Matrices

L'objet des questions qui suivent est de programmer un ensemble de fonctions (qui formeront un module) destinées à agir sur des matrices. Parmi les points abordés figurent la résolution de systèmes linéaires, la recherche de noyau et d'image…

On écrira dans un seul fichier, de nom `matrices.py` les fonctions qui suivent. Il est bien sûr permis (et recommandé !) de me soumettre plusieurs fois votre recherche, au fur et à mesure de votre avancement, tout comme poser toutes les questions que vous voulez.

Une matrice sera représentée sous la forme d'une liste de listes, et un vecteur colonne sous la forme d'une liste. Toutes les matrices considérées seront, dans un premier temps, à coefficients entiers, mais dans un second temps, on rajoutera à l'aide d'un module usuel de python les nombres rationnels. (Difficile d'y échapper dès lors qu'on cherche à résoudre un système d'équations, fût-il à coefficients entiers…)

1 Premières fonctions

Question 1. Niveau 2. Ecrire une fonction d'entête `identite(n)` qui admet un argument entier `n` (supposé non nul) et qui renvoie la matrice identité d'ordre `n`. Exemple :

```
>>> identite(2), identite(3)
    ([[1, 0], [0, 1]], [[1, 0, 0], [0, 1, 0], [0, 0, 1]])
```

Question 2. Niveau 2. Ecrire une fonction d'entête `evalue(A, X)` qui admet deux arguments : une matrice A représentée ar `A` (liste de listes) et une liste `X` qui représente un vecteur colonne X, qui suppose A disposer d'autant de colonnes que X compte de lignes (il n'est pas nécessaire de s'en assurer), et qui renvoie la liste qui représente le vecteur AX.

Ici comme par la suite, la matrice n'est pas supposée carrée, et on pourra donc commencer par introduire les variables locales `n` (le nombre de lignes de A) et `p` le nombre de ses colonnes.

(compte-tenu de la représentation choisie d'une matrice : le nombre de lignes de A est le nombre de termes (listes) de `A`, et le nombre de colonnes est donc le nombre de termes de chacune des listes contenues dans `A`)

Question 3. Niveau 2. Ecrire une fonction `produit(A, B)` qui admet deux arguments : des matrices `A` et `B` dont on suppose que la première compte autant de colonnes que la seconde compte de lignes, qui calcule et renvoie la matrice AB.

Question 4. Niveau 2. Reprendre la fonction précédente (on pourra renommer celle-ci `produitOld`) pour que, selon que `B` soit une matrice ou un vecteur colonne, renvoie ou bien la matrice AB, ou bien le vecteur colonne AB. (Ceci rendra alors caduque la fonction `evalue` bien sûr)

2 Opérations élémentaires (sur les lignes et/ou colonnes)

Question 5. Niveau 1. D'une matrice donnée M, on cherche à effectuer l'opération élémentaire suivante :

$$L_i \leftrightarrow L_j$$

qui vous l'avez compris échange les lignes d'indices i et j. Ecrire une fonction d'entête `echangeL(M, i, j)` qui prend pour arguments une matrice `M`, deux indices de lignes qu'on supposera distincts `i` et `j`, et qui modifie en place `M`. La fonction n'est pas supposée renvoyer de valeur (autre que `None`).

Question 6. Niveau 2. On cherche à implémenter l'opération (pas complètement élémentaire) suivante, où partant d'une matrice M, on lui applique l'opération suivante, où i et j désignent deux indices distincts de lignes, a et b des scalaires (il pourra s'agir d'entiers ou de rationnels, mais on n'a pas à s'en préoccuper dans le code qui suit) :

$$L_i \leftarrow a\,L_i + b\,L_j$$

Ecrire alors une fonction d'entête `combiL(M, i, j, a, b)` qui réalise ce travail. Bien sûr, on modifiera M en place, et la fonction ne renverra pas de valeur.

3 Recherche de rang - pivot de Gauss

Une manière classique de déterminer le rang d'une matrice, rarement la plus efficace à la main, mais qu'il est assez facile de programmer, est de transformer celle-ci par des opérations élémentaires en une matrice dite échelonnée, c'est-à-dire prenant la forme :

$$\begin{pmatrix} 0 & \cdots & 0 & p_1 & * & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & * \\ \vdots & & \vdots & 0 & \cdots & 0 & p_2 & * & \cdots & \cdots & \cdots & * \\ \vdots & & \vdots & \vdots & & & & & & & & \vdots \\ \vdots & & \vdots & 0 & \cdots & \cdots & \cdots & \cdots & 0 & p_r & * & \cdots & * \\ \vdots & & \vdots & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ \vdots & & \vdots & \vdots & & & & & & & & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \end{pmatrix}$$

où $p_1, \cdots, p_r$ sont des scalaires non nuls. Une telle matrice est, on le reconnaît aussitôt par ses vecteurs lignes, de rang r, et comme les opérations élémentaires effectuées ne modifient pas le rang, ce rang r est aussi celui de la matrice dont on était parti.

Exemple commenté de traitement : on part de $M = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 8 \end{pmatrix}$. On s'appuie sur le premier coefficient de la première colonne M, lequel est non nul, et on fait des opérations élémentaires pour que les coefficients qui le suivent sur cette colonne deviennent nuls, ainsi on effectue : $L_2 \leftarrow L_2 - 2\,L_1$ et $L_3 \leftarrow L_3 - 3\,L_1$ et on parvient à la nouvelle matrice $\begin{pmatrix} 1 & 2 & 3 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix}$. On passe alors à la deuxième colonne, où, à partir de la deuxième ligne, tous les coefficients sont nuls, et on arrive à la troisième colonne où le premier coefficient non nul se situe à la troisième ligne : on réalise alors l'opération élémentaire $L_2 \leftrightarrow L_3$ et on voit qu'on en a terminé.

On voit qu'il y a deux grandes étapes (répétées) : la recherche d'un pivot à partir d'une position donnée (c-à-d un coefficient non nul), puis un échange éventuel de deux lignes et quelques opérations élémentaires suivantes pour mettre à zéro tous les coefficients situés sous ce pivot (à la même colonne)

Question 7. Niveau 2. On va ici écrire une fonction qui va nous servir à chercher le pivot suivant, à partir d'une position donnée dans une matrice. A partir des indices de ligne i et de colonne j au sein d'une matrice M, on cherche, si elle existe, la position (k, l) (ligne k et colonne l) du premier indice non nul de M. Premier au sens que on cherche à minimiser l'indice de colonne l, puis l'indice de ligne k. Exemple, sur la matrice $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 1 & 0 & 3 \end{pmatrix}$, depuis la position $(1, 1)$ en numérotation python (on numérote à partir de 0...), alors la position recherchée serait $(3, 1)$ et non pas $(1, 2)$ ou $(2, 2)$.

Dans le cas où depuis la position (i, j) ne figurent plus que des zéros, alors on renverra (n, p) où n est le nombre de lignes de M et p son nombre de colonnes. Compléter alors la fonction suivante qui réalise ce travail :

```
def coefficientNonNulDepuis(M, i, j):
    n, p = len(M), len(M[0])
    for ...:
        for ...:
            ...

    # aucun coefficient non nul trouvé, alors :
    return (n, p)
```

Question 8. Niveau 3. En vous aidant des fonctions précédentes (questions 5, 6, 9) écrire une fonction d'entête `rang(M)` qui transforme dans un premier temps M en une matrice échelonnée (en ne faisant que des opérations élémentaires sur les lignes, c'est important pour ce qui suivra), puis qui déduit de la matrice obtenue son rang, et qui renvoie donc celui-ci. Exemple d'utilisation :

```
>>> M = [[1, 2, 3], [2, 4, 6], [3, 6, 8]]
>>> rang(M)
    2
>>> M
    [1, 2, 3], [0, 0, -1], [0, 0, 0]]
```

4 Image d'une matrice

On rappelle que l'image d'une matrice est l'espace vectoriel qu'engendre sa famille de vecteurs colonnes. Celle-ci forme donc une famille génératrice, mais pas forcément donc une base. On va tâcher d'obtenir une base de l'image d'une matrice.

On sait que des opérations élémentaires sur une matrice n'en changent pas le rang. Pour ce qui est du noyau et de l'image, il faut faire un peu attention :

- Une opération élémentaire sur les colonnes telle que, par exemple, $C_i \leftarrow C_i + a\,C_j$ ne change ni le rang, ni l'image. En effet, si l'image de A est, par définition $\mathrm{Vect}(C_1, \cdots, C_p)$ en supposant que A ait p colonnes, alors l'image de la matrice obtenue après cette opération élémentaire est $\mathrm{Vect}(C_1, \cdots, C_i + a\,C_j, \cdots, C_p)$ dont on devine qu'il est inclus dans le premier, or il a la même dimension…

- En revanche, une opération élémentaire sur les lignes modifie tous les vecteurs colonnes, et on voit bien qu'il n'y a aucune raison pour que l'image de la matrice obtenue soit la même que l'image de la matrice de départ.

  Par exemple, si $A = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$ après l'opération $L_2 \leftarrow L_2 - L_1$, on obtient la matrice $A' = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$ qui, si elle a le même rang, n'a pas la même image que la précédente. (On voit bien que les vecteurs $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ et $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ dont le premier engendre $\mathrm{Im}(A)$ et le second $\mathrm{Im}(A')$ ne sont pas colinéaires.)

- Pour ce qui concerne le noyau, on rappelle déjà que $\mathrm{Ker}\, A = \{X \in \mathbb{R}^p | A\,X = 0\}$ et que, si $X = \begin{pmatrix} x_1 \\ \vdots \\ x_p \end{pmatrix} \in \mathbb{R}^p$, alors $X \in \mathrm{Ker}\, A$ si, et seulement si $x_1\,C_1 + \cdots + x_p\,C_p = 0$ (en notant $C_1, \cdots, C_p$ les vecteurs colonnes de A). Une opération élémentaire sur les lignes de A n'invalide pas cette relation, et ceci indique que si A' s'obtient à partir de A et d'une opération élémentaire sur les lignes, alors $\mathrm{Ker}\, A \subset \mathrm{Ker}\, A'$. De plus, il y a égalité des dimensions et donc une opération élémentaire sur le lignes ne modifie pas le noyau.

- En revanche, une opération élémentaire sur les colonnes a toute chance de modifier le noyau, le mieux est de voir un exemple : si $A = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$, alors le noyau de A est la droite vectorielle $\mathrm{Vect}\left(\begin{pmatrix} 1 \\ -1 \end{pmatrix}\right)$. Mais si A' se déduit de A et de l'opération élémentaire $C_2 \leftarrow C_2 - C_1$, alors $A' = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}$ et elle admet pour noyau $\mathrm{Vect}\left(\begin{pmatrix} 0 \\ 1 \end{pmatrix}\right)$.

En résumé : une opération élémentaire sur les lignes de A ne change ni le rang, ni le noyau, mais a priori modifie l'image, tandis qu'une opération élémentaire sur les colonnes de A ne change ni le rang ni l'image, mais a priori modifie le noyau.

Du coup, voici une stratégie pour déterminer, à peu de frais, l'image d'une matrice :

- On transpose celle-ci (on rappelle que cette opération ne change pas le rang)

- On transforme la matrice obtenue, par des opérations élémentaires sur les lignes, en une matrice échelonnée (C'est comme si donc on travaillait sur les vecteurs colonnes de la matrice initiale)

- Si r est le rang de la matrice initiale (comme celui de la matrice transformée et échelonnée A'), alors les r premiers vecteurs lignes de la matrice obtenue forment une base de $\mathrm{Im}\, A$.

Question 9. Niveau 2. Ecrire une fonction d'entête `transpose(M)` qui prend pour unique argument une matrice et qui renvoie la matrice transposée de `M`.

Question 10. Niveau 2. A l'aide des questions 8 et 9, écrire une fonction d'entête `Im(M)` qui prend pour argument une matrice `M` et qui renvoie une liste de vecteurs formant une base de l'image de `M`. Par exemple :

```
>>> Im([[1, 2, 3], [2, 4, 6], [3, 6, 8]])
    [[1, 2, 3], [0, 0, -1]]

>>> Im([[1, 1], [2, 2]])
    [[1, 2]]
```

(Selon votre code, une autre réponse peut également être valable)

5 Systèmes d'équations linéaires, noyau d'une matrice

Comme indiqué en préambule, tôt ou tard, il nous faut introduire des nombres rationnels. On va s'appuyer sur le type `Fraction` que l'on importe ainsi : `from fractions import Fraction`

(Une autre possibilité, tout à fait équivalente, serait de faire appel au type `rational` que comprend le module de calcul symbolique `sympy` de python.)

On peut alors créer des nombres rationnels de la manière suivante :

```
>>> x, y = Fraction(1, 2), Fraction(1, 3)
>>> x + y
    Fraction(5, 6)
>>> print(x+y)
    5/6
>>> Fraction(6, -8)
    Fraction(-3, 4)
```

(Les fractions sont automatiquement réduites sous forme irréductible)

Attention : en écrivant `5/6`, c'est le nombre flottant 0.833333333333334 qui est calculé, et non le rationnel exactement représenté par un numérateur de 5 et un dénominateur de 6. Il convient donc d'introduire `Fraction(5, 6)` si c'est ce nombre rationnel, exactement représenté, qu'on veut obtenir.

Question 11. Nous allons par les calculs qui suivent obtenir des listes de nombres rationnels (qui représentent des vecteurs colonnes). Il ne sera en fait pas rare que la majorité, si ce n'est tous, des coefficients soient en fait des entiers. Il serait alors plus agréable de convertir, dans une telle liste, tous les rationnels de dénominateur 1 en l'entier correspondant. Ecrire donc une fonction `simplifieRationnels(L)` qui réalise cela. On modifiera `L` en place et on renverra `None`.

```
>>> L = [Fraction(5,1), Fraction(7,3), Fraction(3,1), 5]
>>> simplifieRationnels(L)
>>> L
    [5, Fraction(7,3), 3, 5]
```

Indication : tout objet python de type nombre (`int, float, complex, Fraction`) a parmi ses variables d'instance `numerator` et `denominator`. (Exemple : si `x` est une référence sur l'entier 5, alors `x.numerator` vaut 5 et `x.denominator` vaut 1. Sans surprise, `Fraction(7,3).denominator` vaut 3…)

Exemple. On va résoudre à la main, pour bien se remémorer la méthode, l'équation $AX=B$ pour $A=\begin{pmatrix}1&2&3\\2&4&6\\3&6&8\end{pmatrix}$ et $B=\begin{pmatrix}0\\0\\1\end{pmatrix}$.
On note que les opérations élémentaires sur les lignes correspondent à multiplier, par la gauche, par une matrice élémentaire de transvection, permutation, dilatation, or, si P est une matrice inversible, alors $PAX=PB$ et $AX=B$ sont équivalents, ce qui donne du sens aux calculs suivants :

$$\left(\begin{array}{ccc}1&2&3\\2&4&6\\3&6&8\end{array}\right)\quad\left(\begin{array}{c}0\\0\\1\end{array}\right)$$

$$L_2\leftarrow L_2-2L_1$$

$$\left(\begin{array}{ccc}\boxed{1}&2&3\\0&0&0\\3&6&8\end{array}\right)\quad\left(\begin{array}{c}0\\0\\1\end{array}\right)$$

$$L_3\leftarrow L_3-3L_1$$

$$\left(\begin{array}{ccc}\boxed{1}&2&3\\0&0&0\\0&0&-1\end{array}\right)\quad\left(\begin{array}{c}0\\0\\1\end{array}\right)$$

$$L_2\leftrightarrow L_3$$

$$\left(\begin{array}{ccc}\boxed{1}&2&3\\0&0&\boxed{-1}\\0&0&0\end{array}\right)\quad\left(\begin{array}{c}0\\1\\0\end{array}\right)$$

La première partie, le pivot à proprement parler, est terminée. On voit que le rang du système vaut 2. On a mis en évidence ici deux inconnues principales (la première et la troisième) et une secondaire (la deuxième).

Une première méthode pour décrire ses solutions est de choisir une valeur arbitraire pour l'inconnue secondaire, et d'en déduire les inconnues principales, or $\begin{pmatrix} \boxed{1} & 2 & 3 \\ 0 & 0 & \boxed{-1} \\ 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} x \\ y \\ z \end{pmatrix}=\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$ si et seulement si $z=-1$, $x+2\,y-3=0$ et ainsi l'ensemble des solutions est $\left\{\begin{pmatrix} 3-2\,y \\ y \\ -1 \end{pmatrix}|y\in\mathbb{R}\right\}=\left\{\begin{pmatrix} 3 \\ 0 \\ -1 \end{pmatrix}+y\begin{pmatrix} -2 \\ 1 \\ 0 \end{pmatrix}|y\in\mathbb{R}\right\}$. On reconnaît la droite passant par $\begin{pmatrix} 3 \\ 0 \\ -1 \end{pmatrix}$ et dirigée par $\begin{pmatrix} -2 \\ 1 \\ 0 \end{pmatrix}$.

Une deuxième méthode (qui est celle qu'on va mettre en pratique dans la fonction qui suit) est de déterminer d'abord une solution particulière, pour faire simple celle pour laquelle les inconnues secondaires prennent la valeur 0, puis de déterminer une base du noyau (qui est cet espace vectoriel qu'on ajoute à notre solution particulière pour en déduire l'ensemble des solutions) en choisissant de donner aux inconnues secondaires des valeurs parmi 0 et 1 (où exactement l'une d'elles vaut 1…)

Ici, la solution particulière recherchée est de la forme $\begin{pmatrix} x \\ 0 \\ z \end{pmatrix}$ et il vient $z=-1$ et $x=3$ (on trouve z et x dans cet ordre, en remontant notre système triangulaire)

Puis on cherche x et z tels que $\begin{pmatrix} \boxed{1} & 2 & 3 \\ 0 & 0 & \boxed{-1} \\ 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} x \\ 1 \\ z \end{pmatrix}=\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$ et on voit que $(x,z)=(-2,0)$ convient (ici aussi, on trouve d'abord z puis x)

De ce fait, l'ensemble des solutions est la droite passant par $\begin{pmatrix} 3 \\ 0 \\ -1 \end{pmatrix}$ et dirigée par $\begin{pmatrix} -2 \\ 1 \\ 0 \end{pmatrix}$. (Sans surprise, on retrouve la même chose qu'à la première méthode…)

Question 12. Niveau 2. On écrit ici une petite fonction utilitaire pour la grosse fonction qui va suivre. Etant donnés $n>0$ un entier, et `L` une liste d'entiers entre, au sens large, 0 et $n-1$, la fonction d'entête `complement(n, L)` devra renvoyer la liste formée d'exactement tous les entiers entre 0 et $n-1$ qui ne sont pas dans `L`.

Exemple :

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

Question 13. Niveau 4. On va écrire une fonction qui résout un système linéaire, présenté sous la forme d'une équation matricielle $A\,X=B$ d'inconnue $X\in\mathbb{R}^p$. Une telle équation est, on le sait, ou bien incompatible, ou bien admet un ensemble de solutions qui prend la forme $X_0+\operatorname{Ker}A$ où X_0 est une solution particulière.

La fonction est assez longue. La première partie consiste à reprendre l'échelonnement de A comme on l'avait fait à la question 10, par des opérations élémentaires sur les lignes, à deux nuances près. La première : on effectuera les mêmes opérations sur B que sur A. (Une autre solution serait d'incorporer B dans A, en rajoutant une colonne, mais pour plus de lisibilité, on gardera séparées A et B)

Une seconde modification : à chaque nouveau pivot trouvé (obtenu par la fonction `coefficientNonNulDepuis` de la question 9) on gardera trace de l'indice de colonne du pivot, car l'inconnue correspondante fait partie des inconnues principales. Je vous invite pour ce faire à introduire une liste `inconnuesPrincipales`, initialement vide, et de la remplir une à une des indices de ces inconnues. (Dans l'exemple présenté précédemment, on aurait à la fin obtenu la liste `[0, 2]`.)

Une fois ce premier travail effectué, on connaît le rang de A, mettons r, et seules les r premières lignes de A (on parle ici de la matrice obtenue après ces opérations, laquelle, informatiquement parlant, est toujours la liste représentée par `A`) sont non identiquement nulles. Une première conséquence est que si B (lui aussi modifié) ne comporte pas que des zéros à partir de la ligne d'indice r (on compte à partir de 0, il s'agit donc de la $r+1$-ième ligne), c'est que le système est incompatible.

Dans ce cas, on renvoit `None` et le travail est achevé.

Sinon, le système admet bel et bien des solutions. En notant $i_1<i_2<\cdots<i_r$ les inconnues principales, le système prend désormais la forme (en python, on retirera 1 aux indices) :

$$\begin{cases} a_{1,i_1}x_{i_1}+\ a_{1,i_1+1}x_{i_1+1}+\cdots+\cdots+\cdots+a_{1,p}x_p &= b_1 \\ a_{2,i_2}x_{i_2}+\cdots+\cdots+a_{2,p}x_p &= b_2 \\ \cdots &= \cdots \\ a_{r,i_r}x_{i_r}+\cdots+a_{r,p}x_p &= b_r \end{cases}$$

et on rappelle que les « pivots » $a_{1,i_1},\cdots,a_{r,i_r}$ sont tous non nuls. On obtient donc en premier lieu x_{i_r}, puis $x_{i_{r-1}}$ et ainsi de suite jusqu'à x_{i_1}, et ce en fonction des valeurs choisies pour les inconnues secondaires.

On calcule déjà X_0 une solution particulière. On choisit pour valeur 0 pour chacune des inconnues secondaires. En pratique, on part alors d'une liste `X0` comprenant p fois la valeur 0. Puis on remonte notre système pour déterminer les valeurs de chacune des inconnues principales, en commençant par la dernière (cela signifie qu'à chaque étape on modifie une valeur dans `X0`,

Je vous conseille à ce point d'introduire une seconde liste `inconnuesSecondaires` laquelle sera formée de tous les indices parmi $[\![0, p-1]\!]$ qui ne font pas partie de `inconnuesPrincipales` (bien sûr, on peut faire appel à `complement`)

On reprend, autant de fois que d'inconnues secondaires, les mêmes calculs que pour la solution particulière X_0 de l'équation complète, à ceci près que le second membre est désormais pris nul, et qu'au lieu de partir d'un vecteur colonne X initalement nul, on partira de X dont toutes les composantes sont nulles, sauf une, pour un indice de `inconnuesSecondaires`.

On construit ainsi une base de la direction $\operatorname{Ker} A$ de l'ensemble des solutions.

On peut alors renvoyer le couple `(X0, Direction)` où `X0` est une liste représentant une solution particulière, et `Direction` une liste de listes, représentant une base de la direction. Par exemple :

```
>>> A = [[1, 2, 3], [2, 4, 6], [3, 6, 8]]
>>> B = [0, 0, 1]
>>> resout(A, B)
    ([3, 0, -1], [[-2, 1, 0]])
```

Si on ne fait pas appel à `simplifieRationnels`, alors on obtiendra vraisemblablement plutôt, ce qui est moins élégant :

```
([Fraction(3, 1), 0, Fraction(-1, 1)], [[Fraction(-2, 1), 1, Fraction(0, 1)]])
```

Question 14. Niveau 2. Déduire de la fonction précédente une fonction d'entête `def Ker(A)` qui étant donnée une matrice A renvoie une base de son noyau.