Informatique en CPGE (2015-2016)
Corrigé TD 8 : systèmes linéaires

1 Résolution d'un système tridiagonal

Nous supposons n=10.

1. Ecrire la définition des vecteurs a, b et c, trois listes de longueur n, en utilisant le procédé suivant :
   - donner des valeurs aléatoires réelles entre 0 et 1 aux éléments de a et c ;
   - corriger : a[0]=0 ; c[n-1]=0 ;
   - calculer : b[i] = a[i]+c[i]+1, 0<= i <n.
2. Ecrire la définition de la matrice M du système en utilisant une liste de n listes de longueur n.
3. Ecrire la définition du vecteur F de taille n, défini par : F[i] = a[i]+b[i]+c[i], 0<= i <n.
4. Résoudre le système avec la fonction solve de scipy.linalg.
   Quels valeurs doit-on obtenir pour X[i] ? Tester le programme avec différentes valeurs de n.

```
from scipy.linalg import solve
from random import random

n=10

a=[random() for i in range(n)]
c=[random() for i in range(n)]
a[0]=0
c[n-1]=0
b=[a[i]+c[i]+1 for i in range(n)]

M=[n*[0] for i in range(n)]

for i in range(n):
    M[i][i]=b[i]
for i in range(1,n):
    M[i][i-1]=a[i]
for i in range(n-1):
    M[i][i+1]=c[i]

F=[a[i]+b[i]+c[i] for i in range(n)]

X=solve(M,F)

#print(X) X=(1,1,1,1, ... , 1)
```

2 Décomposition LU

1. Ecrire une fonction **factor_LU** qui prend en argument trois listes a, b et c comme ci-dessus et renvoie deux listes b1 et c1 construites avec les relations de récurrence données plus haut.

2. Ecrire une fonction **res_LU** qui prend en argument la liste a, les deux listes renvoyées par la fonction **factor_LU** et la liste représentant le second membre du système et renvoie la liste x solution du système.
3. Ecrire une fonction **solution** qui prend en argument quatres listes a, b, c, et F représentant le système et renvoie la liste x solution du système.
4. Tester le programme avec les listes définies dans la partie 1.
5. Comparer les temps d'exécution pour n=1000, n=5000. Commentaires ?

```
def factor_LU(a,b,c):
    c[0]/=b[0]
    for i in range(1,len(a)):
        b[i]-=a[i]*c[i-1]
        c[i]/=b[i]
    return b,c

def res_LU(a,b,c,f):
    n=len(a)
    x=n*[0]
    x[0]=f[0]/b[0]
    for i in range(1,n):
        x[i]=(f[i]-a[i]*x[i-1])/b[i]
    for i in range(n-2,-1,-1):
        x[i]-=c[i]*x[i+1]
    return x

def solution(a,b,c,f):
    b,c=factor_LU(a,b,c)
    u=res_LU(a,b,c,f)
    return u

from time import time
st=time()
X=solution(a,b,c,F)
print(time()-st)
```

Avec n=1000, on obtient la solution en 0,25 s avec la fonction **solve** et 0,001 s avec la méthode LU.
Avec n=5000, on obtient la solution en 14 s avec la fonction **solve** et 0,008 s avec la méthode LU.
Avec n=7000, on obtient la solution en 50 s avec la fonction **solve** et 0,01 s avec la méthode LU.
Avec n=9000, le programme plante et affiche "MemoryError" avec la fonction **solve** alors qu'avec la méthode LU la solution est donnée en 0,013 s !