TELECOM Nancy *(ESIAL)*
Maths Numériques

TP2 : systèmes linéaires

Exercice 1 *Mini module d'algèbre linéaire*

Le but de cet exercice est de coder un petit module d'algèbre linéaire appelé syslin. L'exercice consiste à essayer d'écrire un "vrai" module python en essayant de documenter les fonctions un minimum (à l'aide de docstrings) et comprenant aussi un minimum de tests. On importera le module numpy en utilisant `import numpy as np`.

Ce module comprendra :

1. une fonction d'entête :

```
def lu_fact(A):
    """
    A est une matrice (n,n) dont on calcule
      la factorisation LU simple.

    Retourne
      * le tableau LU qui contient la factorisation de maniere compacte
      * un entier donnant le statut de calcul
        -1 si A n'est pas carree (noter qu'il est possible de
           conduire une factorisation sur une matrice rectangulaire)
         0 si la factorisation se passe (a priori) bien
         k > 0 si le pivot naturel à l'étape k est nul.
    """
```

qui effectue la factorisation $A = LU$ de la matrice A.

Remarques :

(a) le tableau d'entrée ne doit pas être modifié, la factorisation sera faite dans un tableau obtenu initialement par copie : `LU = A.copy()` ;

(b) les dimensions peuvent se récupérer avec `n,m = np.shape(A)`.

Pour tester cette première fonction, vous pouvez écrire une fonction `test1_lu_fact` en utilisant les données de l'exercice 3 du TD 2.

```
def donnees_exercice3_td2():
    """
    Retourne
        * la matrice A
        * le second membre b
        * la solution x du système Ax=b
        * les facteurs L et U (A = LU)

    du systeme lineaire de l'exercice 3 du td2
    """
    A = np.array([[1,2,3,4],[1,4,9,16],[1,8,27,64],[1,16,81,256]],float)
    b = np.array([2,10,44,190],float)
    x = np.array([-1,1,-1,1],float)
    L = np.array([[1,0,0,0],[1,1,0,0],[1,3,1,0],[1,7,6,1]],float)
    U = np.array([[1,2,3,4],[0,2,6,12],[0,0,6,24],[0,0,0,24]],float)
    return A, b, x, L, U
```

En vérifiant que vous obtenez les mêmes facteurs L et U. Pour extraire ces facteurs du stockage compact de la factorisation, on pourra utiliser :

```
LL = np.eye(4,4) + np.tril(LU,-1)
UU = np.triu(LU)
```

L'exécution des tests peut être faite à la fin du module après l'instruction :

```
if __name__ == '__main__':
    test1_lu_fact()
```

En général il est recommandé d'écrire les tests à part (et en s'aidant d'un module dédié aux tests) mais ce "truc" permet de tout avoir (code, doc et tests) dans le même fichier. En exécutant ce script "directement" (par exemple via une instruction `python3 syslin.py` dans une console), il est donc considéré comme "main" et la partie après le if est exécutée.

2. une autre fonction d'entête :

```
def lu_solve(LU,b):
    """
    Retourne la solution du système Ax = b où LU est la
    factorisation (en stockage compact) LU de la matrice A
    (obtenue à l'aide de la fonction lu_fact).
    """
```

pour résoudre un système linéaire, lorsque l'on connaît la factorisation $A = LU$. Pour tester votre fonction vous pouvez écrire une autre fonction `test1_lu_solve` pour essayer de retrouver la solution du système linéaire de l'exercice 3 de la feuille 2 de TD.

3. Essayer de trouver d'autres tests possibles.