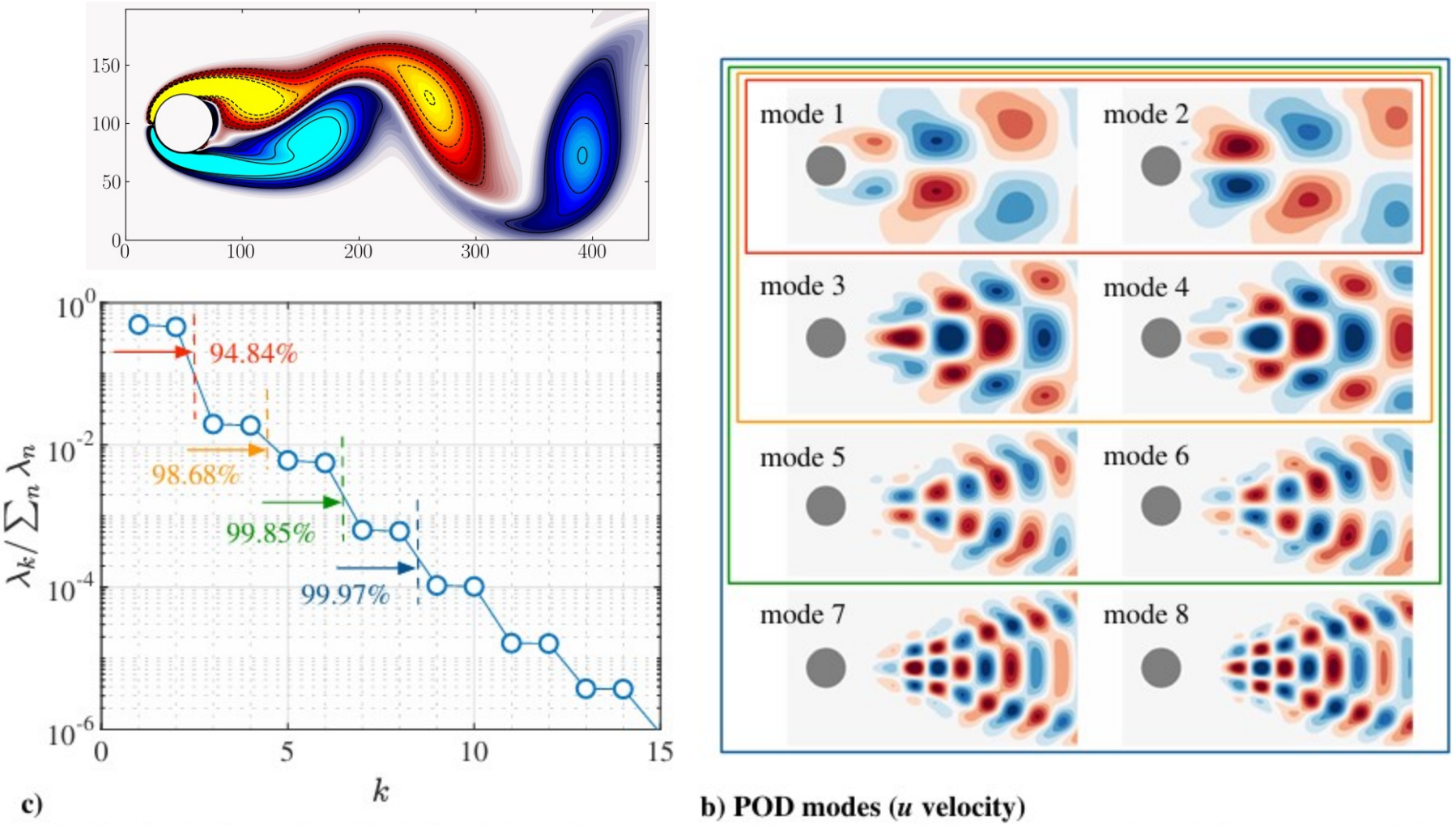


Fig. 2 POD analysis of cylinder flow: a) original flowfield under study (vorticity shown), b) first eight dominant POD modes, and c) amount of KE of unsteadiness captured by the POD modes.

AIAA journal, 2020, 58(3): 998-1022.

POD methods

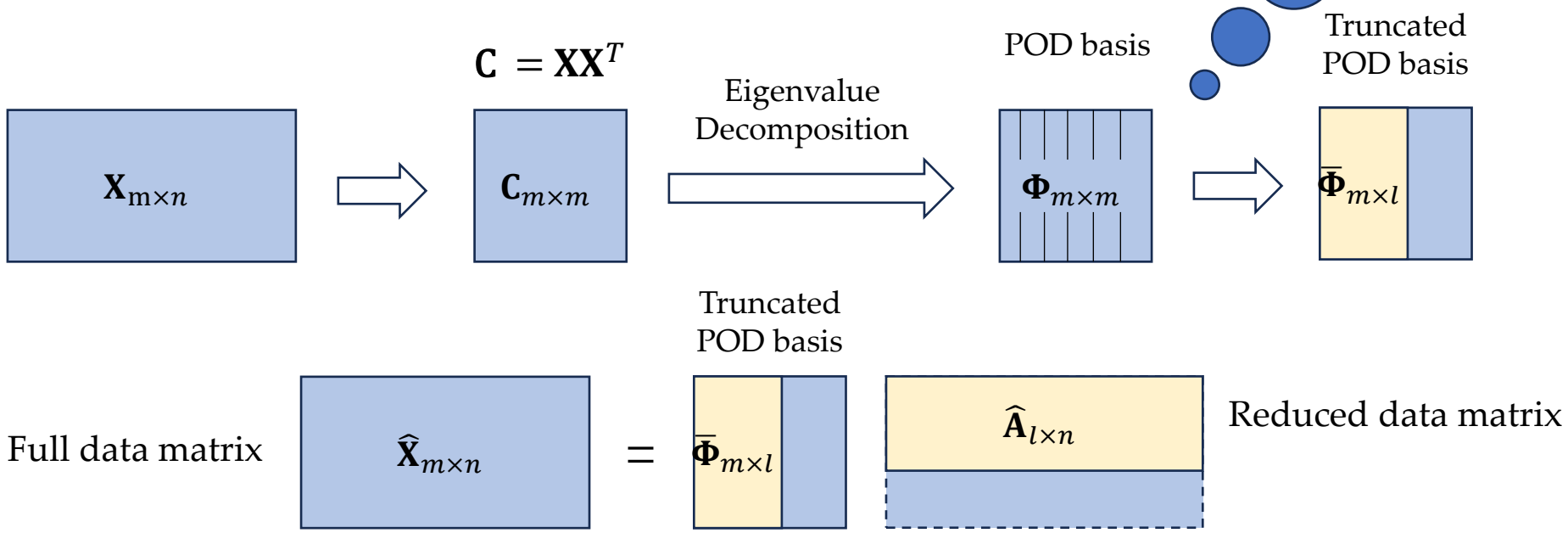
➢ Eigenvalue decomposition of covariance matrix

- **Process**

1. Compute the covariance matrix of the data.
2. Calculate eigenvalues and eigenvectors of the covariance matrix.
3. Sort eigenvectors by eigenvalues in descending order.
4. Project the data onto the selected eigenvectors to reduce dimensionality.

The rank of the covariance matrix **C** is limited to **min(m, n)**, and the number of non-zero eigenvalues is also constrained to **min(m, n)**. The eigenvectors corresponding to these non-zero eigenvalues define the principal components.

Think about m>n



POD methods

➢ Singular Value Decomposition (SVD)

Given a matrix **X**, the **Singular Value Decomposition** (SVD) is a factorization of the matrix into three components:

$$\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$$

Where:

- **X** is an $m \times n$ matrix (in your case, with m features and n cases/observations),
- **U** is an $m \times m$ orthogonal matrix whose columns are the **left singular vectors** of **X**,
- **Σ** is an m x n diagonal matrix (with non-negative real numbers on the diagonal), containing the **singular values** of **X**
- $\mathbf{V}^T$ is an n x n orthogonal matrix whose rows are the **right singular vectors** of **X.**

POD methods

➢ Understanding the Components

1. Left Singular Vectors (U):

- The columns of $\mathbf{U}$ are the eigenvectors of $\mathbf{X}\mathbf{X}^T$, and the columns of $\mathbf{U}$ can be interpreted as the POD basis functions that are orthogonal to each other and capture the most significant modes or directions of variability in the data.

2. Right Singular Vectors (V):

- The rows of $\mathbf{V}^T$ (i.e., the columns of $\mathbf{V}$) are the eigenvectors of $\mathbf{X}^T\mathbf{X}$.
- The $\mathbf{X}\mathbf{X}^T$ and $\mathbf{X}^T\mathbf{X}$ have the same non-zero eigenvalues and they are related to the singular values of $\mathbf{X}$.

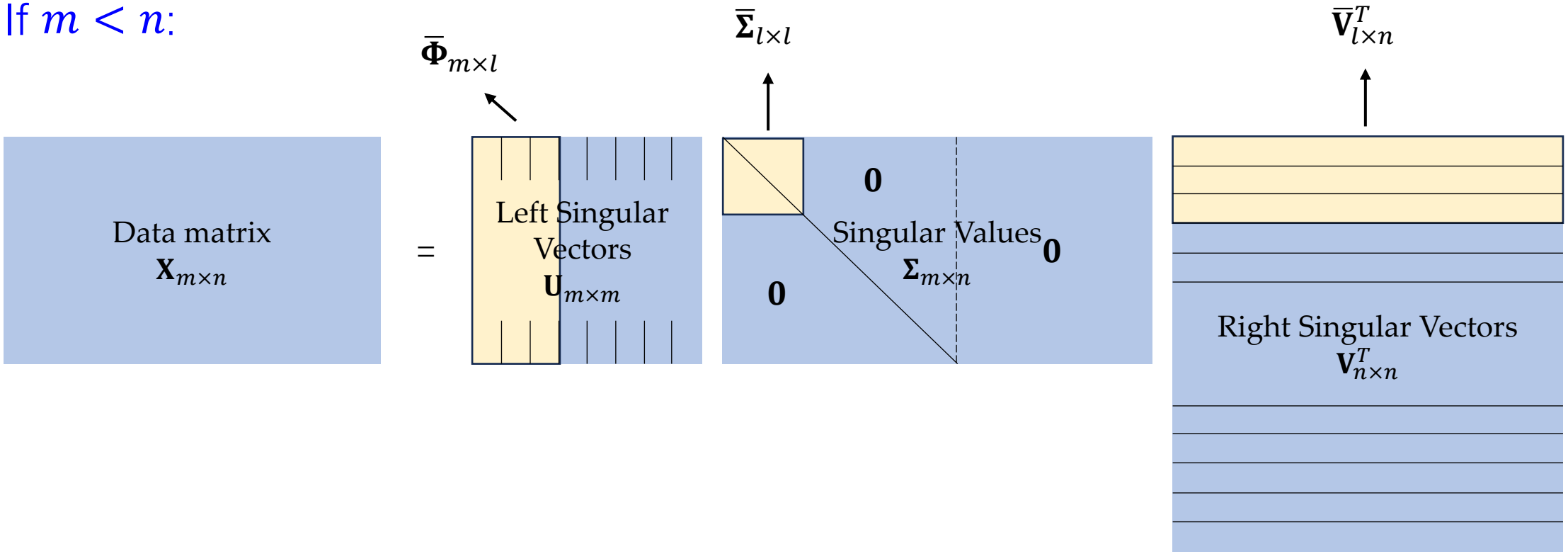
3. Singular Values (Σ):

- The matrix $\mathbf{\Sigma}$ is diagonal with the singular values $\sigma_1, \sigma_2, \ldots, \sigma_r$ on its diagonal, and the number of singular values r is equal to $\min(m, n)$.
- The singular values σ_i are the square roots of the non-zero eigenvalues λ_i of $\mathbf{X}\mathbf{X}^T$ or $\mathbf{X}^T\mathbf{X}$.

POD methods

➢ Singular Value Decomposition (SVD)

If $m < n$:

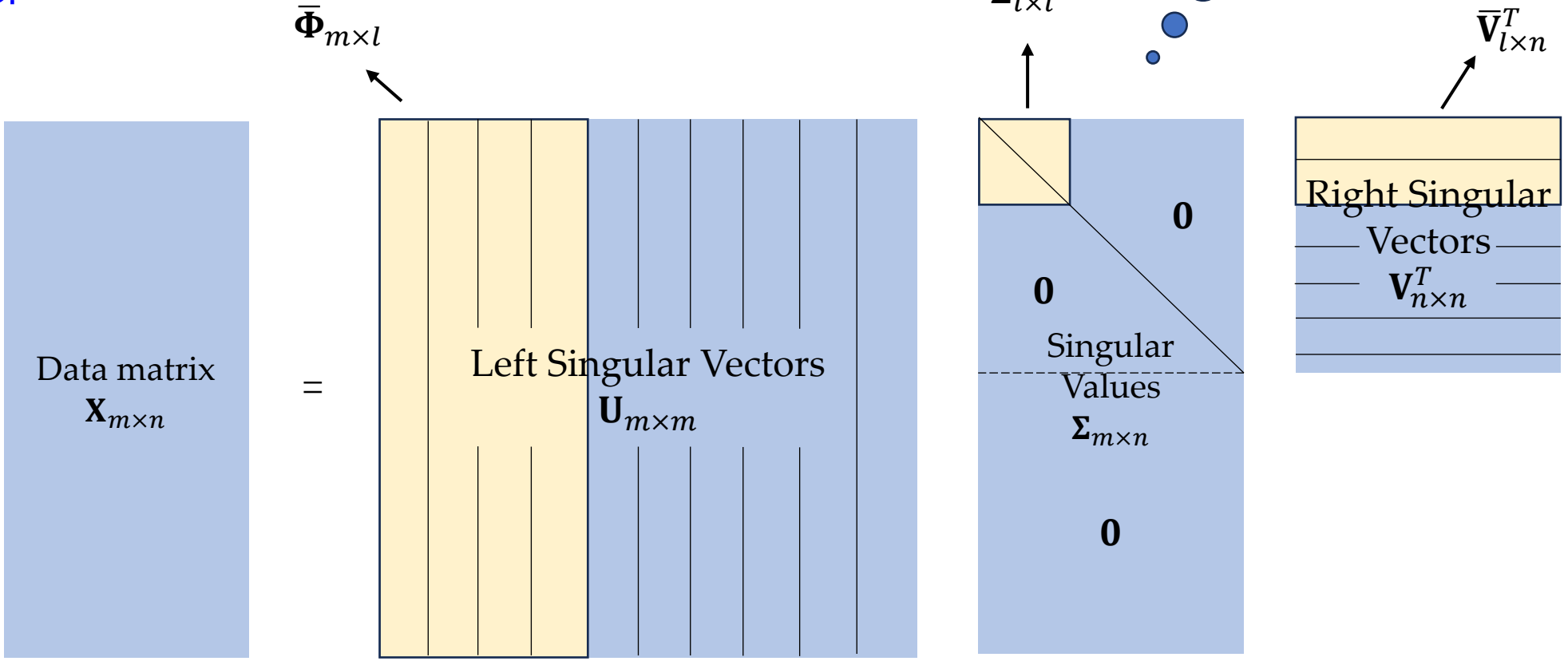


POD methods

➢ **Singular Value Decomposition (SVD)**

$\widehat{\mathbf{A}} = \bar{\mathbf{\Sigma}}_{l\times l}\bar{\mathbf{V}}^T_{l\times n}$

If $m > n$:



POD methods

- **SVD vs PCA**

1) $\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T \qquad \mathbf{X}\mathbf{X}^T\mathbf{U} = \mathbf{U}\boldsymbol{\Lambda}$

$$\mathbf{X}\mathbf{X}^T\mathbf{U} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T\mathbf{V}\boldsymbol{\Sigma}^T\mathbf{U}^T\mathbf{U} = \mathbf{U}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^T = \mathbf{U}\boldsymbol{\Lambda} \qquad \therefore \quad \boldsymbol{\Sigma}\boldsymbol{\Sigma}^T = \boldsymbol{\Lambda}$$

2) $\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T \qquad \mathbf{X}^T\mathbf{X}\mathbf{V} = \mathbf{V}\boldsymbol{\Lambda}$

$$\mathbf{X}^T\mathbf{X}\mathbf{V} = \mathbf{V}\boldsymbol{\Sigma}^T\mathbf{U}^T\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T\mathbf{V} = \mathbf{V}\boldsymbol{\Sigma}^T\boldsymbol{\Sigma} = \mathbf{V}\boldsymbol{\Lambda} \qquad \therefore \quad \boldsymbol{\Sigma}^T\boldsymbol{\Sigma} = \boldsymbol{\Lambda}$$

Example

```python
import numpy as np
from scipy.linalg import svd

# Define input data matrix X
X = np.array([
    [1, 5, 3, 3],
    [1, 4, 4, 3],
    [1, 5, 5, 4]
], dtype=float)

# Display the original data matrix
print("Original Data Matrix:\n", X)
```

```
Original Data Matrix:
 [[1. 5. 3. 3.]
 [1. 4. 4. 3.]
 [1. 5. 5. 4.]]
```

https://github.com/Dr-Ning-An/FEM_Course/blob/main/POD_Example/SVD_POD_reduction.ipynb

Example

```
# Perform Singular Value Decomposition (SVD) on the data matrix
U, S, VT = svd(X)

# Calculate the POD basis (U matrix from SVD)
pod_basis = U
print("\nPOD Basis / Left Singular Vectors:\n", pod_basis)

print("\nSingular Values:\n",S)

print("\nTranspose of Right Singular Vectors:\n",VT)
```

```
POD Basis / Left Singular Vectors:
 [[ 0.53248416  0.84624208  0.01830206]
 [ 0.52545445 -0.31352754 -0.7909476 ]
 [ 0.66359494 -0.43078397  0.61161011]]

Singular Values:
 [12.30834838  1.20802645  0.21267859]

Transpose of Right Singular Vectors:
 [[ 0.13986714  0.65664483  0.57012076  0.47351565]
 [ 0.08437777  0.68142582 -0.71960655 -0.10346813]
 [-0.75717737 -0.06690608 -0.23901616  0.60421625]
 [ 0.63245553 -0.31622777 -0.31622777  0.63245553]]
```

Example

```
# Calculate energy content of each singular value
energy_content = (S**2) / np.sum(S**2)

# Calculate the cumulative energy content
cumulative_energy = np.cumsum(energy_content)

# Determine the number of POD basis vectors needed to achieve the desired error
desired_error = 0.001
num_basis_vectors = np.sum(cumulative_energy < (1 - desired_error)) + 1

# Truncate the POD basis
truncated_pod_basis = U[:, :num_basis_vectors]
print(f"\nTruncated POD Basis (using {num_basis_vectors} vectors):\n", truncated_pod_basis)
```

```
Truncated POD Basis (using 2 vectors):
 [[ 0.53248416  0.84624208]
 [ 0.52545445 -0.31352754]
 [ 0.66359494 -0.43078397]]
```

Example

```
# Calculate the reduced matrix using the truncated POD basis
reduced_matrix = truncated_pod_basis.T @ X
print("\nReduced Matrix:\n", reduced_matrix)

# Reconstruct the original matrix from the reduced matrix
reconstructed_matrix = truncated_pod_basis @ reduced_matrix
print("\nReconstructed Matrix:\n", reconstructed_matrix)

# Calculate the reconstruction error
error = np.linalg.norm(X - reconstructed_matrix)
print(f"\nReconstruction Error: {error}")
```

```
Reduced Matrix:
 [[ 1.72153355  8.08221328  7.01724497  5.82819558]
 [ 0.10193057  0.82318041 -0.86930375 -0.12499224]]

Reconstructed Matrix:
 [[1.00294728 5.00026043 3.00093036 2.99764811]
 [0.87262942 3.98874522 3.95979327 3.10163982]
 [1.09849089 5.0087029  5.03109036 3.92140574]]

Reconstruction Error: 0.2126785923857775
```

Acknowledgment and References

Acknowledgment:

- *This document was created by an undergraduate student Yichen Pu and subsequently modified by Dr. Ning An.*

References:

- Greenacre M, Groenen P J F, Hastie T, et al. Principal component analysis[J]. Nature Reviews Methods Primers, 2022, 2(1): 100.
- Buljak V. Inverse analyses with model reduction: proper orthogonal decomposition in structural mechanics[M]. Springer Science & Business Media, 2011.
- Brunton S L, Kutz J N. Data-driven science and engineering: Machine learning, dynamical systems, and control[M]. Cambridge University Press, 2022.