ICA algorithms and rotations methods

Week 13 - Advanced Topic 1

Introduction to ICA and Rotation

The problem, in general terms

Given a linear combination of N distinct signals, i.e., x_j as

$$\vdots$$
$$\mathbf{x_j} = a_{j1}\mathbf{s_1} + a_{j2}\mathbf{s_2} + \ldots + a_{jN}\mathbf{s_N}$$
$$\vdots$$

determine the original N signals $\mathbf{s_1}$ to $\mathbf{s_N}$.

In matrix form: $\mathbf{x} = \mathbf{A}\mathbf{s}$

Trivial solution? :) $\mathbf{s} = \mathbf{A}^{-1}\mathbf{x}$, assuming the sensors to be such that $\mathbf{A}$ is nonsingular

How to find both source signals **s** and mixing matrix **A**?

Introduction to ICA and Rotation

Goal

To find a set of statistically independent components from a set of observed data.

How ICA works

Input data: A mixture of signals, **x**

Model assumption: The data is a linear combination of independent source signals, **x** = **As**

Output data: The statistically independent source signals, **s**

Introduction to ICA and Rotation

Rotation methods are used to enhance or improve the separation of independent components.

- **Rotation**: A mathematical operation that reorients the independent components in a new coordinate system.
- **Goal**: To find a rotation that maximizes statistical independence between components, making them easier to interpret.

ICA algorithms

JADE (Joint Approximation Diagonalization of Eigen-matrices)

Separates components using higher-order statistics and joint diagonalization

Strengths:

Robust for identifying independent sources & handles higher-order statistics explicitly

Limitations:

Computationally expensive for large data sets & sensitive to noise and outliers

Workflow Outline:

1. Center and whiten data
2. Compute fourth-order statistics, aka *cumulants*
3. Iteratively perform joint diagonalization to extract independent components until convergence

Extended ICA

Generalizes ICA to handle super- and sub-Gaussian sources (sparse and uniform)

Strengths:

Handles diverse source distributions & adaptive selection of contrast function

Limitations:

More computationally expensive than standard ICA & sensitive to choice of adaption criteria

Workflow Outline:

1. Center and whiten data
2. Adaptively select contrast functions based on source type
3. Iteratively optimize weights to separate independent components until convergence

Fast ICA (Fast Independent Component Analysis)

Separate components by maximizing non-Gaussianity

Strengths:

Fast and efficient & handles non-Gaussian signals well

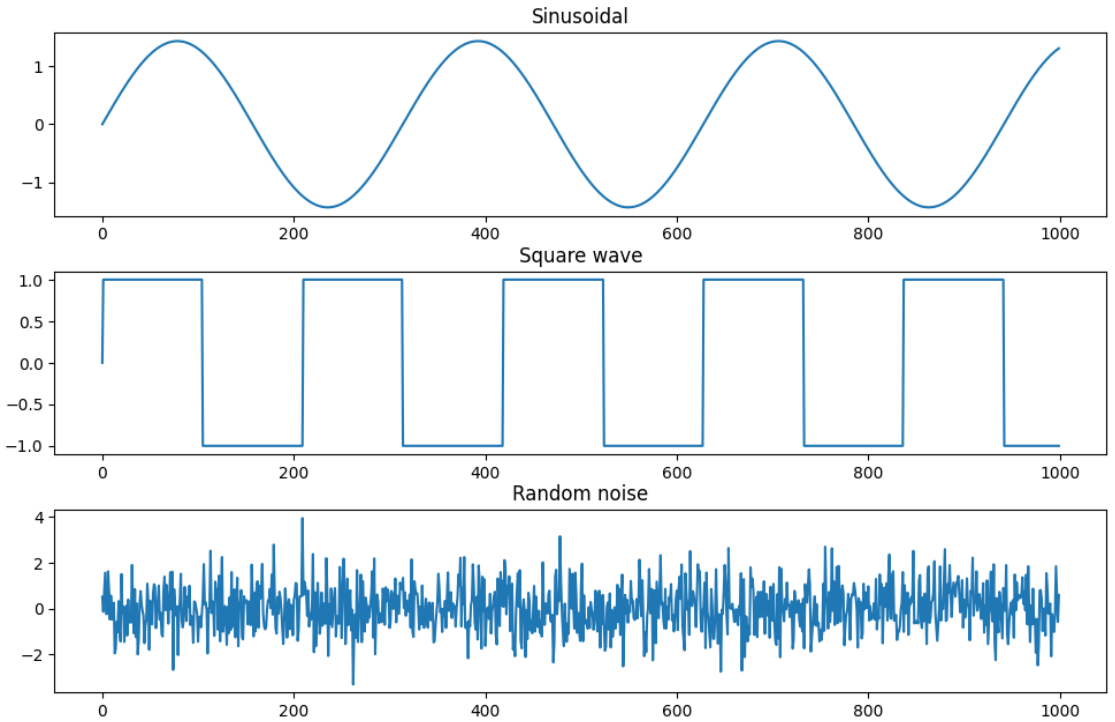
Limitations:

Sensitive to initial conditions & assumes statistical independence and non-Gaussian

Workflow Outline:

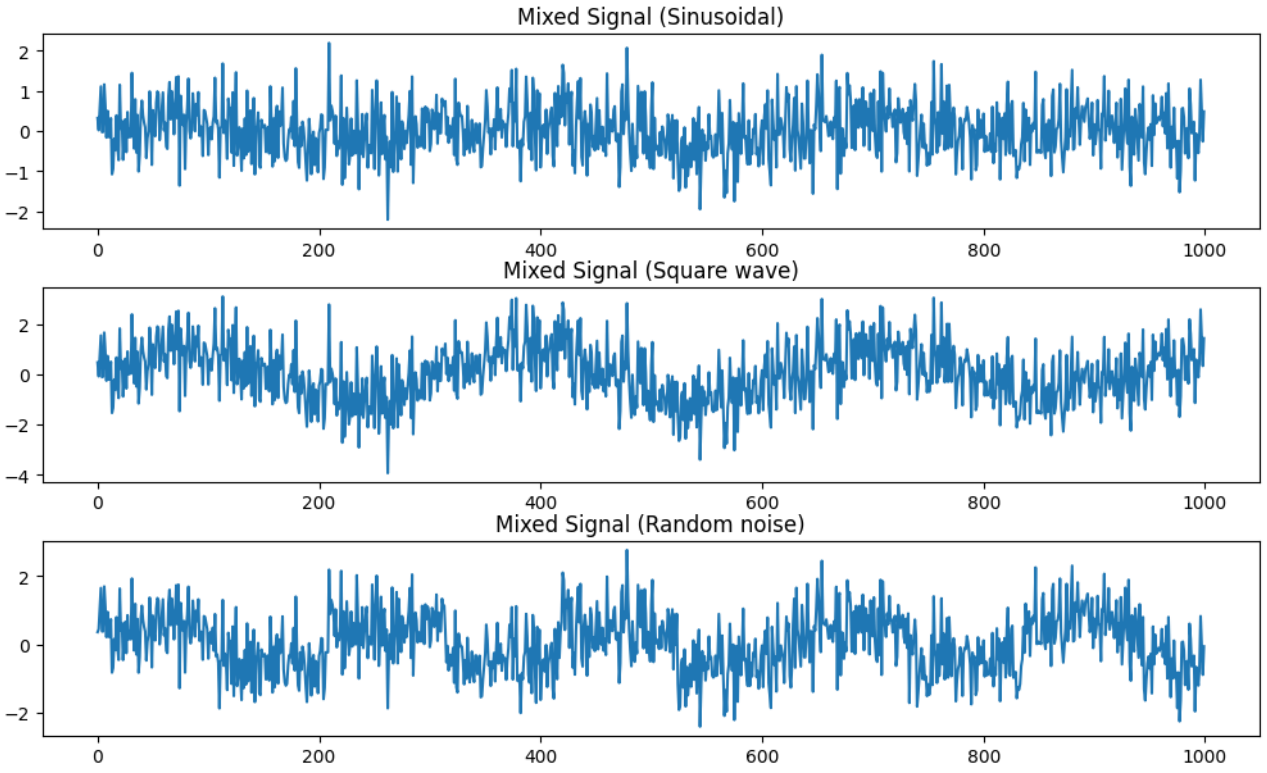
1. Center and white data, initialize a set of random weights
2. Iteratively update (mutually orthogonal) weights until they stabilize

FastICA



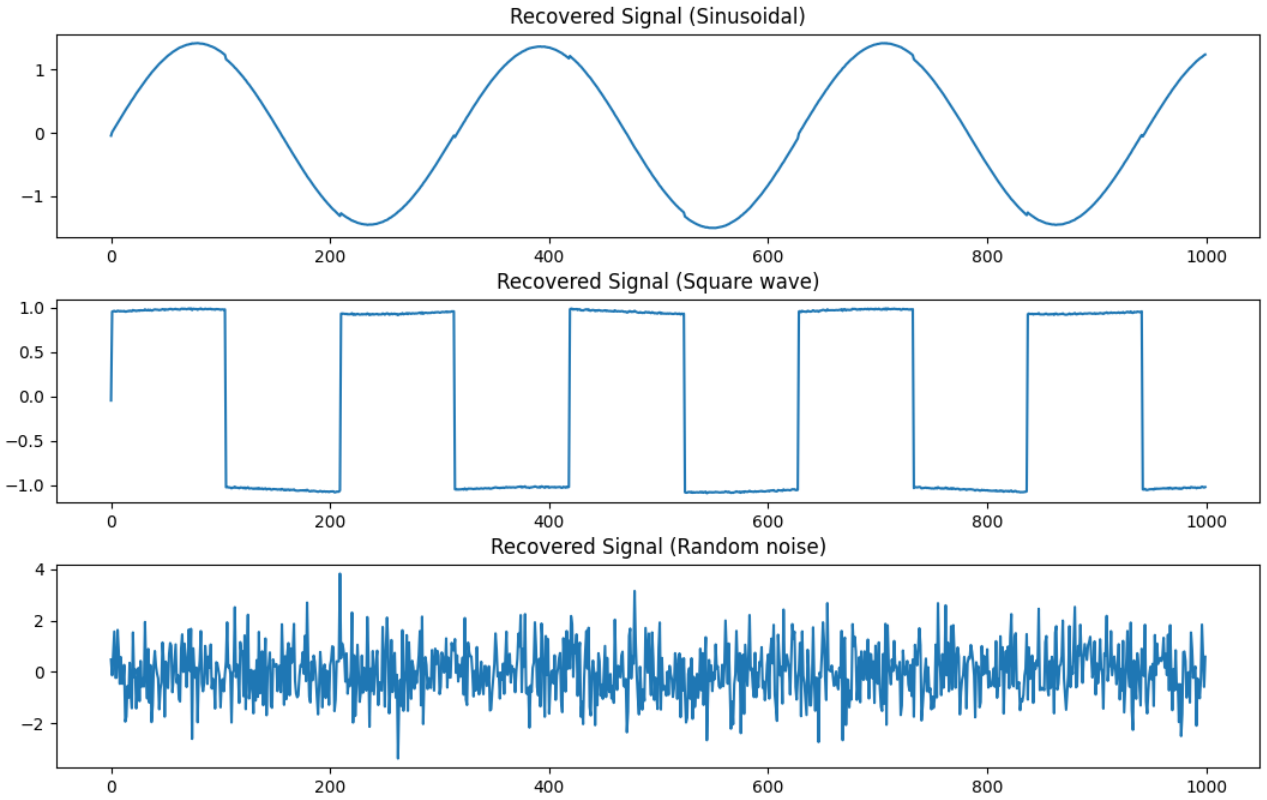

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import FastICA
from scipy.stats import kurtosis

# Generate synthetic data: 3 mixed signals
np.random.seed(42)
time = np.linspace(0, 10, 1000)
s1 = np.sin(2 * time) # Signal 1: Sinusoidal
s2 = np.sign(np.sin(3 * time)) # Signal 2: Square wave
s3 = np.random.normal(size=len(time)) # Signal 3: Random noise
S = np.c_[s1, s2, s3]
S /= S.std(axis=0) # Standardize data
```

FastICA




```
# Mix the signals using a random mixing matrix
A = np.random.rand(3, 3) # Mixing matrix
X = np.dot(S, A.T) # Mixed signals
```

FastICA




```
# Perform Independent Component Analysis (ICA)
ica = FastICA(n_components=3, random_state=42)
S_ica = ica.fit_transform(X) # Recovered signals
A_ica = ica.mixing_ # Estimated mixing matrix
```

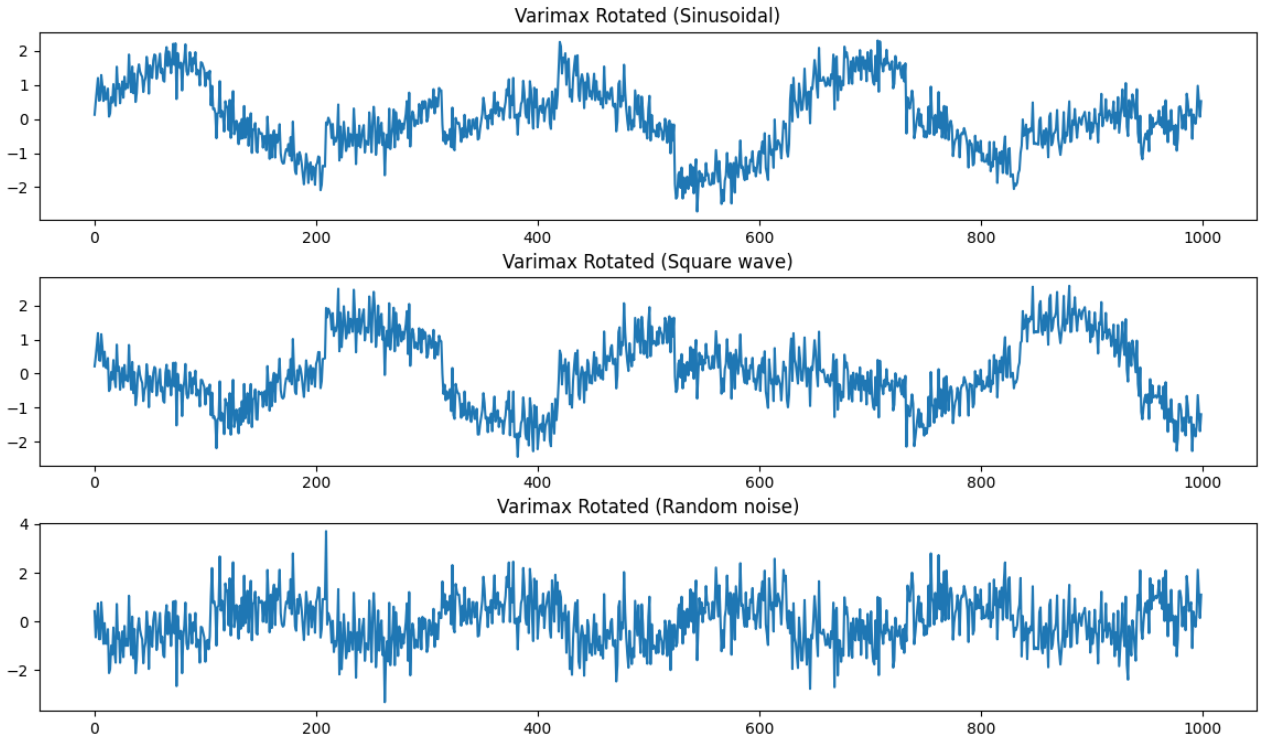
Rotation methods

Rotation Methods - Varimax

The **Varimax** rotation is mathematical algorithm that maximizes high- and low-value factor loadings and minimizes mid-value factor loadings.

- **Normalization** (factors matrix).
- **Rotation matrix**: Apply rotation matrix in normalized data.

Rotation Method - Varimax




```
def varimax(Phi, gamma=1.0, q=20, tol=1e-6):
    p, k = Phi.shape
    R = np.eye(k)
    d = 0
    for i in range(q):
        d_old = d
        Lambda = np.dot(Phi, R)
        u, s, vh = np.linalg.svd(np.dot(Phi.T, Lambda ** 3 -
(gamma / p) * np.dot(Lambda,
np.diag(np.sum(Lambda ** 2, axis=0)))))
        R = np.dot(u, vh)
        d = np.sum(s)
    if d_old != 0 and (d - d_old) < tol:
        break
return np.dot(Phi, R)
```

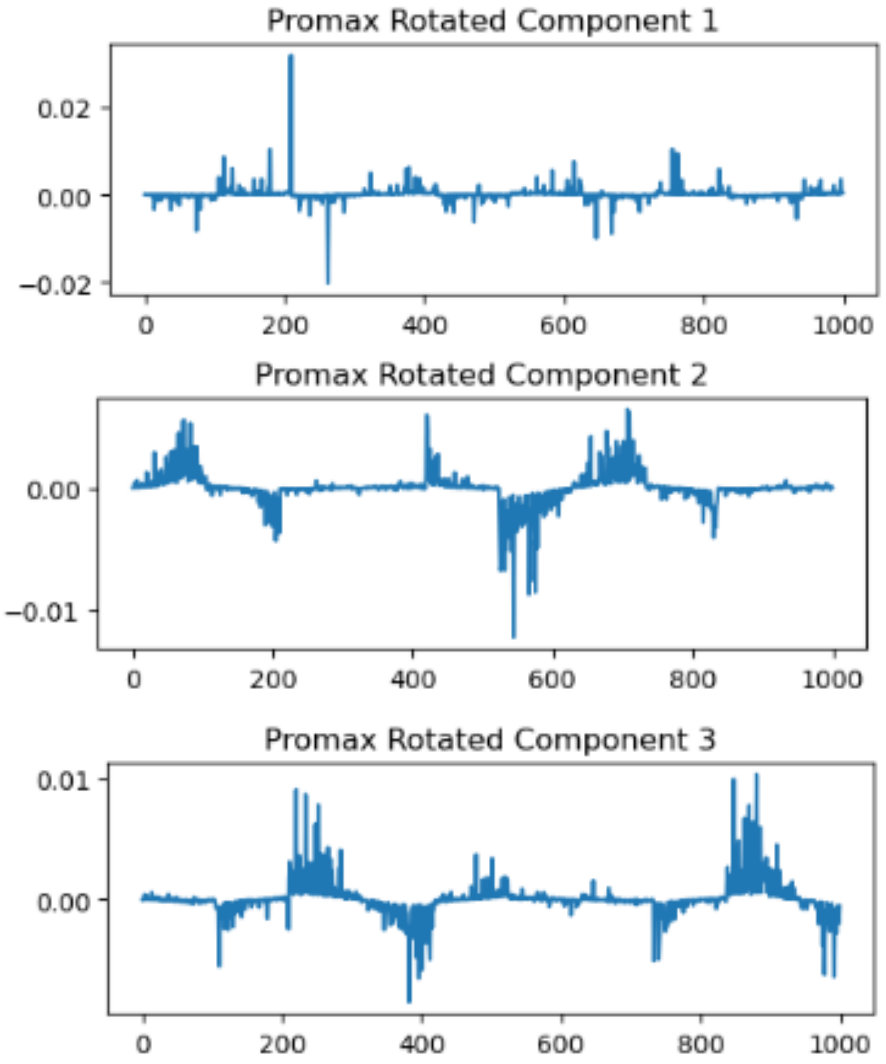
Rotation Methods - Promax

The **Promax** rotation is an oblique rotation method, which allows factors to correlate. It is often used after an orthogonal rotation like Varimax. The implementation involves two steps:

- **Perform Varimax rotation** (to simplify the factor structure).
- **Apply Promax rotation**: Compute power transformation and refine the rotation matrix.

An **oblique rotation method** is a technique in factor analysis or signal decomposition where the rotated factors or components are allowed to be correlated with each other. This is in contrast to **orthogonal rotation methods**, where factors are constrained to remain uncorrelated (i.e., they are at 90° angles in the multidimensional space).

Rotation Methods - Promax




```python
def promax(Phi, power=4):
    """
    Perform Promax (oblique) rotation on a matrix.
    :param Phi: Matrix to be rotated (e.g., ICA components after Varimax rotation).
    :param power: Exponent used in the power transformation (default is 4).
    :return: The matrix after Promax rotation.
    """
    # Step 1: Perform Varimax rotation to simplify the factor structure
    Phi_varimax = varimax(Phi)

    # Step 2: Compute the power transformation of the Varimax-rotated matrix
    # This transformation emphasizes larger factor loadings while suppressing smaller ones.
    U = np.sign(Phi_varimax) * (np.abs(Phi_varimax) ** power)

    # Step 3: Compute the oblique rotation matrix
    # Compute gamma, which aligns the transformed matrix U with the Varimax-rotated matrix
    gamma = np.dot(Phi_varimax.T, U)

    # Compute the oblique transformation matrix P
    # This involves inverting gamma^T * gamma and multiplying by gamma^T
    P = np.dot(np.linalg.inv(np.dot(gamma.T, gamma)), gamma.T)

    # Apply the oblique transformation to the power-transformed matrix
    Phi_promax = np.dot(U, P)

    # Return the Promax-rotated matrix
    return Phi_promax
```

Rotation Method - QUARTIMAX (quarti = four)

-Objective to maximize sum of all loadings raised to power four

-For **x = As** and rotation matrix **R**, let **G** = **AR** with elements g_{ij}

-The objective is to maximize square variance, or:

$$Q = \sum_{j=1}^{m} \sum_{i=1}^{n} \left(g_{ij}^4\right)$$

Limitation:

It simplifies well rows (variables) but is prone to "general factor": the situation where a single factor (column) dominates the solution, with many variables loading significantly onto it

Strength:

Quartimax minimizes the number of factors needed to explain a variable

J. O. Nauhaus and C. Wrigley, "*The quartimax method: An analytical approach to orthogonal simple structure*," Britain Journal of Statistical Psychology, vol. 7, no. 2, pp. 81–91, Nov. 1954.

Rotation Methods - Quartimax

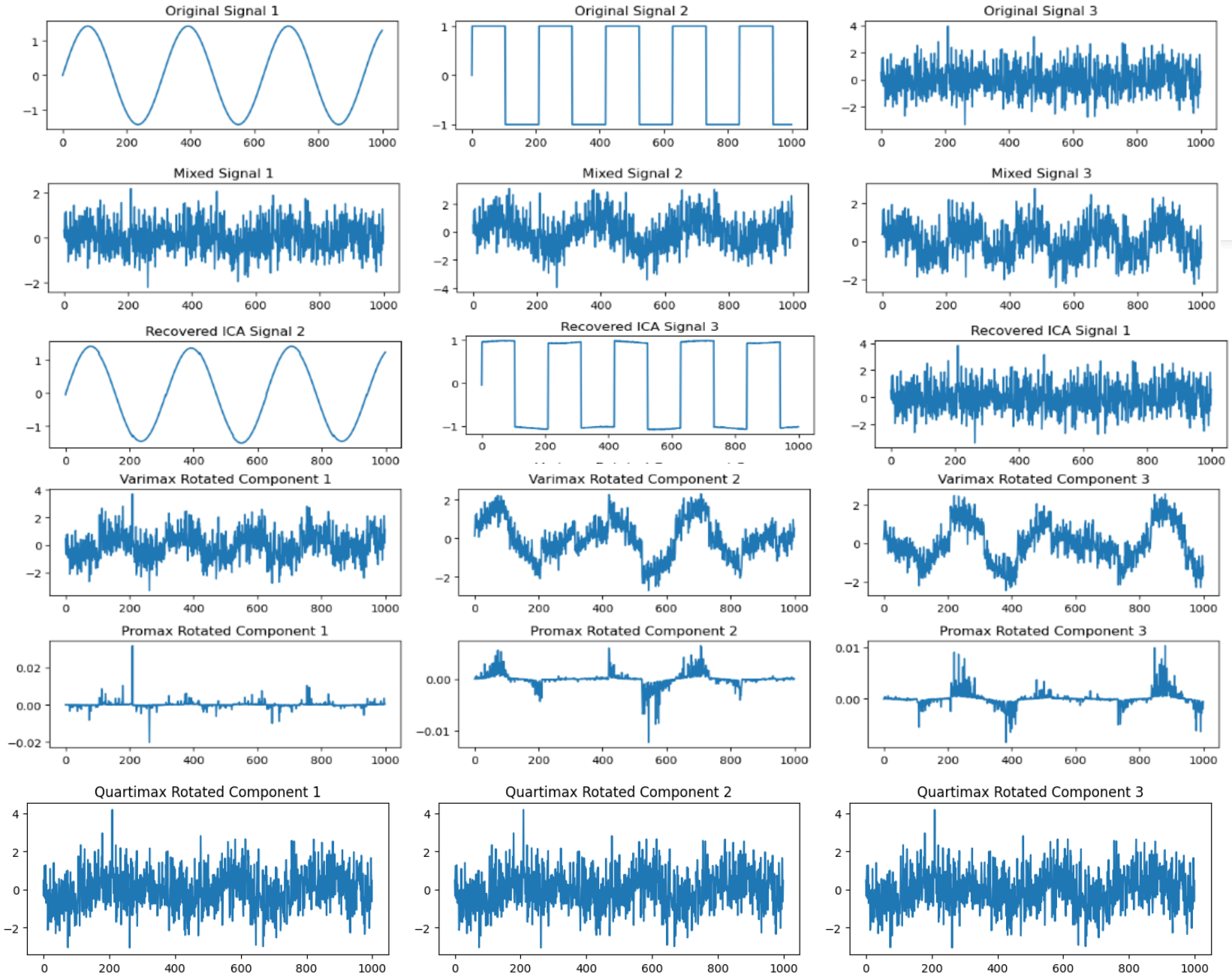
```
n_iterations = 100
Q = np.zeros(n_iterations)

def quartimax(Phi, n_iterations = n_iterations):
    global Q # Tracking iterations of objective function

    i,j = Phi.shape
    R = np.eye(j)
    step = 0.1

    def objective(Phi, R):
        return np.sum(np.dot(Phi,R)**4)

    def gradient(Phi, R):
        # return 4 * np.dot(Phi.T, np.multiply(np.dot(Phi,R), np.dot(Phi,R)**3))
        return 4 * np.dot(Phi.T, np.dot(Phi,R)**3) # Simplified, smoother

    for k in range(n_iterations):
        grad = gradient(Phi, R)
        R += step * grad
        R /= np.linalg.norm(R, axis=0) # Needed to normalize rotation matrix
        Q[k] = objective(Phi, R)

    return np.dot(Phi, R)
```

- Our method implements a **gradient descent** algorithm, where the gradient is

$$\frac{\partial Q}{\partial \mathbf{R}} = 4\mathbf{A}^{\top}\left(\mathbf{G} \circ \mathbf{G}^{3}\right)$$

- … and returns the **rotated loading matrix**

FastICA – Varimax – Promax - Quartimax



Original signals used for mixing

Mixed signals

Signals recovered by FastICA

Varimax

Promax

Quartimax

Conclusions

- **ICA**:
  - Suitable for signal processing and domains
  - Require statistical independence.
- **Varimax**:
  - Best analysis rotation method for uncorrelated factors.
  - Simplifies factor interpretation.
  - A widely used algorithm.
- **Promax**:
  - Suitable when correlated factors are expected or acceptable.
  - Balances between simplicity and realistic models.
- **Quartimax**:
  - Simplifies variable-wise loadings.
  - "Sacrifices" factor interpretability.
  - Less preferred due to its focus on variable reduction rather than factor clarity.

Sources

- J. O. Nauhaus and C. Wrigley, "*The quartimax method: An analytical approach to orthogonal simple structure*," Britain Journal of Statistical Psychology, vol. 7, no. 2, pp. 81–91, Nov. 1954.
- Neuhaus, Jack O., and Charles Wrigley. "The quartimax method: An analytic approach to orthogonal simple structure 1." *British Journal of Statistical Psychology* 7.2 (1954): 81-91.
- Forina, Michelle, et al. "Methods of varimax rotation in factor analysis with applications in clinical and food chemistry." *Journal of Chemometrics* 3.S1 (1989): 115-125.
- Chawla, M. P. S. "PCA and ICA processing methods for removal of artifacts and noise in electrocardiograms: A survey and comparison." *Applied Soft Computing* 11.2 (2011): 2216-2226.

Thank you!