Algèbre linéaire et bilinéaire I
TDTP$_{12}$ – Pivot de Gauss

6 Décembre 2022

On souhaite faire une fonction Python qui calcule le **déterminant d'une matrice carrée d'ordre** n. Pour simplifier, **toutes les matrices sont considérées inversibles**.

Comme exemples, on prendra

$$A_1 = \begin{bmatrix} 3 & 7 & -7 & 4 \\ 1 & -2 & -7 & 2 \\ -3 & 5 & -3 & -3 \\ 6 & 0 & -4 & 4 \end{bmatrix} \in \mathrm{M}_4(\mathbb{R}) \quad \text{et} \quad A_2 = \begin{bmatrix} 3 & 7 & -7 & 4 \\ 0 & 0 & 0 & 1 \\ -3 & 5 & -3 & -3 \\ 6 & 0 & -4 & 4 \end{bmatrix} \in \mathrm{M}_4(\mathbb{R}).$$

Partie 1 : manipulation de matrices

1. Importer la bibliothèque `sympy`. Recopier le code ci-dessous :

```

```

2. Pour les matrices, on utilise le type `Matrix` de la bibliothèque `sympy`. Recopier le code ci-dessous et l'exécuter :

```
A = Matrix([[1,2],[3,4],[5,6]])
display(A)
display(A.shape)
```

Que fait la fonction `display` ? ....

Que fait la méthode `.shape` ? ....

3. Recopier les lignes de code suivantes et les exécuter :

```
B = zeros(2,2)
display(B)
B1 = B
B2 = B.copy()
B1[0,0] = 1
B2[0,0] = 2
display(B1,B2)
display(B)
```

Si on veut faire des opérations à partir de la matrice B sans la changer, faut-il les faire sur la matrice `B1 = B` ou sur la matrice `B2 = B.copy` ? ....

4. Compléter le tableau suivant (les lignes de la matrice A sont appelées L_i) :

Opération élémentaire sur la matrice A	Code correspondant
Dilatation : $L_i \longleftarrow aL_i$ avec $a \in \mathbb{R}$	`A[i,:] =`
Transvection : $L_i \longleftarrow L_i + bL_j$ avec $b \in \mathbb{R}$	`A[i,:] =`
	`A[i,:],A[j,:] = A[j,:],A[i,:]` `A[:,i],A[:,j] = A[:,j],A[:,i]`

Partie 2 : calcul du déterminant d'une matrice triangulaire

5. Soit $M = [m_{i,j}]_{(i,j)\in\{1,\dots,n\}^2} \in \mathrm{M}_n(\mathbb{K})$ une matrice triangulaire. On a

$$\det M = \dots\dots\dots\dots\dots$$

6. Compléter puis implémenter la fonction suivante pour qu'elle calcule le déterminant d'une matrice triangulaire :

```python
def determinantTriangulaire(M):
    d = 1
    for ........................
        ............
    return d
```

Tester cette fonction avec $M = \begin{bmatrix} 2 & 3 & 7 \\ 0 & -4 & 5 \\ 0 & 0 & 9 \end{bmatrix}$ et vérifier avec le résultat de la fonction `det`.

Partie 3 : pivot de Gauss

Le pivot de Gauss consiste à se ramener à une matrice triangulaire à l'aide d'opérations élémentaires sur les lignes.

7. Appliquer ci-dessous et à la main, la première étape de l'algorithme du pivot de Gauss sur la matrice A_2 :

$$A_2 = \begin{bmatrix} 3 & 7 & -7 & 4 \\ 0 & 0 & 0 & 1 \\ -3 & 5 & -3 & -3 \\ 6 & 0 & -4 & 4 \end{bmatrix} \longrightarrow \begin{bmatrix} & & & \\ \\ \\ \\ \end{bmatrix}$$

Quelle opération doit-on faire pour continuer l'algorithme ? ....

8. Créer une fonction `echange(A,k)` qui a pour arguments une matrice $A = [a_{i,j}]$ et un entier k, qui cherche un coefficient $a_{i,k} \neq 0$ pour $i > k$ et qui échange les lignes L_k et L_i de A. Tester la fonction sur une copie de A_2 avec $k = 1$.

9. Écrire la fonction `pivot(A)` en Python qui correspond au pseudo-code suivant :

```
pivot(A)
n ← nombre de lignes/colonnes de A
faire une copie B de A
```

```
d ← 1
pour k allant de 1 à n
        si b_{k,k} = 0
                echange(B,k)
                d ← -d
        pour i allant de k+1 à n
                L_i ← L_i − (b_{i,k}/b_{k,k}) L_k
retourner B et d
```

10. **Application du pivot de Gauss**
    (a) Dans la fonction `pivot`, à quoi correspond `d` ? ....
    En déduire l'expression du déterminant d'une matrice A en fonction des résultats `B` et `d` de `pivot(A)` : ....
    (b) Écrire une fonction `determinant(A)` qui calcule le déterminant d'une matrice A en utilisant les fonctions `pivot` et `determinantTriangulaire`.
    (c) Calculer le déterminant des matrices A_1 et A_2 avec cette fonction. Comparer avec le résultat de la fonction `det`.

Partie 4 : efficacité du pivot de Gauss

Essayons maintenant de comprendre en quoi le pivot de Gauss est « plus efficace » que d'utiliser la formule

$$\forall A = [a_{i,j}] \in \mathrm{M}_n(\mathbb{K}), \quad \det A = \sum_{s \in \mathfrak{S}_n} \varepsilon(s) \prod_{j=1}^{n} a_{s(j),j}. \tag{1}$$

Pour mesure l'*efficacité* d'un algorithme, on compte le nombre de multiplications effectuées.

11. Rappeler le nombre d'éléments du groupe symétrique $\mathfrak{S}_n$: ....
    Combien de multiplications fait-on lors du calcul de $\det A$ par la formule (1) ? ....
    Pour $n = 10$, cela fait ....

Sur Moodle, récupérer le fichier correspondant au TP qui contient une fonction `determinant2(n)` qui choisit au hasard une matrice de $\mathrm{M}_n(\mathbb{R})$, qui calcule son déterminant avec la méthode du pivot de Gauss et renvoie le nombre de multiplications qui ont été effectuées.

12. Éxécuter `determinant2(10)` et comparer ce résultat avec la question précédente.
13. Pour n allant de 2 à 40, tracer le résultat de `determinant2(n)` en fonction de n **en échelle logarithmique** (on pourra utiliser la fonction `loglog` de la bibliothèque `matplotlib.pyplot`). Que remarque-t-on ? ....
14. Tracer sur le même graphe la fonction $n \mapsto n^3$. Que remarque-t-on ? ....
    ....

Ainsi, le nombre de multiplications nécessaires pour calculer un déterminant avec la méthode du pivot de Gauss est de l'ordre de .... ce qui est bien meilleur que ....

Les scientifiques estiment qu'il y a environ 10^{80} atomes (1 suivi de 80 zéros...) dans l'univers observable. Comparer ce nombre avec le nombre de multiplications nécessaires pour calculer le déterminant d'une matrice carrée de taille 60×60, d'abord avec la formule (1) puis avec la fonction `determinant2` ....

Partie 5 : résolution de systèmes linéaires (approfondissement)

On souhaite résoudre des systèmes linéaires de la forme

$$AX = Y \tag{2}$$

d'inconnue $X \in \mathrm{M}_{n,1}(\mathbb{K})$, avec $A \in \mathrm{M}_n(\mathbb{K})$ et $Y \in \mathrm{M}_{n,1}(\mathbb{K})$ données.

15. Écrire une fonction Python qui résout (2) dans le cas où A est triangulaire supérieure.
    Tester votre fonction avec la matrice M de la partie 2 en comparant avec la fonction `linsolve`.
16. Écrire une fonction Python qui résout (2) en utilisant la méthode du pivot de Gauss en se ramenant au cas d'une matrice triangulaire.
    Tester votre fonction avec les matrices A_1 et A_2 en comparant avec la fonction `linsolve`.