Chapitre 16

Calcul matriciel

Sommaire

16.1 Espaces de matrices

16.1.1 Matrices à n lignes et p colonnes

Définition 16.1.1 (matrices à n lignes, à p colonnes, et à coefficients dans $\mathbb{K}$)
Une *matrice* A de type (n,p) (avec n,p dans $\mathbb{N}^*$) est une application de $\{1,\dots,n\}\times\{1,\dots,p\}$ dans $\mathbb{K}$.
On note souvent $a_{i,j}$ (« coefficient d'indice i,j ») l'image du couple (i,j) par l'application A.
On écrit alors $A=(a_{i,j})_{\substack{i=1..n\\ j=1..p}}$, ou plus simplement $A=(a_{ij})$.

On note $\mathcal{M}_{n,p}(\mathbb{K})$ l'ensemble des matrices de type (n,p) à coefficients dans $\mathbb{K}$.

Pour décrire un élément A de $\mathcal{M}_{n,p}(\mathbb{K})$, on dispose les coefficients dans un tableau à n lignes et p colonnes, le coefficient $a_{i,j}$ venant se placer à l'intersection de la i-ème ligne et de la j-ème colonne.

Par exemple, la matrice A de $\mathcal{M}_{2,3}(\mathbb{K})$ définie par : $\begin{cases} a_{1,1}=5, & a_{1,2}=3, & a_{1,3}=2\\ a_{2,1}=7, & a_{2,2}=4, & a_{2,3}=1 \end{cases}$ est $A=\begin{pmatrix}5&3&2\\7&4&1\end{pmatrix}$.

Finalement, c'est ce tableau lui-même qu'on appelle une matrice. On note souvent $[A]_{i,j}=a_{i,j}$.

On dit donc qu'un élément A de $\mathcal{M}_{n,p}(\mathbb{K})$ est une « matrice à n lignes et à p colonnes ».

Voici comment on peut former cette matrice dans `Python`, après avoir chargé le module `numpy` :

```
>>> import numpy as np                    # charge le module numpy et le renomme np
>>> A = np.array([[5,3,2],[7,4,1]]); A    # forme et affiche une matrice d'entiers
array([[5, 3, 2],
       [7, 4, 1]])
```

On peut convertir les coefficients entiers de la matrice A précédente au format `'float'` ou `'complex'` :

```
>>> A.astype(float)
array([[ 5.,  3.,  2.],
       [ 7.,  4.,  1.]])
```

```
>>> A.astype(complex)
array([[ 5.+0.j,  3.+0.j,  2.+0.j],
       [ 7.+0.j,  4.+0.j,  1.+0.j]])
```

Attention !! En `Python`, les indices de ligne et de colonne commencent à 0.

```
>>> A[1,0]                # la ligne d'indice 1 est la deuxième ligne
7                         # la colonne d'indice 0 est la première colonne
```

Important Dans tous les exemples de ce chapitre, on supposera que le module `numpy` a été importé.

Avec `Python`, il est facile de créer une matrice dont le terme général répond à une formule donnée.
On peut par exemple créer la liste des listes de ces valeurs (en utilisant une syntaxe « en compréhension ») et convertir le résultat en un « array » :

```
>>> A = np.array([[10*i+j for j in range(10)] for i in range(5)])
>>> print(A)     # la matrice de terme général 10i+j, avec 0<=i<=4, et 0<=j<=9
[[ 0  1  2  3  4  5  6  7  8  9]
 [10 11 12 13 14 15 16 17 18 19]
 [20 21 22 23 24 25 26 27 28 29]
 [30 31 32 33 34 35 36 37 38 39]
 [40 41 42 43 44 45 46 47 48 49]]
```

Voici une autre méthode pour former la matrice précédente.
Ici les coefficients sont des flottants : si on veut des entiers, il suffit de rajouter l'option `dtype='int'`.

```
>>> B = np.fromfunction(lambda x,y: 10*x+y, (5,10))
>>> print(B)
[[  0.   1.   2.   3.   4.   5.   6.   7.   8.   9.]
 [ 10.  11.  12.  13.  14.  15.  16.  17.  18.  19.]
 [ 20.  21.  22.  23.  24.  25.  26.  27.  28.  29.]
 [ 30.  31.  32.  33.  34.  35.  36.  37.  38.  39.]
 [ 40.  41.  42.  43.  44.  45.  46.  47.  48.  49.]]
```

On peut aussi créer des matrices aléatoires :

```
>>> A = np.random.rand(4,5)   # quatre lignes, cinq colonnes, coeffs dans [0,1]
>>> print(A)
[[ 0.4672486   0.42452394  0.55652697  0.50122566  0.45039697]
 [ 0.10245706  0.39642783  0.18395313  0.43191611  0.59490124]
 [ 0.97133547  0.06577487  0.33274442  0.39404957  0.34601463]
 [ 0.60141568  0.01085386  0.69123586  0.33062893  0.68429281]]
```

Matrices constantes, matrice nulle

Une *matrice constante* $A = (a_{i,j})$ de $\mathcal{M}_{n,p}(\mathbb{K})$ est définie par $a_{i,j} = \lambda$ pour tous i, j.
En particulier, la *matrice nulle* $A = (a_{i,j})$ de $\mathcal{M}_{n,p}(\mathbb{K})$ est définie par $a_{i,j} = 0$ pour tous i, j.
Voici quelques façons de créer des matrices constantes avec `Python` :

```
>>> np.zeros([3, 4])
array([[ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.]])
```

```
>>> np.ones([3, 4])
array([[ 1.,  1.,  1.,  1.],
       [ 1.,  1.,  1.,  1.],
       [ 1.,  1.,  1.,  1.]])
```

Ici la somme (ou le produit) d'une matrice et d'une constante s'effectue terme à terme :

```
>>> np.zeros([3, 4]) + 8
array([[ 8.,  8.,  8.,  8.],
       [ 8.,  8.,  8.,  8.],
       [ 8.,  8.,  8.,  8.]])
```

```
>>> np.ones([3, 4]) * 8
array([[ 8.,  8.,  8.,  8.],
       [ 8.,  8.,  8.,  8.],
       [ 8.,  8.,  8.,  8.]])
```

« Matrices-ligne » et « matrices-colonne »

Les éléments de $\mathcal{M}_{1,p}(\mathbb{K})$ sont dits *matrices-ligne*.
Ceux de $\mathcal{M}_{p,1}(\mathbb{K})$ sont dits *matrices-colonne*.

On peut identifier (en restant prudent) :
– le vecteur $(a_1, a_2, \ldots, a_p)$ de $\mathbb{K}^p$
– la matrice-ligne $(a_1\ a_2\ \ldots\ a_p)$ de $\mathcal{M}_{1,p}(\mathbb{K})$
– la matrice-colonne $\begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{pmatrix}$ de $\mathcal{M}_{p,1}(\mathbb{K})$

```
>>> # on définit un vecteur
>>> np.array([1,3,5])
array([1, 3, 5])

>>> # on définit une matrice-ligne
>>> np.array([[1,3,5]])
array([[1, 3, 5]])

>>> # on définit une matrice-colonne
>>> np.array([[1],[3],[5]])
array([[1],
       [3],
       [5]])
```

16.1.2 Structure d'espace vectoriel de $\mathcal{M}_{n,p}(\mathbb{K})$

Définition 16.1.2 (somme de deux matrices, produit par un scalaire)
Soit $A = (a_{ij})$ et $B = (b_{ij})$ deux matrices de $\mathcal{M}_{n,p}(\mathbb{K})$, et λ un scalaire.
On définit les matrices $C = A + B$ et $D = \lambda A$ dans $\mathcal{M}_{n,p}(\mathbb{K})$ de la manière suivante :
Pour tous indices i et j : $c_{ij} = a_{ij} + b_{ij}$ et $d_{ij} = \lambda a_{ij}$.

Proposition 16.1.1 (structure d'espace vectoriel de $\mathscr{M}_{n,p}(\mathbb{K})$)
L'ensemble $\mathcal{M}_{n,p}(\mathbb{K})$ est un espace vectoriel sur $\mathbb{K}$, de dimension np.

L'élément neutre est la matrice nulle, notée habituellement 0.
L'opposé de la matrice $A = (a_{ij})$ est la matrice notée $-A$, de terme général $-a_{ij}$.

Base canonique

On fixe n et p dans $\mathbb{N}^*$, et on se place dans l'espace vectoriel $\mathcal{M}_{n,p}(\mathbb{K})$.

Soit i dans $\{1, \dots, n\}$ et j dans $\{1, \dots, p\}$.

On note $E_{i,j}$ la matrice dont tous les coefficients sont nuls, sauf celui d'indice (i, j) qui vaut 1.

La famille des np matrices $(E_{i,j})$, pour $1 \leqslant i \leqslant n$ et $1 \leqslant j \leqslant p$ est une base de $\mathcal{M}_{n,p}(\mathbb{K})$.

On l'appelle la « base canonique » de $\mathcal{M}_{n,p}(\mathbb{K})$.
Pour toute matrice $M = (a_{ij})$ de $\mathcal{M}_{n,p}(\mathbb{K})$, on a $M = \sum_{i=1}^{n} \sum_{j=1}^{p} a_{ij} E_{ij}$.

Voici par exemple les six matrices de la base canonique de $M_{2,3}(\mathbb{K})$:

$$E_{1,1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad E_{1,2} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad E_{1,3} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$
$$E_{2,1} = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \quad E_{2,2} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \quad E_{2,3} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Pour toute matrice $A = \begin{pmatrix} a & b & c \\ d & e & f \end{pmatrix}$ de $M_{2,3}(\mathbb{K})$, on a effectivement (et de façon unique) :

$$A = a \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} + b \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} + c \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} + d \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} + e \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} + f \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Voici une fonction `Python` qui renvoie une matrice de la base canonique de $\mathcal{M}_{n,p}(\mathbb{K})$ (attention encore une fois à la numérotation des lignes et des colonnes qui commence à 0 dans les tableaux `Python`) :

```
>>> def E(n, p, i, j, format='int'):
        a = np.zeros([n,p],format); a[i-1,j-1] = 1; return a
```

On calcule ici deux fois $E_{2,4}$ dans $\mathcal{M}_{3,5}(\mathbb{K})$ (d'abord en format `'int'`, puis en format `'float'`) :

```
>>> E(3, 5, 2, 4)
array([[0, 0, 0, 0, 0],
       [0, 0, 0, 1, 0],
       [0, 0, 0, 0, 0]])
```

```
>>> E(3, 5, 2, 4, 'float')
array([[ 0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  1.,  0.],
       [ 0.,  0.,  0.,  0.,  0.]])
```

Avec `Python`, l'addition des matrices et le produit par des scalaires s'effectuent de façon très naturelle :

```
>>> A = np.random.randint(10, size=(3,4))  # coeffs aléatoires entiers dans [0,10[
>>> B = np.random.randint(10, size=(3,4))  # une autre matrice aléatoire
```

```
>>> print(A)
[[1 3 4 2]
 [0 5 5 1]
 [2 1 9 7]]
```

```
>>> print(B)
[[3 5 4 7]
 [1 2 3 1]
 [3 8 8 0]]
```

```
>>> print(A+B)
[[ 4  8  8  9]
 [ 1  7  8  2]
 [ 5  9 17  7]]
```

```
>>> print(10*A+B)
[[13 35 44 27]
 [ 1 52 53 11]
 [23 18 98 70]
```

16.1.3 Matrices carrées

Définition 16.1.3 (matrices carrées d'ordre n)
On appelle *matrice carrée* d'ordre n (ou de taille n) toute matrice de type (n, n).
On note $\mathcal{M}_n(\mathbb{K})$ (plutôt que $\mathcal{M}_{n,n}(\mathbb{K})$) l'ensemble des matrices carrées d'ordre n à coefficients dans $\mathbb{K}$.

Diagonale d'une matrice carrée

Soit $A = (a_{i,j})$ une matrice carrée d'ordre n.
Les coefficients $a_{i,i}$ (appelés *coefficients diagonaux*) de A forment la *diagonale* de A.
Les coefficients $a_{i,j}$ tels que $j < i$ sont donc *en dessous* de cette diagonale.
Les coefficients $a_{i,j}$ tels que $i < j$ sont *au dessus* de celle-ci.

On forme ici une matrice A de $\mathcal{M}_5(\mathbb{R})$, à coefficients entiers choisis aléatoirement dans $[0, 9[$.
On voit que la fonction `diagonal` du module `numpy` permet de récupérer la diagonale de A.
Remarque : il y a une deuxième diagonale dans une matrice carrée, mais seule celle qui débute « en haut en gauche » et se termine « en bas à droite » est importante (on le verra plus loin).

```
>>> A = np.random.randint(10, size=(4,4))
>>> print(A)
[[6 3 4 3]
 [4 1 3 9]
 [7 0 9 6]
 [4 8 6 4]]
>>> A.diagonal()  # ou encore np.diag(A)
array([6, 1, 9, 4])
```

Matrices diagonales

Une matrice $A = (a_{i,j})$ de $\mathcal{M}_n(\mathbb{K})$ est dite *diagonale* si on a $a_{i,j} = 0$ pour tous indices distincts i et j.
Ainsi A est diagonale si les seuls coefficients non nuls *éventuels* de A sont ses coefficients diagonaux.
La matrice nulle de $\mathcal{M}_n(\mathbb{K})$ est donc un cas (très) particulier de matrice diagonale.

Matrice identité

La *matrice identité d'ordre n* est la matrice diagonale de $\mathcal{M}_n(\mathbb{K})$ de coefficients diagonaux égaux à 1.
Cette matrice est notée I_n.
On remarque que pour tous indices i et j, on a : $[\mathrm{I}_n]_{i,j} = \delta_{i,j}$ (notation de Kronecker).

```
>>> np.diag([1,4,7,2])
array([[1, 0, 0, 0],
       [0, 4, 0, 0],
       [0, 0, 7, 0],
       [0, 0, 0, 2]])
```

```
>>> np.eye(4)
array([[ 1., 0., 0., 0.],
       [ 0., 1., 0., 0.],
       [ 0., 0., 1., 0.],
       [ 0., 0., 0., 1.]])
```

```
>>> np.eye(4,dtype='int')
array([[1, 0, 0, 0],
       [0, 1, 0, 0],
       [0, 0, 1, 0],
       [0, 0, 0, 1]])
```

Matrices scalaires

Les matrices carrées de la forme $A = \lambda I_n$, c'est-à-dire les matrices diagonales dont tous les coefficients diagonaux sont égaux, sont dites *matrices scalaires*.

```
>>> (5+2j)*np.eye(3)
array([[ 5.+2.j,  0.+0.j,  0.+0.j],
       [ 0.+0.j,  5.+2.j,  0.+0.j],
       [ 0.+0.j,  0.+0.j,  5.+2.j]])
```

Matrices triangulaires ou strictement triangulaires

Soit $A = (a_{i,j})$ une matrice de $\mathcal{M}_n(\mathbb{K})$.

On dit que A est *triangulaire supérieure* si les coefficients situés *sous* la diagonale sont nuls.

On dit que A est *triangulaire inférieure* si les coefficients situés *au-dessus* de la diagonale sont nuls.

Ainsi A est triangulaire inférieure si $(i < j \Rightarrow a_{i,j} = 0)$, et *triangulaire supérieure* si $(j < i \Rightarrow a_{i,j} = 0)$.

La matrice A est dite strictement triangulaire si elle est triangulaire et si de plus sa diagonale est nulle.

Avec `Python`, on forme ici une matrice aléatoire A, carrée d'ordre 4, à coefficients dans $[\![1, 9]\!]$.

Les fonctions `triu` et `tril` (avec `u` pour « up » et `l` pour « low ») permettent d'extraire de A une matrice triangulaire supérieure ou (éventuellement strictement, avec un argument optionnel).

```
>>> A=np.random.randint(1,10,size=(4,4))
>>> print(A)
[[4 6 2 7]
 [9 7 2 7]
 [3 4 5 1]
 [9 8 7 7]]
```

```
>>> np.triu(A)
array([[4, 6, 2, 7],
       [0, 7, 2, 7],
       [0, 0, 5, 1],
       [0, 0, 0, 7]])
```

Matrice triangulaire supérieure

```
>>> np.triu(A,1)
array([[0, 6, 2, 7],
       [0, 0, 2, 7],
       [0, 0, 0, 1],
       [0, 0, 0, 0]])
```

Matrice strictement triangulaire supérieure

```
>>> np.tril(A)
array([[4, 0, 0, 0],
       [9, 7, 0, 0],
       [3, 4, 5, 0],
       [9, 8, 7, 7]])
```

Matrice triangulaire inférieure

```
>>> np.tril(A,-1)
array([[0, 0, 0, 0],
       [9, 0, 0, 0],
       [3, 4, 0, 0],
       [9, 8, 7, 0]])
```

Matrice strictement triangulaire inférieure

Toujours en `Python`, la fonction `tri` permet de construire des matrices (strictement) triangulaires inférieures dont les coefficients sous-diagonaux valent 1 :

```
>>> np.tri(4,4,dtype='int')
array([[1, 0, 0, 0],
       [1, 1, 0, 0],
       [1, 1, 1, 0],
       [1, 1, 1, 1]])
```

```
>>> np.tri(4,4,-1,dtype='int')
array([[0, 0, 0, 0],
       [1, 0, 0, 0],
       [1, 1, 0, 0],
       [1, 1, 1, 0]])
```

16.2 Produit matriciel

16.2.1 Produit des matrices

Définition 16.2.1 (définition du produit de deux matrices)

Soit $A = (a_{ik})$ dans $\mathcal{M}_{n,p}(\mathbb{K})$ et $B = (b_{kj})$ dans $\mathcal{M}_{p,q}(\mathbb{K})$.

On définit $M = AB$, dans $\mathcal{M}_{n,q}(\mathbb{K})$, par : $\forall\, i \in \{1, \dots, n\},\ \forall\, j \in \{1, \dots, q\},\ m_{ij} = \sum_{k=1}^{p} a_{ik}\, b_{kj}$.

Interprétation et visualisation

Le terme de la i-ième ligne et de la j-ième colonne de $M = AB$ est donc obtenu en sommant les produits des termes de même rang dans la i-ième ligne L_i de A et dans la j-ième colonne C_j de B, selon le schéma ci-après (on a encadré les coefficients de A et de B utiles au calcul du coefficient c_{ij}).

$$\mathrm{L}_i \to \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1k} & \dots & a_{1p} \\ \vdots & \vdots & & \vdots & & \vdots \\ \boxed{a_{i1}} & \boxed{a_{i2}} & \dots & \boxed{a_{ik}} & \dots & \boxed{a_{ip}} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nk} & \dots & a_{np} \end{pmatrix} \begin{pmatrix} b_{11} & \dots & \boxed{b_{1j}} & \dots & b_{1q} \\ b_{21} & \dots & \boxed{b_{2j}} & \dots & b_{2q} \\ \vdots & & \vdots & & \vdots \\ b_{k1} & & \boxed{b_{kj}} & & b_{kq} \\ \vdots & & \vdots & & \vdots \\ b_{p1} & \dots & \boxed{b_{pj}} & \dots & b_{pq} \end{pmatrix} = \begin{pmatrix} m_{11} & \dots & m_{1j} & \dots & m_{1q} \\ \vdots & & \vdots & & \vdots \\ m_{i1} & \dots & \boxed{m_{ij}} & \dots & m_{iq} \\ \vdots & & \vdots & & \vdots \\ m_{n1} & \dots & m_{nj} & \dots & m_{nq} \end{pmatrix}$$

$\mathrm{C}_j \downarrow$

Pour former le coefficient $m_{i,j}$, on met en regard la ligne L_i de A et la colonne C_j de B, et on somme les produits deux à deux des a_{ik} et b_{kj} de même position k, avec $1 \leqslant k \leqslant p$

On verra plus loin les raisons pour lesquelles on choisit de définir de cette façon le produit matriciel.

Condition impérative de compatibilité

Le produit AB n'est possible que si le nombre de colonnes de A est égal au nombre de lignes de B.
On obtient alors une matrice ayant autant de lignes que A, et autant de colonnes que B.

En résumé : $\boxed{\big(\text{matrice de type } (n,p)\big) \times \big(\text{matrice de type } (p,q)\big) = \big(\text{matrice de type } (n,q)\big)}$

Avec `Python` on forme ici deux matrices aléatoires, A dans $\mathcal{M}_{4,3}(\mathbb{R})$, l'autre dans $\mathcal{M}_{3,5}(\mathbb{R})$:

```
>>> A=np.random.randint(10,size=(4,3))
>>> print(A)
[[2 2 7]
 [6 3 8]
 [4 3 3]
 [7 7 0]]
```

```
>>> B=np.random.randint(10,size=(3,5))
>>> print(B)
[[6 8 1 2 6]
 [5 6 2 8 7]
 [5 6 8 4 9]]
```

On voit qu'on peut calculer AB et le résultat est une matrice de type $(4, 5)$.
Mais le produit BA est impossible à calculer et conduirait à l'erreur de format suivante :
`ValueError: objects are not aligned`

```
>>> print(A.dot(B))    # ou np.dot(A,B)
[[ 57  70  62  48  89]
 [ 91 114  76  68 129]
 [ 54  68  34  44  72]
 [ 77  98  21  70  91]]
```

En `Python`, attention à l'opération `*` qui désigne en fait le produit « terme à terme » des tableaux. Dans l'exemple ci-après, on effectue le produit terme à terme de deux matrices A et B de taille 4×3.
Le résultat est encore une matrice 4×3 dont le terme général est $c_{i,j} = a_{i,j}b_{i,j}$. Ce produit (qui peut avoir son utilité) n'a rien à voir avec le produit matriciel dont nous parlons dans ce chapitre !

```
>>> print(A)
[[9 2 4]
 [4 3 7]
 [9 3 5]
 [6 1 9]]
```

```
>>> print(B)
[[3 2 2]
 [1 8 1]
 [6 8 6]
 [4 5 2]]
```

```
>>> print(A*B)
[[27  4  8]
 [ 4 24  7]
 [54 24 30]
 [24  5 18]]
```

Attention, ce n'est pas le « vrai » produit matriciel !

16.2.2 La non-commutativité du produit

Les produits AB et BA ne sont simultanément possibles que si A est de type (n,p) et B de type (p,n).

La matrice AB est alors carrée d'ordre n, tandis que BA est carrée d'ordre p.

Si $n \neq p$, les matrices AB et BA, de formats différents, sont évidemment distinctes !

Si A et B sont toutes deux carrées d'ordre n, alors AB et BA sont encore carrées d'ordre n, mais là encore on a en général $AB \neq BA$. Dans le cas contraire (très rare), on dit que les matrices carrées A et B *commutent* (on en reparle plus loin).

Exemple avec `Python` : on utilise ici la fonction `arange` (qui forme le vecteur des éléments d'une suite arithmétique, ici la suite $0, 1, \dots, 5$) et la fonction `reshape` (qui redéfinit la taille du tableau, du moment que le nombre de coefficients est le même).

```
>>> A = np.arange(6).reshape(3,2); print(A)
[[0 1]
 [2 3]
 [4 5]]
```

```
>>> B = np.arange(6).reshape(2,3)
>>> print(B)
[[0 1 2]
 [3 4 5]]
```

On a donc formé une matrice A de type $(3,2)$, et une matrice B de type $(2,3)$.

Les deux produits AB et BA sont possibles, mais AB est carrée d'ordre 3 et BA est carrée d'ordre 2 :

```
>>> print(A.dot(B))
[[ 3  4  5]
 [ 9 14 19]
 [15 24 33]]
```

```
>>> print(B.dot(A))
[[10 13]
 [28 40]]
>>>
```

Autre exemple, on forme ici deux matrices A et B carrées d'ordre 3 :

```
>>> A = np.random.rand(3,3); print(A)
[[ 0.75596295  0.96396382  0.45077587]
 [ 0.67323071  0.04498405  0.88736757]
 [ 0.86576073  0.03023288  0.42663428]]
```

```
>>> B = np.random.rand(3,3); print(B)
[[ 0.68788001  0.35191549  0.98365263]
 [ 0.38822915  0.80922831  0.75737939]
 [ 0.92474908  0.11545089  0.3735433 ]]
```

On constate effectivement que les produits AB et BA sont différents (c'est le cas général !) :

```
>>> print(np.dot(A,B))
[[ 1.31110522  1.09814436  1.64207559]
 [ 1.30115841  0.37577006  1.02776536]
 [ 1.00180644  0.37839522  1.03387196]]
```

```
>>> print(np.dot(B,A))
[[ 1.60853994  0.70866068  1.04201804]
 [ 1.49399354  0.43353898  1.2162113 ]
 [ 1.10020024  0.90791139  0.67866832]]
```

16.2.3 Propriétés du produit matriciel

Proposition 16.2.1 (bilinéarité du produit matriciel)
Soit A, B dans $\mathcal{M}_{n,p}(\mathbb{K})$, soit C, D dans $\mathcal{M}_{p,q}(\mathbb{K})$, et soit λ, μ deux scalaires.
Alors on a les égalités $A(\lambda C + \mu D) = \lambda AC + \mu AD$ et $(\lambda A + \mu B)C = \lambda AC + \mu BC$.

Cette propriété peut s'énoncer de la façon suivante :

- Si on fixe A dans $\mathcal{M}_{n,p}(\mathbb{K})$, l'application $M \mapsto AM$ est linéaire de $\mathcal{M}_{p,q}(\mathbb{K})$ dans $\mathcal{M}_{n,q}(\mathbb{K})$.
- Si on fixe A dans $\mathcal{M}_{p,q}(\mathbb{K})$, l'application $M \mapsto MA$ est linéaire de $\mathcal{M}_{n,p}(\mathbb{K})$ dans $\mathcal{M}_{n,q}(\mathbb{K})$.
- On peut résumer : l'application $(M,N) \mapsto MN$ est *bilinéaire* de $\mathcal{M}_{n,p}(\mathbb{K}) \times \mathcal{M}_{p,q}(\mathbb{K})$ dans $\mathcal{M}_{n,q}(\mathbb{K})$.

Proposition 16.2.2 (associativité du produit matriciel)
Soit A dans $\mathcal{M}_{n,p}(\mathbb{K})$, soit B dans $\mathcal{M}_{p,q}(\mathbb{K})$, et soit C dans $\mathcal{M}_{q,r}(\mathbb{K})$.
Alors, on a l'égalité $A(BC) = (AB)C$ dans $\mathcal{M}_{n,r}(\mathbb{K})$.

On définit ici trois matrices A, B, C, de coefficients aléatoires dans $\{-1, 0, 1\}$.

```
>>> A = np.random.randint(-1,2,size=(3,4))
>>> B = np.random.randint(-1,2,size=(4,3))
>>> C = np.random.randint(-1,2,size=(3,2))
```

On a donc choisi A dans $\mathcal{M}_{3,4}(\mathbb{R})$,
B dans $\mathcal{M}_{4,3}(\mathbb{R})$, et C dans $\mathcal{M}_{3,2}(\mathbb{R})$.
Voici les trois matrices A, B, C.

```
>>> print(A)
[[ 0  1  1  0]
 [ 0 -1  1  0]
 [ 0  1  1 -1]]
>>>
```

```
>>> print(B)
[[ 0  1  0]
 [ 0  1  1]
 [ 1  0  0]
 [ 1 -1  0]]
```

```
>>> print(C)
[[ 0  1]
 [ 0  1]
 [-1 -1]]
>>>
```

On affiche alors AB (qui est dans $\mathcal{M}_{3,3}(\mathbb{R})$), puis $(AB)C$ (qui est dans $\mathcal{M}_{3,2}(\mathbb{R})$).
Ensuite on affiche alors BC (qui est dans $\mathcal{M}_{4,2}(\mathbb{R})$), puis $A(BC)$ (qui est dans $\mathcal{M}_{3,2}(\mathbb{R})$).
On constate que les deux matrices $A(BC)$ et $(AB)C$ sont égales dans $\mathcal{M}_{3,2}(\mathbb{R})$:

```
>>> print(A.dot(B))
[[ 1  1  1]
 [ 1 -1 -1]
 [ 0  2  1]]
>>> print((A.dot(B)).dot(C))
[[-1  1]
 [ 1  1]
 [-1  1]]
>>>
```

```
>>> print(B.dot(C))
[[ 0  1]
 [-1  0]
 [ 0  1]
 [ 0  0]]
>>> print(A.dot(B.dot(C)))
[[-1  1]
 [ 1  1]
 [-1  1]]
```

16.2.4 Une justification du produit matriciel

Une situation particulière mais importante

L'application qui à $u = (x_1, x_2, \dots, x_p)$ associe $U = \begin{pmatrix} x_1 \\ \vdots \\ x_p \end{pmatrix}$ est un isomorphisme de $\mathbb{K}^p$ sur $\mathcal{M}_{p,1}(\mathbb{K})$.
Cet isomorphisme permet d'identifier u et U.

Soit A un élément de $\mathcal{M}_{n,p}(\mathbb{K})$. L'application $U \mapsto V = AU$ est linéaire de $\mathcal{M}_{p,1}(\mathbb{K})$ dans $\mathcal{M}_{n,1}(\mathbb{K})$.
À un isomorphisme près, il s'agit donc d'une application linéaire de $\mathbb{K}^p$ dans $\mathbb{K}^n$.

Considérons par exemple la matrice $A = \begin{pmatrix} 2 & 1 & 0 & 3 \\ 0 & 4 & 1 & 5 \\ 1 & 6 & 2 & 0 \end{pmatrix}$, élément de $\mathcal{M}_{3,4}(\mathbb{R})$.

Si $U = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$ est dans $\mathcal{M}_{4,1}(\mathbb{R})$ (disons dans $\mathbb{R}^4$), $V = AU = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$ est dans $\mathcal{M}_{3,1}(\mathbb{R})$ (disons dans $\mathbb{R}^3$).

Le produit matriciel donne : $\begin{pmatrix} 2 & 1 & 0 & 3 \\ 0 & 4 & 1 & 5 \\ 1 & 6 & 2 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 2x_1 + x_2 + 3x_4 \\ 4x_2 + x_3 + 5x_4 \\ x_1 + 6x_2 + 2x_3 \end{pmatrix}$ donc $\begin{cases} y_1 = 2x_1 + x_2 + 3x_4 \\ y_2 = 4x_2 + x_3 + 5x_4 \\ y_3 = x_1 + 6x_2 + 2x_3 \end{cases}$

À isomorphisme près, on a donc défini $f : u = (x_1, x_2, x_3, x_4) \mapsto v = (y_1, y_2, y_3)$ linéaire de $\mathbb{R}^4$ dans $\mathbb{R}^3$.

On dira que la matrice A est *canoniquement associée* à l'application linéaire f.

Dans le cas général : on peut légitimement identifier une application linéaire de $\mathbb{K}^p$ dans $\mathbb{K}^n$ avec une matrice A de $\mathcal{M}_{n,p}(\mathbb{K})$. Si on note U, V les matrices colonnes représentant respectivement u dans $\mathbb{K}^p$ et v dans $\mathbb{K}^n$, cela nous permet d'identifier les égalités $v = f(u)$ et $V = AU$.

Identification d'une matrice-ligne et d'une forme linéaire

Un élément L de $\mathcal{M}_{1,n}(\mathbb{K})$ (une matrice-ligne de largeur n) s'identifie à une forme linéaire sur $\mathbb{K}^n$.

Par exemple, considérons la matrice ligne $A = \begin{pmatrix} a & b & c & d \end{pmatrix}$.

Pour toute matrice colonne $U = \begin{pmatrix} x \\ y \\ z \\ t \end{pmatrix}$, on a : $AX = \begin{pmatrix} a & b & c & d \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ t \end{pmatrix} = ax + by + cz + dt$.

La matrice ligne A s'identifie donc à la forme linéaire f définie sur $\mathbb{K}^4$ par $f(x, y, z, t) = ax+by+cz+dt$.

Justification du produit matriciel

Considérons deux applications linéaires $f : \mathbb{K}^n \to \mathbb{K}^p$ et $g : \mathbb{K}^p \to \mathbb{K}^q$.

L'application $h = g \circ f$ est linéaire de $\mathbb{K}^n$ dans $\mathbb{K}^q$.

– à l'application f est canoniquement associée une matrice A de $\mathcal{M}_{p,n}(\mathbb{K})$.
– à l'application g est canoniquement associée une matrice B de $\mathcal{M}_{q,p}(\mathbb{K})$.
– à l'application h est canoniquement associée une matrice C de $\mathcal{M}_{q,n}(\mathbb{K})$.

Pour tous vecteurs u dans $\mathbb{K}^n$, v dans $\mathbb{K}^p$ et w dans $\mathbb{K}^q$, on a l'équivalence : $w = h(u) \Leftrightarrow w = g(f(u))$.
Matriciellement, $w = h(u)$ s'écrit $W = CU$, et $w = g(f(u))$ s'écrit $W = B(AU)$.

Précisément, la définition du produit matriciel est choisie pour qu'on ait l'égalité $BA = C$.

Autrement dit, si on attache aux matrices A et B deux applications linéaires f et g, le produit BA est la matrice attachée à l'application linéaire $g \circ f$.

On résume ça en imaginant que le diagramme : $\mathbb{K}^n \xrightarrow[A]{f} \mathbb{K}^p \xrightarrow[B]{g} \mathbb{K}^q$ se « contracte » en $\mathbb{K}^n \xrightarrow[BA]{g\circ f} \mathbb{K}^q$.

Prenons un exemple simple, en restant en dimension 2 :

$\begin{cases} f : (x, y) \to (x + y, 2x - y) \\ g : (x, y) \to (x + 3y, x - y) \end{cases}$ sont attachées à $A = \begin{pmatrix} 1 & 1 \\ 2 & -1 \end{pmatrix}$ et $B = \begin{pmatrix} 1 & 3 \\ 1 & -1 \end{pmatrix}$. On a $BA = \begin{pmatrix} 7 & -2 \\ -1 & 2 \end{pmatrix}$.

On vérifie que $g\circ f : (x, y) \to (7x-2y, -x+2y)$ est attachée à $C = \begin{pmatrix} 7 & -2 \\ -1 & 2 \end{pmatrix}$ et qu'on a bien $C = BA$.

16.2.5 Produits, lignes et colonnes

Dans le calcul du produit AB de deux matrices, on met en rapport les lignes de A et les colonnes de B.

Il est alors commode de se représenter une matrice quelconque sous la forme :

– de la superposition de n matrices-ligne $\mathrm{L}_1, \mathrm{L}_2, \ldots, \mathrm{L}_n$ de même largeur.
– de la juxtaposition de p matrices-colonne $\mathrm{C}_1, \mathrm{C}_2, \ldots, \mathrm{C}_p$ de même hauteur.

On pourra alors noter symboliquement $\left(\begin{array}{c} \mathrm{L}_1 \\ \hline \mathrm{L}_2 \\ \hline \vdots \\ \hline \mathrm{L}_n \end{array}\right)$ et $\left(\begin{array}{c|c|c|c} \mathrm{C}_1 & \mathrm{C}_2 & \cdots & \mathrm{C}_p \end{array}\right)$

Le terme d'indice (i,j) de $\left(\begin{array}{c}\mathrm{L}_1\\\hline\mathrm{L}_2\\\hline\vdots\\\hline\mathrm{L}_n\end{array}\right)\left(\mathrm{C}_1\,\middle|\,\mathrm{C}_2\,\middle|\,\cdots\,\middle|\,\mathrm{C}_p\right)$ est le « produit scalaire » de L_i par C_j.

Avec `Python`, on forme ici les matrices A dans $\mathcal{M}_{4,5}(\mathbb{R})$ et B dans $\mathcal{M}_{5,4}(\mathbb{K})$.

```
>>> A=np.random.randint(-5,5,size=(4,5))
>>> print(A)
[[ 1 -4 -1 -3  4]
 [-1  1  4  4 -5]
 [ 3 -3 -1 -1  1]
 [-1 -1  2  0  1]]
>>>
```

```
>>> B=np.random.randint(-5,5,size=(5,4))
>>> print(B)
[[-1 -4 -3 -3]
 [-1  1  1  0]
 [-2  1 -2 -5]
 [ 2 -5  2  2]
 [-1 -5 -3 -3]]
```

On extrait la ligne d'indice 3 de A (attention encore une fois à la numérotation `Python` qui commence à 0 : il s'agit donc ici, avec nos notations, de la ligne L_4).

On extrait également la colonne d'indice 2 de B (donc c'est la colonne C_3 !).

On forme ensuite le produit scalaire de cette ligne et de cette colonne.

Le résultat, égal à -5, est effectivement le coefficient de position $(3,2)$ de $C=AB$ (donc c'est $c_{4,3}$!) :

```
>>> A[3,:]   # quatrième ligne !!!
array([-1, -1,  2,  0,  1])
>>> B[:,2]   # troisième colonne !!!
array([-3,  1, -2,  2, -3])
>>> np.vdot(A[3,:],B[:,2])
-5
```

```
>>> A.dot(B)  # ou np.dot(A,B)
array([[ -5, -14, -23, -16],
       [  5,  14,  19,   6],
       [ -1, -16, -15,  -9],
       [ -3,   0,  -5, -10]])
>>>
```

Si A est un élément de $\mathcal{M}_{n,p}(\mathbb{K})$ et si les x_k sont scalaires, on peut écrire :

$$\underbrace{\left(\mathrm{C}_1\,\middle|\,\mathrm{C}_2\,\middle|\,\cdots\,\middle|\,\mathrm{C}_p\right)}_{A}\begin{pmatrix}x_1\\x_2\\\vdots\\x_p\end{pmatrix}=\sum_{j=1}^{p}x_j\mathrm{C}_j\quad\text{et}\quad\begin{pmatrix}x_1&x_2&\cdots&x_n\end{pmatrix}\underbrace{\begin{pmatrix}\mathrm{L}_1\\\hline\mathrm{L}_2\\\hline\vdots\\\hline\mathrm{L}_n\end{pmatrix}}_{A}=\sum_{i=1}^{n}x_iL_i$$

De même, les produits matriciels MA et AN peuvent s'écrire :

$$M\underbrace{\left(\mathrm{C}_1\,\middle|\,\mathrm{C}_2\,\middle|\,\cdots\,\middle|\,\mathrm{C}_p\right)}_{A}=\left(M\mathrm{C}_1\,\middle|\,M\mathrm{C}_2\,\middle|\,\cdots\,\middle|\,M\mathrm{C}_p\right)\quad\text{et}\quad\underbrace{\begin{pmatrix}\mathrm{L}_1\\\hline\mathrm{L}_2\\\hline\vdots\\\hline\mathrm{L}_n\end{pmatrix}}_{A}N=\begin{pmatrix}\mathrm{L}_1N\\\hline\mathrm{L}_2N\\\hline\vdots\\\hline\mathrm{L}_nN\end{pmatrix}$$

Ces idées peuvent être mises à profit pour optimiser le calcul d'un produit AB de deux matrices :

– si la matrice A est « simple », on privilégiera un calcul de AB par lignes.

– si la matrice B est « simple », on privilégiera un calcul de AB par colonnes.

Posons par exemple $A=\begin{pmatrix}a&b&c&d\\a'&b'&c'&d'\\a''&b''&c''&d''\end{pmatrix}$ et $B=\begin{pmatrix}1&2&0\\0&1&-1\\1&0&0\\1&0&1\end{pmatrix}$.

On a ici intérêt à effectuer le calcul de AB colonne par colonne :

$$AB = \begin{pmatrix} a & b & c & d \\ a' & b' & c' & d' \\ a'' & b'' & c'' & d'' \end{pmatrix} \begin{pmatrix} 1 & 2 & 0 \\ 0 & 1 & -1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix} = \left(\begin{array}{c|c|c|c} \mathrm{C}_1 & \mathrm{C}_2 & \mathrm{C}_3 & \mathrm{C}_4 \end{array}\right) \begin{pmatrix} 1 & 2 & 0 \\ 0 & 1 & -1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

$$= \left(\begin{array}{c|c|c} \mathrm{C}_1 + \mathrm{C}_3 + \mathrm{C}_4 & 2\mathrm{C}_1 + \mathrm{C}_2 & -\mathrm{C}_2 + \mathrm{C}_4 \end{array}\right) = \left(\begin{array}{c|c|c} a+c+d & 2a+b & -b+d \\ a'+c'+d' & 2a'+b' & -b'+d' \\ a''+c''+d' & 2a''+b'' & -b''+d'' \end{array}\right)$$

De même, on a intérêt à effectuer le produit BA ligne par ligne :

$$AB = \begin{pmatrix} 1 & 2 & 0 \\ 0 & 1 & -1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} a & b & c & d \\ a' & b' & c' & d' \\ a'' & b'' & c'' & d'' \end{pmatrix} = \begin{pmatrix} 1 & 2 & 0 \\ 0 & 1 & -1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix} \left(\begin{array}{c} \mathrm{L}_1 \\ \hline \mathrm{L}_2 \\ \hline \mathrm{L}_3 \end{array}\right)$$

$$= \left(\begin{array}{c} \mathrm{L}_1 + 2\mathrm{L}_2 \\ \hline \mathrm{L}_2 - \mathrm{L}_3 \\ \hline \mathrm{L}_1 \\ \hline \mathrm{L}_1 + \mathrm{L}_3 \end{array}\right) = \left(\begin{array}{cccc} a+2a' & b+2b' & c+2c' & d+2d' \\ \hline a'-a'' & b'-b'' & c'-c'' & d'-d'' \\ \hline a & b & c & d \\ \hline a+a'' & b+b'' & c+c'' & d+d'' \end{array}\right)$$

16.2.6 Produits de matrices de la base canonique

Proposition 16.2.3 (produits de matrices de la base canonique)
On se donne les ensembles $\mathcal{M}_{n,p}(\mathbb{K})$, $\mathcal{M}_{p,q}(\mathbb{K})$ et $\mathcal{M}_{n,q}(\mathbb{K})$, avec n, p, q dans $\mathbb{N}^*$.
Soit $E_{i,j}$ (resp. $E'_{j,k}$, $E''_{i,k}$) les matrices de la base canonique de $\mathcal{M}_{n,p}(\mathbb{K})$ (resp. $\mathcal{M}_{p,q}(\mathbb{K})$, $\mathcal{M}_{n,q}(\mathbb{K})$).
Si les indices j et j' sont distincts, alors le produit $E_{i,j}\,E'_{j',k}$ est égal à la matrice nulle de $\mathcal{M}_{n,q}(\mathbb{K})$.
Si $j' = j$, alors le produit $E_{i,j}\,E'_{j,k}$ est égal à $E''_{i,k}$.

Plus généralement, quand on effectue le produit $A\,E_{i,j}$ le résultat est une matrice nulle à l'exception de sa j-ème colonne qui est la copie de la i-ème colonne de A.
De même $E_{i,j}A$ est nulle à l'exception de sa i-ème ligne qui est la copie de la j-ème ligne de A.
Voici un exemple, où on réutilise la fonction `E` de la sous-section 16.1.2 :

```
>>> print(A)
[[-3  2  0  3  1]
 [ 0  1 -5 -4  4]
 [ 3  2  2  1 -3]
 [-4  3 -1  1 -3]
 [-4 -1  3  0  4]]
```

Une matrice de $\mathcal{M}_{5,5}(\mathbb{R})$

```
>>> print(A.dot(E(5,5,2,4)))
[[ 0  0  0  2  0]
 [ 0  0  0  1  0]
 [ 0  0  0  2  0]
 [ 0  0  0  3  0]
 [ 0  0  0 -1  0]]
```

Produit par $E_{2,4}$ à droite
C_2 a été copiée en C_4

```
>>> print(E(5,5,2,4).dot(A))
[[ 0  0  0  0  0]
 [-4  3 -1  1 -3]
 [ 0  0  0  0  0]
 [ 0  0  0  0  0]
 [ 0  0  0  0  0]]
```

Produit par $E_{2,4}$ à gauche
L_4 a été copiée en L_2

16.3 Calculs sur les matrices carrées

16.3.1 L'anneau $(\mathcal{M}_n(\mathbb{K}), +, \times)$

Proposition 16.3.1 (l'anneau $\mathcal{M}_n(\mathbb{K})$)
Muni de la somme et du produit des matrices, l'ensemble $\mathcal{M}_n(\mathbb{K})$ possède une structure d'anneau.
L'anneau $(\mathcal{M}_n(\mathbb{K}), +, \times)$ est non commutatif dès que $n \geqslant 2$.

- Le neutre additif de l'anneau $\mathcal{M}_n(\mathbb{K})$ est la matrice nulle, notée 0 (ou éventuellement 0_n).
  Le neutre multiplicatif de l'anneau $\mathcal{M}_n(\mathbb{K})$ est la matrice identité I_n.
- Si $n = 1$, on identifie une matrice $A = (a)$ avec l'unique scalaire qu'elle contient.
  Dans ce cas, $\mathcal{M}_1(\mathbb{K})$ s'identifie donc au corps (commutatif) $\mathbb{K}$.
- Il est facile de former des matrices qui ne commutent pas dans $\mathcal{M}_2(\mathbb{K})$.
  Par exemple si $A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ et $B = \begin{pmatrix} 0 & 2 \\ 1 & 0 \end{pmatrix}$, alors $AB = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}$ et $BA = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$.
  Si $n \geqslant 2$, l'anneau $\mathcal{M}_n(\mathbb{K})$ est non commutatif, comme on le voit en « étendant » l'exemple précédent :
  si $A = \begin{pmatrix} 0 & 1 & 0 & \cdots \\ 1 & 0 & 0 & \cdots \\ 0 & 0 & 0 & \ldots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$ et $B = \begin{pmatrix} 0 & 2 & 0 & \cdots \\ 1 & 0 & 0 & \cdots \\ 0 & 0 & 0 & \ldots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$ alors $AB = \begin{pmatrix} 1 & 0 & 0 & \cdots \\ 0 & 2 & 0 & \cdots \\ 0 & 0 & 0 & \ldots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$ et $BA = \begin{pmatrix} 2 & 0 & 0 & \cdots \\ 0 & 1 & 0 & \cdots \\ 0 & 0 & 0 & \ldots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$

Proposition 16.3.2 (produits de matrices diagonales ou triangulaires)
Soit A et B deux matrices de $\mathcal{M}_n(\mathbb{K})$.
On suppose que A et B sont diagonales (resp. triangulaires supérieures, triangulaires inférieures).
Alors $C = AB$ est encore diagonale (resp. triangulaire supérieure, triangulaire inférieure).
De plus les coefficients diagonaux c_{ii} de C vérifient $c_{ii} = a_{ii}b_{ii}$.

Propriétés diverses

- Les matrices triangulaires supérieures forment un sous-espace de $\mathcal{M}_n(\mathbb{R})$, de dimension $\dfrac{n(n+1)}{2}$.
  Le résultat précédent nous dit que c'est aussi un sous-anneau de $\mathcal{M}_n(\mathbb{R})$.
  Il en est bien sûr de même pour les matrices triangulaires inférieures.
  Les matrices diagonales forment un sous-espace de $\mathcal{M}_n(\mathbb{R})$ (et un sous-anneau) de dimension n.
- La proposition précédente s'étend (récurrence facile) à un nombre quelconque de matrices carrées (toutes diagonales, ou toutes triangulaires supérieures, ou toutes triangulaires inférieures).
  Cela s'applique donc aux puissances A^p (avec p dans $\mathbb{N}^*$) d'une matrice diagonale ou triangulaire.
- Si deux matrices A et B sont triangulaires (disons supérieurement) et si l'une d'elle est strictement triangulaire, alors leur produit AB est strictement triangulaire.
- On ne peut rien dire du produit de deux matrices triangulaires, l'une inférieure et l'autre supérieure.
  Par exemple, si $A = \begin{pmatrix} 1 & 2 \\ 0 & 3 \end{pmatrix}$ et $B = \begin{pmatrix} 4 & 0 \\ 5 & 6 \end{pmatrix}$, alors $AB = \begin{pmatrix} 14 & 12 \\ 15 & 18 \end{pmatrix}$ et $BA = \begin{pmatrix} 4 & 8 \\ 5 & 28 \end{pmatrix}$.

En Python, on voit ici une matrice A de $\mathcal{M}_4(\mathbb{R})$, ainsi que son carré A^2 et son cube A^3 : si un calcul est nécessaire pour les coefficients « sur-diagonaux », les coefficients diagonaux sont immédiats :

```
>>> print(A)
[[ 2  3 -2  4]
 [ 0 -1  1  0]
 [ 0  0  1  5]
 [ 0  0  0  3]]
```

```
>>> A2 = A.dot(A); print(A2)
[[ 4  3 -3 10]
 [ 0  1  0  5]
 [ 0  0  1 20]
 [ 0  0  0  9]]
```

```
>>> A3 = A.dot(A2); print(A3)
[[ 8  9 -8 31]
 [ 0 -1  1 15]
 [ 0  0  1 65]
 [ 0  0  0 27]]
```

Une matrice triangulaire très particulière

On note ici T_n la matrice strictement triangulaire dont tous les coefficients sont nuls, sauf ceux situés immédiatement au-dessus de la diagonale et qui valent 1.

Voici une fonction `Python` pour définir une telle matrice :

```
def T(n):
    return np.array([[1 if j==i+1 else 0 for j in range(n)] for i in range(n)])
```

Voici par exemple la matrice $J = T_4$, ainsi que J^2 et J^3. On voit très bien comment la « sur-diagonale » de coefficients égaux à 1 « remonte » quand l'exposant augmente (la prochaine étape est $J^4 = 0$).

Remarque : en mode Python interactif, on désigne par `_` le résultat du calcul précédent.

```
>>> J = T(4); J
array([[0, 1, 0, 0],
       [0, 0, 1, 0],
       [0, 0, 0, 1],
       [0, 0, 0, 0]])
```

La matrice $J = T_4$

```
>>> J.dot(_)
array([[0, 0, 1, 0],
       [0, 0, 0, 1],
       [0, 0, 0, 0],
       [0, 0, 0, 0]])
```

Le carré J^2 de J

```
>>> J.dot(_)
array([[0, 0, 0, 1],
       [0, 0, 0, 0],
       [0, 0, 0, 0],
       [0, 0, 0, 0]])
```

Le cube J^3 de J

Les matrices triangulaires $J = T_n$ reviennent souvent.

Le passage de A au produit JA revient à « décaler vers le haut » les lignes de A.

Le passage de A au produit AJ revient à « décaler vers la droite » les parallèles à la diagonale de A.

Pour l'exemple ci-dessous, on a fabriqué la matrice A de la manière suivante :

```
A = np.arange(16).reshape(4,4)
```

On multiplie ensuite A sur sa gauche, puis sur sa droite, par la matrice J de l'exemple précédent :

```
>>> A
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12],
       [13, 14, 15, 16]])
```

La matrice A

```
>>> J.dot(A)
array([[ 5,  6,  7,  8],
       [ 9, 10, 11, 12],
       [13, 14, 15, 16],
       [ 0,  0,  0,  0]])
```

Le produit par J à gauche remonte les lignes vers le haut

```
>>> A.dot(J)
array([[ 0,  1,  2,  3],
       [ 0,  5,  6,  7],
       [ 0,  9, 10, 11],
       [ 0, 13, 14, 15]])
```

Le produit par J à droite décale les colonnes vers la droite

16.3.2 La formule du binôme

Le résultat suivant est un cas particulier d'une situation rencontrée dans 11.3.3 :

Proposition 16.3.3 (formule du binôme pour deux matrices carrées qui commutent)
Soit A et B deux matrices de $\mathcal{M}_n(\mathbb{K})$. On suppose que A et B commutent.
Alors, pour tout entier naturel p, on a : $(A+B)^p = \sum_{k=0}^{p} \binom{p}{k} A^k B^{p-k} = \sum_{k=0}^{p} \binom{p}{k} A^{p-k} B^k$.

Remarques

– L'hypothèse selon laquelle $AB = BA$ est essentielle. Si elle n'est pas réalisée, alors le développement de $(A+B)^p$ est totalement différent (et beaucoup plus compliqué).

Par exemple, si $AB \neq BA$, on a : $\begin{cases} (A+B)^2 = A^2 + AB + BA + B^2 \\ (A+B)^3 = A^3 + A^2B + ABA + AB^2 + BA^2 + BAB + B^2A + B^3 \end{cases}$

Toujours si $AB \neq BA$ on a : $(A+B)(A-B) = A^2 - AB + BA - B^2$, donc $(A+B)(A-B) \neq A^2 - B^2$.

– Les matrices scalaires $\lambda \mathrm{I}_n$ commutent avec toutes les matrices de $\mathcal{M}_n(\mathbb{K})$.

Pour tout A de $\mathcal{M}_n(\mathbb{K})$, on peut donc écrire : $(A+\lambda I_n)^p = \sum_{k=0}^{p} \binom{p}{k} \lambda^{p-k} A^k = \sum_{k=0}^{p} \binom{p}{k} \lambda^k A^{p-k}$.

– Si $A_1, \dots, A_m$ commutent deux à deux dans $\mathcal{M}_n(\mathbb{K})$, alors $\left(\sum_{k=1}^{m} A_k\right)^2 = \sum_{k=1}^{m} A_k^2 + 2 \sum_{1 \leqslant i < j \leqslant m} A_i A_j$

– Si A et B commutent et si $p \geqslant 1$, alors $(AB)^p = A^p B^p$, sinon $(AB)^p = ABAB \cdots AB$ (p fois).

16.3.3 Matrices nilpotentes ou diviseurs de zéro

Si $J = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$, on a $J^2 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$, $J^3 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$, et $J^p = 0$ pour tout $p \geqslant 4$.

On exprime cette propriété en disant que J est nilpotente, et que son indice de nilpotence est 4.

Définition 16.3.1 (matrices nilpotentes)
Soit A une matrice de $\mathcal{M}_n(\mathbb{K})$.
On dit que la matrice A est *nilpotente* s'il existe un entier $p \geqslant 1$ tel que $A^p = 0$.
Le plus petit entier $p \geqslant 1$ tel que $A^p = 0$ est appelé *indice de nilpotence* de A.

Avec cette définition, on convient que la matrice nulle est nilpotente d'indice 1.

Proposition 16.3.4 (condition suffisante pour qu'une matrice soit nilpotente)
Toute matrice strictement triangulaire est nilpotente (la réciproque est fausse).

L'exemple de $A = \begin{pmatrix} 1 & 1 \\ -1 & -1 \end{pmatrix}$, qui vérifie $A^2 = 0$, montre effectivement que la réciproque est fausse.

En revanche : si A est triangulaire, elle est nilpotente si et seulement si elle est *strictement* triangulaire.

Proposition 16.3.5 (condition nécessaire sur l'indice de nilpotence)
Soit A une matrice nilpotente de $\mathcal{M}_n(\mathbb{K})$.
Alors l'indice de nilpotence de A est inférieur ou égal à n. On est donc certain que $A^n = 0$.

Par la contraposée : si A est dans $\mathcal{M}_n(\mathbb{K})$ et si $A^n \neq 0$, il est certain que A n'est pas nilpotente !

Proposition 16.3.6 (somme ou produit de deux matrices nilpotentes)
Soit A et B deux matrices nilpotentes de $\mathcal{M}_n(\mathbb{K})$.
Si A et B commutent, alors AB et $A+B$ sont nilpotentes.

L'hypothèse selon laquelle $AB = BA$ est essentielle.
Par exemple si $A = \begin{pmatrix} 1 & 1 \\ -1 & -1 \end{pmatrix}$ et $B = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$ alors $AB = \begin{pmatrix} 0 & 1 \\ 0 & -1 \end{pmatrix}$ et $BA = \begin{pmatrix} -1 & -1 \\ 0 & 0 \end{pmatrix}$.
On trouve $(AB)^2 = \begin{pmatrix} 0 & 1 \\ 0 & -1 \end{pmatrix} = AB$ et $A+B = \begin{pmatrix} 1 & 0 \\ -1 & -1 \end{pmatrix}$ (donc ni AB ni $A+B$ ne sont nilpotentes).

Définition 16.3.2 (matrices diviseurs de zéro)
Soit A une matrice de $\mathcal{M}_n(\mathbb{K})$. On dit que A est un *diviseur de zéro* s'il existe une matrice B non nulle dans $\mathcal{M}_n(\mathbb{K})$ telle que $AB = 0$ ou $BA = 0$.

Avec la définition précédente, on convient que la matrice nulle est un diviseur de 0.
Par exemple, si $A = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$, $B = \begin{pmatrix} 1 & -1 \\ 1 & -1 \end{pmatrix}$ et $C = \begin{pmatrix} 1 & 1 \\ -1 & -1 \end{pmatrix}$, alors $BA = AC = 0$.
La matrice A est donc un diviseur de zéro (l'une des deux égalités $BA = 0$ ou $AC = 0$ suffisait).
En revanche, on notera que $BA = 0$ mais que $AB = \begin{pmatrix} 2 & -2 \\ 2 & -2 \end{pmatrix}$ n'est pas la matrice nulle.
On montrera que s'il existe $B \neq 0$ telle que $AB = 0$ alors, il existe $C \neq 0$ telle que $CA = 0$ (et réciproquement). Il n'y a donc pas de matrice qui soit diviseur de zéro « d'un seul coté ».

Proposition 16.3.7 (nilpotent ⇒ diviseur de zéro)
Soit A une matrice de $\mathcal{M}_n(\mathbb{K})$.
Si A est nilpotente, alors A est un diviseur de zéro. La réciproque est fausse.

Bien sûr si A est nilpotente d'indice $p \geqslant 1$, il suffit d'écrire $0 = A^p = AA^{p-1}$ (avec $A^{p-1} \neq 0$) pour réaliser que A est un diviseur de zéro.
La matrice $A = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$ est un diviseur de zéro mais elle n'est pas nilpotente.

Non simplifiabilité pour le produit, pour une matrice diviseur de zéro

Si A est un diviseur de zéro, les implications $\begin{cases} AM = AN \Rightarrow M = N \\ MA = NA \Rightarrow M = N \end{cases}$ sont fausses.

En effet, si $AB = 0$ avec $B \neq 0$, et si $N = M + B$, alors $AM = AN$ (bien que M et N soient distinctes).

On exprime cela en disant qu'une matrice diviseur de zéro *n'est pas simplifiable* pour le produit.

16.3.4 Calcul des puissances d'une matrice carrée

On présente ici quelques méthodes usuelles de calcul de A^p avec A dans $\mathcal{M}_n(\mathbb{K})$ et p dans $\mathbb{N}$.

▷ Utilisation de la formule du binôme

Pour calculer A^p, il est parfois possible d'écrire $A = B + C$, et d'utiliser la formule du binôme.
La condition indispensable est bien sûr l'égalité $BC = CB$.
Il faut que le calcul des puissances de B et C soit beaucoup plus facile que celui de A !

Un cas fréquent est celui où $B = \lambda I_n$ (matrice scalaire) et où C est nilpotente.

Si $C^m = 0$, alors, pour tout p : $A^p = \sum_{k=0}^{p} \binom{p}{k} C^k B^{p-k} = \sum_{k=0}^{p} \binom{p}{k} \lambda^{p-k} C^k = \sum_{k=0}^{m-1} \binom{p}{k} \lambda^{p-k} C^k$.

Par exemple si $C^3 = 0$, alors pour $p \geqslant 3$ on a : $(I+C)^p = I + pC + \dfrac{p(p-1)}{2} C^2 + \dfrac{p(p-1)(p-2)}{6} C^3$.
NB : cette formule reste vraie si $0 \leqslant p \leqslant 2$ (car les termes « excédentaires » sont nuls).

Premier exemple

Soit $A = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$. La matrice A s'écrit $A = I_4 + T$ avec $T = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$.

On a facilement : $T^2 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$, $T^3 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$ puis $T^k = 0$ si $k \geqslant 4$.

On en déduit : $A^p = I + pT + \dfrac{p(p-1)}{2} T^2 + \dfrac{p(p-1)(p-2)}{6} T^3 = \begin{pmatrix} 1 & p & \frac{p(p-1)}{2} & \frac{p(p-1)(p-2)}{6} \\ 0 & 1 & p & \frac{p(p-1)}{2} \\ 0 & 0 & 1 & p \\ 0 & 0 & 0 & 1 \end{pmatrix}$.

Deuxième exemple

Voici un autre d'exemple d'utilisation de la formule du binôme.

La matrice $A = \begin{pmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 2 \end{pmatrix}$ s'écrit $A = I_4 + J$ avec $J = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$.

On trouve facilement $J^2 = 4J$, donc $J^k = 4^{k-1} J$ pour $k \geqslant 1$ (mais attention, c'est faux si $k = 0$).

On obtient finalement (bien vérifier toutes les étapes du calcul) :

$$A^p = \sum_{k=0}^{p} \binom{p}{k} T^k = I_4 + \frac{1}{4}\Big(\sum_{k=1}^{p} \binom{p}{k} 4^k \Big) J = I_4 + \frac{5^p - 1}{4} J = \frac{1}{4} \begin{pmatrix} 5^p + 3 & 5^p - 1 & 5^p - 1 & 5^p - 1 \\ 5^p - 1 & 5^p + 3 & 5^p - 1 & 5^p - 1 \\ 5^p - 1 & 5^p - 1 & 5^p + 3 & 5^p - 1 \\ 5^p - 1 & 5^p - 1 & 5^p - 1 & 5^p + 3 \end{pmatrix}$$

▷ Utilisation d'une récurrence

Il arrive que les coefficients des premières puissances de A satisfassent à une formule simple.

Il reste à établir si cette formule est vraie pour toutes les puissances de A, à l'aide d'une récurrence.

Par exemple, si $R(\theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$ (rotation d'angle θ), on trouve $R(\theta)R(\varphi) = R(\theta+\varphi)$.

Une récurrence évidente donne alors $R(\theta)^p = R(p\,\theta) = \begin{pmatrix} \cos(p\,\theta) & -\sin(p\,\theta) \\ \sin(p\,\theta) & \cos(p\,\theta) \end{pmatrix}$ pour tout p.

▷ Utilisation d'un polynôme annulateur

Supposons par exemple qu'une matrice A vérifie $A^3 - 2A^2 - A + 2I_n = 0 \quad (1)$
On dit que $P = \mathrm{X}^3 - 2\mathrm{X}^2 - \mathrm{X} + 2$ est un *polynôme annulateur* de A, l'égalité (1) s'écrivant $P(A) = 0$.
Mais le polynôme P se factorise en $P = (\mathrm{X}-1)(\mathrm{X}+1)(\mathrm{X}-2)$.

Pour calculer A^p, on écrit la division euclidienne $\mathrm{X}^p = Q(\mathrm{X})P(\mathrm{X}) + a_p\mathrm{X}^2 + b_p\mathrm{X} + c_p \quad (2)$.

Dans cette division, on remplace X par 1, -1 et 2 et on trouve $\begin{cases} a_p + b_p + c_p = 1 \\ a_p - b_p + c_p = (-1)^p \\ 4a_p + 2b_p + c_p = 2^p \end{cases}$

On trouve alors : $a_p = -\dfrac{1}{2} + \dfrac{(-1)^p}{6} + \dfrac{2^p}{3}$, $b_p = \dfrac{1}{2} - \dfrac{(-1)^p}{2}$ et $c_p = 1 + \dfrac{(-1)^p}{3} - \dfrac{2^p}{3}$.

Dans la division (2), on remplace X par A et on trouve : $A^p = Q(A)P(A) + a_pA^2 + b_pA + c_pI_n$.

Du fait que $P(A) = 0$, cela se simplifie en $A^p = a_pA^2 + b_pA + c_pI_n$.

On a ainsi obtenu A^p en fonction de A^2, de A et de la matrice identité.

▷ Puissances entières avec `Python`

La fonction (rudimentaire) suivante calcule les puissances A^k d'une matrice A, avec $k \geqslant 1$.

```
def powmat(A, k):
    B = A
    for i in range(1, k): B = B.dot(A)
    return B
```

```
>>> A = np.tri(4,dtype='int')
>>> A
array([[1, 0, 0, 0],
       [1, 1, 0, 0],
       [1, 1, 1, 0],
       [1, 1, 1, 1]])
```

```
>>> powmat(A,2)
array([[1, 0, 0, 0],
       [2, 1, 0, 0],
       [3, 2, 1, 0],
       [4, 3, 2, 1]])
>>>
```

```
>>> powmat(A,10)
array([[  1,  0,  0, 0],
       [ 10,  1,  0, 0],
       [ 55, 10,  1, 0],
       [220, 55, 10, 1]])
>>>
```

16.3.5 Matrices inversibles

Définition 16.3.3 (le groupe des matrices carrées inversibles d'ordre n)
On note $\mathrm{GL}_n(\mathbb{K})$ l'ensemble des matrices de $\mathcal{M}_n(\mathbb{K})$ qui sont inversibles pour le produit.
L'ensemble $\mathrm{GL}_n(\mathbb{K})$ est un groupe (non commutatif si $n \geqslant 2$) pour le produit des matrices.
En fait $\mathrm{GL}_n(\mathbb{K})$ est le groupe des inversibles de l'anneau $\mathcal{M}_n(\mathbb{K})$ (voir 11.3.1).
On l'appelle souvent « le groupe linéaire d'indice n ».

Rappel : si A et B sont inversibles, alors AB est inversible et $(AB)^{-1} = B^{-1}A^{-1}$ (l'ordre est important !).

Rappelons que dire que A est inversible dans $\mathcal{M}_n(\mathbb{K})$, c'est dire qu'il existe B telle que $AB = BA = I_n$.
Le produit de $\mathcal{M}_n(\mathbb{K})$ n'étant pas commutatif, il semble indispensable de vérifier $AB = I_n$ et $BA = I_n$.
La proposition 16.3.9 va nous montrer que l'une de ces deux égalités implique l'autre.

Une formule pour l'inverse d'une matrice carrée d'ordre 2

Il n'est pas toujours facile de voir au premier coup d'œil si une matrice carrée A est inversible ou non.
En revanche, c'est facile pour les matrices carrées d'ordre 2 :

Proposition 16.3.8 (inversibilité d'une matrice carrée d'ordre 2)
Soit $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ *dans* $\mathcal{M}_2(\mathbb{K})$. *Soit* $\Delta = ad - bc$ *le « déterminant » de* A.
Alors A *est inversible si et seulement si* Δ *est non nul. Dans ce cas,* $A^{-1} = \dfrac{1}{\Delta}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$.

Par exemple, avec $A = \begin{pmatrix} 1 & 4 \\ 2 & 10 \end{pmatrix}$, on a $\Delta = 2$, donc $A^{-1} = \dfrac{1}{2}\begin{pmatrix} 10 & -4 \\ -2 & 1 \end{pmatrix}$.
On reprend cet exemple avec `Python`, le temps de mettre en garde contre une erreur facile à commettre !
L'expression Python $1/A$ désigne en effet non pas l'inverse A^{-1} de A, mais la matrice dont les coefficients sont les inverses, terme à terme, de ceux de A.
Pour calculer l'inverse de A avec `Python`, il faut utiliser la syntaxe `np.linalg.inv(A)`.

```
>>> A = np.array([[1,4],[2,10]])
>>> print(A)
[[ 1  4]
 [ 2 10]]
```

```
>>> print(1/A)
[[ 1.    0.25]
 [ 0.5   0.1 ]]
```

```
>>> B = np.linalg.inv(A)
>>> print(B)
[[ 5.  -2. ]
 [-1.   0.5]]
```

Avec la matrice B ainsi obtenue, on vérifie qu'on a bien les égalités $AB = BA = I_2$.
La matrice B et les produits AB et BA sont obtenus en mode `'float'`.

```
>>> A.dot(B)
array([[ 1., 0.],
       [ 0., 1.]])
```

```
>>> B.dot(A)
array([[ 1., 0.],
       [ 0., 1.]])
```

Inverser une matrice à coefficients rationnels avec `Python`

Si une matrice carrée inversible A est à coefficients entiers (ou plus généralement rationnels), alors son inverse est à coefficients rationnels.
Avec `Python`, le calcul de l'inverse de A provoque un passage en mode `float`.
Il est alors assez délicat de retrouver l'expression exacte (sous forme de rationnels) de la matrice A^{-1}.

C'est tout de même possible, en utilisant la fonction suivante dont le rôle est de convertir les coefficients flottants d'une matrice en leur équivalent rationnel (avec une précision donnée) :

```
>>> def tofrac(M, d=1000):  # ici d est le dénominateur maximum
        from fractions import Fraction
        return [[Fraction(y).limit_denominator(d) for y in x] for x in M]
```

Par exemple, on définit une matrice A d'ordre 3, on calcule son inverse, et on convertit le résultat avec notre fonction `tofrac` :

```
>>> A = np.array([[1,3,2],[4,1,6],[3,2,5]]); A    # une matrice A d'ordre 3
array([[1, 3, 2],
       [4, 1, 6],
       [3, 2, 5]])
>>> B = np.linalg.inv(A); B                       # son inverse, en mode float
array([[ 2.33333333,  3.66666667, -5.33333333],
       [ 0.66666667,  0.33333333, -0.66666667],
       [-1.66666667, -2.33333333,  3.66666667]])
>>> tofrac(B)                                     # convertit le résultat en rationnels
[[Fraction(7, 3), Fraction(11, 3), Fraction(-16, 3)],
 [Fraction(2, 3), Fraction(1, 3), Fraction(-2, 3)],
 [Fraction(-5, 3), Fraction(-7, 3), Fraction(11, 3)]]
```

Le résultat précédent nous dit donc que si $A = \begin{pmatrix} 1 & 3 & 2 \\ 4 & 1 & 6 \\ 3 & 2 & 5 \end{pmatrix}$, alors $A^{-1} = \frac{1}{3}\begin{pmatrix} 7 & 11 & -16 \\ 2 & 1 & -2 \\ -5 & -7 & 11 \end{pmatrix}$.

On en vient maintenant à plusieurs caractérisations équivalentes de l'inversibilité d'une matrice carrée :

Proposition 16.3.9 (caractérisation des matrices inversibles)
Soit A une matrice de $\mathcal{M}_n(\mathbb{K})$, avec $n \geqslant 1$.
Les conditions suivantes sont équivalentes :
- *la matrice A est inversible dans $\mathcal{M}_n(\mathbb{K})$*
- *il existe une matrice B de $\mathcal{M}_n(\mathbb{K})$ telle que $AB = I_n$ (on a alors $B = A^{-1}$)*
- *il existe une matrice B de $\mathcal{M}_n(\mathbb{K})$ telle que $BA = I_n$ (on a alors $B = A^{-1}$)*
- *la seule matrice-colonne X telle que $AX = 0$ est la colonne nulle*
- *la seule matrice-ligne X telle que $XA = 0$ est la ligne nulle*
- *la matrice A n'est pas un diviseur de zéro*

Proposition 16.3.10 (inversibilité des matrices triangulaires)
Soit A une matrice triangulaire supérieure (resp. inférieure).
Alors est inversible si et seulement si ses coefficients diagonaux a_{ii} sont non nuls.
Son inverse A^{-1} est alors triangulaire supérieure (resp. inférieure).
De plus les coefficients diagonaux de A^{-1} sont les inverses des a_{ii}.

- Cas particulier : une matrice diagonale D est inversible si et seulement si ses coefficients diagonaux d_{ii} sont tous non nuls. La matrice D^{-1} est alors la matrice diagonale dont les coefficients diagonaux

sont les inverses respectifs des d_{ii}.

– Si la matrice A est triangulaire inversible, alors la propriété disant que A^p est triangulaire (et du même coté que A) avec pour coefficients diagonaux les a_{ii}^p est encore valable pour les exposants négatifs.

Soit A dans $\mathcal{M}_n(\mathbb{K})$. Notons $C_1, C_2, \dots, C_n$ les colonnes de A, et $L_1, L_2, \dots, L_n$ les lignes de A.

Avec ces notations, on peut écrire : $A = \left(\begin{array}{c} L_1 \\ \hline L_2 \\ \hline \vdots \\ \hline L_n \end{array}\right) = \left(\begin{array}{c|c|c|c} C_1 & C_2 & \cdots & C_n \end{array}\right)$

On sait (voir sous-section 16.2.5) que si les λ_k sont des scalaires, on peut écrire :

$$\left(\begin{array}{c|c|c|c} C_1 & C_2 & \cdots & C_n \end{array}\right)\begin{pmatrix}\lambda_1\\ \lambda_2\\ \vdots\\ \lambda_n\end{pmatrix} = \sum_{j=1}^{n}\lambda_j C_j \quad \text{et} \quad \begin{pmatrix}\lambda_1 & \lambda_2 & \cdots & \lambda_n\end{pmatrix}\left(\begin{array}{c} L_1 \\ \hline L_2 \\ \hline \vdots \\ \hline L_n \end{array}\right) = \sum_{i=1}^{n}\lambda_i L_i$$

On peut alors écrire de nouvelles caractérisations de l'inversibilité d'une matrice carrée :

Proposition 16.3.11 (caractérisation de l'inversibilité d'une matrice carrée)
Soit A une matrice de $\mathcal{M}_n(\mathbb{K})$, avec $n \geqslant 1$. Les conditions suivantes sont équivalentes :
– *la matrice A est inversible dans $\mathcal{M}_n(\mathbb{K})$*
– *les lignes de A sont linéairement indépendantes*
– *les colonnes de A sont linéairement indépendantes*

La proposition précédente permet donc de caractériser la non inversibilité !

Proposition 16.3.12 (caractérisation de la non inversibilité d'une matrice carrée)
Soit A une matrice de $\mathcal{M}_n(\mathbb{K})$, avec $n \geqslant 1$. Les conditions suivantes sont équivalentes :
– *la matrice A n'est pas inversible dans $\mathcal{M}_n(\mathbb{K})$*
– *les lignes de A forment une famille liée*
– *les colonnes de A forment une famille liée*

Voici donc des conditions *suffisantes* pour dire, au premier coup d'œil, qu'une matrice est non inversible :
– si elle contient une ligne nulle, ou une colonne nulle
– si elle contient deux lignes (ou deux colonnes) proportionnelles

Considérons par exemple $A = \begin{pmatrix}1 & 3 & 4\\ 2 & 1 & 3\\ 3 & 2 & 5\end{pmatrix}$. On « voit » que ses colonnes sont liées car $C_3 = C_1 + C_2$.
La matrice A n'est donc pas inversible. Plus précisément $AX = 0$, avec $X = \begin{pmatrix}1\\ 1\\ -1\end{pmatrix}$.
Puisque A n'est pas inversible, ses lignes L_1, L_2, L_3 doivent être liées.
Par le calcul, on trouve $5L_3 = L_1 + 7L_2$, ce qui était difficile à voir « au premier coup d'œil » !

16.3.6 Calcul de l'inverse d'une matrice inversible

▷ Inversion d'une matrice par résolution d'un système

Soit A un élément de $\mathcal{M}_n(\mathbb{K})$, et soit X, B deux matrices-colonne de hauteur n.
Si le système $AX = B$, d'inconnue X, possède une solution unique alors A est inversible.
La résolution de ce système fournit d'ailleurs A^{-1} car $AX = B \Leftrightarrow X = A^{-1}B$.

Considérons par exemple la matrice $A = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$.
On a les équivalences :

$$A\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \end{pmatrix} \Leftrightarrow \begin{cases} y + z = a \\ x + y + z = b \\ x + z = c \end{cases} \Leftrightarrow \begin{cases} x = -a + b \\ y = b - c \\ z = a - b + c \end{cases} \Leftrightarrow \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -1 & 1 & 0 \\ 0 & 1 & -1 \\ 1 & -1 & 1 \end{pmatrix}\begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

Ce calcul prouve que A est inversible et que $A^{-1} = \begin{pmatrix} -1 & 1 & 0 \\ 0 & 1 & -1 \\ 1 & -1 & 1 \end{pmatrix}$.

▷ Utilisation d'un polynôme annulateur

Supposons par exemple qu'une matrice A vérifie $A^3 - 2A^2 - A + 2I_n = 0 \quad (1)$
On dit que $P = \mathrm{X}^3 - 2\mathrm{X}^2 - \mathrm{X} + 2$ est un *polynôme annulateur* de A, l'égalité (1) s'écrivant $P(A) = 0$.
Mais (1) s'écrit $A(A^2 - 2A + 2I_n) = -2I_n$ et prouve que A est inversible avec $A^{-1} = -\dfrac{1}{2}(A^2 - 2A + 2I_n)$.

Autre exemple : avec $A = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & -1 \end{pmatrix}$, on a $A^2 = \begin{pmatrix} 4 & 0 & 0 & -2 \\ 0 & 4 & 0 & 2 \\ 0 & 0 & 4 & 2 \\ -2 & 2 & 2 & 4 \end{pmatrix}$ et $A^3 = \begin{pmatrix} 2 & 6 & 6 & 6 \\ 6 & 2 & -6 & -6 \\ 6 & -6 & 2 & -6 \\ 6 & -6 & -6 & -10 \end{pmatrix}$.

On remarque que $A^3 = 6A - 4I_4$. Ainsi $A(6I_4 - A^2) = 4I_4$.
On en déduit que A est inversible et que $A^{-1} = \dfrac{1}{4}(6I_4 - A^2) = \dfrac{1}{2}\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$.

▷ Exposants négatifs

Supposons qu'on ait trouvé une formule donnant $A^p = \varphi(p)$ en fonction de l'entier $p \geqslant 0$. On peut chercher à prouver que cette formule est encore valable pour les exposants négatifs, à condition que A soit inversible. Il suffit alors de prouver que pour tout p de $\mathbb{N}$, $\varphi(p)\varphi(-p) = \mathrm{I}_n$.

Soit par exemple $A = \begin{pmatrix} 1 & 2\lambda & 0 \\ 0 & 1 & 2\lambda \\ 0 & 0 & 1 \end{pmatrix} = I + 2\lambda T$, avec $T = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$, $T^2 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ et $T^3 = 0$.

La formule du binôme donne $A^p = \varphi(p) = I + 2p\lambda T + 2p(p-1)\lambda^2 T^2 = \begin{pmatrix} 1 & 2p\lambda & 2p(p-1)\lambda^2 \\ 0 & 1 & 2p\lambda \\ 0 & 0 & 1 \end{pmatrix}$

Mais $\varphi(p)\varphi(-p) = \Big(I + 2p\lambda T + 2p(p-1)\lambda^2 T^2\Big)\Big(I - 2p\lambda T + 2p(p+1)\lambda^2 T^2\Big) = I$ (grâce à $T^3 = 0$).

Ainsi $\varphi(-p) = \Big(\varphi(p)\Big)^{-1}$, donc la formule $A^p = \varphi(p)$ est encore valable pour les exposants négatifs.

16.4 Transposition

16.4.1 Propriétés de la transposition

Définition 16.4.1 (transposée d'une matrice)
Soit A une matrice de $\mathcal{M}_{n,p}(\mathbb{K})$, de terme général $a_{i,j}$.
On appelle *transposée* de A et on note $A^\top$ la matrice B de $\mathcal{M}_{p,n}(\mathbb{K})$ de terme général $b_{i,j} = a_{j,i}$.

Remarque : on peut utiliser les deux notations $A^\top$ ou $A^{\mathtt{t}}$.

Si par exemple $A = \begin{pmatrix} 1 & 4 & 2 & 3 \\ 8 & 4 & 3 & 6 \\ 7 & 1 & 0 & 5 \end{pmatrix}$, alors $A^\top = \begin{pmatrix} 1 & 8 & 7 \\ 4 & 4 & 1 \\ 2 & 3 & 0 \\ 3 & 6 & 5 \end{pmatrix}$.

Voici comment on peut transposer une matrice avec `Python`. On notera les deux syntaxes possibles.

```
>>> A = np.array([[1,4,2,3],[8,4,3,6],[7,1,0,5]]);
>>> A
array([[1, 4, 2, 3],
       [8, 4, 3, 6],
       [7, 1, 0, 5]])
```

```
>>> A.transpose()  # ou A.T
array([[1, 8, 7],
       [4, 4, 1],
       [2, 3, 0],
       [3, 6, 5]])
```

Proposition 16.4.1 (linéarité et bijectivité de la transposition)
La transposition des matrices induit un isomorphisme de $\mathcal{M}_{n,p}(\mathbb{K})$ sur $\mathcal{M}_{p,n}(\mathbb{K})$.
Plus précisément, si $n = p$, la transposition est un automorphisme involutif de $\mathcal{M}_n(\mathbb{K})$.

On retiendra que si A, B ont le même format, et pour tous λ, μ de $\mathbb{K}$, on a : $\begin{cases} (\lambda A + \mu B)^\top = \lambda A^\top + \mu B^\top \\ \left(A^\top\right)^\top = A \end{cases}$

Proposition 16.4.2 (transposée d'un produit de deux matrices)
Pour A dans $\mathcal{M}_{n,p}(\mathbb{K})$, et B dans $\mathcal{M}_{p,q}(\mathbb{K})$, on a l'égalité : $(AB)^\top = B^\top A^\top$ (Attention à l'ordre !)

On peut généraliser à un produit de k matrices : $({}^\top A_1 A_2 \cdots A_k) = A_k^\top\, A_{k-1}^\top \cdots A_2^\top\, A_1^\top$.

Proposition 16.4.3 (transposition de l'inverse d'une matrice carrée)
Si A est une matrice carrée inversible, alors la matrice $A^\top$ est inversible et $(A^\top)^{-1} = (A^{-1})^\top$.

On peut retenir que l'inverse de la transposée, c'est la transposée de l'inverse.

Proposition 16.4.4
Pour toute matrice A de $\mathcal{M}_n(\mathbb{K})$ et tout entier naturel k, on a : $\left(A^k\right)^\top = (A^\top)^k$.
Si A est inversible, cette égalité s'étend aux entiers strictement négatifs.

16.4.2 Matrices symétriques ou antisymétriques

Définition 16.4.2 (matrices carrées symétriques)
Une matrice $A = (a_{i,j})$ de $\mathcal{M}_n(\mathbb{K})$ est dite *symétrique* si elle vérifie $A^\top = A$.
Cela équivaut donc à dire que $a_{j,i} = a_{i,j}$ pour tous indices i et j.

Dire que A est symétrique, c'est donc dire qu'elle est « symétrique par rapport à sa diagonale ».

Une matrice symétrique A est définie de façon unique par la donnée de ses coefficients $a_{i,j}$ avec $j \geqslant i$ (c'est-à-dire qui sont « sur » ou « au-dessus » de la diagonale).

Définition 16.4.3 (matrices carrées antisymétriques)
Une matrice $A = (a_{i,j})$ de $\mathcal{M}_n(\mathbb{K})$ est dite *antisymétrique* si $A^\top = -A$.
Cela équivaut donc à dire que $a_{j,i} = -a_{i,j}$ pour tous indices i et j.

Si A est antisymétrique, alors ses coefficients diagonaux sont nuls.

Une matrice antisymétrique A est définie de façon unique par la donnée de ses coefficients $a_{i,j}$ avec $j > i$ (c'est-à-dire qui sont « strictement au-dessus » de la diagonale).

Par exemple, $A = \begin{pmatrix} 4 & 1 & 0 & 8 \\ 1 & 3 & 2 & 5 \\ 0 & 2 & 7 & 6 \\ 8 & 5 & 6 & 3 \end{pmatrix}$ est symétrique, et $B = \begin{pmatrix} 0 & -1 & 3 & 6 \\ 1 & 0 & 2 & -4 \\ -3 & -2 & 0 & 7 \\ -6 & 4 & -7 & 0 \end{pmatrix}$ est antisymétrique.

Cela n'a pas de sens de parler de symétrie ou d'antisymétrie pour une matrice qui n'est pas carrée !

Quelques propriétés
- si A est inversible et symétrique alors A^{-1} est symétrique.
  si A est inversible et antisymétrique alors A^{-1} est antisymétrique.
- si A est symétrique alors A^k est symétrique pour tout k de $\mathbb{N}$ (et k dans $\mathbb{Z}$ si A est inversible).
- si A est antisymétrique, alors A^k est symétrique si k est pair et antisymétrique si k est impair.

Notation
On note $\mathcal{S}_n(\mathbb{K})$ l'ensemble des matrices de $\mathcal{M}_n(\mathbb{K})$ qui sont symétriques.

On note $\mathcal{A}_n(\mathbb{K})$ l'ensemble des matrices de $\mathcal{M}_n(\mathbb{K})$ qui sont antisymétriques.

Proposition 16.4.5 (sous-espaces des matrices symétriques ou antisymétriques)
Les ensembles $\mathcal{S}_n(\mathbb{K})$ et $\mathcal{A}_n(\mathbb{K})$ sont deux sous-espaces supplémentaires de $\mathcal{M}_n(\mathbb{K})$.
La dimension de $\mathcal{S}_n(\mathbb{K})$ est $\frac{1}{2}n(n+1)$ et celle de $\mathcal{A}_n(\mathbb{K})$ est $\frac{1}{2}n(n-1)$.
Toute matrice M de $\mathcal{M}_n(\mathbb{K})$ s'écrit de manière unique $M = S + A$, avec S dans $\mathcal{S}_n(\mathbb{K})$ et A dans $\mathcal{A}_n(\mathbb{K})$.
Les matrices S et A sont données par : $S = \frac{1}{2}(M + M^\top)$ et $A = \frac{1}{2}(M - M^\top)$.

Les fonctions `sym` et `asym` renvoient les composantes symétrique et antisymétrique d'une matrice M :

```
>>> def sym(M): return (M+M.T)/2
```

```
>>> def asym(M): return (M-M.T)/2
```

```
>>> M=np.array([[1,3,4],[2,7,6],[4,1,2]])
>>> print(M)
[[1 3 4]
 [2 7 6]
 [4 1 2]]
```

```
>>> S = sym(M)
>>> print(S)
[[ 1.   2.5  4. ]
 [ 2.5  7.   3.5]
 [ 4.   3.5  2. ]]
```

```
>>> A = asym(M)
>>> print(A)
[[ 0.   0.5  0. ]
 [-0.5  0.   2.5]
 [ 0.  -2.5  0. ]]
```

Les deux fonctions suivantes (`randsym` et `randasym`) permettent de fabriquer des matrices symétriques ou antisymétriques aléatoires d'ordre n (avec $n = 4$ par défaut). Elles forment en fait, respectivement, la composante symétrique ou antisymétrique d'une matrice aléatoire d'ordre n à coefficients entiers dans un intervalle donné $[a, b]$ (avec $[a, b] = [-5, 5]$ par défaut).

Ici les matrices renvoyées sont dans le type `int`.

Attention à la « division par 2 » qu'il faut forcer dans le type `float` (on utilise donc `/` et non pas `//`) avant de revenir dans le type `int` avec la fonction `astype` :

```
>>> def randsym(n=4, a=-5, b=5):
        m = np.random.randint(a, b+1, size=(n,n))
        return ((m+m.T)/2).astype('int')

>>> def randasym(n=4, a=-5, b=5):
        m = np.random.randint(a, b+1, size=(n,n))
        return ((m-m.T)/2).astype('int')
```

Voici quelques exemples :

```
>>> randsym()
array([[-1, -2,  0,  0],
       [-2, -5,  4, -1],
       [ 0,  4,  5,  1],
       [ 0, -1,  1, -1]])
```

```
>>> randasym()
array([[ 0,  2,  3,  1],
       [-2,  0,  1,  0],
       [-3, -1,  0,  3],
       [-1,  0, -3,  0]])
```

```
>>> randsym(a=0,b=1)
array([[0, 0, 1, 0],
       [0, 1, 1, 0],
       [1, 1, 0, 1],
       [0, 0, 1, 0]])
```

```
>>> randsym(6,-9,9)
array([[ 7,  2,  5,  5, -1,  0],
       [ 2,  1, -3, -1,  0, -4],
       [ 5, -3,  6, -2, -6, -1],
       [ 5, -1, -2, -3, -6, -5],
       [-1,  0, -6, -6, -9, -1],
       [ 0, -4, -1, -5, -1, -2]])
```

```
>>> randasym(6,-99,99)
array([[  0,   4, -62,  -5, -45,  27],
       [ -4,   0, -30, -70,  24,  38],
       [ 62,  30,   0, -33,  15, -61],
       [  5,  70,  33,   0,   8, -60],
       [ 45, -24, -15,  -8,   0,  28],
       [-27, -38,  61,  60, -28,   0]])
```

16.5 Calculs par blocs

16.5.1 Décompositions en blocs

▷ Commençons par quelques exemples

La matrice $A = \begin{pmatrix} 2 & 5 & 3 & 1 & 6 & 7 \\ 4 & 1 & 5 & 6 & 8 & 9 \\ 1 & 0 & 4 & 5 & 1 & 3 \\ 7 & 9 & 6 & 2 & 1 & 5 \\ 1 & 5 & 8 & 2 & 7 & 4 \end{pmatrix}$ est « décomposable en blocs » de plusieurs manières, par exemple :

$A=\left(\begin{array}{ccc|ccc}2&5&3&1&6&7\\4&1&5&6&8&9\\\hline1&0&4&5&1&3\\7&9&6&2&1&5\\1&5&8&2&7&4\end{array}\right)=\begin{pmatrix}M&N\\P&Q\end{pmatrix}$, où $M=\begin{pmatrix}2&5&3\\4&1&5\end{pmatrix}$, $N=\begin{pmatrix}1&6&7\\6&8&9\end{pmatrix}$, $P=\begin{pmatrix}1&0&4\\7&9&6\\1&5&8\end{pmatrix}$, $Q=\begin{pmatrix}5&1&3\\2&1&5\\2&7&4\end{pmatrix}$

On pourrait aussi décomposer en $A=\left(\begin{array}{cccc|cc}2&5&3&1&6&7\\4&1&5&6&8&9\\1&0&4&5&1&3\\\hline7&9&6&2&1&5\\1&5&8&2&7&4\end{array}\right)$, ou même en $A=\left(\begin{array}{cc|ccc|c}2&5&3&1&6&7\\4&1&5&6&8&9\\\hline1&0&4&5&1&3\\7&9&6&2&1&5\\\hline1&5&8&2&7&4\end{array}\right)$

▷ Décompositions en blocs avec `Python`

On reprend la matrice initiale A, et on forme les matrices M, P, N, Q comme indiqué ci-dessus.

Rappelons le principe du « slicing » qui dit que l'expression `A[a:b;c:d]` renvoie le bloc formé des lignes dont l'indice est dans $[a, b[$ et des colonnes dont l'indice est dans $[c, d[$.

Dans cette syntaxe, le premier indice vaut 0 par défaut, et le deuxième vaut $+\infty$ par défaut.

```
>>> A
array([[2, 5, 3, 1, 6, 7],
       [4, 1, 5, 6, 8, 9],
       [1, 0, 4, 5, 1, 3],
       [7, 9, 6, 2, 1, 5],
       [1, 5, 8, 2, 7, 4]])
>>>
```

```
>>> M = A[:2,:3]; M
array([[2, 5, 3],
       [4, 1, 5]])
>>> P = A[2:,:3]; P
array([[1, 0, 4],
       [7, 9, 6],
       [1, 5, 8]])
```

```
>>> N = A[:2,3:]; N
array([[1, 6, 7],
       [6, 8, 9]])
>>> Q = A[2:,3:]; Q
array([[5, 1, 3],
       [2, 1, 5],
       [2, 7, 4]])
```

Voici ensuite deux façons de reconstituer A à partir des blocs M, N, P, Q :

```
>>> np.bmat([[M,N],[P,Q]])
matrix([[2, 5, 3, 1, 6, 7],
        [4, 1, 5, 6, 8, 9],
        [1, 0, 4, 5, 1, 3],
        [7, 9, 6, 2, 1, 5],
        [1, 5, 8, 2, 7, 4]])
```

```
>>> np.bmat('M N; P Q')
matrix([[2, 5, 3, 1, 6, 7],
        [4, 1, 5, 6, 8, 9],
        [1, 0, 4, 5, 1, 3],
        [7, 9, 6, 2, 1, 5],
        [1, 5, 8, 2, 7, 4]])
```

Attention : les résultats sont au format « matrix », une sous-classe de « array ». Il y a quelques différences de syntaxe (notamment pour le produit matriciel), et il est préférable de s'en tenir au format `array`.

On pouvait par exemple écrire : `np.asarray(np.bmat('M N; P Q'))`

▷ Forme générale d'une décomposition en blocs

Plus généralement, soit A une matrice quelconque de $\mathcal{M}_{n,p}(\mathbb{K})$.

On peut écrire $A=\left(\begin{array}{c|c|c|c|c|c}A_{11}&A_{12}&\dots&A_{1k}&\dots&A_{1q}\\\hline\vdots&\vdots&&\vdots&&\vdots\\\hline A_{i1}&A_{i2}&\dots&A_{ik}&\dots&A_{iq}\\\hline\vdots&\vdots&&\vdots&&\vdots\\\hline A_{m1}&A_{m2}&\dots&A_{mk}&\dots&A_{mq}\end{array}\right)$ où les $A_{i,j}$ sont elles-mêmes des matrices.

On a ainsi décomposé A en mq blocs.

La contrainte est que chaque bloc $A_{i,j}$ soit de taille (n_i, p_j), avec $\sum_{i=1}^{m} n_j = n$, et $\sum_{j=1}^{q} p_j = p$.

Certaines décompositions par blocs reviennent plus souvent que d'autres.

Une matrice A peut par exemple être décomposée en quatre blocs : $A = \begin{pmatrix} M & N \\ P & Q \end{pmatrix}$

▷ Décompositions en lignes ou en colonnes

Soit A dans $\mathcal{M}_{n,p}(\mathbb{K})$. Notons $\mathrm{C}_1, \mathrm{C}_2, \ldots, \mathrm{C}_p$ les colonnes de A, et $\mathrm{L}_1, \mathrm{L}_2, \ldots, \mathrm{L}_n$ les lignes de A.

Les écritures $A = \left(\begin{array}{c} \mathrm{L}_1 \\ \hline \mathrm{L}_2 \\ \hline \vdots \\ \hline \mathrm{L}_n \end{array}\right) = \left(\begin{array}{c|c|c|c} \mathrm{C}_1 & \mathrm{C}_2 & \cdots & \mathrm{C}_p \end{array}\right)$ sont des décompositions particulières de A.

▷ Matrices diagonales ou triangulaires par blocs

Désignons par $D_1, D_2, \ldots, D_m$ une succession de m matrices carrées (pas nécessairement de même taille, certaines d'entre elles pouvant même être réduites à un seul scalaire).

On note également $*$ des blocs matriciels quelconques.

Si $A = \begin{pmatrix} D_1 & 0 & \ldots & 0 \\ 0 & D_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & D_m \end{pmatrix}$ et $B = \begin{pmatrix} D_1 & * & \ldots & * \\ 0 & D_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & * \\ 0 & \ldots & 0 & D_m \end{pmatrix}$ on dit que $\begin{cases} A \text{ est « diagonale par blocs »} \\ B \text{ est « triangulaire par blocs »} \end{cases}$

16.5.2 Opérations sur les décompositions en blocs

▷ Addition par blocs

Soit A et B dans $\mathcal{M}_{n,p}(\mathbb{K})$, décomposées en blocs : $A = \begin{pmatrix} M & N \\ P & Q \end{pmatrix}$ et $B = \begin{pmatrix} M' & N' \\ P' & Q' \end{pmatrix}$

On demande ici que ces deux décompositions soient « compatibles pour la somme », c'est-à-dire que M et M' aient le même format (il en est alors de même pour N et N', pour P et P', pour Q et Q').

Si ces conditions sont réunies, alors $\lambda A + \mu B = \begin{pmatrix} \lambda M + \mu M' & \lambda N + \mu N' \\ \lambda P + \mu P' & \lambda Q + \mu Q' \end{pmatrix}$.

Cette propriété se généralise à des décompositions en blocs quelconques, sous réserve bien sûr que les blocs qui se correspondent aient exactement la même taille.

▷ Produit par blocs

Soit A dans $\mathcal{M}_{n,p}(\mathbb{K})$, B dans $\mathcal{M}_{p,q}(\mathbb{K})$, décomposées en blocs : $A = \begin{pmatrix} M & N \\ P & Q \end{pmatrix}$ et $B = \begin{pmatrix} M' & N' \\ P' & Q' \end{pmatrix}$

On souhaite effectuer le produit AB en exploitant cette décomposition.

On voudrait en fait pouvoir écrire : $AB = \begin{pmatrix} M & N \\ P & Q \end{pmatrix} \begin{pmatrix} M' & N' \\ P' & Q' \end{pmatrix} = \begin{pmatrix} MM' + NP' & MN' + NQ' \\ PM' + QP' & PN' + QQ' \end{pmatrix}$

Pour cela, il faut déjà pouvoir effectuer le produit MM' des blocs M et M'.

Supposons par exemple que le bloc M soit de type (m, r), avec les inégalités $\begin{cases} 1 \leqslant m \leqslant n \\ 1 \leqslant r \leqslant p \end{cases}$

Supposons également que M' soit de type (r,s), avec $\begin{cases}1\leqslant r\leqslant p\\1\leqslant s\leqslant q\end{cases}$

Avec ces hypothèses (qui permettent effectivement de calculer MM'), on a :

$$\begin{cases}M\in\mathcal{M}_{m,r}(\mathbb{K})\\M'\in\mathcal{M}_{r,s}(\mathbb{K})\end{cases}\quad\begin{cases}N\in\mathcal{M}_{m,p-r}(\mathbb{K})\\P'\in\mathcal{M}_{p-r,s}(\mathbb{K})\end{cases}\quad\text{donc}\quad\begin{cases}MM'\in\mathcal{M}_{m,s}(\mathbb{K})\\NP'\in\mathcal{M}_{m,s}(\mathbb{K})\end{cases}\quad\text{donc } MM'+NP'\in\mathcal{M}_{m,s}(\mathbb{K}).$$

$$\begin{cases}M\in\mathcal{M}_{m,r}(\mathbb{K})\\N'\in\mathcal{M}_{r,q-s}(\mathbb{K})\end{cases}\quad\begin{cases}N\in\mathcal{M}_{m,p-r}(\mathbb{K})\\Q'\in\mathcal{M}_{p-r,q-s}(\mathbb{K})\end{cases}\quad\text{donc}\quad\begin{cases}MN'\in\mathcal{M}_{m,q-s}(\mathbb{K})\\NQ'\in\mathcal{M}_{m,q-s}(\mathbb{K})\end{cases}\quad\text{donc } MN'+NQ'\in\mathcal{M}_{m,q-s}(\mathbb{K}).$$

$$\begin{cases}P\in\mathcal{M}_{n-m,r}(\mathbb{K})\\M'\in\mathcal{M}_{r,s}(\mathbb{K})\end{cases}\quad\begin{cases}Q\in\mathcal{M}_{n-m,p-r}(\mathbb{K})\\P'\in\mathcal{M}_{p-r,s}(\mathbb{K})\end{cases}\quad\text{donc}\quad\begin{cases}PM'\in\mathcal{M}_{n-m,s}(\mathbb{K})\\QP'\in\mathcal{M}_{n-m,s}(\mathbb{K})\end{cases}\quad\text{donc } PM'+QP'\in\mathcal{M}_{n-m,s}(\mathbb{K}).$$

$$\begin{cases}P\in\mathcal{M}_{n-m,r}(\mathbb{K})\\N'\in\mathcal{M}_{r,q-s}(\mathbb{K})\end{cases}\quad\begin{cases}Q\in\mathcal{M}_{n-m,p-r}(\mathbb{K})\\Q'\in\mathcal{M}_{p-r,q-s}(\mathbb{K})\end{cases}\quad\text{donc}\quad\begin{cases}PN'\in\mathcal{M}_{n-m,q-s}(\mathbb{K})\\QQ'\in\mathcal{M}_{n-m,q-s}(\mathbb{K})\end{cases}\quad\text{donc } PN'+QQ'\in\mathcal{M}_{n-m,q-s}(\mathbb{K}).$$

On vérifie alors qu'effectivement : $AB=\begin{pmatrix}M&N\\P&Q\end{pmatrix}\begin{pmatrix}M'&N'\\P'&Q'\end{pmatrix}=\begin{pmatrix}MM'+NP'&MN'+NQ'\\PM'+QP'&PN'+QQ'\end{pmatrix}$

Cette propriété se généralise à des décompositions en un nombre quelconque de blocs, sous réserve que tous les produits de blocs qui sont nécessaires à la construction du résultat soient possibles :

Proposition 16.5.1 (théorème du produit par blocs)
Soit A dans $\mathcal{M}_{n,p}(\mathbb{K})$, et B dans $\mathcal{M}_{p,q}(\mathbb{K})$.
On suppose que A et B sont décomposées en blocs : $A=(A_{i,k})_{\substack{1\leqslant i\leqslant r\\1\leqslant k\leqslant s}}$ et $B=(B_{k,j})_{\substack{1\leqslant k\leqslant s\\1\leqslant j\leqslant t}}$.
Alors $M=AB$ se décompose en blocs $M=(M_{i,j})_{\substack{1\leqslant i\leqslant r\\1\leqslant j\leqslant t}}$, avec $\begin{cases}\forall\, i\in\{1,\dots,r\}\\\forall\, j\in\{1,\dots,t\}\end{cases}\quad M_{ij}=\sum_{k=1}^{s}A_{ik}\,B_{kj}$

▷ Cas des matrices triangulaires par blocs

Soit A et B deux matrices carrées, triangulaires par blocs.

On peut donc écrire $A=\begin{pmatrix}D_1&*&\dots&*\\0&D_2&\ddots&\vdots\\\vdots&\ddots&\ddots&*\\0&\dots&0&D_m\end{pmatrix}$ et $B=\begin{pmatrix}\Delta_1&*&\dots&*\\0&\Delta_2&\ddots&\vdots\\\vdots&\ddots&\ddots&*\\0&\dots&0&\Delta_m\end{pmatrix}$

On suppose ici que D_i et Δ_i ont la même taille. Les blocs marqués $*$ sont sans importance.

Avec ces notations, on montre que :

$$AB=\begin{pmatrix}D_1\Delta_1&*&\dots&*\\0&D_2\Delta_2&\ddots&\vdots\\\vdots&\ddots&\ddots&*\\0&\dots&0&D_m\Delta_m\end{pmatrix},\quad A^p=\begin{pmatrix}D_1^p&*&\dots&*\\0&D_2^p&\ddots&\vdots\\\vdots&\ddots&\ddots&*\\0&\dots&0&D_m^p\end{pmatrix},\quad A^{-1}=\begin{pmatrix}D_1^{-1}&*&\dots&*\\0&D_2^{-1}&\ddots&\vdots\\\vdots&\ddots&\ddots&*\\0&\dots&0&D_m^{-1}\end{pmatrix}$$

La dernière égalité doit être précisée : elle exprime que A est inversible si et seulement si chacun des blocs diagonaux D_i est inversible. L'inverse de A est alors triangulaire par blocs, les blocs diagonaux de A^{-1} étant, respectivement, les inverses des blocs diagonaux de A.