TP L3, Méthode de la puissance et descente de gradient

28 mars 2025

1 Méthode de la puissance

On cherche à trouver la plus grande valeur propre d'une matrice $M \in \mathbb{R}^{n\times n}$ dont on suppose ici que $\lambda_1 > |\lambda_2|, \cdots, |\lambda_n|$ et le vecteur propre associé $v_1 \in \mathbb{R}^n$. Pour cela on définit la suite suivante

$$u_0 \in \mathbb{R}^n, \quad u_{n+1} = \frac{Mu_n}{\|Mu_n\|}$$

et sous certaines conditions on a

$$\lim_{n\to\infty} u_n = v_1 \quad \text{et} \quad \lim_{n\to\infty} \langle u_n, Mu_n\rangle = \lambda_1 \tag{1}$$

1. Écrire une fonction PuissanceIteree(u0,M,n) qui calcule les n premiers éléments de la suite u_n et les valeurs $\langle u_n, Mu_n\rangle$.
2. Pour chacun des matrices suivantes tester la méthode et discuter si cela fonctionne ou non

$$M_1 = \begin{pmatrix} 15 & -2 & 2 \\ 1 & 10 & -3 \\ -2 & 1 & 0 \end{pmatrix}, \quad M_2 = \begin{pmatrix} 1 & -3 \\ -3 & 1 \end{pmatrix}, \quad M_3 = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

3. On suppose ici que M une matrice diagonale $M = \mathrm{diag}(\lambda_1, \cdots, \lambda_n)$ avec $|\lambda_1| > |\lambda_2|, \cdots, |\lambda_n|$ et $u_0 = (1, \cdots, 1)/\sqrt{n}$. Démontrer que dans ce cas, on a bien la convergence (1).
4. Calculer (sans python puis en utilisant python) les valeurs propres et les vecteurs propres de la matrice M_2, on note $u_n^{(2)}$ la suite construite avec M_2.
5. Montrer que pour $u_0^{(2)} = (1, 0)$,il existe $C > 0$

$$\|u_0^{(2)} - v_1\| \leq C\frac{1}{2^n} \quad \text{et} \quad |\langle u_0^{(2)}, M_2 u_0^{(2)}\rangle - \lambda_1| \leq C\frac{1}{2^n}$$

6. Ilustrer ce résultat en traçant dans un graphe $\log\|u_0^{(2)} - v_1\|$, $\log|\langle u_0^{(2)}, M_2 u_0^{(2)}\rangle - \lambda_1|$.
7. Expliquer le comportement de la suite $u_n^{(3)}$ construite avec M_3.

2 La descente de gradient

La descente de gradient est une idée très naturel. Pour trouver le minimum d'une fonction de $f : \mathbb{R}^n \to \mathbb{R}$, on calcul son gradient (la différentielle de f) et on se déplace dans la direction indiqué par le gradient de tel sorte à diminuer la fonction :

$$u_0 \in \mathbb{R}^n, \quad u_{n+1} = u_n - \eta\nabla f(u_n)$$

où $\eta > 0$ est un paramètre du modèle qui donne la « taille des sauts» à réaliser à chaque fois. On cherche ici à calculer l'inverse

1. Écrire une méthode DescenteGradient(u0,df,η,n) qui calcule les n premiers éléments de la suite. On va utiliser cette méthode pour calculer l'inverse d'une matrice. Pour cela on définit la fonction

$$f(x) = \frac{1}{2}\langle x, Ax\rangle - bx$$

avec $A \in \mathbb{R}^{n\times n}$ une matrice symétrique dont toutes les valeurs propres sont positive et $b \in \mathbb{R}^n$. On a ici $\nabla f(x) = Ax - b$, dans ce cas le minimum est atteint lorsque $Ax - b = 0$ c'est à dire $x = A^{-1}b$.

2. Essayer la descente de gradient sur cette fonction avec

$$A = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

et $\eta = 0,1$ et $u_0 = (0,0)$. On note (i_n) la suite obtenue.

3. Tracer l'évolution de $(i_n) \in \mathbb{R}^2$ dans un graphe en deux dimension. (si il y a le temps ajouter les lignes de niveau de la fonction)

4. Même questions pour la fonction $g : \mathbb{R}^2 \to \mathbb{R}$

$$g(x,y) = ((x+1)^2 + (y+1)^2 - 1)((x-1)^2 + (y-1)^2)$$

   (a) Calculer le gradient dg

   (b) Appliquer une descente de gradient pour g pour $u_0 = (\frac{3}{2}, \frac{7}{2})$ et pour $u_0 = (0,2)$. Qu'observez vous ?

   (c) Faire un graphe en deux dimension et tracer les deux suites et les lignes de niveau pour g.

3 Commandes python

Algèbre linéaire

— import numpy as np
— import numpy.linalg as npl (une bibliothèque pour de algèbre linéaire)
— import scipy.linalg as spl (même chose)
— np.dot(A,B) ou np.dot(A,u) : produit de deux matrices ou d'une matrice et un vecteur.
— np.norm(u) : la norme du vecteur.
— np.transpose(A) : la transposition d'une matrice (ou d'un vecteur)
— npl.eig(A) : diagonalise la matrice
— npl.det(A) : calcul le déterminant
— npl.solve(M,b) : résoud le système linéaire $Mx = b$.

Bout de code pour tracer les lignes de niveau

```
«
i1 = np.linspace(−3,3,100)
i2 = np.linspace(−3 ,3,100)
x1mesh , x2mesh = np.meshgrid(i1,i2)
fmesh = f ( [x1mesh , x2mesh])
plt.contour(...)
»
```