Algorithme du pivot de Gauss

D.Malka – MPSI 2015-2016 – Lycée Saint-Exupéry

Avec pivot naturel

```python
from pylab import*
from numpy.linalg import solve
from random import*

#FONCTIONS
def matrice_alea(n,p):
    """
    Genere une matrice a coeff entiers aleatoires, type : array
    n,p: dimensions de la matrice, type : int
    A : array de dimension (n,p)
    """
    M=zeros((n,p))
    for i in range(n):
        for j in range(p):
            M[i,j]=randint(1,100)
    return M

def transvect(A,i,j,c):
    """
    Transvection entre les lignes i et j : Lj<-Lj-c*Lj,
    Realise les transvection en place
    A : marice carre 2D, type : 2D-array, dim=(n,n)
    i,j : int, indice des lignes
    c : type :float
    Retour : None
    """
    p=A.shape[1]#nb de colonnes = longueur d'une ligne
    for k in range(p):
        A[j,k]=A[j,k]-c*A[i,k]
    return None

def triang(A,Y):
    """
    Calcule le systeme triangulaire superieur TX=Yp <=> AX=Y par la methode du pivot de gauss partiel
     T : matrice 2D carre triangulaire superieure, type : 2D-array, dim=(n,n)
    Yp : matrice 2D, type : 2D-array , dim=(n,1)
    T : matrice 2D carre triangulaire superieure, type : 2D-array, dim=(n,n)
    Yp : matrice 2D, type : 2D-array , dim=(n,1)
    """
    T=A[:,:]
    Yp=Y[:,:]
    n,p=Y.shape
    for i in range(n):
        pivot=T[i,i]#par permutation de ligne aii est le grand des aki
        for j in range(i+1,n):
            c=T[j,i]/pivot
            # Pour tout j>i, Lj<-Lj-aji/aii*Lj
            transvect(T,i,j,c)
            transvect(Yp,i,j,c)
    return T,Yp

def remonte(T,Yp):
    """
    Resout le systeme triangulaire superieur TX=Yp
    T : matrice 2D carre triangulaire superieure, type : 2D-array, dim=(n,n)
    Yp : matrice 2D, type : 2D-array , dim=(n,1)
```

```
    X : matrice 2D, type : 2D-array, dim=(n,1)
    """
    n=len(Yp)
    X=zeros((n,1))
    for i in range(n):
        s=0
        for k in range(n-i,n):
            s=s+T[n-i-1,k]*X[k]
        X[n-i-1,0]=1/T[n-i-1,n-i-1]*(Yp[n-i-1,0]-s)
    return X

def resol(A,Y):
    T,Yp=triang(A,Y)
    X=remonte(T,Yp)
    return X
```

Avec pivot partiel

```
from pylab import*
from numpy.linalg import solve
from random import*

#FONCTIONS
def matrice_alea(n,p):
    """
    Genere une matrice a coeff entiers aleatoires, type : array
    n,p: dimensions de la matrice, type : int
    A : array de dimension (n,p)
    """
    M=zeros((n,p))
    for i in range(n):
        for j in range(p):
            M[i,j]=randint(1,100)
    return M

def transvect(A,i,j,c):
    """
    Transvection entre les lignes i et j : Lj<-Lj-c*Lj,
    Realise les transvection en place
    A : marice carre 2D, type : 2D-array, dim=(n,n)
    i,j : int, indice des lignes
    c : type :float
    Retour : None
    """
    p=A.shape[1]#nb de colonnes = longueur d'une ligne
    for k in range(p):
        A[j,k]=A[j,k]-c*A[i,k]
    return None

def permute_ligne(A,i,j):
    """
    Permute , en place, les lignes i et je de le matrice A
    A: matrice carre 2D, type: 2D-array, dim=(n,n)
    i,j : indice des lignes, type : int
    Retour : None
    """
    p=A.shape[1]##nb de colonnes = longueur d'une ligne
    for k in range(p):
        A[i,k],A[j,k]=A[j,k],A[i,k]#permutation des valeurs de A[i,k],A[j,k]
    return None

def rech_pivot(A,Y,i):
    """
    Methode du pivot partiel. Positionne a la ligne i le plus grand pivot de la colonne i.
    Echange la ligne k avec la ligne j ou aik=max(aij) pour j de i a n-1. Echange pour Y aussi;
    A: matrice carre 2D, type : 2D-array, dim=(n,n)
    Y: matrice 2D, type : 2D-array, dim=(n,1)
    i: indice de ligne, type : int
    """
    n=A.shape[0]
    k=i
    for j in range(i+1,n):
        if abs(A[j,i])>abs(A[k,i]):
            k=j
    permute_ligne(A,i,k)
    permute_ligne(Y,i,k)
    return None

def triang(A,Y):
    """
    Calcule le systeme triangulaire superieur TX=Yp <=> AX=Y par la methode du pivot de gauss partiel
     T : matrice 2D carre triangulaire superieure, type : 2D-array, dim=(n,n)
    Yp : matrice 2D, type : 2D-array , dim=(n,1)
    T : matrice 2D carre triangulaire superieure, type : 2D-array, dim=(n,n)
    Yp : matrice 2D, type : 2D-array , dim=(n,1)
    """
    T=A[:,:]
    Yp=Y[:,:]
    n,p=Y.shape
    for i in range(n):
        rech_pivot(T,Yp,i)#pivot partiel : plus grand aki de la colonne i
        pivot=T[i,i]#par permutation de ligne aii est le grand des aki
```

```
        for j in range(i+1,n):
            c=T[j,i]/pivot
            # Pour tout j>i, Lj<-Lj-aji/aii*Lj
            transvect(T,i,j,c)
            transvect(Yp,i,j,c)
    return T,Yp

def remonte(T,Yp):
    """
    Resout le systeme triangulaire superieur TX=Yp
    T : matrice 2D carre triangulaire superieure, type : 2D-array,
        dim=(n,n)
    Yp : matrice 2D, type : 2D-array , dim=(n,1)
    X : matrice 2D, type : 2D-array, dim=(n,1)
    """
    n=len(Yp)
    X=zeros((n,1))
    for i in range(n):
        s=0
        for k in range(n-i,n):
            s=s+T[n-i-1,k]*X[k]
        X[n-i-1,0]=1/T[n-i-1,n-i-1]*(Yp[n-i-1,0]-s)
    return X

def resol(A,Y):
    T,Yp=triang(A,Y)
    X=remonte(T,Yp)
    return X
```