L3 maths

Durée : 2h

I. Introduction

Dans cet énoncé, on vous demande de vous appuyer sur les définitions et propriétés qui sont rappelées ci-dessous (et sur la définition du produit de matrices, que vous connaissez). Vous ne devez pas appuyer vos preuves sur d'autres propriétés du déterminant et/ou de l'inverse, sauf si vous les prouvez préalablement.

Pour $n \in \mathbb{N}^*$, on considère les matrices réelles de taille $n \times n$, et on rappelle que le déterminant est la fonction $\begin{cases} \mathcal{M}_n(\mathbb{R}) \to \mathbb{R} \\ (a_{i,j})_{1 \leqslant i,j \leqslant n} \mapsto \sum_{\sigma \in S_n} \epsilon(\sigma) \prod_{i=1}^{n} a_{i,\sigma(i)} \end{cases}$ où S_n désigne l'ensemble des bijections de $[\![1;n]\!]$ dans lui même, et $\epsilon(\sigma) = \prod_{1 \leqslant i < j \leqslant n} \frac{\sigma(j) - \sigma(i)}{j - i}$ désigne la signature de la permutation σ.

On admet que $\forall A \in \mathcal{M}_n(\mathbb{R}), \forall B \in \mathcal{M}_n(\mathbb{R}), \det(AB) = \det(A)\det(B)$.

On admet enfin que pour A et B des matrices de $\mathcal{M}_n(\mathbb{R})$, on a l'équivalence : $AB = \mathbb{I} \iff BA = \mathbb{I}$, où $\mathbb{I}$ désigne la matrice identité de $\mathcal{M}_n(\mathbb{R})$. On dit qu'une matrice $A \in \mathcal{M}_n(\mathbb{R})$ est inversible s'il existe une matrice B telle que $AB = \mathbb{I}$, et dans ce cas la matrice B s'appelle "l'inverse de A", noté A^{-1}.

1. Propriétés élémentaires

1. Soient M et N deux matrices inversibles de $\mathcal{M}_n(\mathbb{R})$. Démontrer que MN et NM sont toutes deux inversibles, et déterminer leur inverse.
2. Montrer que si le déterminant d'une matrice est nul, alors cette matrice n'est pas inversible.
3. On désigne par $\mathbb{R}^n$ l'ensemble des matrices colonnes à n coefficients réels (matrices de taille $n \times 1$). Soit $A \in \mathcal{M}_n(\mathbb{R})$. Montrer que si $\forall X \in \mathbb{R}^n$, $AX = 0$, alors A est la matrice nulle (c'est à dire que tous ses coefficients sont nuls).

2. Matrice triangulaire

4. Lorsque $a_{1,1}$, $a_{2,2}$ et $a_{3,3}$ sont tous non-nuls, déterminer l'ensemble des valeurs de x_1, x_2 et x_3 qui satisfont :

$$\begin{cases} a_{1,1}x_1 + a_{1,2}x_2 + a_{1,3}x_3 = y_1 \\ \qquad\quad a_{2,2}x_2 + a_{2,3}x_3 = y_2 \\ \qquad\qquad\quad a_{3,3}x_3 = y_3 \end{cases}$$

(en fonction des paramètres $a_{1,1}$, $a_{1,2}$, $a_{1,3}$, $a_{2,2}$, $a_{2,3}$, $a_{3,3}$, y_1, y_2 et y_3, qui sont tous réels).

5. Montrer que si on autorise un (ou plusieurs) des paramètres $a_{1,1}$, $a_{2,2}$ et $a_{3,3}$ a être nul, alors il arrive qu'il n'y ait pas de solution, et il arrive aussi qu'il y ait une infinité de solutions.

Dans la suite, on désigne par $A = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ 0 & a_{2,2} & a_{2,3} \\ 0 & 0 & a_{3,3} \end{pmatrix}$ la matrice associée à ce système linéaire.

6. Déterminer la valeur de $\det(A)$.
7. En utilisant les questions précédentes, déterminer à quelle condition la matrice A est inversible, et l'expression de son inverse.
8. Pour $n \in \mathbb{N}^*$ arbitraire, on considère une matrice triangulaire $T = \begin{pmatrix} t_{1,1} & t_{1,2} & t_{1,3} & \dots \\ 0 & t_{2,2} & t_{2,3} & \dots \\ \vdots & \ddots & \ddots & \\ 0 & \dots & 0 & t_{n,n} \end{pmatrix}$
   (a) Déterminer le déterminant de T.
   (b) Déterminer une condition nécessaire et suffisante pour que T soit inversible et montrer que son inverse $U = (u_{i,j})_{1 \leqslant i,j \leqslant n}$ est alors définie par :

$$u_{i,j} = \begin{cases} 0 & \text{si } j < i \\ 1/t_{i,i} & \text{si } j = i \\ -\sum_{k=i+1}^{j} t_{i,k} u_{k,j} / t_{i,i} & \text{si } j > i \end{cases} \tag{1}$$

Indication : une des étapes pourra consister à montrer que sous cette condition, les coefficients de la matrice TU satisfont $(TU)_{i,j} = t_{i,i}u_{i,j} + \sum_{k \in [\![i+1;j]\!]} t_{i,k}u_{k,j}$ lorsque $i \geqslant j$.

3. Implémentation en python

On manipulera les matrices en python comme des listes de listes.
Par exemple `M=[[1,2,3],[2,4,6],[4,8,12]]` désigne la matrice $M = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 4 & 8 & 12 \end{pmatrix}$.

9. Écrire une fonction `triangulaire` en python, telle que `triangulaire(M)` renvoie `True` si la matrice `M` est triangulaire, c'est à dire que `M[i][j]` est nul lorsque $i > j$, que toutes les lignes on la même taille et que cette taille est égale au nombre de lignes (c'est à dire que la matrice doit en fait être "carré et triangulaire"). Si la matrice n'est pas "carré et triangulaire", la fonction doit renvoyer `False`.
10. Compléter la définition ci-après de la fonction `det_triang` qui calcule le déterminant d'une matrice triangulaire (et produit un message d'erreur si la matrice n'est pas triangulaire).

```
def det_triang(M):
    if not triangulaire(M): raise ValueError("Matrice pas triangulaire")
    d=1
    for i in range(len(M)):
        d=......
    return d
```

11. Si `M` est de taille $n \times n$, combien de multiplications et combien d'additions effectue `det_triang(M)` pour calculer le déterminant de `M` ?
12. Écrire une fonction `inv_triang` qui calcule l'inverse d'une matrice si celle-ci est triangulaire et inversible (et renvoie un message d'erreur dans le cas contraire).

II. Pivot de Gauss

L'algorithme du "pivot de gauss" repose sur des opérations élémentaires sur les lignes et les colonnes des matrices.

Chacune de ces opérations élémentaires peut s'écrire comme la multiplication par une matrice précise.

13. Déterminer les étapes effectuées par l'ordinateur et le résultat renvoyé par python (s'il renvoie un résultat) quand on calcule `o1([[1,2],[3,6]],0,1)` et `o2([[1,2],[3,6]],0,1,2)` (où `o1` et `o2` sont définies ci-dessous :)

```
def o1(M,a,b):
    m=[]
    for i in range(len(M)):
        if   i==a: m.append(M[b])
        elif i==b: m.append(M[a])
        else  :    m.append(M[i])
    return m

def o2(M,a,b,f):
    m=[]
    for i in range(len(M)):
        if   i==b:
            l=[]
            for j in range(len(M)):
                l.append(M[b][j]-f*M[a][j])
        else:
            l=M[i]
        m.append(l)
    return m
```

Soit `M` $\in \mathcal{M}_n(\mathbb{R})$, `a` et `b` deux entiers de $[\![0;n[\![$ distincts et `f` $\in \mathbb{R}$, on pose `A=o1(M,a,b)` et `B=o2(M,a,b,f)`.

14. Énoncer l'opération sur les lignes (ou les colonnes) qui donne `A` en fonction de `M`. Énoncez aussi la multiplication matricielle qui correspond à cette opération élémentaire.
    Même question pour l'opération qui donne `B` en fonction de `M`.

Lorsque `M` est inversible, on pose `C` = $\mathtt{A}^{-1}$ et `D` = $\mathtt{B}^{-1}$

15. Exprimer la relation matricielle entre `C` et `M`, ainsi que la relation entre `D` et `M`.
16. Réciproquement, écrire deux fonction en python : `inv1` qui calcule $\mathtt{M}^{-1}$ à partir de `C` et `inv2` qui calcule $\mathtt{M}^{-1}$ à partir de `D`. En d'autres termes, `inv1(C,a,b)` renvoie l'inverse de la matrice `M` telle que `C` soit l'inverse de `o1(M,a,b)`, et `inv2(D,a,b)` renvoie l'inverse de la matrice `M` telle que `D` soit l'inverse de `o2(M,a,b,f)`.

La méthode du "pivot de Gauss" consiste à utiliser ces opérations sur les lignes et les colonnes pour calculer l'inverse d'une matrice `M`, comme ci-dessous :

```
def inverse(M):
   for j in range(len(M)):
      if M[j][j]==0:
         for i in range(j+1,len(M)):
            if M[i][j]!=0: return inv1(inverse(o1(M,i,j)),i,j)
         raise ValueError("Matrice pas inversible")
      for i in range(j+1,len(M)):
         if M[i][j]!=0:
            return inv2(inverse(o2(M,j,i,M[i][j]/M[j][j])),j,i,M[i][j]/M[j][j])
   # si on arrive à la fin de ces boucles, c'est que chaque M[i][i] est !=0
   #                                 et que pour i>j, chaque M[i][j] est =0
   return inv_triang(M)
```

17. Quand on exécute `inverse([[0,-1,4],[1,3,-8],[-2,3,12]])`, quels appels récursif de la fonction `inverse` ont successivement lieu ?
18. Écrire de même une fonction qui calcule le déterminant d'une matrice à partir d'opérations sur les lignes et les colonnes.
19. Quand on calcule ainsi le déterminant d'une matrice de taille $n \times n$, combien y a-t-il au maximum d'additions et de multiplications ? En comparaison combien y en aurait-il si on utilisait directement la définition $\sum_{\sigma \in S_n} \epsilon(\sigma) \prod_{i=1}^{n} a_{i,\sigma(i)}$? En supposant que le temps de calcul soit donné par le nombre de multiplications et d'additions, ce Pivot de Gauss est-il plus rapide que l'utilisation directe de la définition du déterminant ?