CCP PSI 2022 Informatique

1 code

```
## Ecrit par Manuel Bricard. Relu par Sylvain Pelletier.

##### Assemblage et deformation de pieces automobiles
from numpy import *

## Partie 1 : posage des pieces
# pour les tests
P0=[2.,3.,2.,3.]
P1=[3.,1.,2.,3.]
Delta_x=0.1

# Q1
P2=[3.,3.,2.,2.]

# Q2
def retourne(Ls) :
    return [-Ls[k] for k in range(len(Ls)-1,-1,-1)]
#print(retourne(P2))

# Q3
"Il y a collision pour le posage au points 2 et 3"

# Q4
def droite(Ls,n,p) :
    coefDir=(Ls[n]-Ls[p])/((n-p)*Delta_x)
    ordoOrigine=Ls[n]-n*(Ls[n]-Ls[p])/(n-p)
```

```
    return coefDir,ordoOrigine
#print(droite(P0,0,1))

# Q5
def posage(Ls1,Ls2,n,p) : # Ls2 est retournee mais cela ne change rie
    # on calcule l'ecart entre la droite les points pour les deux dro
    # avec une piece a plat, la droite a donc une equation differente
    coef1,ordo1=droite(Ls1,n,p) # equation de la droite associee a la
    coef2,ordo2=droite(Ls2,n,p) # idem avec la piece 2
    #
    altMin=10000 # on pourrait prendre inf en important la bibliotheq
    altMax=-10000 # idem
    for i in range(len(Ls1)) :
        # calcul de l'altitude pour Ls1 puis Ls2, on considere a chaq
        # que les pieces sont posees a plat
        alt1=(coef1*i*Delta_x+ordo1) - Ls1[i]  # l'altitude est la di
        alt2= Ls2[i] - (coef2*i*Delta_x+ordo2)   # c'est different po
        print(alt1,alt2)
        alt=alt2+alt1
        if alt>altMax : # recherche du min et du max, il faudra au
moins 2 points
            altMax=alt
        elif alt<altMin :
            altMin=alt
    return [altMax,altMin]
print(posage(P0,retourne(P1),0,1))
# Q6
def posage_opt(Ls1,Ls2) :
    dMax=-10
    absPoints=(-1,-1) # initialisation sur des valeurs non pertinente
    for i in range(len(Ls1)-1) : # construction des couples de positi
        for j in range(i+1,len(Ls1)) :
            altMaxi,altMini=posage(Ls1,Ls2,i,j)
            if altMini>=0 :
                if altMaxi>dMax :
                    dMax=altMaxi
                    absPoints=(i,j)
    return absPoints
```

```
#print(posage_opt(P0,retourne(P1)))
" Soit n le nombre de sommets, la double boucle en i et j est en O(n2


## Partie 2 Maillage de surface
T=[[0,1,2],[2,3,0]]
noeud=array([[0.,0.],[0.,2.],[2.,2.],[2.2,0.],[0.4,1.]])
# Q7
"On commence par decouper (0,1,2) en (0,1,4),(1,2,4),(0,2,4)"
"On voit alors que (0,2,4) et (0,2,3) ne repecte pas Delaunay, on tou
"pour obtenir au final (0,1,4),(1,2,4),(0,3,4),(2,3,4)"

# Q8
def addT(indN,indT) :
    triangle=T[indT]
    T.remove(triangle)
    for i in range(len(triangle)) :
        nouvTri=triangle[:] # on copie le triangle pour le modifier
        nouvTri[i]=indN
        T.append(nouvTri)
addT(4,0)
#print(T)

# Q9
"Ledge contiendra les sommets de l'arete commune actuelle, Lflip cont
"Proposition 1 : on ajoute trop de sommets, quand on verra un sommet
"OK Proposition 2 : premiere boucle on range les bons sommets de T2 d
"Proposition 3 : Ce serait bien si les sommets communs etaient situes

# Q10
"Mathematiquement, un produit vectoriel n'a pas de signe puisque c'es
"Ce qu'utilise l'enonce est le fait que les points sont tous dans un
"sur z qui est le determinant de la matrice des coordonnees faites av
"Le signe de ce determinant donnant la direction du produit on retrou
"cross() calcule bien un produit vectoriel en dimension 3"
"Pour eviter les problemes d'ordre j'aurais plutot utilise (AB,AM)*(A
def insideT(indN,T1) :
    T=T1[:] # on copie le triangle pour pouvoir modifier l'ordre des
```

```
    for i in range(3) : # on doit faire 3 verifications, on changera
        v1=noeud[T[1]]-noeud[T[0]]
        v2=noeud[indN]-noeud[T[0]]
        v3=noeud[T[2]]-noeud[T[0]]
        if cross(v1,v2)*cross(v1,v3)<0 : # l'enonce suppose que le no
            return False
        T=T[1:]+[T[0]] # on permute l'ordre des sommets pour changer
    return True
print(insideT(4,[0,1,2]))
print(insideT(3,[0,1,2]))

# Q11
"""je pense que la definition de checkedge() est erronee, l'enonce di
Cette fonction doit d'abord verifier si le noeud insere est inclus da
voisins.
Mais les parametres sont le nouveau sommet et un triangle le contenan
construit sur le cote oppose au nouveau sommet. Ce nouveau sommet ser
des autres voisins
Une formulation correcte serait :
Cette fonction doit d'abord verifier si le noeud insere est inclus da
voisin construit sur le cote oppose au sommet traite. (puis parler du
"petite erreur de syntaxe, la fonction definie initialement est circl
"indA est l'indice du triangle construit sur le cote oppose au sommet
"C,R sont les caracteristiques du cercle circonscrit au triangle d'in
"le test agit si le sommet d'indice indN est dans ce cercle circonscr

"la ligne 3 gere le cas ou l'on est au bord du maillage, et le triang
"""ligne6,7,8 :
puisque indN est dans le cercle du triangle oppose on utilise flip()
ligne 6 :
    flip(indT,indA)
pour la suite on propage la verification sur les voisins du voisin, e
ligne 7 et 8 :
    checkedge(indN,indT) # le triangle d'indice indT a change
    checkedge(indN,indA) # celui d'indice indA aussi
"""
"je me pose pas mal de questions sur la construction de delaunay dans
"au fur et a mesure, une modification risquant d'impacter les modific
```

```
# Q12
def delaunay(noeud) :
    T=[[0,1,2],[2,3,0]]
    for i in range(4,len(noeud)) :
        for j in range(len(T)) :
            if insideT(i,T[j]) : # parmi tous les triangle existant o
                addT(i,j)
                for k in range(1,4) : # on verifie que les nouveaux t
                    checkedge(i,T[-k])
    return T

## partie 3
"ok pour la resolution de systeme d'edo (mais rien n'est demande)"

# Q13

x = linspace(0,L,n)
uth = (-p0*x*x/2 + (Fx+L*p0)*x)/(E*S)
plot(x,uth)
show() # pas donne dans l'annexe donc probablement non attendu


"La mise en place est tres longue"
# Q14
def phi(k,x) : # x est un vecteur global est un flottant local, bof !
    pas=L/N
    if x < (k-1)*pas or x>(k+1)*pas :  # la variable globale doit etr
        return 0
    elif x<k*pas : # else non obligatoire en raison du return
        return (x-(k-1)*pas)/pas   # thales ou autre
    else :
        return ((k+1)*pas-x)/pas

# Q15
" phi' vaut 0,1/n2 ou -1/n2 en fonction de la valeur de x, on prendra
"""
pour i=j  : le produit vaut 1/n2 sur ](i-1)*L/n,(i+1)*L/n] et 0 aille
```

```
pour i=j=0 : le produit vaut 1/n2 sur [0,L/n] et 0 ailleurs , meme si
pour i=j=n : le produit vaut 1/n2 sur [(n-1)*L/n,L] et 0 ailleurs
pour |i-j|=1 : le produit vaut -1/n2 sur [(i+j-1)/2*L/n,(i+j+1)/2*L/n
pour les autres : 0
"""
"""
Pour les integrales on multiplie par la longueur de l'intervalle
pour i=j : 2*L/n^3
pour i=j=0 et i=j=n : L/n^3
pour |i-j|=1 : -L/n^3
pour les autres : 0
"""

# Q16

"En identifiant avec ce qui precede : C=L/n et alpha=2"

# Q17

"A = E*S*M et B = P + Fx * vecteurColonne([0,....,0,1])"
A = E*S*M
col=zeros((1,N)) # construction du vecteur colonne
col[0,N-1]=1
B = P + Fx*col

# Q18
"ligne 3 : pourquoi len(M) et pas len(R) ?"

"Les lignes 2 a 9 construise la matrice etendue du systeme : la matri
"mais Syst est une liste de liste et pas un array, c'est bof !"

# Q19
"Le tableau Syst est en cours de triangularisation, triangulaire supe
"La question n'est pas du tout evidente. Le fait de prendre le plus g
"d'arrondi lors du calcul de mu. Il permet aussi d'eviter de prendre
"alors qu'il devrait etre nul"
def pivot(M,ind) :
    """M est une matrice, ind est un numero de colonne de cette matri
```

```
    La fonction recherche le plus grand element en valeur absolu de c
    maxi=-1 # on prend la valeur absolue des elements, les valurs uti
    indMaxi=-1
    for i in range(ind,len(M)) :
        val=abs(M[i][ind])
        if val>maxi :
            maxi=val
            indMaxi=i
    return indMaxi

# Q20

mu=-Syst[k][i]/Syst[i][i]
def combinaison(Syst,k,i,x) :
    for j in range(len(Syst[k])) : # on aurait pu se contenter de ne
        Syst[k][j]+=mu*Syst[i][j]

# Q21
for k in range(len(Syst)-1,-1,-1) : # Comme on travaille avec des arr
    X[k]=Syst[k][len(Syst)]
    for i in range(k+1,len(Syst)) :
        X[k]-=Syst[k][i]*X[i]
    X[k]=X[k]/Syst[k][k]

# Q22
"La resolution de AU=B se fait avec"
U=resolution(A,B)

def uEF(vect,f,x) :
    res=0
    for k in range(len(vect)) :
        res+=vect[k]*f(k,x)
    return res
# a utiliser avec uEF(U,phi,x)
## Partie 4

# Q23
" La porte du bas semble legere plus claire, donc avec une deformatio
```

"␣dans␣la␣cellule.␣Ceci␣dit␣le␣calcul␣de␣la␣deformation␣signifie−t−il

 # Q24
"""
J'ai un probleme avec le tableau et le graphique. Le tableau dit que
maximal est important.
Mais le graphique dit le contraire.
Pour repondre a la question : avec un maillage de 4957 noeuds, le pre
(si l'on suppose que plus il y a de noeuds, plus le resultat est prec

 # Q25
"Peut−etre␣qu'avec␣5000␣elements␣on␣est␣assez␣fin␣pour␣avoir␣au␣moins