Class 3

Power System Event Classification with Convolutional Neural Network Using PMU Data
– Power System Event Detection Using Graph Signal Processing

Outline

- What is Graph Signals
- Graph Laplacian
- Graph Fourier Transform (GFT)
- Event Detection by Graph Fourier Transform
- Assignment walk-through

Outline

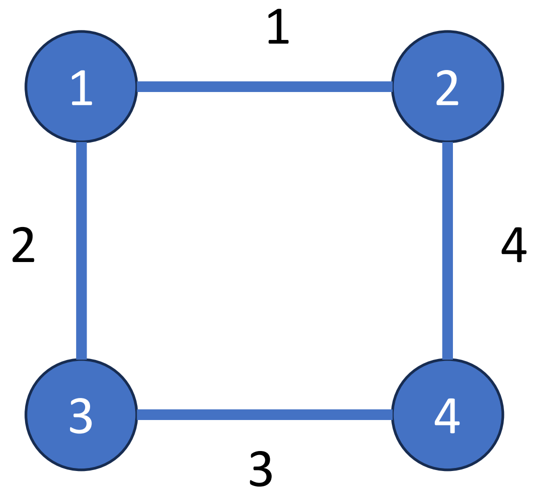
- What is Graph Signal
- Graph Laplacian
- Graph Fourier Transform (GFT)
- Event Detection by Graph Fourier Transform
- Assignment walk-through

Graphs and Adjacency Matrices

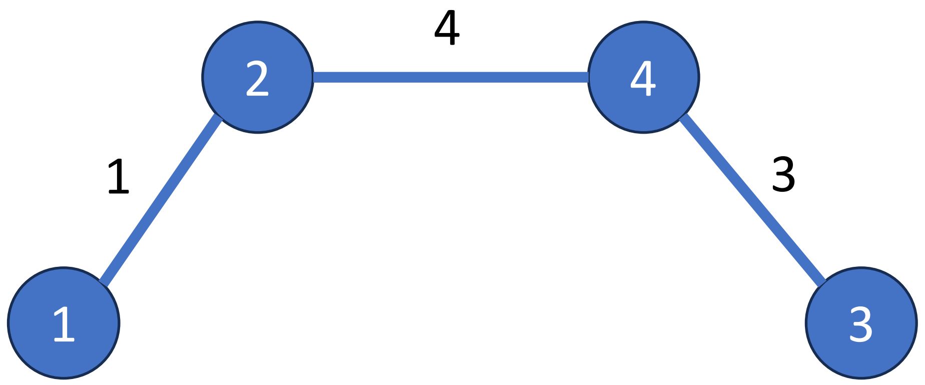
- A graph is a triplet $G = (V, E, W)$
  - $V = \{1, 2, \ldots, N\}$ is a finite set of N nodes or vertices.
  - $E \subseteq \{\mathrm{V} \times V\}$ is a set of edges defined as pairs (n, m)
  - $W\colon\ E \to \mathbb{R}$ is a map from the set of edges to scalar values, $(n, m) \to W_{nm}$

- Adjacency Matrices
  - The Adjacency Matrices $A \subseteq \mathbb{R}^{N \times N}$ is defined as
  - $A_{nm} = \begin{cases} W_{nm}, & if\ (n, m) \in E \\ 0, & otherwiese \end{cases}$

Adjacency Matrices -- examples



$$A = \begin{bmatrix} 0 & 1 & 2 & 0 \\ 1 & 0 & 0 & 4 \\ 2 & 0 & 0 & 3 \\ 0 & 4 & 3 & 0 \end{bmatrix}$$



$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 4 \\ 0 & 0 & 0 & 3 \\ 0 & 4 & 3 & 0 \end{bmatrix}$$

In Power System

Node: Different PMUs

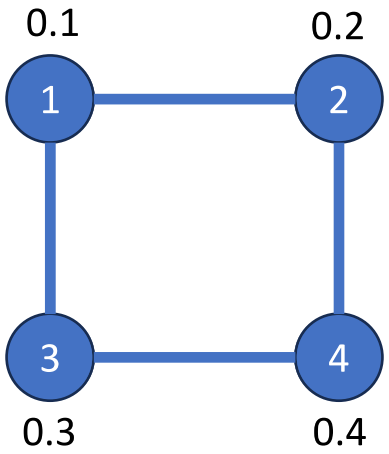
Edges: correlation between PMUs

Graph Signals

Graph signals are mappings $x\colon\ V \to \mathbb{R}$.

Defined on the graph vertices.

x_n represents the signal value at the n-th vertex in V.



$$x = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.3 \\ 0.4 \end{bmatrix}$$

In Power System

Node: Different PMUs

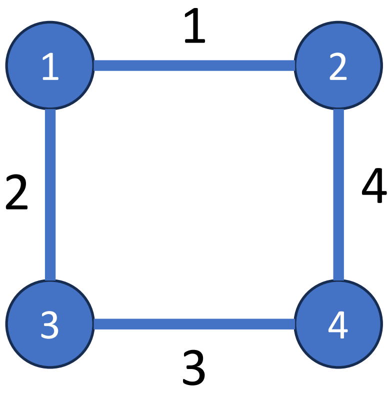
Graph Signals: PMUs measurement

Outline

- What is Graph Signals
- Graph Laplacian
- Graph Fourier Transform (GFT)
- Event Detection by Graph Fourier Transform
- Assignment walk-through

Degree of node and degree matrix

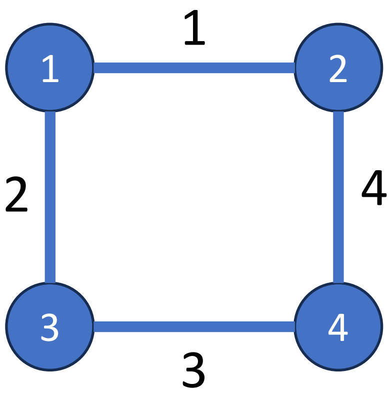
- Degree of a node
  - The degree of a node is the sum of the weights of its incident edges.
  - Given a weighted and undirected graph $G = (V, E, W)$
  - The degree of node i, $deg(i)$ is defined as $deg(i) = \sum_{j \in \mathcal{N}(i)} \mathrm{W}_{ij}$
    - $\mathcal{N}(i)$ is the set of neighbors of node i
  - Using the adjacency matrix A
    - $deg(i) = \sum_j A_{ij}$
  - The degree matrix $D \subseteq R^{N \times N}$ is a diagonal matrix, $D_{ii} = \mathrm{deg}(i)$



$$\mathrm{D} = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 7 \end{bmatrix}$$

Laplacian of Graph

- Given a graph G, with the adjacency matrix A and degree matrix D
- The Laplacian matrix $L \subseteq \mathbb{R}^{N\times N}$ is defined as
  - $L = D \ - A$
- Equivalently, L can be defined elementally as:
  - $L_{ij} = \begin{cases} \deg(i) = \sum_{j\in\mathcal{N}(i)} \mathrm{W}_{ij}\,, & if\ i = j \\ -W_{ij}, & if\ j \in \mathcal{N}(i) \\ 0, & otherwiese \end{cases}$



$$L = \begin{bmatrix} 3 & -1 & -2 & 0 \\ -1 & 5 & 0 & -4 \\ -2 & 0 & 5 & -3 \\ 0 & -4 & -3 & 7 \end{bmatrix}$$

Outline

- What is Graph Signals
- Graph Laplacian
- Graph Fourier Transform (GFT)
- Event Detection by Graph Fourier Transform
- Assignment walk-through

Graph Fourier Transform (GFT)

- Laplacian matrix $L \subseteq R^{N\times N}$ has a complete set of orthonormal eigenvectors:

Left Eigenvector

$$L = U\Lambda U^T$$

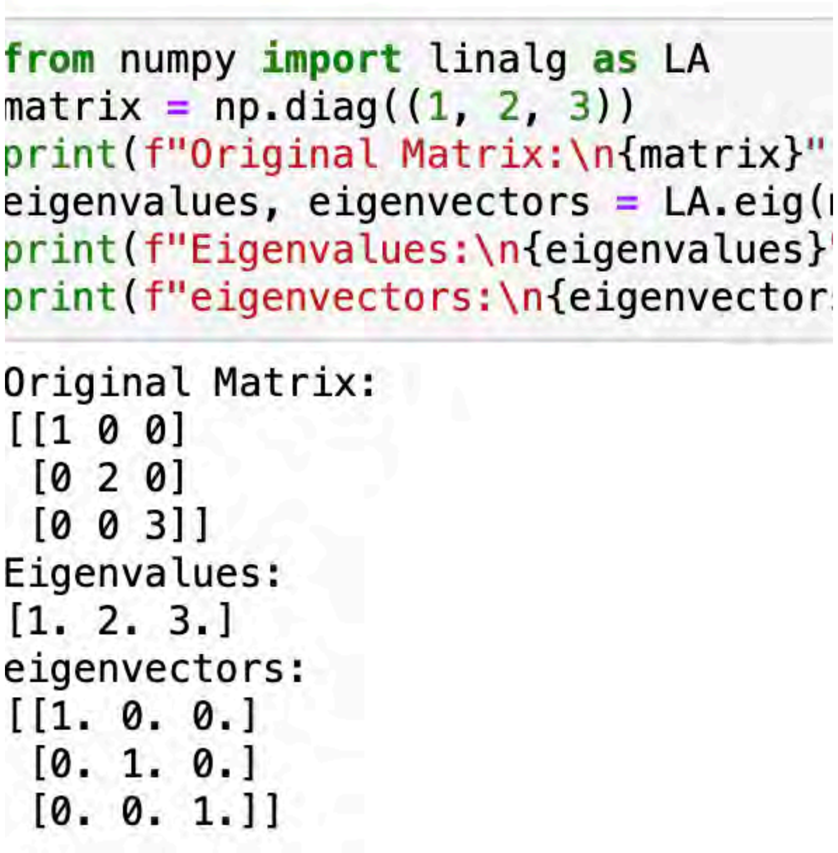
Right Eigenvector

$$L = \begin{bmatrix} u_1 & \ldots & u_N \end{bmatrix} \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_N \end{bmatrix} \begin{bmatrix} u_1^T \\ \vdots \\ u_N^T \end{bmatrix}$$

- Eigenvalues $(\lambda_1, \lambda_2, \ldots \lambda_N)$ are usually sorted in ascending order: $\lambda_1 < \lambda_2 < \ \ldots < \ \lambda_N$

Get the eigenvectors and eigenvalues of the matrix in Python

- The eigenvectors and eigenvalues can be computed by a numerical computing library, NumPy .
- “numpy.linalg.eig” computes the eigenvalues and right eigenvectors of a square matrix.
- Here is an example of the “numpy.linalg.eig”.
- More resource: https://numpy.org/doc/stable/reference/generated/numpy.linalg.eig.html



```
from numpy import linalg as LA
matrix = np.diag((1, 2, 3))
print(f"Original Matrix:\n{matrix}"
eigenvalues, eigenvectors = LA.eig(
print(f"Eigenvalues:\n{eigenvalues}"
print(f"eigenvectors:\n{eigenvector
```

```
Original Matrix:
[[1 0 0]
 [0 2 0]
 [0 0 3]]
Eigenvalues:
[1. 2. 3.]
eigenvectors:
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

Graph Fourier Transform (GFT)

- Given the orthonormal eigenvectors U of the graph Laplacian L and the graph signal x

GFT represent the signal's spectral content in the graph domain

$$L = U\Lambda U^T$$

Eigenvectors associated with lower eigenvalues: smoother variations in the signal

- The **Graph Fourier Transform (GFT)** of x is defined as

$$\tilde{x} = U^T x = [u_1 \quad \cdots \quad u_N]^T x$$

Each coefficient indicates the contribution of the corresponding eigenvector to the original signal

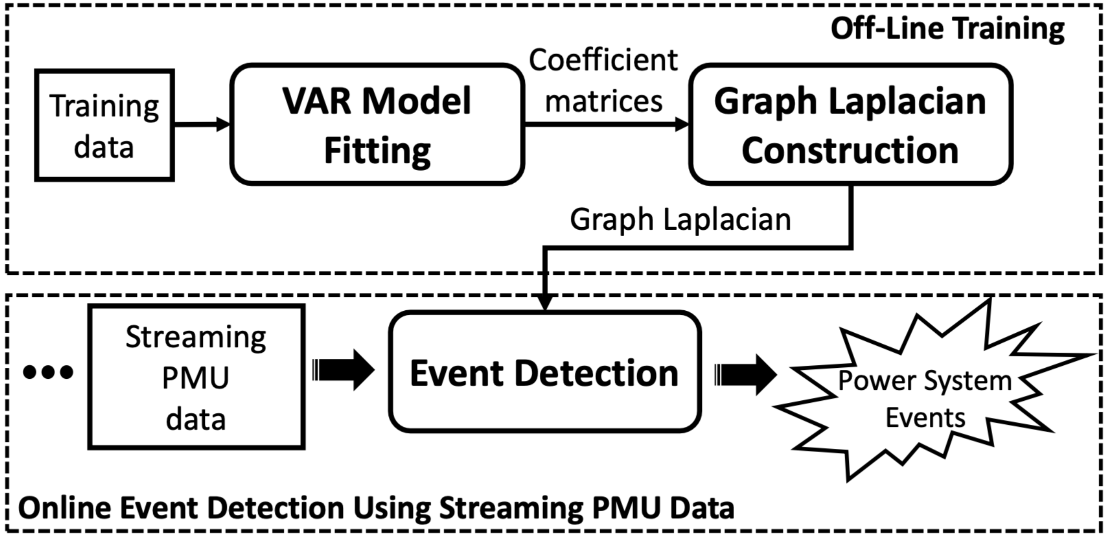
Eigenvectors associated with higher eigenvalues: more oscillatory or rapid variations in the signal

Outline

- What is Graph Signals
- Graph Laplacian
- Graph Fourier Transform (GFT)
- Event Detection by Graph Fourier Transform
- Assignment walk-through

Power System Event Detection by Graph Signal Processing

- The event detection in the power system contains two stages:
  - 1. The offline training captures the spatial and temporal correlation between PMUs' data and constructs the Graph Laplacian matrix.
  - 2. The online detection algorithm uses the built Graph Laplacian matrix and Graph Fourier Transform (GFT) to quantify the original signals. Then, it computes event indicators, based on this quantification, facilitating the detection of both normal and abnormal events.



Calculate the spatial correlation between different PMUs

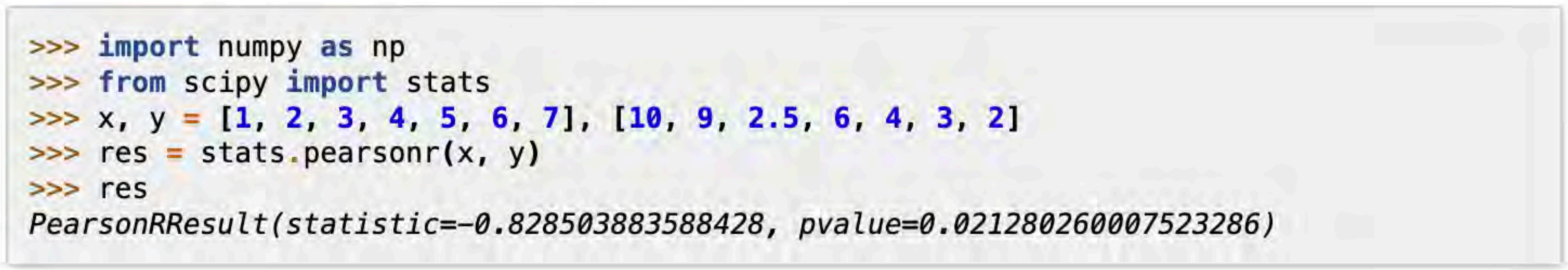
- Pearson correlation coefficients
  - The Pearson correlation coefficient (PCC) is a correlation coefficient that measures linear correlation between two sets of data.
  - Suppose x is a series of values from PMU i, and y is another series of values from PMU j. Then we can have:

$$r_{ij} = \frac{\sum_{k=1}^{n}(x_k - \bar{x})(y_k - \bar{y})}{\sqrt{\sum_{k=1}^{n}(x_k - \bar{x})^2}\sqrt{\sum_{k=1}^{n}(y_k - \bar{y})^2}}$$

- $P \subseteq \mathbb{R}^{M\times M}$, where M is the number of PMUs, P_{ij} indicates the Pearson correlation between PMU i and PMU j.

Calculate the spatial correlation between different PMUs (Code example)

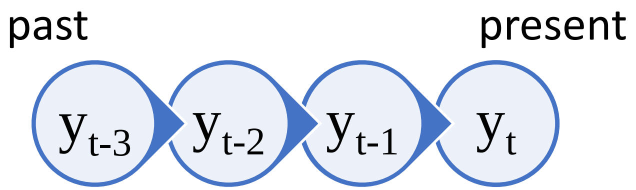
- The scipy, a scientific computing library in Python, provides us with a ready-made function for calculating the Pearson correlation.
- Here is a simple example of how to calculate the Pearson correlation in Python.



```
>>> import numpy as np
>>> from scipy import stats
>>> x, y = [1, 2, 3, 4, 5, 6, 7], [10, 9, 2.5, 6, 4, 3, 2]
>>> res = stats.pearsonr(x, y)
>>> res
PearsonRResult(statistic=-0.828503883588428, pvalue=0.021280260007523286)
```

Calculate the Temporal correlation between different PMUs

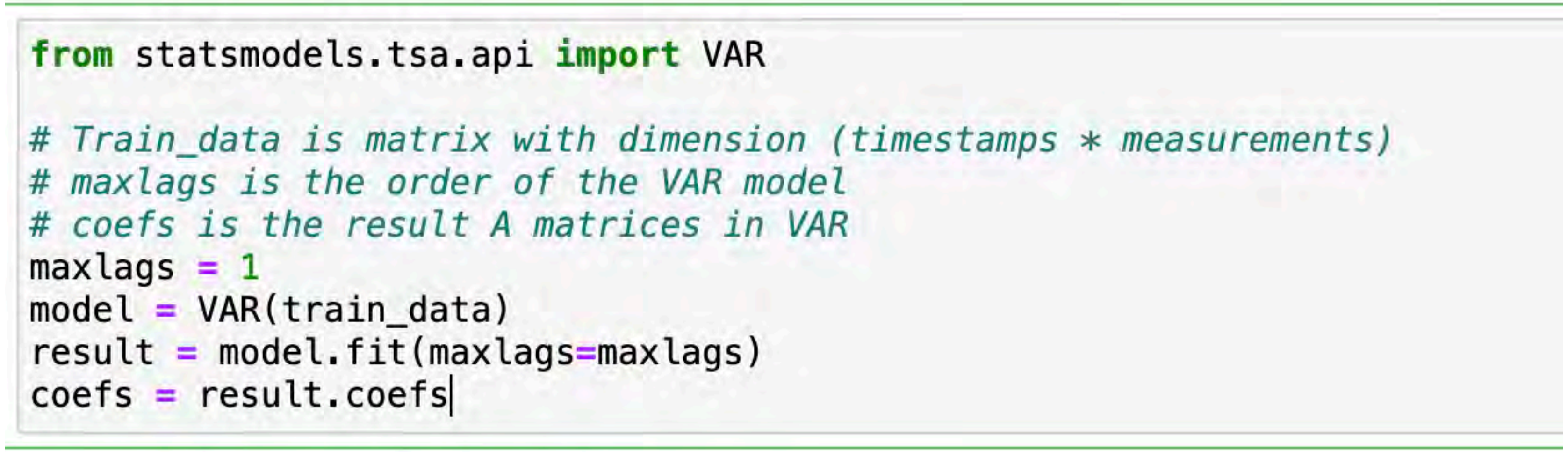
- Vector autoregression (VAR) is a statistical model used to capture the relationship between multiple quantities as they change over time.
- A p-th-order VAR model is written as:
  - $y_t = c + T_1 y_{t-1} + T_2 y_{t-2} + \ \ldots + T_p y_{t-p} + e_t$



- y_t is a vector of the different PMUs' measurement at timestamp t
- T_k denotes coefficient of VAR model (fixed values over time)
- In this course, we only care about the first-order VAR model, which means we only need to get the matrix T_1.
- $T_1 \subseteq \mathbb{R}^{M \times M}$, where M is the number of PMUs, $T_{1_{ij}}$ indicate the temporal correlation between PMU i and PMU j.

Calculate the temporal correlation between different PMUs (Code example)

- The “statsmodels” library in Python offers a wide range of statistical models, including the VAR model, among others.
- Here is a simple example of how to fit the VAR model to target data in Python.



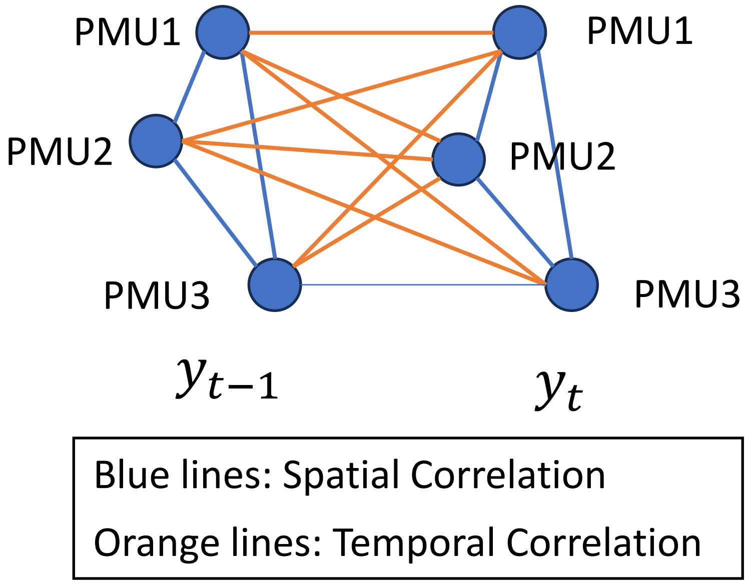
```
from statsmodels.tsa.api import VAR

# Train_data is matrix with dimension (timestamps * measurements)
# maxlags is the order of the VAR model
# coefs is the result A matrices in VAR
maxlags = 1
model = VAR(train_data)
result = model.fit(maxlags=maxlags)
coefs = result.coefs
```

Construct the Graph Laplacian Matrix

- Graph of the PMU series data
- Adjacency matrix A

$$A = \begin{bmatrix} P & T_1 \\ T_1 & P \end{bmatrix}$$

- Based on the graph structure, adjacency matrix A and the corresponding degree matrix D are derived. With these matrices, Graph Laplacian Matrix L can be calculated.



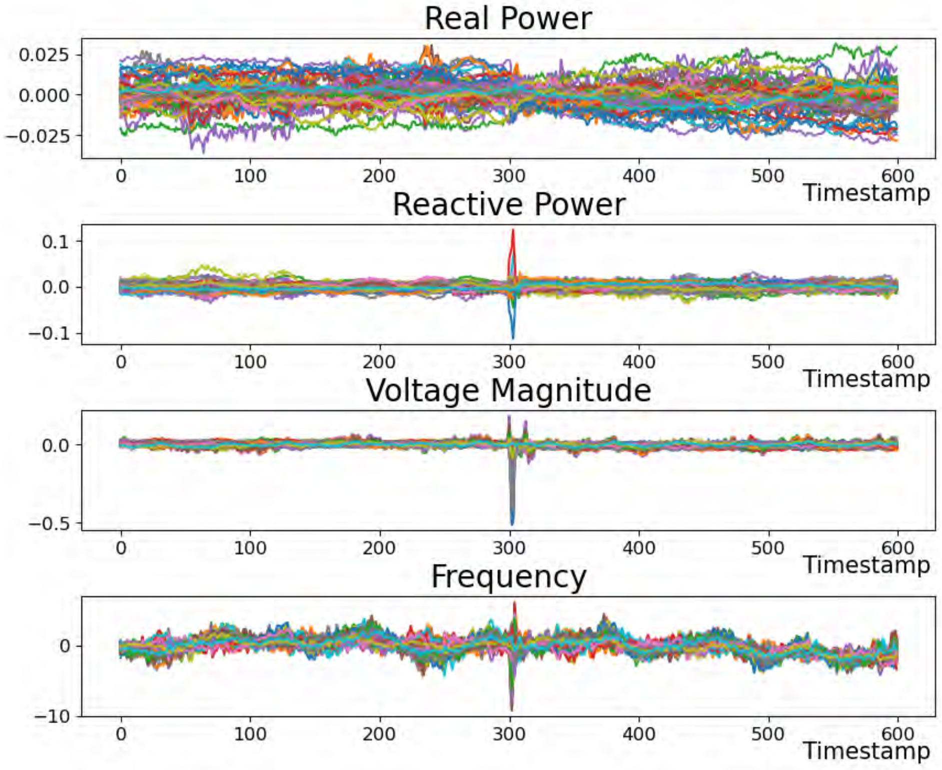
Blue lines: Spatial Correlation

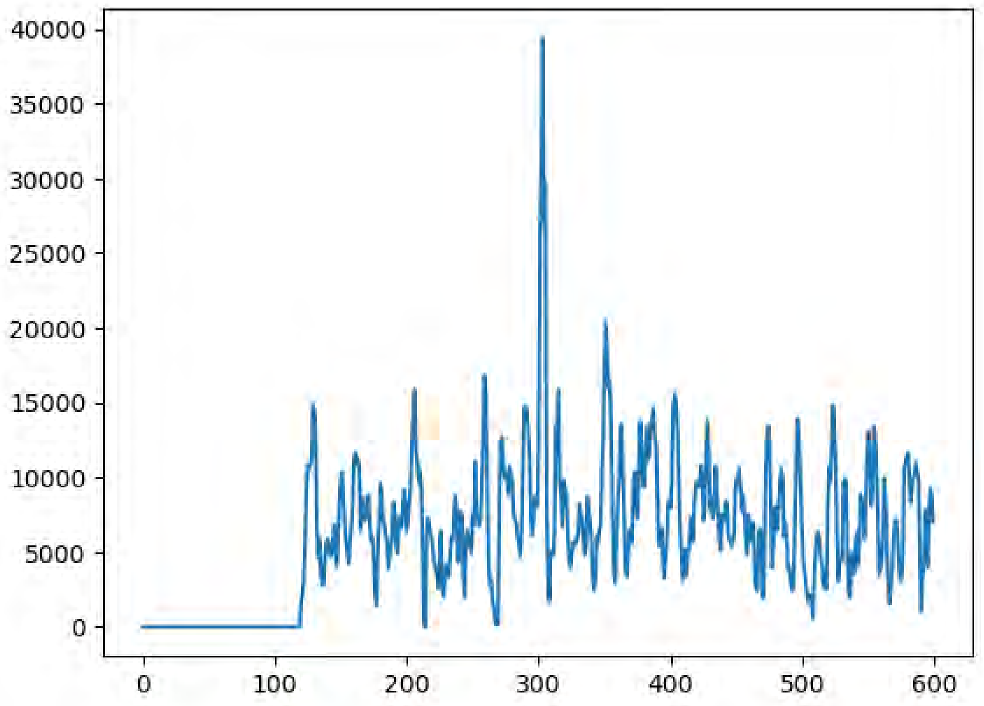
Orange lines: Temporal Correlation

Graph Fourier Transform (GFT) and Abnormal Measurement Indicator (AMI)

- The Laplacian matrix $L \subseteq \mathbb{R}^{N \times N}$ is:
  - $L = D \ - A$
- Calculate the eigenvectors and eigenvalues
  - $L = U \Lambda U^T$
- Graph Fourier Transform (GFT)
  - $\tilde{x} = U^T x = [u_1 \quad \cdots \quad u_N]^T x$
- Abnormal Measurement Indicator
  - Summation of the absolute value of processed signal $\tilde{x}$
  - $AMI = diag(\Lambda)^T |\tilde{x}| = \ \sum_{k=1}^{N} \lambda_k |\tilde{x}_k|$

Example of the Abnormal Measurement Indicator (AMI)



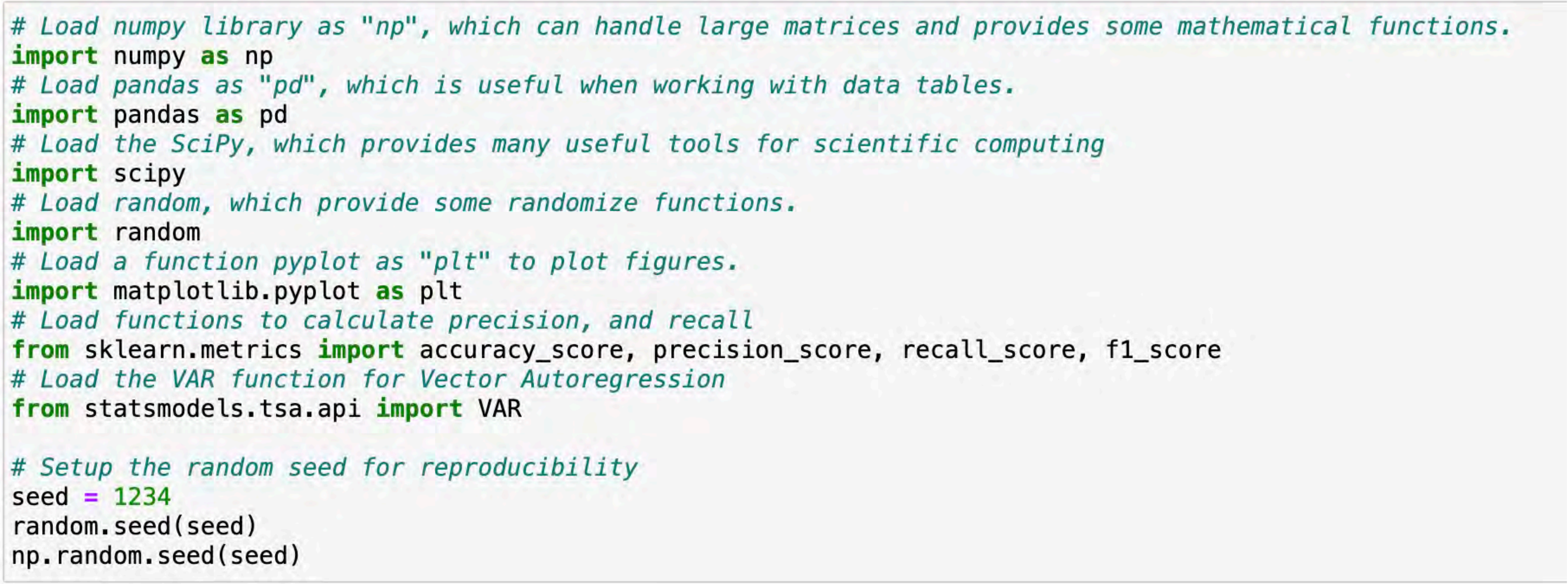


Outline

- What is Graph Signals
- Graph Laplacian
- Graph Fourier Transform (GFT)
- Event Detection by Graph Fourier Transform
- Assignment walk-through

Load Needed Libraries/Packages

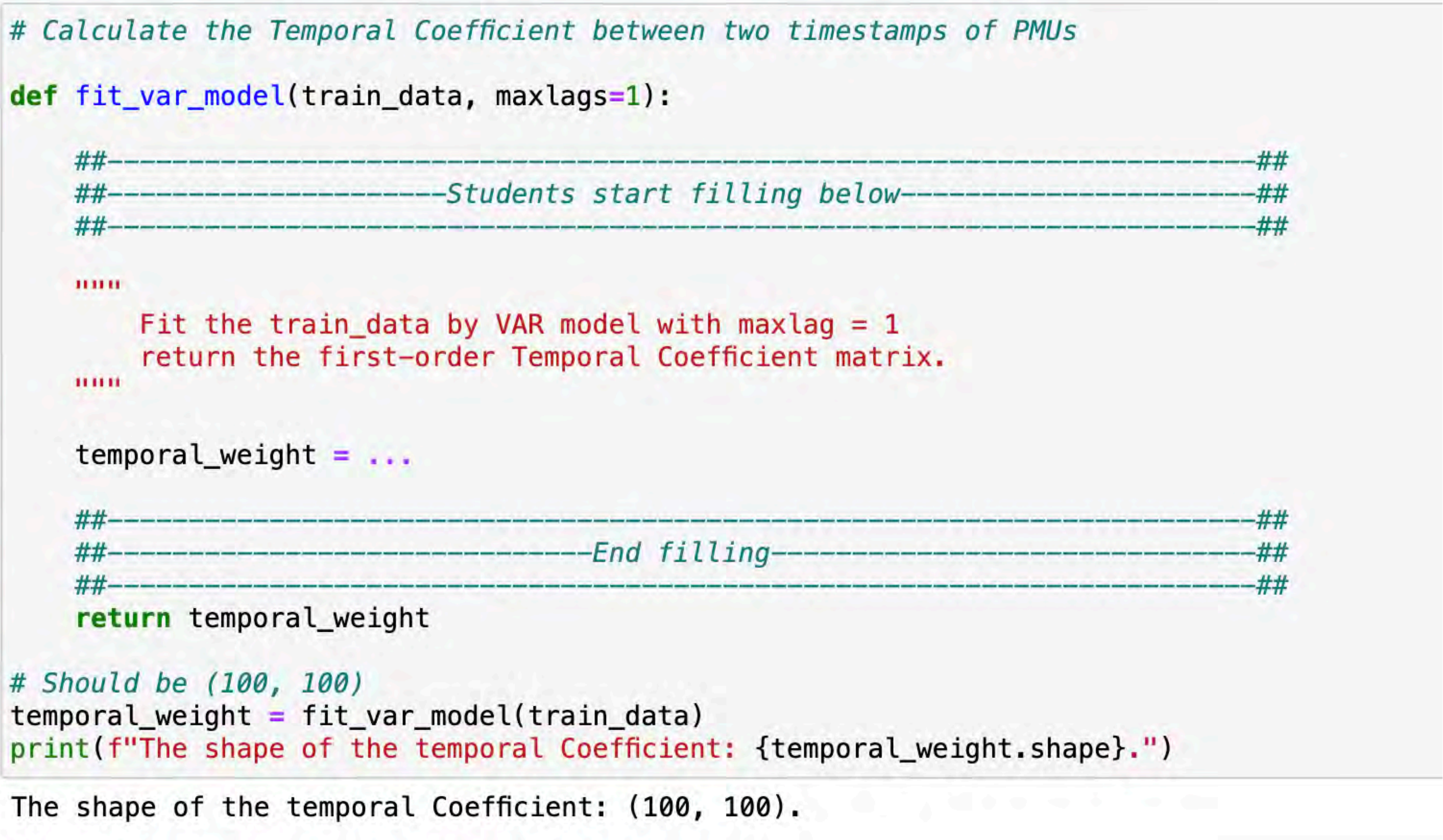
- To start, we need to load the libraries and packages that will be used in the code.
- Similar to class 1, we need new packages for new evaluation metrics, early stopping, and VAR model.



```
# Load numpy library as "np", which can handle large matrices and provides some mathematical functions.
import numpy as np
# Load pandas as "pd", which is useful when working with data tables.
import pandas as pd
# Load the SciPy, which provides many useful tools for scientific computing
import scipy
# Load random, which provide some randomize functions.
import random
# Load a function pyplot as "plt" to plot figures.
import matplotlib.pyplot as plt
# Load functions to calculate precision, and recall
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
# Load the VAR function for Vector Autoregression
from statsmodels.tsa.api import VAR

# Setup the random seed for reproducibility
seed = 1234
random.seed(seed)
np.random.seed(seed)
```

Calculate Temporal Coefficient

- In this task, you are required to calculate the Temporal Coefficient between different PMUs by using the VAR model.

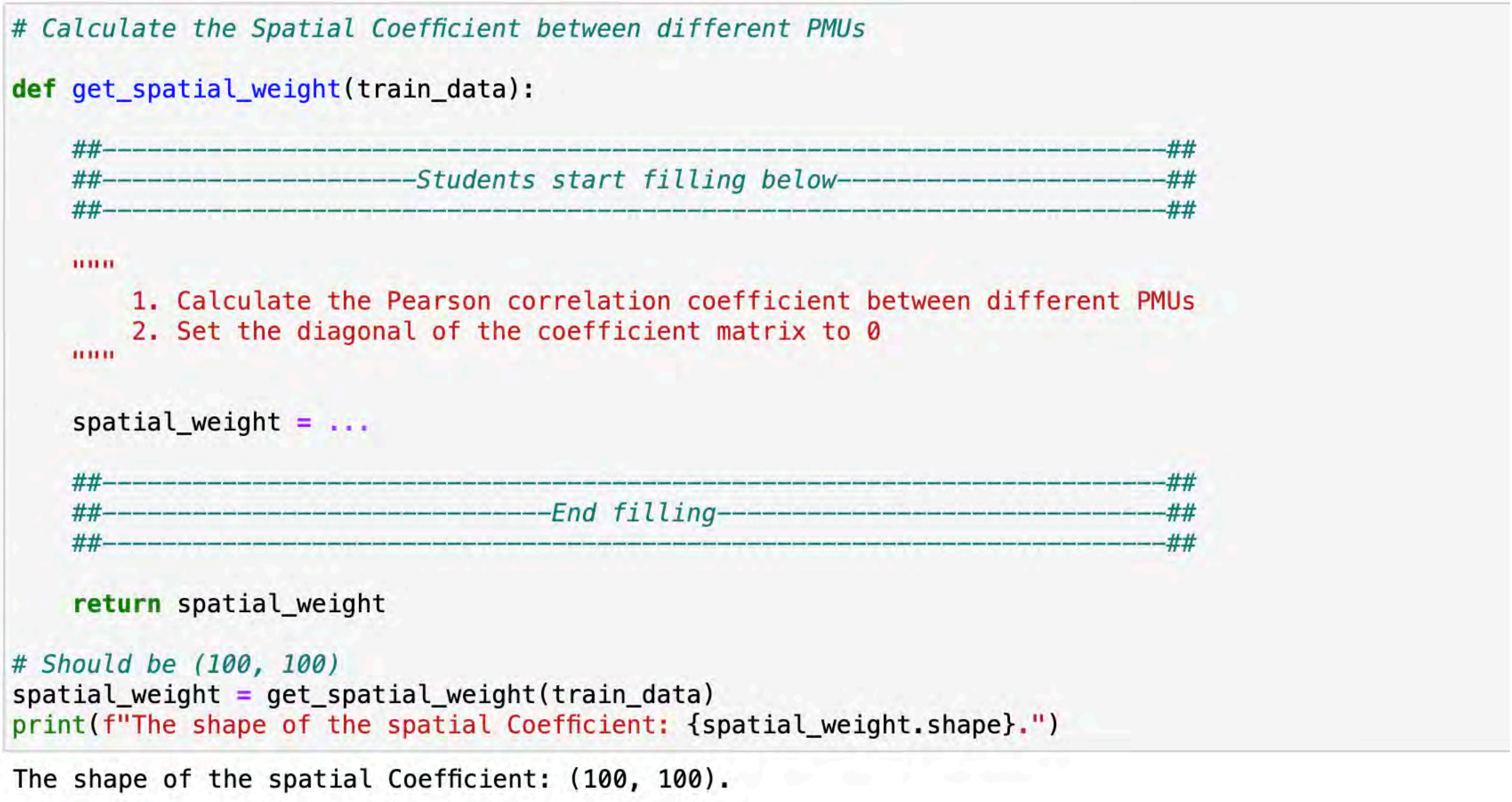


```python
# Calculate the Temporal Coefficient between two timestamps of PMUs

def fit_var_model(train_data, maxlags=1):

    ##-----------------------------------------------------------------------##
    ##---------------------Students start filling below----------------------##
    ##-----------------------------------------------------------------------##

    """
        Fit the train_data by VAR model with maxlag = 1
        return the first-order Temporal Coefficient matrix.
    """

    temporal_weight = ...

    ##-----------------------------------------------------------------------##
    ##------------------------------End filling------------------------------##
    ##-----------------------------------------------------------------------##
    return temporal_weight

# Should be (100, 100)
temporal_weight = fit_var_model(train_data)
print(f"The shape of the temporal Coefficient: {temporal_weight.shape}.")
```

```
The shape of the temporal Coefficient: (100, 100).
```

Calculate Spatial Coefficient

- In this task, you are required to calculate the Spatial Coefficient between different PMUs by Pearson correlation coefficients.

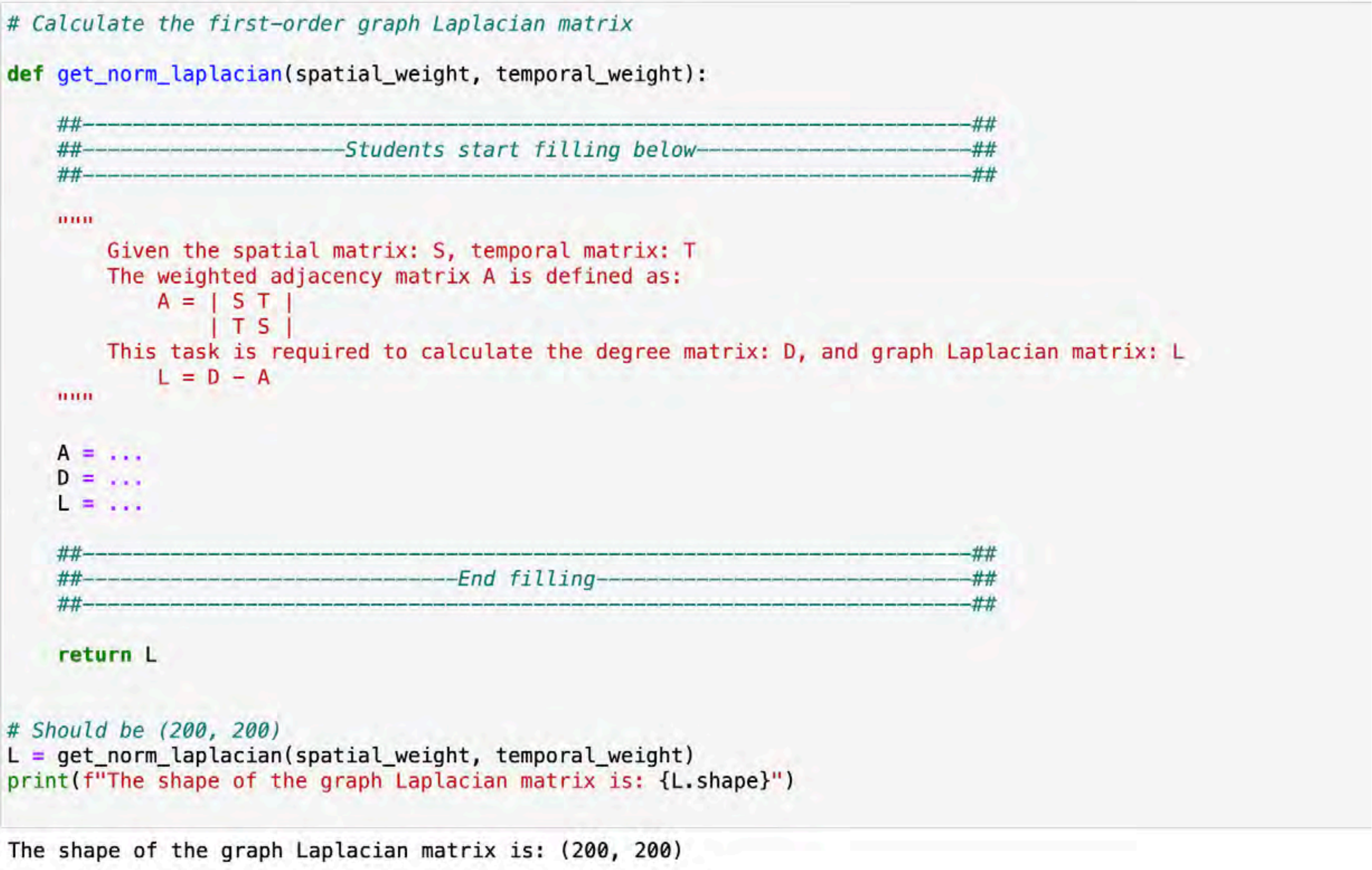


```
# Calculate the Spatial Coefficient between different PMUs

def get_spatial_weight(train_data):

    ##-----------------------------------------------------------------------##
    ##---------------------Students start filling below----------------------##
    ##-----------------------------------------------------------------------##

    """
        1. Calculate the Pearson correlation coefficient between different PMUs
        2. Set the diagonal of the coefficient matrix to 0
    """

    spatial_weight = ...

    ##-----------------------------------------------------------------------##
    ##------------------------------End filling------------------------------##
    ##-----------------------------------------------------------------------##

    return spatial_weight

# Should be (100, 100)
spatial_weight = get_spatial_weight(train_data)
print(f"The shape of the spatial Coefficient: {spatial_weight.shape}.")
```

```
The shape of the spatial Coefficient: (100, 100).
```

Calculate Graph Laplacian matrix

- In this task, you are required to calculate the graph Laplacian matrix from spatial and temporal coefficients

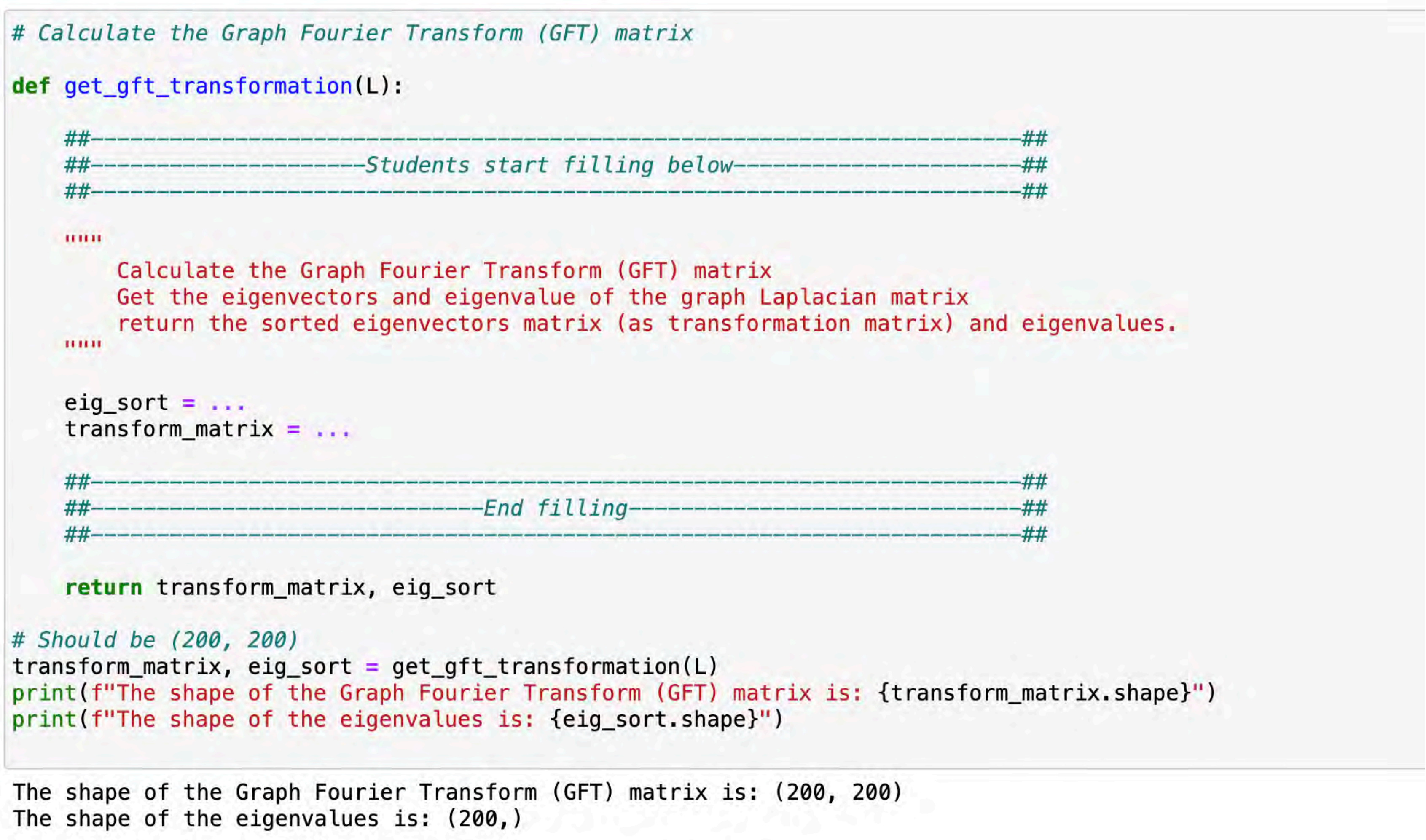


```
# Calculate the first-order graph Laplacian matrix

def get_norm_laplacian(spatial_weight, temporal_weight):

    ##-----------------------------------------------------------------------##
    ##---------------------Students start filling below----------------------##
    ##-----------------------------------------------------------------------##

    """
        Given the spatial matrix: S, temporal matrix: T
        The weighted adjacency matrix A is defined as:
            A = | S T |
                | T S |
        This task is required to calculate the degree matrix: D, and graph Laplacian matrix: L
            L = D - A
    """

    A = ...
    D = ...
    L = ...

    ##-----------------------------------------------------------------------##
    ##------------------------------End filling------------------------------##
    ##-----------------------------------------------------------------------##

    return L


# Should be (200, 200)
L = get_norm_laplacian(spatial_weight, temporal_weight)
print(f"The shape of the graph Laplacian matrix is: {L.shape}")
```

```
The shape of the graph Laplacian matrix is: (200, 200)
```

Calculate Graph Fourier Transform (GFT) matrix

- In this task, you are required to calculate the Graph Fourier Transform (GFT) matrix from graph Laplacian

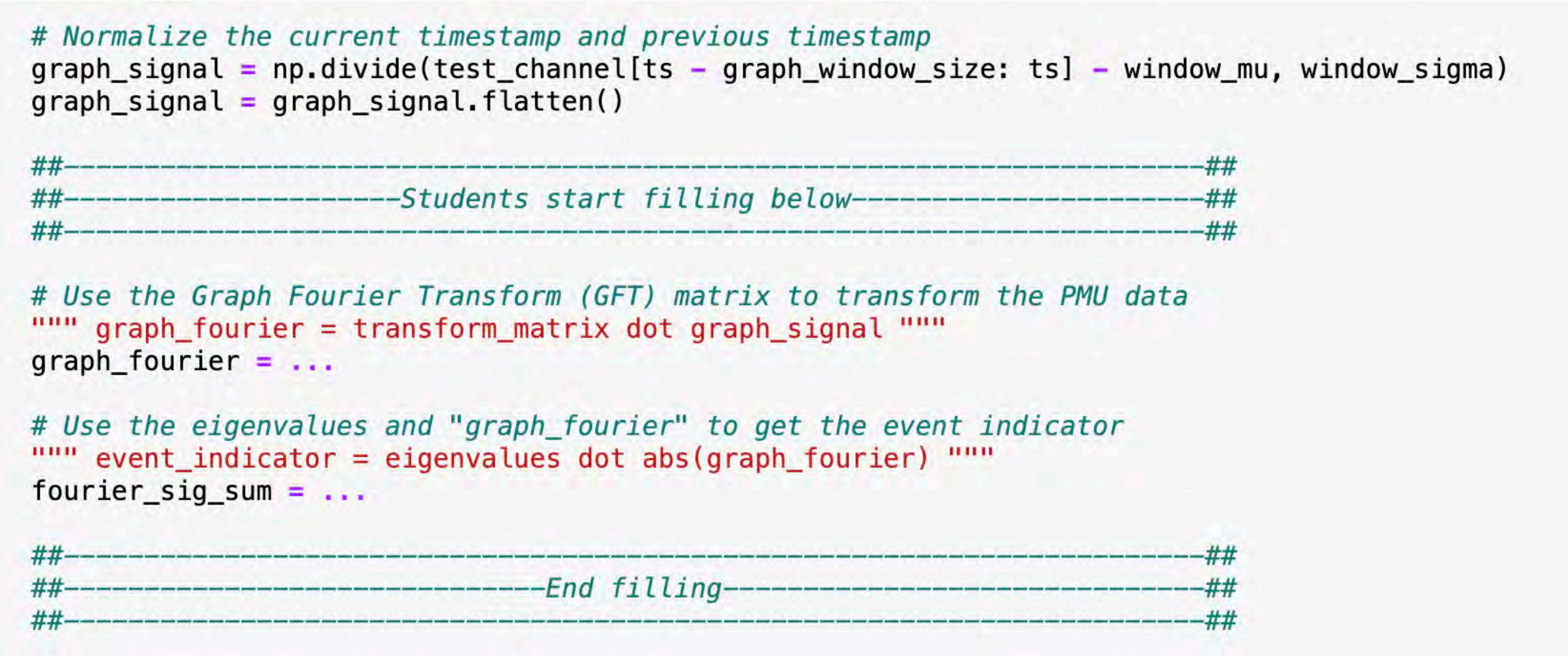


```
# Calculate the Graph Fourier Transform (GFT) matrix

def get_gft_transformation(L):

    ##-----------------------------------------------------------------------##
    ##---------------------Students start filling below----------------------##
    ##-----------------------------------------------------------------------##

    """
        Calculate the Graph Fourier Transform (GFT) matrix
        Get the eigenvectors and eigenvalue of the graph Laplacian matrix
        return the sorted eigenvectors matrix (as transformation matrix) and eigenvalues.
    """

    eig_sort = ...
    transform_matrix = ...

    ##-----------------------------------------------------------------------##
    ##------------------------------End filling------------------------------##
    ##-----------------------------------------------------------------------##

    return transform_matrix, eig_sort

# Should be (200, 200)
transform_matrix, eig_sort = get_gft_transformation(L)
print(f"The shape of the Graph Fourier Transform (GFT) matrix is: {transform_matrix.shape}")
print(f"The shape of the eigenvalues is: {eig_sort.shape}")
```

```
The shape of the Graph Fourier Transform (GFT) matrix is: (200, 200)
The shape of the eigenvalues is: (200,)
```

Calculate event indicator by transformation matrix and eigenvalues

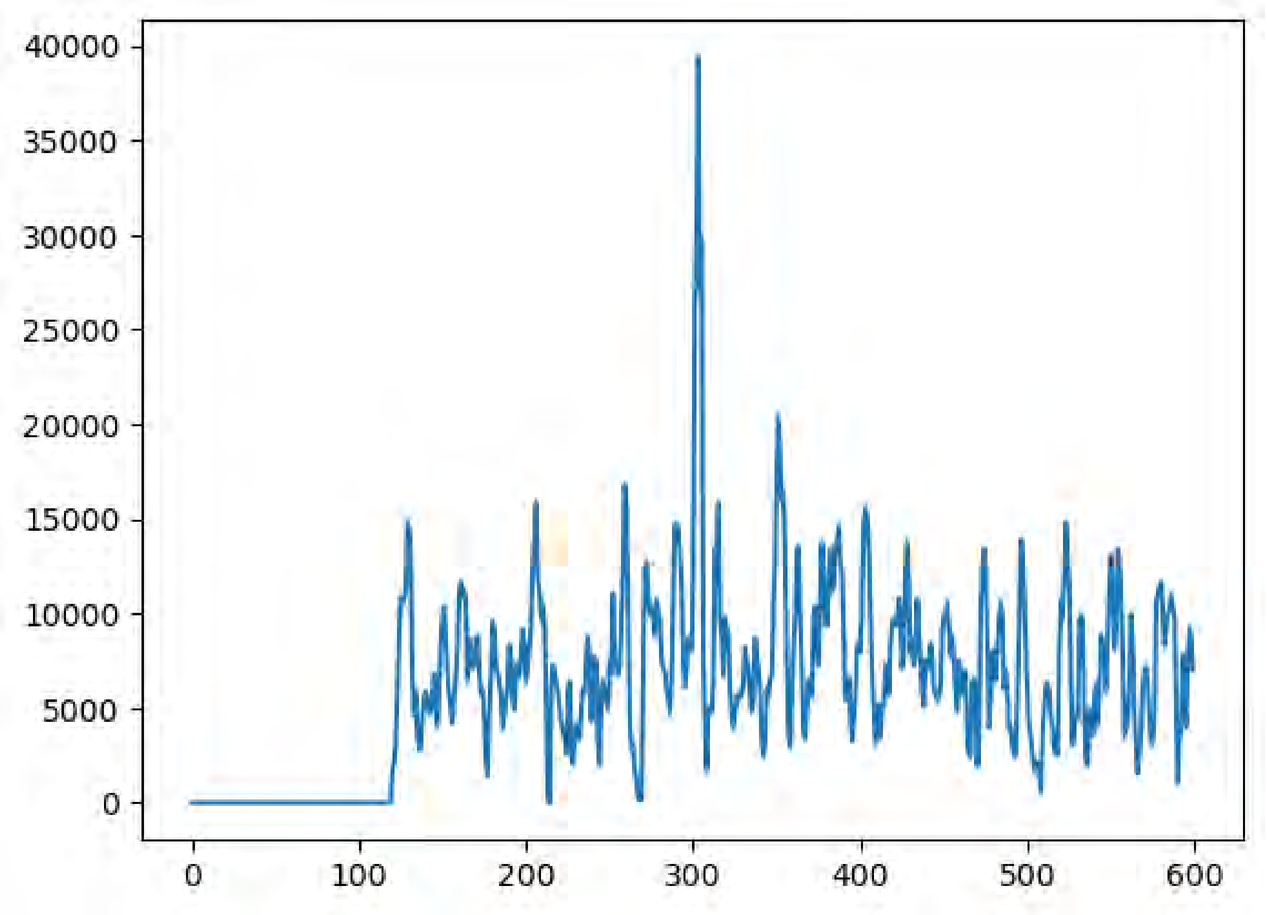
- In this task, you are required to calculate the abnormal event indicator (AMI that we learned) by the graph signal, Graph Fourier Transform (GFT) matrix, and eigenvalues.



```python
# Normalize the current timestamp and previous timestamp
graph_signal = np.divide(test_channel[ts - graph_window_size: ts] - window_mu, window_sigma)
graph_signal = graph_signal.flatten()

##-----------------------------------------------------------------------##
##---------------------Students start filling below----------------------##
##-----------------------------------------------------------------------##

# Use the Graph Fourier Transform (GFT) matrix to transform the PMU data
""" graph_fourier = transform_matrix dot graph_signal """
graph_fourier = ...

# Use the eigenvalues and "graph_fourier" to get the event indicator
""" event_indicator = eigenvalues dot abs(graph_fourier) """
fourier_sig_sum = ...

##-----------------------------------------------------------------------##
##------------------------------End filling------------------------------##
##-----------------------------------------------------------------------##
```

Result Example

- Here is an example of the abnormal event indicator results.
- Codes related to data processing and visualization are provided in this homework.


40000
35000
30000
25000
20000
15000
10000
5000
0
0
100
200
300
400
500
600