Programme pour simulation Matlab (fonction puis programme principal)

Ce programme a été utilisé pour réaliser la figure 2a.

```
function [a] = Hf(u,theta,b,C)
    a=(-
1+sqrt(cos(theta)^2+(u/b+sin(theta))^2))*sin(acos(cos(theta)/(sqrt(c
os(theta)^2+(u/b+sin(theta))^2))))+C*b*(acos(cos(theta)/(sqrt(cos(th
eta)^2+(u/b+sin(theta))^2)))-
theta)/sqrt(cos(theta)^2+(u/b+sin(theta))^2)*(cos(theta)/(sqrt(cos(t
heta)^2+(u/b+sin(theta))^2)))

% Résolution pour 5 cellules

clear all
clc
clf
format long

%définition des paramètres

%sens fibre
f5=0;
f0=3;

%sens anti-fibre
%f5=-3
%f0=0

b=1;
C=1;
theta=pi/16;
k=1;

%définition des variables du système
syms u1
syms u2
syms u3
syms u4

%résolution du système (Hf est la fonction qui calcule l'expression qui est
%réitérée dans le système)
[x1,x2,x3,x4]=solve(f0+k*u2 == u1*k+2*b*k*Hf(u1,theta,b,C)...
    ,k*u1+k*u3 == u2*k+2*b*k*Hf(u2,theta,b,C)...
    ,u4*k+k*u2 == u3*k+2*b*k*Hf(u3,theta,b,C)...
    ,f5+k*u3 == u4*k+2*b*k*Hf(u4,theta,b,C))
x0=f0/k;
x5=f5/k;

%Tracé des solutions sous la forme de vecteurs
```

```
U=[x5,x4,x3,x2,x1,x0];
x=[1,2,3,4,5,6];
y=[1,1,1,1,1,1];
subplot(1,2,1)
plot(x,y,'.')
quiver(x,y,U,[0,0,0,0,0,0],0.5)
title('Champ de déplacement dans le matériau')
xlabel('Numéro de cellule')
axis ([0 8 0 2])


%Tracé du déplacement de la cellule en fonction du numéro de cellule
subplot(1,2,2)
plot([0,1,2,3,4,5],[x0,x1,x2,x3,x4,x5],'*')
grid
title('Déplacement de chaque cellule pour une traction dans le sens
choisi')
xlabel('Numéro de cellule')
ylabel('Déplacement (mm)')
```

Programme pour simulation Python

Ce programme a été utilisé pour réaliser la figure 2b.

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.optimize as op
from math import *
from numpy import *

def sensfibre(u): #Cette fonction résoud un système de n équations,
n étant à préciser dans la fonction et représentant le nombre de
cellules de nptre matériau, elle revoie le vecteur solution, dans le
cas où l'on tire dans le sens défini comme étant le sens des fibres
n=5 #nombre de cellules du matériau
theta=pi/16 #theta est fixé et correspond à l'angle que font les
branches du matériau avec l'horizontale
C=5 #C homogène à un couple par unité de longueur
b=5 #b représente la longueur des ressorts utilisés pour la
modélisation
k=3.75 #représente la constante de raideur de ces ressorts
#Tous ces paramètres sont ajustables pour permettre de s'approcher
de nos résulatats expérimentaux
f=[1 for i in range (n+2)]
f[0]=3.6 #représente la force=déplacement appliqué à la première
cellule, en mm
f[-1]=0 #symbolise le fait que l'on maintient l'autre extrémité du
matériau fixe
```

```
sol=[u[0]*k+2*b*k*(-
1+sqrt(cos(theta)**2+(u[0]/b+sin(theta))**2))*sin(acos(cos(theta)/(s
qrt(cos(theta)**2+(u[0]/b+sin(theta))**2))))+C*b*(acos(cos(theta)/(s
qrt(cos(theta)**2+(u[0]/b+sin(theta))**2)))-
theta)/sqrt(cos(theta)**2+(u[0]/b+sin(theta))**2)*(cos(theta)/(sqrt(
cos(theta)**2+(u[0]/b+sin(theta))**2)))-f[0]*k-
k*u[1]]+[u[i]*k+2*b*k*(-
1+sqrt(cos(theta)**2+(u[i]/b+sin(theta))**2))*sin(acos(cos(theta)/(s
qrt(cos(theta)**2+(u[i]/b+sin(theta))**2))))+C*b*(acos(cos(theta)/(s
qrt(cos(theta)**2+(u[i]/b+sin(theta))**2)))-
theta)/sqrt(cos(theta)**2+(u[i]/b+sin(theta))**2)*(cos(theta)/(sqrt(
cos(theta)**2+(u[i]/b+sin(theta))**2)))-k*u[i-1]-u[i+1]*k for i in
range(1,n)]+[u[n]*k+2*b*k*(-
1+sqrt(cos(theta)**2+(u[n]/b+sin(theta))**2))*sin(acos(cos(theta)/(s
qrt(cos(theta)**2+(u[n]/b+sin(theta))**2))))+C*b*(acos(cos(theta)/(s
qrt(cos(theta)**2+(u[n]/b+sin(theta))**2)))-
theta)/sqrt(cos(theta)**2+(u[n]/b+sin(theta))**2)*(cos(theta)/(sqrt(
cos(theta)**2+(u[n]/b+sin(theta))**2)))-k*u[n-1]-f[-1]] #système de
n équations
return(sol)
def sensantifibre(u): #la fonction est similaire à la précédente,
cette fois on tire dans le sens défini comme étant le sens anti-
fibres
n=7
theta=pi/16 #comme les fibres sont dans l'autre sens , le nouveau
theta à considérer est pi-pi/16, cela revient à mettre le signe -
devant les cos présents dans l'équation
C=1
b=1
k=1
f=[1 for i in range (n+2)]
f[0]=0
f[-1]=-15 #le déplacement imposé est négatif puisqu'il est orienté
sol=[u[0]*k+2*b*k*(-
1+sqrt(cos(theta)**2+(u[0]/b+sin(theta))**2))*sin(-
acos(cos(theta)/(sqrt(cos(theta)**2+(u[0]/b+sin(theta))**2))))+C*b*(
-acos(cos(theta)/(sqrt(cos(theta)**2+(u[0]/b+sin(theta))**2)))-
theta)/sqrt(cos(theta)**2+(u[0]/b+sin(theta))**2)*(-
cos(theta)/(sqrt(cos(theta)**2+(u[0]/b+sin(theta))**2)))-f[0]*k-
k*u[1]]+[u[i]*k+2*b*k*(-
1+sqrt(cos(theta)**2+(u[i]/b+sin(theta))**2))*sin(-
acos(cos(theta)/(sqrt(cos(theta)**2+(u[i]/b+sin(theta))**2))))+C*b*(
-acos(cos(theta)/(sqrt(cos(theta)**2+(u[i]/b+sin(theta))**2)))-
theta)/sqrt(cos(theta)**2+(u[i]/b+sin(theta))**2)*(-
cos(theta)/(sqrt(cos(theta)**2+(u[i]/b+sin(theta))**2)))-k*u[i-1]-
u[i+1]*k for i in range(1,n)]+[u[n]*k+2*b*k*(-
1+sqrt(cos(theta)**2+(u[n]/b+sin(theta))**2))*sin(-
acos(cos(theta)/(sqrt(cos(theta)**2+(u[n]/b+sin(theta))**2))))+C*b*(
```

```
-acos(cos(theta)/(sqrt(cos(theta)**2+(u[n]/b+sin(theta))**2)))-
theta)/sqrt(cos(theta)**2+(u[n]/b+sin(theta))**2)*(-
cos(theta)/(sqrt(cos(theta)**2+(u[n]/b+sin(theta))**2)))-k*u[n-1]-
f[-1]]
return(sol)
def vect(n): #Cette fonction définit un vecteur n, n étant le nombre
de cellules, pour fournir à la fonction solve une idée d'où chercher
les solutions
vecteur=[3]
for i in range (n):
vecteur.append(vecteur[i]/2)
return vecteur
def solve(g,u): #prend en argument la fonction sens fibre ou
sensantifibre ainsi que vect, et affiche les solutions sur un graphe
(déplacement en fonction du numéro de la cellule)
plt.plot(op.fsolve(g,u))
def save(g,u): #enregistre les solutions trouvées par solve sous un
fichier texte
np.savetxt('solutions.txt',np.transpose(op.fsolve(g,u)),fmt='%.18e',
newline='\n',delimiter=',')
```