Ecole Polytechnique de Tunisie

TP éléments finis

COMPTE RENDU

Réalisé par :
TRABELSI Ilyes

Enseignante :
REZGUI Taysir

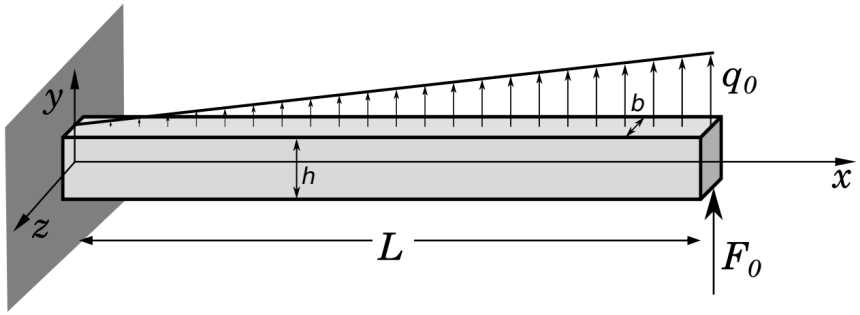
30 mai 2020

Table des matières

1 Introduction

Dans le TP d'élément fini nous allons travailler sur la résolution numérique de problème d'une poutre en flexion pour chercher ses déformations en chaque point de cette poutre.

Pour ce faire on va travailler avec une poutre rectangulaire de la longueur $L = 1m$ et épaisseur h = 150 mm avec $E = 200GPa$ $I = 29/10^6$ m^4 soumise à une densité linéique q_0 et une force ponctuelle en x = L :



2 Matrice de rigidité et vecteur force élémentaire

Le but de cette partie est d'implémenter un script qui permet de calculer la matrice de rigidité et le vecteur force élémentaire d'une poutre en flexion en utilisant des éléments d'Hermite à deux nœuds.

2.1 Implémentation des fonctions psi et ses dérivés secondes

Les fonctions (psi) sont définit de la manière suivante :

$$\psi_1^e(\bar{x}) = 1 - \frac{3\bar{x}^2}{h_e^2} + \frac{2\bar{x}^3}{h_e^3}$$
$$\psi_2^e(\bar{x}) = \bar{x} - \frac{2\bar{x}^2}{h_e} + \frac{\bar{x}^3}{h_e^2}$$
$$\psi_3^e(\bar{x}) = \frac{3\bar{x}^2}{h_e^2} - \frac{2\bar{x}^3}{h_e^3}$$
$$\psi_4^e(\bar{x}) = -\frac{\bar{x}^2}{h_e} + \frac{\bar{x}^3}{h_e^2}$$

En utilisant le langage octave ces fonctions sont programmées de la manière suivante :

```
function y=psi(i,xb,he)
  switch (i)
    case(1)
      y=1-3*(xb./he).^2+2*(xb./he).^3;
    case(2)
      y=xb.-2/he*xb.^2+xb.^3/he^2;
    case(3)
      y=3*(xb./he).^2-2*(xb./he).^3;
    case(4)
      y=-xb.^2/he+xb.^3/he^2;
    endswitch
endfunction
```

```
function y=psisec(i,xb,he)
  switch (i)
    case(1)
      y=-6/he^2+12*xb./he^3;
    case(2)
      y=-4/he+6*xb./he^2;
    case(3)
      y=6/he^2-12*xb./he^3;
    case(4)
      y=-2/he+6*xb./he^2;
    endswitch
endfunction
```

2.2 Matrice de rigidité élémentaire

Dans le repère local où x varie entre 0 et h_e les éléments du matrice K sont définit par :

$$K_{ij}^e = \int_0^{h_e} E(\bar{x}) I_{Gz}(\bar{x}) \psi_i^{e\prime\prime}(\bar{x}) \psi_j^{e\prime\prime}(\bar{x}) d\bar{x}$$

Par un changement du variables dans l'intégrale on peut alors implémenter la fonction [Ke] = Kele(x_1,x_2,E,I)).

```
function [ke]=kele(x1,x2,E,I)
  he=x2-x1;
  k=zeros(4,4);
  xb=0:1/100:he;
  for i=1:4
    for j=1:4
      k(i,j)=E.*I.*trapz(xb,psisec(i,xb,he).*psisec(j,xb,he));
    endfor
  endfor
  ke=k;
endfunction
```

2.3 Vecteur force élémentaire

Dans le repère local où x varie entre 0 et he les éléments du Vecteur force élémentaire sont définit par :

$$f_i^e = \int_0^{h_e} \psi_i^e q d\bar{x}$$

Par conséquent la fonction e function [fe] = fele(x_1,x_2,q) est implémenter comme ceci :

```
function [fe]=fele(x1,x2,q0,L)
   he=x2-x1;
   f=[];
   xb=0:1/100:he;
   for i=1:4
   q=(xb.*q0./L); % charge linéaire
    %=q0 ; % charge constante
    f(i)= trapz( xb,q.*psi(i,xb,he));
    endfor
   fe=f';
 endfunction
```

3 Assemblage et conditions aux limites

Dans cette partie nous allons construire le système matriciel à résoudre pour la poutre en flexion en travaillant sur deux types de maillages :

– Un maillage régulier à éléments d'Hermite P3 avec nh éléments de même longueur $h_1 = h_2 = ... = h_{nh} = L/n_h$. Les noeuds sont donc situés aux positions :

$$x_i = (i-1)\frac{L}{n_h}, \quad i = 1, \ldots, n_h + 1$$

– Un maillage variable à éléments d'Hermite P3 avec nh éléments dont les noeuds sont situés aux positions :

$$x_i = \frac{L}{2}\left(1 - \cos\left(\frac{i-1}{n_h}\pi\right)\right), \quad i = 1, \ldots, n_h + 1$$

3.1 Assemblage de la matrice de rigidité et du vecteur force

Pour imposer les conditions d'équilibre, il est nécessaire d'additionner les 3*eme* et 4eme lignes de l'élément e à la 1ere et 2eme lignes de l'élément e + 1, on aura donc la forme globale de la matrice de rigidité suivante :

$$[K] = \begin{bmatrix} K_{11}^1 & K_{12}^1 & K_{13}^1 & K_{14}^1 & 0 & 0 & \cdots \\ K_{12}^1 & K_{22}^1 & K_{23}^1 & K_{24}^1 & 0 & 0 & \cdots \\ K_{13}^1 & K_{23}^1 & K_{33}^1 + K_{11}^2 & K_{34}^1 + K_{12}^2 & K_{13}^2 & K_{14}^2 & \cdots \\ K_{14}^1 & K_{24}^1 & K_{34}^1 + K_{12}^2 & K_{44}^1 + K_{22}^2 & K_{23}^2 & K_{24}^2 & \cdots \\ 0 & 0 & K_{13}^2 & K_{23}^2 & K_{33}^2 + K_{11}^3 & K_{34}^2 + K_{12}^3 & \cdots \\ 0 & 0 & K_{14}^2 & K_{24}^2 & K_{34}^2 + K_{12}^3 & K_{44}^2 + K_{22}^3 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

Et le vecteur de force généralisée est décrit par :

$$\{f\} = \left\{f_1^1, f_2^1, f_3^1 + f_1^2, f_4^1 + f_2^2, \cdots, f_4^{n_h-1} + f_2^{n_h}, f_3^{n_h}, f_4^{n_h}\right\}^T$$

Donc en utilisant les fonction (fele et kele) développées précédemment on imprimante les fonctions Kele-gene , fele-gene :

```
%1er maillage
function k= kele_gene(n,l,E,I)
  k=zeros(2*n+2,2*n+2);
  x=0:l/n:l;
  l=0;
  for i=1:2:(2*n)
    l=l+1;
    bloc=i:i+3;
    mat=k(bloc,bloc)+kele(x(l),x(l+1),E,I);
    k(bloc,bloc)=mat;
  endfor
endfunction
```

```
 %2eme maillage
function k= kele_gene1(n,l,E,I)
  k=zeros(2*n+2,2*n+2);
  x=zeros(n+1,1);
  for i=1:n+1
    x(i)=l/2*(1-cos((i-1)*pi/n));
  endfor
  l=0;
  for i=1:2:(2*n)
    l=l+1;
    bloc=i:i+3;
    mat=k(bloc,bloc)+kele(x(l),x(l+1),E,I);
    k(bloc,bloc)=mat;
  endfor
endfunction
```

```
%1er maillage
function f= fele_gene(n,l,q0,L)
  f=zeros(2*n+2,1);
  x=0:l/n:l;
  l=0;
  for i=1:2:2*n
    l=l+1;
    bloc=i:i+3;
    vect=f(bloc,1)+fele(x(l),x(l+1),q0,L);
    f(bloc,1)=vect;
  endfor
endfunction
```

```
%2eme maillage
function f= fele_gene1(n,l,q0,L)
  f=zeros(2*n+2,1);
  x=zeros(n+1,1);
  for i=1:n+1
    x(i)=L/2*(1-cos((i-1)*pi/n));
  endfor
  l=0;
  for i=1:2:2*n
    l=l+1;
    bloc=i:i+3;
    vect=f(bloc,1)+fele(x(l),x(l+1),q0,L);
    f(bloc,1)=vect;
  endfor
endfunction
```

3.2 Application des conditions aux limites :

Pour le vecteur déplacement U en x=0 la déformation et rotation sont nulles car il y a une liaison encastrement avec la poutre est le mur (support) et les autres composantes sont inconnues.
Pour le vecteur Q on a les 1er deux composantes sont inconnues (la réactions du support) est la composante avant dernière est égale a F0 (force ponctuelle appliquer en x = L) est les autres composantes sont nulles.

Donc dans le langage octave ces vecteurs sont définis de la manière suivante :

```
U=zeros(2*nh+2,2);

% Les vecteurs U et Q ont deux colonnes

Q=zeros(2*nh+2,2);

% la première définit l'état (connu / inconnu) (1/0)
et la seconde contient les valeurs
%pour U

U(1,1)=1;
U(2,1)=1;
```

```
%pour Q

for i=3:2*nh+2
  Q(i,1)= 1;
endfor
Q(2*nh+1,2)=F0;
```

4 Solution en déplacement et post traitement

Finalement dans cette partie nous allons aborder la résolution du système pour trouver la solution déplacement U = K F.

4.1 Résolution de système

Nous savons que K*U = f + Q donc il suffit d'inverser la matrice K pour trouver la solution :

```
%1er maillage

  x=0:1/nh:1;



% fonction de la 2eme maillage

  function x1=maill_2(nh,L)
      x1=zeros(nh+1,1);
      for i=1:nh+1
        x1(i)=L/2*(1-cos((i-1)*pi/nh));
      endfor
 endfunction
```

```
%deplacement dans les deux maillage

  function [UT,U]=deplacement_maill_1(nh,q0,L,E,I,Q)
    F=Q(3:2*nh+2,2).+fele_gene(nh,1,q0,L)(3:2*nh+2);
    K1=kele_gene(nh,1,E,I)(3:2*nh+2,3:2*nh+2);
    U1=inv(K1)*F;
    UT=zeros(2*nh+2,1);
    UT(3:2*nh+2)=U1;
    U=UT(1:2:2*nh+2);
  endfunction



  function [UT,U]=deplacement_maill_2(nh,q0,L,E,I,Q)
    F=Q(3:2*nh+2,2).+fele_gene1(nh,1,q0,L)(3:2*nh+2);
    K1=kele_gene1(nh,1,E,I)(3:2*nh+2,3:2*nh+2);
    U1=inv(K1)*F;
    UT=zeros(2*nh+2,1);
    UT(3:2*nh+2)=U1;
    U=UT(1:2:2*nh+2);
  endfunction
```

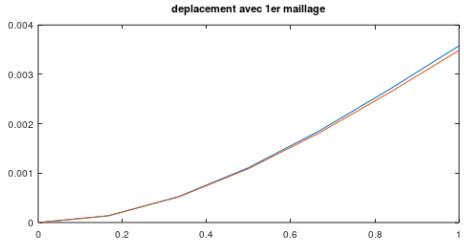
4.2 Post-traitement

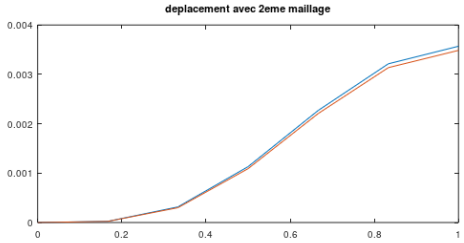
Apres avoir la solution on va comparer la déformation avec la solution analytique suivante :

$$v(x) = \frac{F_0 x^2}{6EI}(3L - x) + \frac{q_0 L^4}{120EI}\left(20\frac{x^2}{L^2} - 10\frac{x^3}{L^3} + \frac{x^5}{L^5}\right)$$

Avec le langage octave on peut comparer en traçant les deux courbes sur la même figure :

```
% déformation théorique

    function v =deformation_theo(x,nh,q0,L,E,I,F0)
      v =(F0/(6*E*I)*x.^2).*(3*L-x)+q0*L^4/(120*E*I)*
      (20*x.^2/L^2-10*x.^3/L^3+x.^5/L^5);
    endfunction

% Utilisant la commande plot pour visualiser les courbes.
```



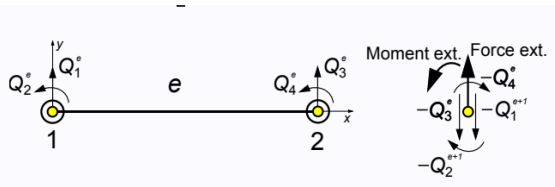


D'après ces figures on peut confirmer notre résolution car les deux courbes sont proche avec les deux maillages.

En utilisant la solution trouvé et la forme local de l'équilibre sur un élément (Q=K*U-F) on peut calculer les variables secondaires dans un élément donné :

```
%détermination de Q dans un element donné

  function q=Q(i,nh,U,E,I,L,q0) % i cad le ieme element
    x=0:1/nh:1;
    n=zeros(nh,1);
    j=1;
    for k=1:nh
      n(k)=j;
      j=j+2;
    endfor
    q=(kele(x(i),x(i+1),E,I)*U(n(i):n(i)+3,1)).
    -fele(x(i),x(i+1),q0,L);
  endfunction
```

A partir des variables secondaires trouves on peut calculer les moments fléchissant et les efforts tranchants aux nœuds :



Par conséquent on a

$$Q_1 = -T_y(0),\ Q_2 = -M_{fz}(0),\ Q_3 = T_y(h_e),\ Q_4 = M_{fz}(h_e)$$

code Apres avoir trouvé les moments fléchissant et les efforts tranchants aux nœuds on les assemble et comparant avec les solutions analytiques donnés par :

$$T_y(x) = F_0 + \frac{q_0 L}{2}\left(1 - \frac{x^2}{L^2}\right)$$

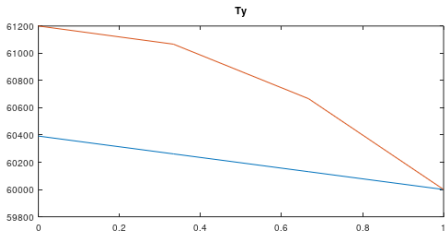
$$M_{fz}(x) = F_0(L - x) + \frac{q_0 L^2}{6}\left(2 - 3\frac{x}{L} + \frac{x^3}{L^3}\right)$$

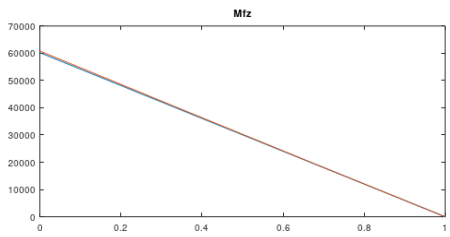
```
% Mfz_pratique


  function m=Mfz_pratique(nh,U,E,I,L,q0)
    m=zeros(nh+1,1);
    m(1)=-Q(1,nh,U,E,I,L,q0)(2);
    for i=2:nh+1
    m(i)=Q(i-1,nh,U,E,I,L,q0)(4);
    endfor
  endfunction


% Ty_partique
```

```
function t=Ty_partique(nh,U,E,I,L,q0)
  t=zeros(nh+1,1);
  t(1)=-Q(1,nh,U,E,I,L,q0)(1);
  for i=2:nh+1
  t(i)=Q(i-1,nh,U,E,I,L,q0)(3);
  endfor
endfunction
```

```
% fonctions théoriques
  function t=Ty_theo(x,I,L,q0,F0)
    t=F0+q0*L/2*(1-(x.^2)./L^2);
  endfunction
```

```
  function m=Mfz_thep(x,I,L,q0,F0)
    m= F0*(L-x).+q0*L^2/6*(2-3*x./L+x.^3/L^3);
  endfunction
   % Utilisant la commande plot pour visualiser les courbes.
```





Il est clair que les courbes sont proches.

5 Conclusion

D'après les courbes tracées on peut confirmer que la méthode numérique pour la résolution des problèmes d'élément finie est précis ce qui nous aide les ingénieurs dans les simulations graphique afin de déterminer certaine caractéristique sur le système étudié pour bien le réaliser.