Network Analysis with NumPy

2025-01-15

Table of contents

1 Network Graph as an Adjacency Matrix

A network topology can be represented as an adjacency matrix A where $A_{ij} = 1$ if there is a link between node i and node j. We'll model a small 5-router backbone and compute some useful properties.

```
import numpy as np

# 5-router backbone adjacency matrix (undirected)
A = np.array([
    [0, 1, 1, 0, 0],
    [1, 0, 1, 1, 0],
    [1, 1, 0, 0, 1],
    [0, 1, 0, 0, 1],
    [0, 0, 1, 1, 0]
])

routers = ['R1', 'R2', 'R3', 'R4', 'R5']
print("Adjacency matrix:")
print(A)
```

2 Degree Matrix and Laplacian

The **degree matrix** D is diagonal with D_{ii} = number of links on router i. The **graph Laplacian** is $L = D - A$. Its eigenvalues reveal connectivity properties: - The number of zero eigenvalues = number of disconnected components - The second smallest eigenvalue (Fiedler value) measures how well-connected the network is

```
D = np.diag(A.sum(axis=1))
L = D - A

eigenvalues, eigenvectors = np.linalg.eigh(L)
eigenvalues = np.round(eigenvalues, 6)
```

```
print("Degree of each router:", dict(zip(routers, D.diagonal().astype(int))))
print("\nLaplacian eigenvalues:", eigenvalues)
print(f"\nFiedler value (algebraic connectivity): {eigenvalues[1]:.4f}")
print("Network is", "connected" if eigenvalues[1] > 0 else "disconnected")
```

3 Link Failure Impact

Remove the R2–R3 link and check if the network stays connected.

```
A_failed = A.copy()
A_failed[1, 2] = 0
A_failed[2, 1] = 0

D_f = np.diag(A_failed.sum(axis=1))
L_f = D_f - A_failed
eigs_failed = np.round(np.linalg.eigh(L_f)[0], 6)

print(f"Fiedler value after R2-R3 failure: {eigs_failed[1]:.4f}")
print("Network is", "connected" if eigs_failed[1] > 0 else "disconnected")
```

4 MIMO Channel Capacity via SVD

In a MIMO system with n_t transmit and n_r receive antennas, the channel matrix $H \in \mathbb{R}^{n_r \times n_t}$. Capacity is computed from the singular values σ_i of H:

$$C = \sum_{i=1}^{\min(n_t, n_r)} \log_2 \left(1 + \frac{\rho}{n_t} \sigma_i^2\right) \quad \text{[bits/s/Hz]}$$

```
np.random.seed(42)
n_t, n_r = 4, 4
SNR_dB = 20
rho = 10 ** (SNR_dB / 10)

# Rayleigh fading channel
H = (np.random.randn(n_r, n_t) + 1j * np.random.randn(n_r, n_t)) / np.sqrt(2)

_, singular_values, _ = np.linalg.svd(H)
capacity = np.sum(np.log2(1 + (rho / n_t) * singular_values**2))

print(f"Singular values: {np.round(singular_values, 4)}")
print(f"MIMO capacity at {SNR_dB} dB SNR: {capacity:.2f} bits/s/Hz")
```