



ALGORITHME DU PIVOT DE GAUSS

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

AVEC PIVOT NATUREL

```
1 from pylab import*
2 from numpy.linalg import solve
3 from random import*
4
5 #FONCTIONS
6 def matrice_alea(n,p):
7     """
8     Genere une matrice a coeff entiers aleatoires, type : array
9     n,p: dimensions de la matrice, type : int
10    A : array de dimension (n,p)
11    """
12    M=zeros((n,p))
13    for i in range(n):
14        for j in range(p):
15            M[i,j]=randint(1,100)
16    return M
17
18 def transvect(A,i,j,c):
19     """
20     Transvection entre les lignes i et j : Lj<-Lj-c*Lj,
21     Realise les transvection en place
22     A : marice carre 2D, type : 2D-array, dim=(n,n)
23     i,j : int, indice des lignes
24     c : type :float
25     Retour : None
26     """
27    p=A.shape[1]#nb de colonnes = longueur d'une ligne
28    for k in range(p):
29        A[j,k]=A[j,k]-c*A[i,k]
30    return None
```

```
31
32 def triang(A,Y):
33     """
34     Calcule le systeme triangulaire superieur TX=Yp <=> AX=Y par la
35     methode du pivot de gauss partiel
36     T : matrice 2D carre triangulaire superieure, type : 2D-array,
37     dim=(n,n)
38     Yp : matrice 2D, type : 2D-array , dim=(n,1)
39     T : matrice 2D carre triangulaire superieure, type : 2D-array,
40     dim=(n,n)
41     Yp : matrice 2D, type : 2D-array , dim=(n,1)
42     """
43    T=A[:,:]
44    Yp=Y[:,:]
45    n,p=Y.shape
46    for i in range(n):
47        pivot=T[i,i]#par permutation de ligne aii est le grand des
48        aki
49        for j in range(i+1,n):
50            c=T[j,i]/pivot
51            # Pour tout j>i, Lj<-Lj-aji/aai*Lj
52            transvect(T,i,j,c)
53            transvect(Yp,i,j,c)
54    return T,Yp
55
56 def remonte(T,Yp):
57     """
58     Resout le systeme triangulaire superieur TX=Yp
59     T : matrice 2D carre triangulaire superieure, type : 2D-array,
60     dim=(n,n)
61     Yp : matrice 2D, type : 2D-array , dim=(n,1)
```

```
57 X : matrice 2D, type : 2D-array, dim=(n,1)
58 """
59 n=len(Yp)
60 X=zeros((n,1))
61 for i in range(n):
62     s=0
63     for k in range(n-i,n):
64         s=s+T[n-i-1,k]*X[k]
```

```
65     X[n-i-1,0]=1/T[n-i-1,n-i-1]*(Yp[n-i-1,0]-s)
66     return X
67
68 def resol(A,Y):
69     T,Yp=triang(A,Y)
70     X=remonte(T,Yp)
71     return X
```

AVEC PIVOT PARTIEL

```

1 from pylab import*
2 from numpy.linalg import solve
3 from random import*
4
5 #FONCTIONS
6 def matrice_alea(n,p):
7     """
8     Genere une matrice a coeff entiers aleatoires, type : array
9     n,p: dimensions de la matrice, type : int
10    A : array de dimension (n,p)
11    """
12    M=zeros((n,p))
13    for i in range(n):
14        for j in range(p):
15            M[i,j]=randint(1,100)
16    return M
17
18 def transvect(A,i,j,c):
19    """
20    Transvection entre les lignes i et j : Lj<-Lj-c*Lj,
21    Realise les transvection en place
22    A : marice carre 2D, type : 2D-array, dim=(n,n)
23    i,j : int, indice des lignes
24    c : type :float
25    Retour : None
26    """
27    p=A.shape[1]#nb de colonnes = longueur d'une ligne
28    for k in range(p):
29        A[j,k]=A[j,k]-c*A[i,k]
30    return None
31
32 def permute_ligne(A,i,j):
33    """
34    Permute , en place, les lignes i et je de le matrice A
35    A: matrice carre 2D, type: 2D-array, dim=(n,n)
36    i,j : indice des lignes, type : int
37    Retour : None
38    """
39    p=A.shape[1]##nb de colonnes = longueur d'une ligne
40    for k in range(p):
41        A[i,k],A[j,k]=A[j,k],A[i,k]#permutation des valeurs de A[i,k

```

```

42    ],A[j,k]
43
44 def rech_pivot(A,Y,i):
45    """
46    Methode du pivot partiel. Positionne a la ligne i le plus grand
47    pivot de la colonne i.
48    Echange la ligne k avec la ligne j ou aik=max(aij) pour j de i a
49    n-1. Echange pour Y aussi;
50    A: matrice carre 2D, type : 2D-array, dim=(n,n)
51    Y: matrice 2D, type : 2D-array, dim=(n,1)
52    i: indice de ligne, type : int
53    """
54    n=A.shape[0]
55    k=i
56    for j in range(i+1,n):
57        if abs(A[j,i])>abs(A[k,i]):
58            k=j
59    permute_ligne(A,i,k)
60    permute_ligne(Y,i,k)
61    return None
62
63 def triang(A,Y):
64    """
65    Calcule le systeme triangulaire superieur TX=Yp <=> AX=Y par la
66    methode du pivot de gauss partiel
67    T : matrice 2D carre triangulaire superieure, type : 2D-array,
68    dim=(n,n)
69    Yp : matrice 2D, type : 2D-array , dim=(n,1)
70    T : matrice 2D carre triangulaire superieure, type : 2D-array,
71    dim=(n,n)
72    Yp : matrice 2D, type : 2D-array , dim=(n,1)
73    """
74    T=A[:,:]
75    Yp=Y[:,:]
76    n,p=Y.shape
77    for i in range(n):
78        rech_pivot(T,Yp,i)#pivot partiel : plus grand aki de la
79        colonne i
80        pivot=T[i,i]#par permutation de ligne aii est le grand des
81        aki

```

```
75     for j in range(i+1,n):
76         c=T[j,i]/pivot
77         # Pour tout j>i, Lj<-Lj-aji/aii*Lj
78         transvect(T,i,j,c)
79         transvect(Yp,i,j,c)
80     return T,Yp
81
82 def remonte(T,Yp):
83     """
84     Resout le systeme triangulaire superieur TX=Yp
85     T : matrice 2D carre triangulaire superieure, type : 2D-array,
86         dim=(n,n)
87     Yp : matrice 2D, type : 2D-array , dim=(n,1)
88     X : matrice 2D, type : 2D-array, dim=(n,1)
89     """
```

```
89     n=len(Yp)
90     X=zeros((n,1))
91     for i in range(n):
92         s=0
93         for k in range(n-i,n):
94             s=s+T[n-i-1,k]*X[k]
95         X[n-i-1,0]=1/T[n-i-1,n-i-1]*(Yp[n-i-1,0]-s)
96     return X
97
98 def resol(A,Y):
99     T,Yp=triang(A,Y)
100    X=remonte(T,Yp)
101    return X
```