

Université Gustave Eiffel

Homogénéisation périodique numérique en élasticité linéaire plane

Janvier 2023

1. Objectifs

Ce document s'adresse à un utilisateur qui dispose d'une part, du solveur CESAR, et, d'autre part, du Pilote de CESAR (voir la section 2 pour leur installation).

Il présente la mise en œuvre de techniques d'homogénéisation numérique "classiques" pour définir les propriétés élastiques homogénéisées (ou macroscopiques) d'un matériau hétérogène périodique.

On s'appuie sur la présentation de la démarche proposée par Dirrenberger et Forest (2010), avec le programme de calcul par éléments finis Zébulon. On pourra également consulter le cours de O. Débordes (2009).

En pratique, on présente un script qui met en œuvre à la fois le langage Pilote et le langage CESAR. Le langage Pilote est essentiellement utilisé pour la création du maillage de la cellule de base en faisant appel au logiciel gmsh. Le langage CESAR permet de modifier ce maillage pour introduire des conditions aux limites périodiques, réaliser des calculs imposant une déformation macroscopique (c'est-à-dire imposant la valeur moyenne du tenseur de déformation sur la cellule étudiée) et finalement extraire les propriétés élastiques homogénéisées.

L'exemple présenté permet d'envisager de réaliser simplement des études paramétriques sur la finesse du maillage, les propriétés des matériaux en jeu, la forme et la disposition des hétérogénéités.

2. Prérequis

Les étapes permettant d'installer les logiciels utiles, à partir d'une installation Ubuntu 20.04 « fraîche », sont listées ci-dessous. Il n'est pas nécessaire de les répéter si elles ont déjà été réalisées.

1/ Solveur CESAR

- Télécharger l'archive auto extractible (cesar.univ-gustave-eiffel.fr, page Téléchargements)
- Exécuter (avec sh ou bash) le script `cesar_bin64_ubuntu-20.04_2022_11_14.run`
- Contacter cesar-lcpc@ifsttar.fr pour obtenir votre fichier de licence SOLVCESV42.KEYC, à coller dans le répertoire `fconfig/` créé par le script d'installation

2/ Pilote de CESAR

- Télécharger l'archive auto extractible (cesar.univ-gustave-eiffel.fr, page Téléchargements)
- Exécuter (avec sh ou bash) le script `cespil_bin64_ubuntu-20.04_2022_11_14.run`
- Suivre les instructions pour terminer l'installation (créer un lien symbolique vers le répertoire d'installation du solveur CESAR et installer `python3-dev`)

3/ Installation de GMSH

Gmsh est un logiciel de maillage par éléments finis développé par Christophe Geuzaine et Jean-François Remacle, publié sous licence GPL (<http://gmsh.info>). Le script proposé l'utilise pour générer le maillage. Gmsh permet également de visualiser les résultats du calcul.

- `sudo apt install gmsh`

4/ Installation de numpy et matplotlib

- `sudo apt install python3-numpy`
- `sudo apt install python3-matplotlib`

3. Exécution du script

On lance l'exécution du script en tapant :

```
/home/user/cespil/bin/run_pilote homog_num.py
```

en adaptant le chemin (en particulier le nom de l'utilisateur `user`) au contexte dans lequel on travaille.

4. Éléments théoriques sommaires

4.1. Grandeurs microscopiques et grandeurs macroscopiques

On considère un matériau hétérogène présentant une structure périodique, décrite par la répétition d'une cellule de base, qu'on suppose dans un premier temps rectangulaire (en 2D). On désigne par Ω le domaine occupé par la cellule de base et on note $\langle f \rangle$ la moyenne d'une grandeur f sur ce domaine :

$$\langle f \rangle = \frac{1}{|\Omega|} \int_{\Omega} f(\underline{x}) d\Omega$$

où \underline{x} désigne le vecteur position à l'intérieur de la cellule.

On définit la contrainte macroscopique $\underline{\Sigma}$ et la déformation macroscopique \underline{E} respectivement comme la moyenne sur la cellule de base des tenseurs de contraintes et de déformations :

$$\underline{\Sigma} = \langle \underline{\sigma} \rangle \quad ; \quad \underline{E} = \langle \underline{\varepsilon} \rangle$$

Le but de la démarche d'homogénéisation est de tirer parti des informations disponibles sur la géométrie de la cellule de base et sur le comportement local reliant $\underline{\sigma}$ et $\underline{\varepsilon}$ pour établir des relations de comportement valables entre $\underline{\Sigma}$ et \underline{E} .

4. 2. Position d'un problème auxiliaire sur la cellule de base

L'étude du comportement du matériau périodique repose sur l'étude du problème auxiliaire suivant, posé sur une cellule de base Ω :

Trouver $(\underline{u}, \underline{\sigma})$ tels que

$$\text{div } \underline{\sigma} = 0 \quad (\text{pas de forces de volume})$$

$$\underline{\sigma} = \underline{\sigma}(\underline{\varepsilon})$$

$$\underline{\varepsilon} = 1/2 (\text{grad } \underline{u} + {}^t\text{grad } \underline{u})$$

$$\underline{u}(\underline{x}) = \underline{\underline{E}}^* \cdot \underline{x} + \underline{v} \quad \forall \underline{x} \in \partial\Omega$$

où $\underline{\underline{E}}^*$ est un tenseur donné, et \underline{v} un vecteur de fluctuations périodiques prenant la même valeur en deux points opposés M et M' ou N et N' du contour $\partial\Omega$ (figure 1) :

$$\underline{v}(M) = \underline{v}(M') ; \underline{v}(N) = \underline{v}(N')$$

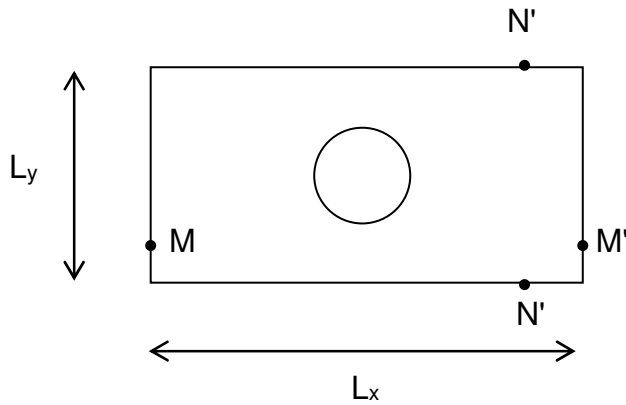


Figure 1 – Maillage de la cellule de base

Dans le cas élastique linéaire, le problème a une solution ; le champ de contraintes $\underline{\sigma}$ est défini de manière unique et le champ de déplacement \underline{u} est défini à une translation d'ensemble près.

Il est facile de montrer que la solution vérifie :

$$\underline{\underline{E}} = \langle \underline{\varepsilon} \rangle = \underline{\underline{E}}^*$$

ce qui montre que le problème auxiliaire revient à rechercher la réponse de la cellule de base lorsque l'on impose la valeur moyenne $\underline{\underline{E}}^*$ du tenseur de déformation $\underline{\varepsilon}$.

4. 3. Lemme de Hill–Mandel

Pour la solution du problème auxiliaire, on montre que

$$\langle \underline{\underline{\sigma}} : \underline{\underline{\varepsilon}} \rangle = \langle \underline{\underline{\sigma}} \rangle : \langle \underline{\underline{\varepsilon}} \rangle = \langle \underline{\underline{\sigma}} \rangle : \underline{\underline{E}}^* = \underline{\underline{\Sigma}} : \underline{\underline{E}}$$

Le travail virtuel de déformation à l'échelle macroscopique $\underline{\underline{\Sigma}} : \underline{\underline{E}}$ apparaît comme la moyenne sur la cellule de base du travail virtuel de déformation $\underline{\underline{\sigma}} : \underline{\underline{\varepsilon}}$ ce qui confère une signification mécanique forte aux moyennes macroscopiques $\underline{\underline{\Sigma}}$ et $\underline{\underline{E}}$.

4. 4. But et principe de mise en œuvre d'une résolution numérique avec CESAR

Le but est de proposer un ensemble d'outils permettant de :

- résoudre numériquement avec CESAR le problème auxiliaire présenté en 3.2 ;
- effectuer le calcul de l'intégrale (ou de la moyenne) des contraintes et des déformations sur tout le maillage pour en déduire les caractéristiques mécaniques « homogénéisées ».

La donnée est la déformation macroscopique imposée. Il faut imposer la relation de "périodicité" entre des points appariés de la frontière de la cellule de base. Dans ce qui suit, on traite la relation linéaire à imposer entre les déplacements de nœuds homologues par pénalisation. Pour chaque paire de nœuds, on introduit deux éléments spéciaux : le premier relie les déplacements horizontaux des deux nœuds, et le deuxième relie les déplacements verticaux.

5. Présentation du problème

L'exemple présenté est tiré de Dirrenberger et Forest (2010), qui présentent la mise en œuvre de calculs d'homogénéisation périodique avec le code Zébulon de l'Ecole des Mines.

On considère un matériau constitué par une matrice renforcée par des fibres cylindriques circulaires toutes parallèles à une même direction Oz. La cellule de base dans le plan perpendiculaire aux fibres est carrée, de côté 1, et le rayon de la fibre est égal à 0.2 (soit une fraction volumique de 12.6%).

La matrice et le matériau constituant la fibre sont élastiques linéaires isotropes, avec :

pour la matrice : $E = 3000 \text{ MPa}$; $\nu = 0,3$

pour la fibre : $E = 50000 \text{ MPa}$; $\nu = 0,3$

Le but du calcul est d'identifier le tenseur d'élasticité macroscopique du matériau renforcé, noté C et défini, dans le cadre des déformations planes, par :

$$\underline{\underline{\Sigma}} = \mathbf{C} : \underline{\underline{E}} \quad \text{avec} \quad {}^t \underline{\underline{\Sigma}} = [\Sigma_{xx}, \Sigma_{yy}, \Sigma_{xy}, \Sigma_{zz}] \quad \text{et} \quad {}^t \underline{\underline{E}} = [E_{xx}, E_{yy}, 2E_{xy}]$$

Le tenseur d'élasticité est de la forme :

$$C = \begin{pmatrix} C_{xxxx} & C_{xxyy} & C_{xxxy} \\ C_{yyxx} & C_{yyyy} & C_{yyxy} \\ C_{xyxx} & C_{xyyy} & C_{xyxy} \\ C_{zzxx} & C_{zzyy} & C_{zzxy} \end{pmatrix}$$

Avec les symétries majeures : $C_{yyxx} = C_{xxyy}$; $C_{xyxx} = C_{xxxy}$; $C_{xyyy} = C_{yyxy}$

L'idée générale est d'imposer une déformation moyenne sur la cellule de base et de calculer la moyenne des contraintes qui en résultent, de manière à identifier les coefficients de C :

- pour $E_{xx} = 1$ et $E_{yy} = E_{xy} = 0$, le calcul permet d'identifier la première colonne de la matrice de C.
- pour $E_{yy} = 1$ et $E_{xx} = E_{xy} = 0$, on identifie la deuxième colonne de la matrice de C.
- pour $E_{xx} = E_{yy} = 0$ et $E_{xy} = 1$, on identifie la troisième colonne de la matrice de C.

On notera que, dans le cas d'une inclusion circulaire centrée dans une cellule carrée, on a d'autres symétries (en particulier, il est clair que les directions x et y jouent le même rôle). Pour cette géométrie particulière, on a de plus :

$$C_{yyyy} = C_{xxxx} ; C_{yyxy} = C_{xxxy} = 0 ; C_{zzyy} = C_{zzxx}$$

si bien que la matrice de C ne contient finalement que 5 termes à identifier :

$$C = \begin{pmatrix} C_{xxxx} & C_{xxyy} & 0 \\ C_{xxyy} & C_{xxxx} & 0 \\ 0 & 0 & C_{xyxy} \\ C_{zzxx} & C_{zzxx} & C_{zzxy} \end{pmatrix}$$

En particulier, dans ce cas, on peut faire le calcul seulement pour le premier et le troisième des tenseurs de déformation proposés.

La suite de ce document présente un exemple de calcul de caractéristiques mécaniques macroscopiques pour un matériau élastique linéaire périodique. On présente le problème étudié, puis la réalisation du calcul, qui consiste à :

- utiliser le pilote de CESAR pour mailler la cellule de base,
- ajouter au maillage des éléments spécifiques permettant d'imposer la périodicité du déplacement,
- constituer les données correspondant à l'application d'une déformation moyenne,
- lancer le calcul et extraire des résultats les modules homogénéisés.

6. Maillage de la cellule de base

6.1. Début du script Python

On commence par importer différents modules, en particulier ceux relatifs au langage CESAR et au langage Pilote :

```
from modele_donnees import *
from modele_cesar import *
import numpy as np
import re
```

On définit le répertoire de travail :

```
rep_trav = './'
```

NB : la ligne ci-dessus doit être adaptée par l'utilisateur.

Le script contient ensuite une fonction charRL dont on détaillera le fonctionnement plus loin.

6.2. Préparation du maillage

La cellule de base est maillée en utilisant gmsh, via la classe DOMAINE et sa méthode `mailler`. On définit d'abord le nom des fichiers utiles pour l'interopérabilité avec gmsh :

```
fic_dom = rep_trav + 'cellule.geo'
fic_mail = rep_trav + 'cellule.msh'
```

On introduit des paramètres qui représentent la longueur et la largeur de la cellule, le rayon de l'inclusion, et la taille des éléments sur le contour extérieur et sur le bord de l'inclusion :

```
# parametres geometriques
LX = 1. ; LY = 1. ; R = 0.2
d = 0.1 # longueur des elements sur le contour de l inclusion
d2 = 0.025 # longueur des elements sur le contour de la cellule de base
```

On définit une séquence d'opérations pour définir

- les points suivants : les coins de la cellule, deux points du diamètre horizontal et le centre du cercle

```
# definition de la geometrie
op = [
# definition des points
    POINT(1, 0., 0., 0., d), POINT(2, LX, 0., 0., d),
    POINT(3, LX, LY, 0., d), POINT(4, 0., LY, 0., d),
    POINT(5, LX/2.-R, LY/2, 0., d2),
    POINT(6, LX/2., LY/2., 0., d2),
    POINT(7, LX/2.+R, LY/2., 0., d2),
```

- les lignes et arcs de cercle reliant ces points :

```
# definition des lignes
    LIGNE( 1, 1, 2), LIGNE(2, 2, 3),
    LIGNE(3, 3, 4), LIGNE( 4, 4, 1),
    CERCLE (5, P1= 5, PC=6, P2=7),
    CERCLE (6, P1= 7, PC=6, P2=5),
```

- deux boucles et deux surfaces pour définir les domaines occupés par la matrice et par l'inclusion,

```
# definition des contours des zones du maillage
BOUCLE(1, 'LIGNE', [1, 2, 3, 4, 6, 5]),
SURFACE(1, [1]),

BOUCLE(2, 'LIGNE', [5, 6]),
SURFACE(2, [2]),
```

- deux blocs surfaciques destinés à être associés aux groupes d'éléments de CESAR :

```
# definition de deux blocs surfaciques
BLOC('1', 'SURFACE', [1]),
BLOC('2', 'SURFACE', [2]),
```

Et quatre blocs-lignes permettant de repérer les bords gauche/droit et haut/bas

```
# bloc 3 = base / bloc 4 = droit / bloc 5 = LY / bloc 6 = gauche
BLOC('3', 'LIGNE', [1]),
BLOC('4', 'LIGNE', [2]),
BLOC('5', 'LIGNE', [3]),
BLOC('6', 'LIGNE', [4]),
```

Pour finir, on choisit des éléments à interpolation quadratique

```
# option : elements quadratiques
OPTION(MAIL_ORDRE_ELEM = 2, MAIL_ORDRE_2_INCOMPLET = 1),
]
```

Il reste à définir le domaine et à le mailler :

```
dom = DOMAINE(NOM = 'dom',
              DIM = 2,
              FIC_OUT = FICHER(NOM = fic_dom, FORMAT = 'GMSH'),
              OPER = op,
              VISU = 'NON')
dom.mailler(FIC_OUT = FICHER(NOM = fic_mail, FORMAT = 'GMSH'))
```

A ce stade, on peut visualiser le maillage constitué par gmsh, dans le fichier cellule.msh.

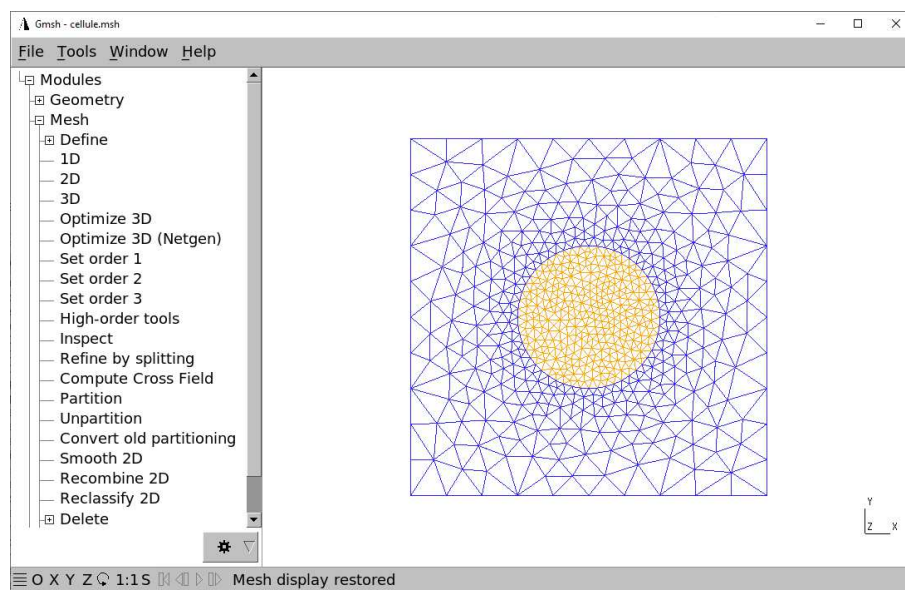


Figure 2 – Maillage de la cellule de base

7. Prise en compte des conditions aux limites périodiques

7. 1. Principe

La suite de la démarche consiste à prendre en compte des conditions aux limites périodiques entre des points M et M' homologues sur les bords gauche et droit (figure 1), et entre les points N et N' sur la base et le bord haut. De manière plus précise, la condition :

$$\underline{v}(M) = \underline{v}(M')$$

se traduit, sur le champ de déplacement lui-même par :

$$\underline{u}(M') = \underline{u}(M) + \underline{\underline{E}}^* \cdot \underline{MM}'$$

En notant Δ_x et Δ_y les deux composantes du vecteur $\underline{\underline{E}}^* \cdot \underline{MM}'$, on a donc deux relations scalaires à imposer :

$$U_x(M') - U_x(M) = \Delta_x$$

$$U_y(M') - U_y(M) = \Delta_y$$

On se propose de prendre en compte ces conditions dans le calcul numérique en les imposant entre les degrés de liberté des nœuds du contour de la cellule de base. Pour cela, on modifie le maillage pour ajouter des éléments « de relation linéaire » (un élément pour chaque couple de nœuds et dans chaque direction de l'espace), et d'autre part on définit des sollicitations nodales adaptées pour prendre en compte le membre de droite (Δ_x ou Δ_y).

7. 2. Relecture du maillage

La première étape consiste à récupérer les informations liées au maillage. On fait le choix dans la suite d'utiliser les fonctionnalités du langage CESAR, et on s'appuie en particulier sur la classe MAIL, qui permet de retrouver les deux blocs surfaciques correspondant aux groupes d'éléments (de massif) et les quatre groupes de facette correspondant aux bords du maillage.

```
from modele_cesar import *
# on recupere le maillage dans le fichier .msh genere par GMSH
mail = MAIL(FIC = fic_mail, FMT = 'GMSH', NDIM = 2,
            FAMI= { '1':'MB', '2':'MB' },
            GF = [ '3', '4', '5', '6' ],
            )
# NB : GF donne acces aux groupes de facettes dans l'objet mail qui permettront
# de definir les bords
```

On peut en particulier récupérer le nombre de nœuds et d'éléments du maillage et les imprimer :

```
nnt = len(mail.vcorg)//mail.ndim
ndim = mail.ndim
nelt = len(mail.pnumel)-1
print("Pour info : "+str(nnt)+" noeuds / "+str(nelt)+" elements")
```

7. 3. Recherche des paires de nœuds à associer

Le Pilote permet de reconstituer le nombre de nœuds situés sur les quatre côtés de la cellule (base, droite, haut, gauche) et la liste des numéros correspondants :

```
nb_nds_base    = mail.gn[3]['NP'] ; nds_base    = mail.gn[3]['NUM']
nb_nds_droite  = mail.gn[4]['NP'] ; nds_droite  = mail.gn[4]['NUM']
nb_nds_haut    = mail.gn[5]['NP'] ; nds_haut    = mail.gn[5]['NUM']
nb_nds_gauche  = mail.gn[6]['NP'] ; nds_gauche  = mail.gn[6]['NUM']
```

Pour constituer les paires de nœuds à associer par les éléments de relation linéaire, pour la base et le bord haut, on classe les nœuds de ces côtés par ordre d'abscisse croissante :

```
list_base = []
for noeud in nds_base :
    list_base.append ( [noeud,mail.vcorg[2*noeud-2]] )
list_base = sorted(list_base, key=lambda node : node[1])

list_haut = []
for noeud in nds_haut :
    list_haut.append ( [noeud,mail.vcorg[2*noeud-2]] )
list_haut = sorted(list_haut, key=lambda node : node[1])
```

On peut alors constituer une liste de tuples contenant l'abscisse des nœuds, le numéro du nœud sur la base et celui du nœud sur le bord haut

```
pairesBH = []
index=0
for noeud in list_base:
    pairesBH.append( (list_base[index][1],list_base[index][0],list_haut[index][0]))
    index += 1
```

On procède de même pour constituer les paires de nœuds situés sur les côtés gauche et droit.

7. 4. Introduction d'éléments de relation linéaire

Pour chaque paire de nœuds, on crée deux nouveaux éléments de relation linéaire à deux nœuds, le premier correspondant à la relation entre les déplacements horizontaux, le deuxième à la relation entre les déplacements verticaux. Le maillage original comporte deux groupes d'éléments ; on ajoute un groupe 3 pour les relations entre déplacements horizontaux et un groupe 4 pour les relations entre déplacements verticaux. Le code Python est relativement simple : on incrémente les listes mail.pnumel, mail.numel, mail.groupe, ainsi que le nombre d'éléments et de groupes.

```
# Definition des nouveaux elements

mail.ngrpe += 2
last_pnumel = mail.pnumel[-1]

for paire in pairesBH :
    nelt=nelt+2
    mail.pnumel=mail.pnumel+[last_pnumel+2, last_pnumel+4]
    mail.numel=mail.numel+[int(paire[2]), int(paire[1])]
    mail.numel=mail.numel+[int(paire[2]), int(paire[1])]
    mail.type=mail.type+['RL ', 'RL ']
    mail.groupe=mail.groupe+[3,4]
    last_pnumel += 4
```

```

for paire in pairesGD :
    nelt=nelt+2
    mail.pnumel=mail.pnumel+[last_pnumel+2, last_pnumel+4]
    mail.type=mail.type+['RL ', 'RL ']
    mail.groupe=mail.groupe+[3,4]
    mail.numel=mail.numel+[int(paire[2]), int(paire[1])]
    mail.numel=mail.numel+[int(paire[2]), int(paire[1])]
    last_pnumel += 4

print("Pour info : "+str(nnt)+" noeuds / "+str(nelt)+" elements")

```

8. Préparation des calculs en langage CESAR

8.1. Coordonnées

On n'a pas modifié les coordonnées des nœuds. L'objet `coor` correspondant est donc facile à reconstituer à partir de l'objet `mail` relu à l'aide de la classe ad hoc :

```

# objet COOR
coor = COOR(M = 0,
            M1 = 0,
            NNT = nnt,
            NDIM = ndim,
            VCORG = mail.vcorg)

```

8.2. Données matériau et caractéristiques des relations linéaires

La matrice et l'inclusion sont supposées élastiques linéaires, et on spécifie les caractéristiques correspondantes. Pour les éléments de relation linéaire, on choisit un coefficient de pénalisation égal à 10000 fois le module de l'inclusion.

```

# caracteristiques de la matrice et de l inclusion
Emat = 3000. ; Pmat = 0.3
Einc = 50000. ; Pinc = 0.3
# coefficient de penalisation pour les RL
K_penal = 10000.*Einc

```

On définit les modèles correspondants :

```

m_matrice = MB(NOMG = 'matrice',
               ACTI = 'A',
               IMOD = 1,
               CARA = MB_ELI( RO = 0., YOUNG = Emat , POISS = Pmat, INAT = 1 )
               )

m_inclusion = MB(NOMG = 'inclusion',
                ACTI = 'A',
                IMOD = 1,
                CARA = MB_ELI( RO = 0., YOUNG = Einc , POISS = Pinc, INAT = 1 )
                )

```

Pour les relations linéaires, on distingue les relations entre degrés de liberté en x et en y.

```
m_RLx = RL(NOMG = 'RLx',
           ACTI = 'A',
           IDL = [1, 1], C = [1., -1.], P = K_penal
           )

m_RLy = RL(NOMG = 'RLy',
           ACTI = 'A',
           IDL = [2, 2], C = [1., -1.], P = K_penal
           )
```

Enfin, on rassemble ces données dans l'objet `elem` :

```
elem = ELEM(M = 0,
            M1 = 0,
            NELT = nelt,
            NGRPE = mail.ngrpe,
            PNUMEL = mail.pnumel,
            NUMEL = mail.numel,
            TYPE = mail.type,
            GROUPE = mail.groupe,
            GRPES = [ m_matrice, m_inclusion, m_RLx , m_RLy ])
```

8. 3. Conditions aux limites en déplacement

Le problème posé sur la cellule de base a une solution unique en contraintes, mais le champ de déplacement n'est défini qu'à un mouvement de corps rigide près.

On bloque en x un nœud situé au milieu du bord gauche de la cellule, et en y le nœud situé au milieu du bord inférieur. Pour retrouver les numéros de nœuds, on exploite le contenu des paires de nœuds réunies dans `pairesBH` et `pairesGD` :

```
for paire in pairesBH :
    coorY=paire[0]
    if abs(float(coorY)-LY/2) < d2/2 :
        blocageY= int(paire[1])

for paire in pairesGD :
    coorX=paire[0]
    if abs(float(coorX)-LX/2) < d2/2 :
        blocageX= int(paire[1])
```

On constitue ensuite l'objet `cond` correspondant aux blocages voulus :

```
cond = COND(M = 2,
            OPT = [COND_NUL(GEN = [GEN_NUL(IGEN = 2,
                                           NP = 1,
                                           NUM = [blocageX],
                                           IDL = [1, 0],
                                           NDLN = 2) ] ),
                  COND_NUL(GEN = [GEN_NUL(IGEN = 2,
                                           NP = 1,
                                           NUM = [blocageY],
                                           IDL = [0, 1],
                                           NDLN = 2) ] ),
                  ])
```

8. 4. *Chargement associé à la déformation moyenne à imposer*

L'introduction des éléments de relation linéaire permet d'introduire dans l'énergie élastique de la structure une contribution du type

$$W_{RL} = 1/2 K_p [(U_y(M') - U_y(M))^2 + (U_y(M') - U_y(M))^2]$$

dont la minimisation revient à imposer l'égalité du déplacement entre les points M et M'. Pour imposer la condition

$$\underline{u}(M') = \underline{u}(M) + \underline{\underline{E}}^* \cdot \underline{MM}'$$

il faut définir des chargements nodaux sur les degrés de liberté du nœud M', égaux aux composantes de $\underline{\underline{E}}^* \cdot \underline{MM}' = \underline{\underline{E}}^* \cdot (L_x \underline{e}_x)$, multipliés par le coefficient de pénalisation K_p , et ajouter les chargements nodaux opposés sur les degrés de liberté du nœud M.

Il faut prendre en compte les paires de nœuds M,M' sur les bords gauche et droit et N,N' sur la base et le bord haut. Le calcul est réalisé, pour une valeur donnée de la déformation qu'on veut imposer $\underline{\underline{E}}^*$, par la fonction Python `charRL` :

```
def charRL(Eps,LX,LY,pairesGD,pairesBH,K_penal) :

# forces nodales sur les ddl des elements RL
# On calcule E : (x2-x1) , on multiplie par k et on charge les deux noeuds opposes
    listnod=[]
    fnodX=[] ; fnodY=[]
    deltaM = np.array ( [ LX , 0. ] )
    force = np.dot ( Eps, deltaM )
    fx= K_penal*force[0] ; fy=K_penal*force[1]

    for paire in pairesGD :
        listnod.extend( [ paire[2] , paire[1] ] )
        fnodX.extend( [fx,-fx] )
        fnodY.extend( [fy,-fy] )

    gX_H=GEN_SOL(IGEN = 2, NP = 2*len(pairesGD) , NUM= listnod , IL = 1, F= fnodX )
    gY_H=GEN_SOL(IGEN = 2, NP = 2*len(pairesGD) , NUM= listnod , IL = 2, F= fnodY )

    listnod=[]
    fnodX=[] ; fnodY=[]
```

```

deltaM = np.array ( [ 0. , LY ] )
force = np.dot ( Eps, deltaM )
fx= K_penal*force[0] ; fy=K_penal*force[1]

for paire in pairesBH :
    listnod.extend( [ paire[2] , paire[1] ] )
    fnodX.extend( [ fx,-fx ] )
    fnodY.extend( [ fy,-fy ] )

gX_V=GEN_SOL(IGEN = 2, NP = 2*len(pairesBH) , NUM= listnod , IL = 1, F= fnodX )
gY_V=GEN_SOL(IGEN = 2, NP = 2*len(pairesBH) , NUM= listnod , IL = 2, F= fnodY )

sol_RL = CHAR_SOL(M1=0, GEN= [ gX_H, gY_H, gX_V, gY_V ] )

return CHAR (M=4, OPT= sol_RL )

```

Le résultat de la fonction est un objet de la classe CHAR que l'on intègre à la définition du jeu de données.

8. 5. Options du module MCNL, constitution des jeux de données et lancement

Bien que le calcul soit linéaire, on utilise ici le module de calcul mécanique non linéaire (MCNL) afin d'utiliser l'option HPE, qui facilite le post-traitement des résultats. On choisit la méthode de résolution dite des contraintes initiales (IMET=1), avec un seul incrément (on applique le chargement affecté d'un coefficient 1.), un nombre d'itérations maximal et une tolérance quelconques : le calcul étant linéaire, on obtient le résultat en une seule itération. On utilise le solveur multifrontal (option MUL), on ajoute l'option DTO pour une visualisation éventuelle des déformations totales, et l'option HPE pour obtenir la valeur de la moyenne des contraintes et des déformations sur la cellule de base.

```

mul = MUL (MOD = 'MCNL', IGND = 1 , IIO = 0 )
dto = DTO (MOD = 'MCNL')
hpe = HPE (MOD = 'MCNL')
mcnl = MCNL (M = 0,
             NINCR = 1, NITER = 500, TOL = 0.001, IMET = 1,
             VCT = [ 1.],
             OPT = [ mul, dto, hpe ] )

```

Pour obtenir les résultats au fichier gmsh et faire remonter dans le répertoire de travail le fichier de sortie au format .list de CESAR, on définit les noms de fichiers nécessaires, et on constitue un jeu de données pour chaque chargement.

```

expo = EXPO (IPGMSH = 1)

ficl_pos = rep_trav + 'cellule_M1.pos'
ficl_list = rep_trav + 'cellule_M1.list'
exe_XX = EXEC(MSH = ficl_pos , LIST = ficl_list)
jeu_XX = JDD(EXEC = exe_XX,
             EXPO = expo,
             COOR = coor,
             ELEM = elem,
             COND = cond,
             CHAR = char_XX,
             MCNL = mcnl
             )

```

On procède de même pour les deux autres tenseurs de déformation que l'on souhaite imposer.

On peut ensuite lancer les calculs :

```
jeu_XX.lancer()  
jeu_YY.lancer()  
jeu_XY.lancer()
```

9. Résultats

On peut visualiser les résultats avec gmsh, par exemple : la figure ci-dessous représente la déformée calculée lorsque l'on impose une déformation moyenne de cisaillement (avec une échelle adaptée pour les déplacements). On note que l'inclusion, plus raide, se déforme moins que la matrice qui l'entoure. On voit également le fait que le déplacement du contour ne correspond pas à une déformation homogène, à cause de la fluctuation périodique \underline{v} .

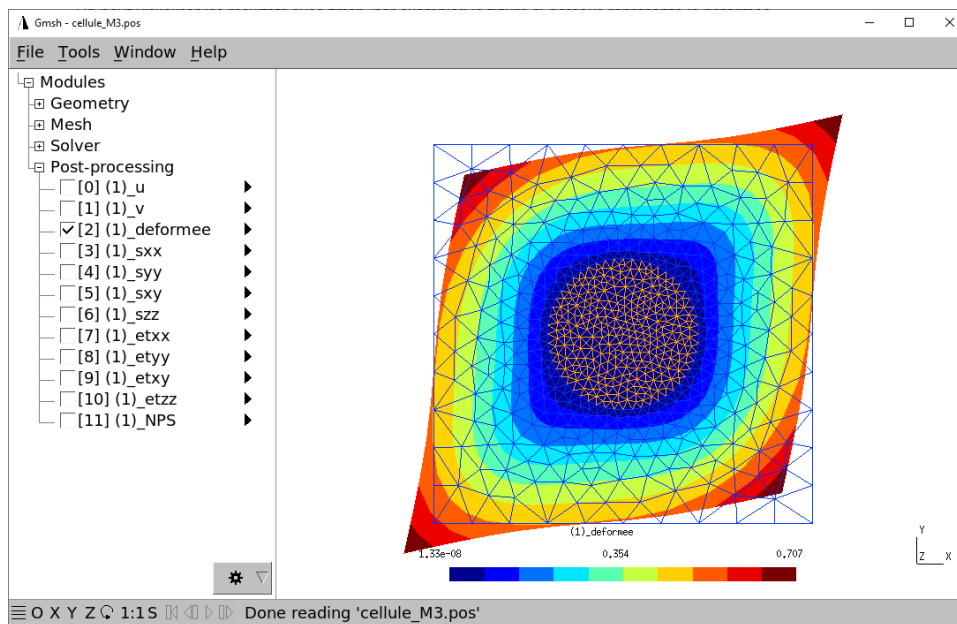


Figure 3 – Exemple de déformée obtenue

La dernière partie du script permet de dépouiller les résultats. L'option HPE permet de calculer l'intégrale et la moyenne sur le maillage des composantes des tenseurs de contrainte et de déformation. Le résultat n'est pas écrit dans le fichier de résultats binaire au format .rsv4, mais seulement dans le fichier de sortie au format list. Quelques lignes permettent de relire les valeurs cherchées, et d'afficher les composantes du tenseur d'élasticité homogénéisé :

```
f = open(fic1_list, "r")  
for line in f :  
    test = re.search(r'.*MOYENNE DES CONTRAINTES.*', line)  
    if (str(test) != 'None') :  
        rep = f.readline().split()  
        CXXX = f"{ float(rep[0]):12.5e} "  
        CYYX = f"{ float(rep[1]):12.5e} "  
        CXYX = f"{ float(rep[2]):12.5e} "  
        CZZX = f"{ float(rep[3]):12.5e} "  
        print("cas 1 : CXXX, CYYX, CXYX, CZZX = ", CXXX, CYYX, CXYX, CZZX)  
f.close()
```

Avec le code analogue pour les deux autres calculs, et en tenant compte du facteur 2 introduit devant Exy dans la définition de $tE = [E_{xx}, E_{yy}, 2E_{xy}]$, on obtient l’affichage suivant :

```
*****
***** SYNTHESE DES RESULTATS : *****
***** MODULES ELASTIQUES HOMOGENEISES *****
*****
cas 1 : CXXXX, CYYXX, CXYXX, CZZXX =  4.82194e+03  2.00079e+03  -7.63245e-04  2.04682e+03
cas 2 : CXXYY, CYYYY, CXYYY, CZZYY =  2.00079e+03  4.82194e+03  -2.88627e-04  2.04682e+03
cas 3 : CXXXY, CYYXY, CXYXY, CZZXY = -7.33320e-04 -4.03678e-04  1.36508e+03  -3.41100e-04
```

Ces valeurs montrent que la solution respecte les symétries attendues (en particulier les égalités entre CXXXX et CYYYY, CXXYY et CYXX, CZZXX et CZZYY, et le fait que CXYXX, CXYYY, CXXXY, CYYXY sont proches de zéro), et les valeurs obtenues pour CXXXX, CYYXX, CZZXX, CXXYY, CYYYY, CZZYY et CXYXY sont identiques à celles données par Dirrenberger et Forest (2010).

10. Conclusion

Le script présenté permet de conduire en Python une étude complète des modules élastiques homogénéisés d’un matériau périodique, dans le cas des déformations planes.

Il pourrait se généraliser sans peine pour obtenir le tenseur complet en trois dimensions.

On a choisi ici de réaliser trois calculs, mais on pourrait regrouper les données dans une seule exécution du solveur CESAR, en définissant trois chargements différents, avec trois incréments et des coefficients appropriés dans le vecteur VCT.

D’autre part, le script permet, moyennant des adaptations plus ou moins simples :

- de prendre en compte une inclusion de forme quelconque (par exemple elliptique), ou un motif qui pourrait représenter un matériau comme la maçonnerie,
- de prendre en compte plus de deux matériaux dans la cellule de base,
- d’étudier la sensibilité des résultats au coefficient de pénalisation,
- d’étudier la sensibilité des résultats à la finesse du maillage.

11. Références

Dirrenberger J, Forest S (2010) Simulation & homogénéisation de microstructures périodiques, http://www.mat.ensmp.fr/Pages/jdirrenb/files/zeb/Notice_Homogeneisation_Zebulon.pdf, 38 p.

Débordes O (2010) Homogénéisation périodique, cours de l'Ecole Centrale de Marseille, 47 p.

12. Annexe : approche alternative « en contraintes »

Dirrenberger et Forest (2010) proposent une autre approche alternative, en contraintes, qui consiste à rechercher l'écart entre le déplacement \underline{u} et le déplacement qui correspondrait à une déformation homogène $\underline{\underline{E}}.x$

Dans le cas élastique linéaire, le problème posé se formule comme suit :

$$\text{div } \underline{\underline{\sigma}} = 0$$

$$\underline{\underline{\sigma}} = \underline{\underline{c}} : \underline{\underline{\varepsilon}}$$

$$\underline{\underline{\varepsilon}} = 1/2 (\text{grad } \underline{u} + {}^t\text{grad } \underline{u})$$

$$\underline{\underline{\varepsilon}} = 1/2 (\underline{\underline{\text{grad}}} \underline{u} + {}^t\underline{\underline{\text{grad}}} \underline{u})$$

où $\underline{\underline{c}}$ est le tenseur des modules d'élasticité (à l'échelle microscopique). Il est facile de voir que ce problème peut se reformuler de la manière suivante

$$\text{div } \underline{\underline{\sigma}}^* = - \text{div } (\underline{\underline{c}} : \underline{\underline{E}}^*)$$

$$\underline{\underline{\sigma}}^* = \underline{\underline{c}} : \underline{\underline{\varepsilon}}^*$$

$$\underline{\underline{\varepsilon}}^* = 1/2 (\underline{\underline{\text{grad}}} \underline{v} + {}^t\underline{\underline{\text{grad}}} \underline{v})$$

où \underline{v} est astreint à vérifier les conditions de périodicité sur le contour.

Cette reformulation consiste simplement à choisir comme inconnue principale la fluctuation périodique \underline{v} et pas le champ de déplacement total \underline{u} . Formellement, on parvient à un problème d'élasticité sur la cellule de base, dans lequel il convient de prendre en compte un chargement volumique égal à $\text{div } (\underline{\underline{c}} : \underline{\underline{E}}^*)$.

L'option SIG du module CHAR (avec IOPT=1) permet de résoudre le problème posé sur \underline{v} , à condition de calculer le produit $(\underline{\underline{c}} : \underline{\underline{E}}^*)$ au moment de la définition des chargements. La mise en œuvre est donc, d'une certaine façon, plus simple que celle consistant à calculer les valeurs des forces nodales à appliquer sur le contour du maillage. Le calcul donne la fluctuation périodique du champ de déplacement.

Le code Python proposé pourrait être adapté très simplement pour mettre en œuvre cette approche.

Attention : Cette option ne permet pas de faire le calcul pour plusieurs tenseurs de déformation imposée dans le même jeu de données (pour des raisons liées au fonctionnement de l'option SIG).