Méthode du pivot de Gauss – Corrigé

Partie n°1: Pivot de Gauss

1.

Python
```
A = [[2,2,-3],[-2,-1,-3],[6,4,4]]
Y = [2,-5,16]
```

2.

Python
```
import copy          # pour utiliser la fonction deepcopy
                     # et réaliser une copie réellement indépendante

def matrice_aug(A,B) :
    """Fonction qui prend en argument un tableau A (matrice carrée nxn : type list )
    et un vecteur B (n éléments, type list )
    et qui renvoie la matrice M =A|B (type : list ).
    """
    n = len(A)
    M=A                      # attention en réalité on modifie A sinon M=copy.deepcopy(A)
    for i in range(0,n):     # boucle permettant de balayer les lignes de la matrice A
        M[i].append(B[i])    # à la fin de chaque ligne, on ajoute le ième élément de B : B[i]
    return M


def chercher_pivot(Mat,ind):
    """prend en argument un tableau (n lignes, n+1 colonnes) de type list et
    un entier ind (compris entre 0 et n-1)
    renvoie l'indice du pivot (type int)"""
    n=len(Mat)           # nombre de lignes
    ind_pivot=ind                                  # la ligne du maximum provisoire
    for k in range(ind+1,n): # recherche du pivot sur la matrice carrée n-ind (en bas à droite)
            if abs(Mat[k][ind])>abs(Mat[ind_pivot][ind]): # comparaison des coefficients
                                 # de la colonne ind en balayant les différentes lignes
                ind_pivot=k              # nouveau maximum provisoire
    return j
```

Python
```
def echanger_lignes(Mat,i,j):
    """
    prend en argument un tableau Mat à deux dimensions (n lignes) de type list et deux entiers i et j
    (compris entre 0 et n-1)
    permute les lignes d'indice i et j.
    ne renvoie rien (None)
    """
    Mat[i], Mat[j] = Mat[j], Mat[i]

def combinaison(Mat,i,j,mu):
    """
    fonction qui prend en arguments un tableau à deux dimensions Mat (n lignes) de type list,
    2 entiers i et j compris entre 0 et n-1
    un flottant mu
    Cette fonction modifie Mat (combinaison linéaire : Li <- Li + mu*Lj)
    Ne renvoie rien (None).
    """
    nc=len(Mat[0])              # nombre de colonnes (la matrice peut être augmentée)
    for k in range(nc):         # on parcourt les colonnes à i et j fixé.
        Mat[i][k] = Mat[i][k]+Mat[j][k]*mu
```

3.

Python
```
def resolution (A0,Y0) :
    """
    prend en argument une matrice carrée A nxn (type list) et un vecteur Y à n éléments.
    Renvoie un vecteur X à n éléments (de type list).
    X est la solution du système linéaire A0.X= Y0
    nécessite la fonction copy.deepcopy (du module copy).
    """
    """
    Etape 1 : copie et création de la matrice augmentée
    """
    A,Y = copy.deepcopy(A0), copy.deepcopy(Y0)  # permet de ne pas modifier les listes d'origine.
    M =matrice_aug(A,Y)
    n = len(A)        # taille du système nb de lignes

    """
    Etape 2 : Mise sous forme triangulaire
    """
    for i in range(n):
        ind_pivot=chercher_pivot(M,i)
        echanger_lignes(M,i,pivot)
        pivot = M[i][i]
        for k in range(i+1,n):              # parcours des différents lignes
            coeff=M[k][i]/pivot             # recherche du coeff pour annuler le terme correspondant
            combinaison(M,k,i,-coeff)       # combinaison linéaire pour annuler
                                            # le terme de la ième colonne
```

Python
```
    """
    Etape 3 : Phase de remontée
    """
    X=[0]*n    # initialisation de la liste X ( solutions )
    for k in range(n-1,-1,-1):       # on balaie la matrice en partant
                                     # de la dernière ligne et en remontant
        somme =0
        for j in range(k+1,n):
            somme +=M[k][j]*X[j]
        X[k] = (M[k][n] - somme)/M[k][k]
    return X
```

4.

Python
```
>>> resolution (A,Y)
[-14.000000000000036, 21.000000000000046, 4.000000000000007]
```

Partie n°2: Application : Passerelle télescopique d'aéroport

Python
```
a= 1.5 # Longueurs en mètres
d = 5
e = 5
h = 4
l = 15
m1 =5000 # masses en kg
m2 =6000
g=10    # accélération de la pesanteur
y0= 16
```

Python

```
Atele=[
    [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-1],
    [0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,-y0],
    [0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,1,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,-a,0,0,0,0,0,0,-2*a,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,2*a,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0],
    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0],
    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0]]

Btele=[0,0,(m1+m2)*g,y0*m2*g+d*m1*g,-e*m2*g,0,0,0,0,e*m2*g,
    0,0,m1*g-d*m1*g/y0 -m1*g,m1*g*l/(2*y0)+d*m2*g/y0 ,e*m2*g,
    0,0,0,m2*g+d*m2*g/y0 ,0,0]
```

Python

```
solution   =  resolution  (Atele , Btele )

Xtele=["X04","Y04","Z04","L04","M04",
    "XA","YA","ZA",
    "YB","ZB",
    "X12","Z12","L12","M12","N12",
    "X23","Y23","Z23","L23","M23",
    "Z30"]

for  i  in  range( len (Xtele )):
     print (Xtele [ i ] , "=", solution [ i ])
```

Python

```
X04 =0.0
Y04 =0.0
Z04 =34375.0
L04 =0.0
M04 =-300000.0
XA =0.0
YA =0.0
ZA =82812.5
YB =0.0
ZB =-117187.5
X12 =0.0
Z12 =-15625.0
L12 =42187.5
M12 =300000.0
N12 =0.0
X23 =0.0
Y23 =0.0
Z23 =78750.0
L23 =0.0
M23 =0.0
Z30 =-75625.0
```