SSNP164 – Joints 2D et 3D pour la loi JOINT_MECA_ENDO

Résumé :

Ce test a pour objet la validation des éléments de joints hydro-mécaniques pour la loi de comportement des barrages JOINT_MECA_ENDO.

On teste à la fois les modélisations mécaniques pures PLAN_JOINT en $2D$ (maille QUAD4, QUAD8) et 3D_JOINT en $3D$ (maille HEXA8, HEXA20 ou PENTA6, PENTA15) en présence de pression imposée. Puis, pour les même éléments quadratiques on teste également les modélisations hydro-mécaniques couplées PLAN_JOINT_HYME et C3D_JOINT_HYME, avec le profil de sous-pression obtenue dans le calcul.

Les éléments sur lesquels s'appuient ces modélisations sont les éléments de joints cohésifs standards.

Modélisation A : PLAN_JOINT maille QUAD4 avec JOINT_MECA_RUPT test de traction
Modélisation B : PLAN_JOINT maille QUAD4 avec JOINT_MECA_RUPT test de cisaillement
Modélisation C :
Modélisation D :
Modélisation E :
Modélisation F :

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

1 Problème de référence

1.1 Géométrie

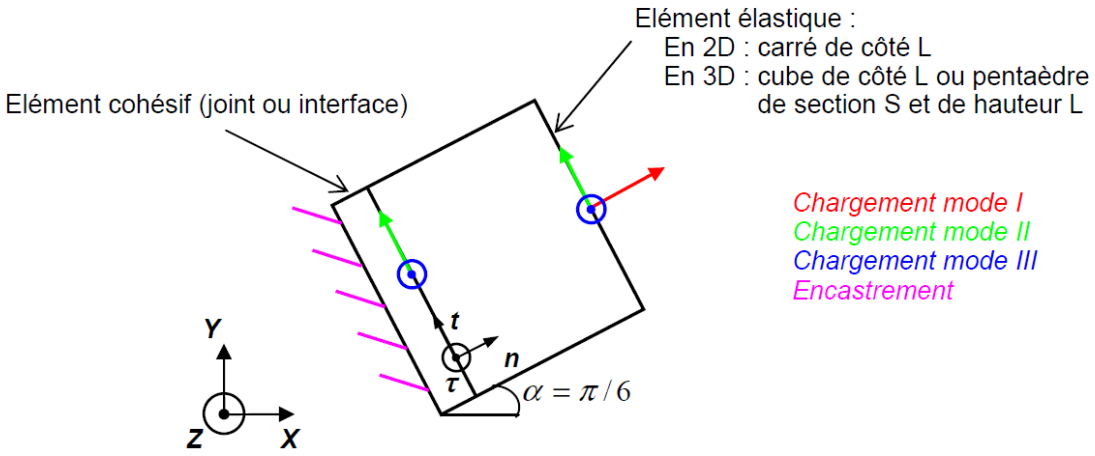


Figure 1 : Représentation du système de deux éléments dans le plan (X, Y).

On choisit $L = 1\,mm$.

1.2 Propriétés du matériau

Cube : élastique
Module d'Young : $E = 3 \times 10^{12}\,Pa$
Coefficient de Poisson : $\nu = 0$

rigidité normale :	$K_N = 10^{12}\,Pa/m$	(mot clé : K_N)
rigidité tangentielle :	$K_T = 2 \times 10^{12}\,Pa/m$	(mot clé : K_T)
Seuil d'activation de l'endommagement :	$\alpha = 10^{-7}$	(mot clé : ALPHA)
paramètre de lissage de rupture fragile	$D1 = 1$	(mot clé : PENA_RUPTURE)
Masse volumique du fluide (uniquement pour modélisations HM)	$\rho_{fluide} = kg/m^3$	(mot clé : RHO_FLUIDE)
Viscosité dynamique du fluide (uniquement pour modélisations HM)	$\mu_{fluide} = Pa.s$	(mot clé : VISC_FLUIDE)
Ouverture de régularisation de l'écoulement (uniquement pour modélisations HM)	$\delta_{min} = m$	mot clé : OUV_MIN)

NB : Les données matériaux n'ont bien entendu pas vocation à représenter un matériau en particulier. Elles sont uniquement destinées à des tests numériques de validation.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

1.3 Conditions aux limites et chargements

Encastrement : Les déplacements imposés sont nuls sur la face de l'élément cohésif opposée à l'élément élastique. Le joint est incliné à 30 degrés par rapport au plan horizontale, ce qui donne les directions de chargement suivantes selon le mode de sollicitation :

En test de traction :

Un déplacement imposé U est appliqué sur la face de l'élément élastique opposée au joint (voir figure 1).

$$DX = 5e^{-4}\cos(30°) \qquad DY = 5e^{-4}\sin(30°) \qquad DZ = 0$$

En test de cisaillement :

Une pression de compression est imposée sur la face de l'élément élastique opposée au joint afin de mettre le joint dans un état de compression.

$$PRES = 10^4\ Pa$$

De plus, un déplacement imposé U est appliqué sur tous les nœuds de la face supérieure du cube élastique.

$$DNOR = 0{,}5$$

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

2 Solution de référence

2.1 Cas général

La loi de comportement JOINT_MECA_ENDO a été développée dans le cadre de la thèse d'Ilaria Fontana soutenue en mars 2022. Le manuscrit de thèse contient tous les éléments théorique de la loi.

On cherche ici à reproduire un test de traction et un test de cisaillement sous contrainte normale constante imposée. On rappelle que les paramètres de la loi sont : la rigidité normale et tangentielle K_n et K_t, le coefficient de frottement μ, la cohésion C, la contrainte de cisaillement maximale T, deux paramètres permettant respectivement de contrôler le pic en cohésion et le pic en cisaillement B_n, B_t, deux paramètres permettant de contrôler l'évolution du pic de contrainte en cisaillement m_1 et m_2 et un paramètre de régularisation de la phase post-pic D_1 (qui permet notamment de s'assurer de l'absent de snap-back). Il est important de noter que K_n, K_t, μ, C et T sont des paramètres « ingénieur » qu'il est possible d'identifier sur des essais expérimentaux. Toutefois, B_n, B_t, m_1, m_2 et D_1 sont des paramètres numériques permettant d'assurer les bonnes propriétés mathématiques de la loi et de contrôler la phase adoucissante. On détermine les valeurs de B_n et de B_t à partir des équations 2.48 et 2.49 du manuscrit de thèse. Dans la version actuelle de la loi, les paramètres m_1 et m_2 sont fixés respectivement à $m_1=3$ et $m_2=0,5$ permettant ainsi d'avoir une « bonne forme » pour la phase post-pic. Concernant le paramètre D_1 on considère qu'il doit être dix fois supérieur à sa valeur minimale $D_{1\,min}$ définie par la condition de non snap-back décrite en page 47 du manuscrit. A noter que d'autres conditions sur les paramètres sont à respecter : une condition sur C et T décrite par l'équation 2.50, une condition sur K_n et K_t décrite par l'équation 2.24 et pour finir il faut vérifier que B_n et B_t soient positifs.

Le script Python présenté en annexe permet de calculer les paramètres numériques de la loi à partir des paramètres ingénieurs. Il est possible d'utiliser ce script dans le sens inverse, c'est-à-dire de calculer les paramètres ingénieurs à partir des paramètres numériques définis au préalable. Dans la version actuelle du cas-test, une fonction Python est introduite en début de chaque fichier de commande permettant de calculer les paramètres numériques de la loi à partir des paramètres dit « ingénieur ».

Pour le test de cisaillement, la solution de référence est donnée par les équations 2?38 et 2.39 du manuscrit de thèse. Pour le test de traction, la solution de référence est donnée par les équations 2.44 et 2.45 du manuscrit de thèse. Chacune de ces équations est exprimé en fonction de la variable d'endommagement alpha.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

3 Modélisation A

Validation du joint $2D$ avec la loi JOINT_MECA_ENDO
On réalise ici un test de traction.

3.1 Caractéristiques de la modélisation

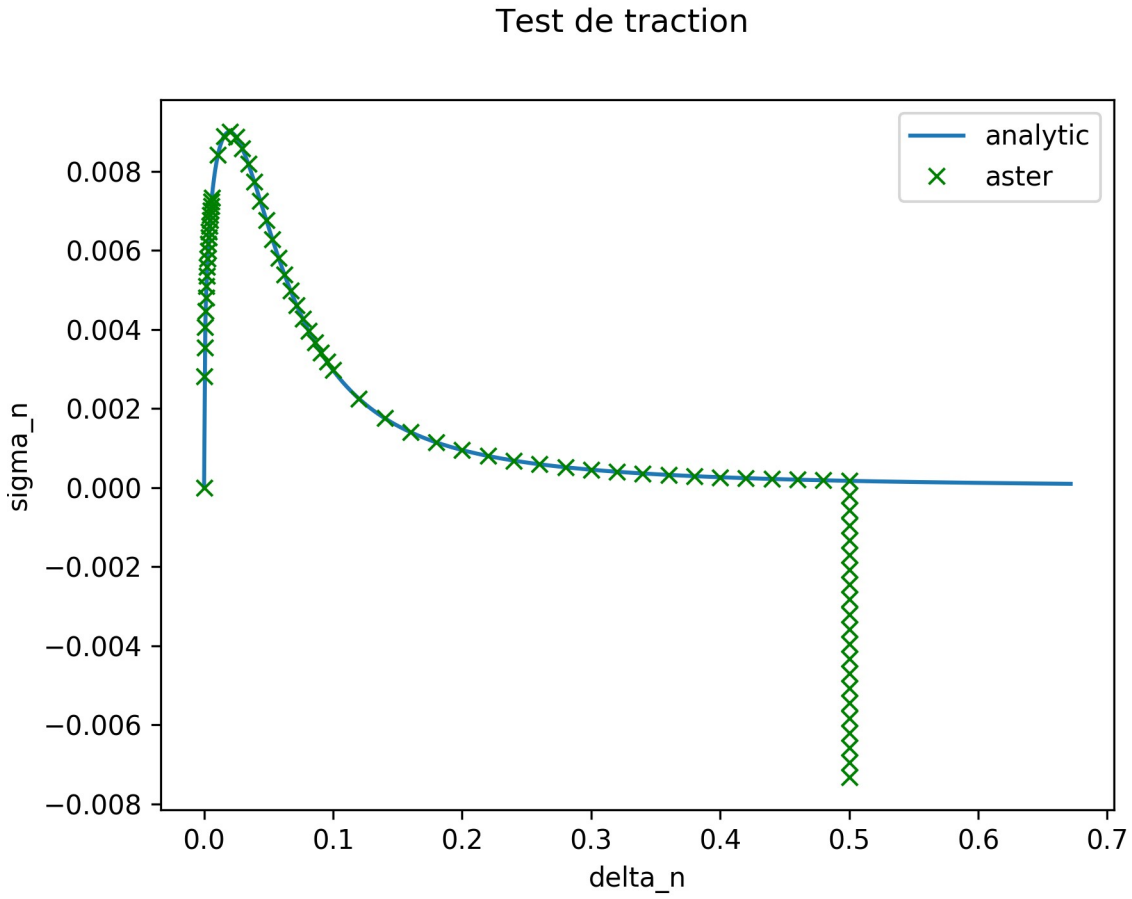
Modélisation en déformations planes D_PLAN pour l'élément élastique.
Modélisation plan pour l'élément de joint (mot clé PLAN_JOINT).

3.2 Caractéristiques du maillage

Nombre de nœuds : 6
L'élément élastique est un QUAD4.
L'élément de joint est un QUAD4 dégénéré (nœuds confondus).

3.3 Grandeurs testées et résultats

On compare la contrainte normale avec la solution de référence à différent instant de chargement qui correspondent à différent état d'endommagement du joint.



Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

Grandeur testée	Référence	Tolérance (%)
FN, à l'instant 20	$\sigma_n = 7333.48152\,Pa$	1
FN, à l'instant 40	$\sigma_n = 2969.57717064\,Pa$	1
FN, à l'instant 43	$\sigma_n = 1404.71701949\,Pa$	1
FN, à l'instant 60	$\sigma_n = 172.0723055535\,Pa$	1

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

4 Modélisation B

Validation du joint 2D avec la loi JOINT_MECA_ENDO
On réalise ici un chargement de cisaillement sous pression normale.

4.1 Caractéristiques de la modélisation

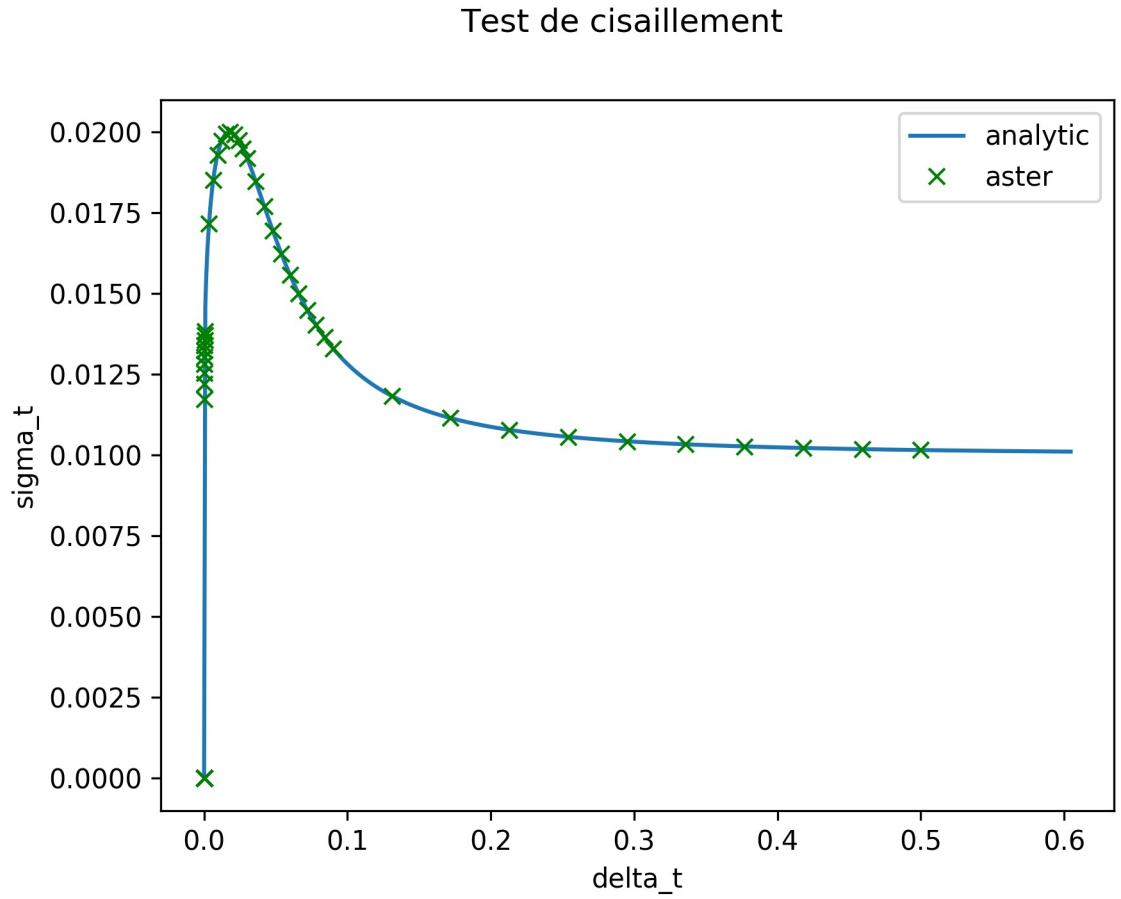
Modélisation en déformations planes D_PLAN pour l'élément élastique.
Modélisation plan pour l'élément de joint (mot clé PLAN_JOINT).

4.2 Caractéristiques du maillage

Nombre de nœuds : 6
L'élément élastique est un QUAD4.
L'élément de joint est un QUAD4 dégénéré (nœuds confondus).

4.3 Grandeurs testées et résultats

On compare la contrainte tangentielle avec la solution de référence à différent instant de chargement qui correspondent à différent état d'endommagement du joint.



Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

Grandeur testée	Référence	Tolérance (%)
FT , à l'instant 21	$\sigma_t = 17163.92 Pa$	1
FT , à l'instant 26	$\sigma_t = 19997.40052853 Pa$	1
FT , à l'instant 40	$\sigma_t = 13299.5301896 Pa$	1
FT, à l'instant 50	$\sigma_t = 10156.4653147 Pa$	1

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

5 Modélisation C

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

6 Modélisation D

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

7 Modélisation E

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

8 Modélisation F

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

9 Synthèse des résultats

Les résultats numériques sont en accord avec la solution analytique. Ces tests permettent de valider les éléments de joint, les éléments de joint HM en $2D$ et $3D$ dans les différents modes d'ouverture.

Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

10 ANNEXE : Script Python pour la solution analytique

```
##############################################
#   JOINT_MECA_ENDO : Test de cisaillement   #
##############################################
from math import *
import numpy as np
import scipy.optimize as opt
import matplotlib.pyplot as plt

#####################################################################
# SELECTION DES DONNEES D'ENTREE :
#####################################################################
# Mode = ['Ilaria', 'CIH']
mode = 'CIH'

if mode == 'Ilaria':
   # Coefficient de frottement
   mu = 1.              # Extrait de ssnp162a ligne 35
   Bn = 137622240.236444   # Extrait de lcejdm ligne 166
   Bt = 32281760.05546218  # Extrait de lcejdm ligne 167
   # Energie dissipee durant l'endommagement :
   D1 = 1.              # Extrait de lcejdm ligne 174

if mode == 'CIH':
   # Cohesion
   C = 1.4e6            # Extrait mail du CIH du 8/02
   #C = 1.e5             # Extrait du test d'Ilaria
   # Contrainte de cisaillement maximale
   T = 1.1e6            # Extrait mail du CIH du 8/02
   #T = 0.9*C            # Extrait du test d'Ilaria
   # Coefficient de frottement
   mu = 0.95            # Extrait mail du CIH du 8/02

# Rigidite normale :
Kn = 1.e12              # Extrait de ssnp162a ligne 30
# Rigidite tangentielle :
Kt = 2.*Kn              # Extrait de ssnp162a ligne 32
# Cohesion residuelle
c_bar = 0.6
# Contrainte de compression constante :
sigma_n_comp = 0.4e6
# Parametres de forme du pic
m1 = 3.                 # Extrait de lcejdm ligne 172
m2 = 0.5                # Extrait de lcejdm ligne 173
# Pas d'integration (alpha de 0 a 1 par pas de step)
step = 0.0001
# Endomagement max (entre 0 et 1)
endo_max = 0.90

#####################################################################
# Definition des fonctions S et R fonction de alpha :
#####################################################################
# Definition de la fonction S(alpha) :
def S(alpha):
   return ((1-alpha)**m1)/(alpha**m2)

# Definition de la fonction derivee de S(alpha) :
def Sp(alpha):
   return ((m2-m1)*alpha-m2)*((1-alpha)**(m1-1))/(alpha**(m2+1))
```


Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

```
# Definition de la derivee seconde de S(alpha) :
def S2p(alpha):
   return ((m1-m2)*(m1-m2-1.)*alpha**2+2.*m2*(m1-m2-1.)*alpha+
         m2*(m2+1.))*(1.-alpha**(m1-2))/alpha**(m2+2)

# Definition de la fonction R(alpha) :
def R(alpha):
   return ((2.*(Sp(alpha))**2)/(S2p(alpha)))-S(alpha)

######################################################################
# Recherche du max de R(alpha) :
######################################################################
def FindMaxR():
   val_max = opt.fmin(lambda alpha: -R(alpha), 1.e-10)
   return R(val_max[0])

######################################################################
# Calcul de Bn, Bt, C et T :
######################################################################
def CalcBn():
   Bn = (-Sp(alpha_max)*T**2)/(2.*D1*S(alpha_max)**2)
   return Bn

def CalcBt():
   Bt = (-Sp(alpha_max)*(C**2-mu**2*T**2))/(2.*D1*S(alpha_max)**2)
   return Bt

def CalcC():
   C = sqrt(2.*D1*(Bn*(mu**2)+Bt))*S(alpha_max)/(sqrt(-Sp(alpha_max)))
   return C

def CalcT():
   T = sqrt(2.*D1*Bn)*S(alpha_max)/sqrt(-Sp(alpha_max))
   return T

######################################################################
# Calcul de delta et sigma en fonction de alpha :
######################################################################
def joint_meca_endo(Kn,Kt,Bn,Bt,m1,m2,mu,c_bar,D1,alpha):
   delta_t = (sqrt(2.*D1/(-(mu**2*Bn+Bt)*Sp(alpha)))*(Kt+(mu**2*Bn+Bt)*S(alpha))/Kt+
         (mu*sigma_n_comp+c_bar)/Kt)
   sigma_t = sqrt(2.*D1*(mu**2*Bn+Bt))*S(alpha)/sqrt(-Sp(alpha))+mu*sigma_n_comp+c_bar
   delta_n = -sigma_n_comp/Kn+mu*sqrt(2.*D1/(-(mu**2*Bn+Bt)*Sp(alpha)))
   delta_n_trac = sqrt(2*D1/(-Bn*Sp(alpha)))*(Kn+Bn*S(alpha))/Kn+c_bar/(mu*Kn)
   sigma_n = S(alpha)*sqrt(2.*D1*Bn)/sqrt(-Sp(alpha))+c_bar/mu
   return delta_t, sigma_t, delta_n, delta_n_trac, sigma_n

######################################################################
# Impression des courbes :
######################################################################
def PlotCurve(x,y,title,label_x,label_y,output_folder_path,filename):
   plt.figure(title)
   plt.plot(x, y, label=title)
   plt.xlabel(label_x)
   plt.ylabel(label_y)
   plt.suptitle(title)
   plt.legend()
   plt.savefig(output_folder_path+filename, dpi=300)
   plt.close()
```


Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

```
if __name__ == '__main__':
    # Calcul de alpha_max :
    alpha_max = (-m2+sqrt(m1*m2/(m1-m2+1)))/(m1-m2)

    # Recherche du maximu de R :
    R_max = FindMaxR()

    # Calculs des parametres en fonction des donnees d'entree :
    if mode == 'Ilaria':
        C = CalcC()
        T = CalcT()
    if mode == 'CIH':
        D1 = max((max(T**2/(2.*Kn),C**2/(2.*Kt))*(-Sp(alpha_max)/(S(alpha_max)**2))*R_max*10.),1.)
        Bn = CalcBn()
        Bt = CalcBt()

    # Calcul de D_min :
    D_min = max(T**2/(2.*Kn),C**2/(2.*Kt))*(-Sp(alpha_max)/(S(alpha_max)**2))*R_max

    # Impression des donnees :
    print '\n'
    print '=====       VALEUR DES PARAMETRES       ====='
    print '-----------------------------------------------'
    print ' Kn :', Kn, '\t\t\t Kt :', Kt
    print ' C :', C, '\t\t\t T :', T
    print 'mu :', mu, '\t\t\t c_bar :', c_bar
    print 'Bn :', Bn, '\t\t Bt :', Bt
    print 'alpha_max :', alpha_max, '\t R_max :', R_max
    print 'D1 :', D1, '\t\t sigma_n_comp :', sigma_n_comp
    print 'm1 :', m1, '\t\t\t m2 :', m2

    # Tests de validation des parametres :
    print '\n'
    print '===== TEST DE VALIDATION DES PARAMETRES ====='
    print '-----------------------------------------------'
    print 'Test D1 > D_min :\t\t\t', D1>=D_min
    if not D1>D_min:
        print '    /!\ D1 = ', D1, 'et D_min =', D_min
    print 'Test mu*T < C < sqrt(2.)*mu*T :\t\t', mu*T < C < sqrt(2.)*mu*T
    if not mu*T < C < sqrt(2.)*mu*T:
        print '    /!\ mu*T = ', mu*T, 'C = ', C, 'sqrt(2.)*mu*T = ', sqrt(2.)*mu*T
    print 'Test Kn > Bn*R_max :\t\t\t', Kn>Bn*R_max
    if not Kn>Bn*R_max:
        print '    /!\ Kn = ', Kn, 'Bn*R_max = ', Bn*R_max
    print 'Test Kt > Bt*R_max :\t\t\t', Kt>Bt*R_max
    if not Kt>Bt*R_max:
        print '    /!\ Kt = ', Kt, 'Bt*R_max = ', Bt*R_max
    print 'Test Bn > 0\t\t\t\t', Bn>0.
    if not Bn>0.:
        print '    /!\ Bn = ', Bn
    print 'Test Bt > 0\t\t\t\t', Bt>0.
    if not Bt>0.:
        print '    /!\ Bt = ', Bt
    print '-----------------------------------------------'
    print '\n'

    # Calculs de delta et sigma :
    list_delta_t = [0.]
    list_sigma_t = [0.]
```


Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)

```
    list_delta_n = [0.]
    list_delta_n_trac = [0.]
    list_sigma_n = [0.]
    for alpha in np.arange(0.+step, endo_max+step, step):
                                    delta_t, sigma_t, delta_n, delta_n_trac, sigma_n =
joint_meca_endo(Kn,Kt,Bn,Bt,m1,m2,mu,c_bar,D1,alpha)
        list_delta_t.append(delta_t*1e3)
        list_sigma_t.append(sigma_t/1.e6)
        list_delta_n.append(delta_n*1e3)
        list_delta_n_trac.append(delta_n_trac*1e3)
        list_sigma_n.append(sigma_n/1.e6)
```


Document diffusé sous licence GNU FDL (http://www.gnu.org/copyleft/fdl.html)