2TSI. TD PYTHON 02

Matrices. Systèmes linéaires. Pivot de Gauss

Exercice 01 : opérations élémentaires classiques

❑ **Méthode 0.1.— Comment écrire en langage Python les opérations élémentaires classiques sur les lignes de matrices**

Il s'agit de créer les fonctions $L_i \leftarrow kL_i$ et $L_i \leftrightarrow L_j$. Aucune fonction prédéfinie n'existe.

- Pour ***effectuer la transvection*** $L_i \leftarrow L_i + kL_j$ sur A, on tape :

```
def transvecligne(A,i,j,k) :
    p = len(A[0])
    for m in range(p) :
        A[i,m] = A[i,m] + k*A[j,m]
```

Une version alternative est

```
def transvecligne(A,i,j,k) :
    A[i] = A[i] + k*A[j]
```

- Pour ***effectuer la dilatation*** $L_i \leftarrow kL_i$ sur A, on tape :

```
def dilatligne(A,i,k) :
    p = len(A[0])
    for m in range(p) :
        A[i,m] = k*A[i,m]
```

Une version alternative est :

```
def dilatligne(A,i,k) :
    A[i] = k*A[i]
```

- Pour ***effectuer la permutation*** $L_i \leftrightarrow L_j$ sur A, on tape :

```
def permutligne(A,i,j) :
    p = len(A[0])
    for m in range(p) :
        temp = A[i,m]; A[i,m] = A[j,m]; A[j,m] = temp
```

Une version alternative est :

```
def permutligne(A,i,j) :
    A[i] , A[j] = np.copy(A[j]) , np.copy(A[i])
```

1. Écrire les fonctions Python qui correspondent aux trois opérations sur les colonnes que l'on notera *transveccolonne*, *dilatcolonne* et *permutcolonne*.
   Indication : pour ***effectuer les mêmes opérations sur les colonnes,*** on adapte sans souci. Attention, cependant, c'est $A[:,j]$ qui désigne la colonne d'indice j.
2. Appliquer alors à transformer la matrice $A = \begin{pmatrix} 1 & -1 & 3 \\ 0 & 1 & 2 \\ -1 & -2 & 0 \end{pmatrix}$ en I_3 en utilisant plusieurs des six fonctions Python précédentes.

T.S.V.P →

Exercice 02 : résolution d'un système linéaire

❑ Méthode 0.2.— Comment résoudre un système linéaire

Il s'agit de résoudre $AX = B$.

- On peut appliquer une ***méthode directe*** lorsque A est carrée et inversible en employant la fonction *solve* du sous-module *numpy.linalg*. On tape $alg.solve(A, B)$ Nous obtenons X sous forme d'un tableau-ligne.
- On peut utiliser la fonction *solve_linear_system* du module *sympy* ***quand le système possède un paramètre***
  Ainsi dans le cas d'un système à deux inconnues x et y, si m est le paramètre dont dèpendent certains des coefficients des deux èquations $a\,x+b\,y=c$ et $d\,x+e\,y=f$, on tape :
  $m = symbols('m')\,;\ x = symbols('x')\,;\ y = symbols('y')$
  $syst = Matrix([[a,b,c],[d,e,f]])\,;\ solve\_linear\_system(syst, x, y)$
- On peut appliquer la ***méthode de Gauss*** et les fonctions construites dans la méthode ou au premier exercice.

1. Utiliser le module *sympy* ou *numpy.linalg* pour résoudre ($m \in \mathbb{R}$) :

$$\left\{\begin{array}{rcl} x+my+z & = & 1 \\ x+y+mz & = & 0 \\ 7x+(4+m)y+(3+2m)z & = & -1 \end{array}\right. , \quad \left\{\begin{array}{rcl} 10^{-20}x+y & = & 1 \\ x+y & = & 2 \end{array}\right. , \quad \left\{\begin{array}{rcl} 10^{-5}x+y & = & 1 \\ x+y & = & 2 \end{array}\right. .$$

   Que remarque-t-on ?

2. Écrire une fonction *systLin* qui résout le système linéaire de matrice A (carrée et inversible) et de second membre B par la méthode du pivot de Gauss. Les tableaux A et B seront les arguments de *systLin*. La fonction ne devra modifier ni la matrice A, ni la matrice B (il faudra en faire des copies, sur lesquelles on fera les opérations). On pourra utiliser les fonctions Python correspondantes aux opérations élémentaires sur les lignes vues à l'exercice 01.
   Tester la fonction *systLin* sur les deux systèmes $\left\{\begin{array}{rcl} 10^{-20}x+y & = & 1 \\ x+y & = & 2 \end{array}\right.$, $\left\{\begin{array}{rcl} 10^{-5}x+y & = & 1 \\ x+y & = & 2 \end{array}\right.$.