# Calcul Numeric

Laboratorul 2.

Vectori și valori proprii

## 1 Metoda puterii

Fie $\lambda$ o valoare proprie a matricii $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ și $\boldsymbol{v} \in \mathbb{R}^n$ vectorul propriu asociat. Atunci

$$\boldsymbol{A}\boldsymbol{v} = \lambda\boldsymbol{v} \tag{1}$$

Definiția de mai sus este neconstructivă, adică nu permite deducerea unei scheme de calcul pentru valorile și vectorii proprii ale unei matrici. Acest lucru se datorează circularității: pentru a afla valorea proprie este necesară cunoașterea vectorului propriu și invers. Prin urmare, sunt necesare alte metode pentru calculul acestor mărimi. Nici proprietatea că valorile proprii sunt rădăcinile polinomului caracteristic $p(\lambda) = det(\lambda\boldsymbol{I} - A)$ nu este foarte utilă în practică, datorită dificultății calculului determinantului unei matrici de dimensiuni mari.

Ceea ce sugerează, însă definiția (1) este faptul că atunci $\boldsymbol{A}$ când înmulțită cu un vector propriu, matricea se comportă ca un scalar. Motivația stă în faptul că vectori proprii sunt direcțiile invariante ale unei matrici pătratice.

Metoda puterii, utilizată pentru calcul a vectorilor proprii, se folosește de această proprietate.
Algoritmul construiește iterativ șirul de matrici $\boldsymbol{A}\boldsymbol{y}, \boldsymbol{A}^2\boldsymbol{y}, \boldsymbol{A}^3\boldsymbol{y}, ...$, șir ce converge către direcția vectorului propriu asociat valorii proprii dominante. Vectorul $\boldsymbol{y}$ este inițializat aleator, iar la fiecare iterație direcția acestuia se modifică, ajungând, atunci când algoritmul converge, pe direcția vectorului propriu al matricii $\boldsymbol{A}$.

Convergența metodei puterii este asigurată în cazul în care $\boldsymbol{A}$ prezintă o valoare proprie dominantă, iar rata de convergență depinde de raportul $\lambda_1/\lambda_2$.

Algoritmul 1 prezintă pașii metodei puterii.

## 2 Valorile proprii ale unui graf

Fie un graf definit de mulțimea de noduri $U = \{u_1, u_2, ..., u_n\}$. Matricea $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ este formată din elementele $\boldsymbol{A}_{i,j}$ având valoarea 1 dacă între nodurile

**Data:** $\boldsymbol{A} \in \mathbb{R}^{n \times n}$
nivelul de toleranță, $tol$
numărul de iterații, $max_{iter}$
**Result:** vectorul propriu asociat valorii proprii dominante, $\boldsymbol{y} \in \mathbb{R}^n$

Se inițializează $\boldsymbol{y}$ cu valori aleatoare, $iter = 0$ și $err = 1$.
**while** $err > tol$ **do**
  **if** $iter > max_{iter}$ **then**
    Stop
  $\boldsymbol{y} = \boldsymbol{y}/\|y\|$
  $\boldsymbol{z} = \boldsymbol{A}\boldsymbol{y}$
  $\boldsymbol{z} = \boldsymbol{z}/\|\boldsymbol{z}\|$
  $err = |1 - |\boldsymbol{z}^\top \boldsymbol{y}||$
  $\boldsymbol{y} = \boldsymbol{z}$
  $iter = iter + 1$

$i$ și $j$ există o muchie, respectiv 0 dacă acestea nu sunt conectate. Grafurile neorientate au matricea de adiacență simetrică.

Un graf se numește complet dacă fiecare pereche de noduri este conectată printr-o muchie. Când se face referire la un subgraf complet, cel mai adesea se folosește noțiunea de clică. Un graf este bipartrit dacă nodurile acestuia formează două mulțimi disjuncte $U$ și $V$ astfel încât diecare muchie conectează un nod din $U$ cu un nod din $V$.

O serie de proprietăți structurale ale grafurilor pot fi deduse din valorile proprii asociate matricii de adiacență. Menționăm câteva astfel de proprietăți:

- Un graf care are exact două valori proprii este un graf complet.
- Dimensiunea celei mai mari clici conținută într-un graf este cel mult $\lambda_{max}+1$, unde $\lambda_{max}$ reprezintă valoarea proprie dominantă.
- Grafurile complete bipartrite au spectrul simetric în origine, respectiv $\lambda_{min} = -\lambda_{max}$.

# 3 Ghid Python

Pentru a rezolva exercițiile din laboratorul de astăzi, aveți nevoie de biblioteca `pickle`, modulele `numpy.linalg` și `matplotlib.pyplot`. De asemenea, este necesară biblioteca `networkx` și, separat, de modulul `networkx.algorithms.approximation`.

Pentru a încărca datele dintr-un fișier `pickle`, utilizați următoarea secvență:

```
with open('nume_fisier.pickle', 'rb') as f:
x1, x2 = pickle.load(f)
```

Pentru a obține o structură de graf în formatul `networkx` pornind de la o matrice de adiacență se poate utiliza funcția
`networkx.convert_matrix.from_numpy_matrix`.

Pentru a estima dimensiunea celei mai mari clici dintr-un graf, se folosește funcția `networkx.algorithms.approximation.clique.max_clique(G)`.

Un graf poate fi vizualizat utilizând `networkx.draw(G, with_labels=True)`.

## 4 Exerciții

1. Implementați algoritmul Metodei Puterii și testați pe o matrice de dimensiune $n = 6$. Verificați rezultatul comparând soluția cu cea obținută de funcția `numpy.linalg.eig()`.

2. Fișierul `grafuri.pickle` conține matricile de adiacență a trei grafuri de dimensiuni și structuri diferite.

    (a) Obțineți grafurile asociate matricilor de adiacență.

    (b) Vizualizați grafurile.

    (c) Calculați valorile proprii ale grafurilor.

    (d) Verificați dacă cele trei grafuri sunt complete, dacă sunt bipartrite și care e dimensiunea celei mai mari clici și specificați aceste proprietăți pentru fiecare graf.