


ES Elements Finis

Optimisation de l'épaisseur d'une structure tridimensionnelle

Leticia Gomes
Pollet Florent

Encadrants : **David Ryckelynck, Nicole Spillane et Pierre-Henri Tournier**

24 novembre 2021

Table des matières

1 Présentation du problème

2 Formulation variationnelle

Etape 1
Mise sous équation

Réponse Blabla

3 Résolution à épaisseur fixée

4 Optimisation de l'épaisseur

4.1 Approximation graphique

4.2 Raffinement de la solution

A Code source

Disponible sur ...

```
load "medit"
include "cube.idp"

// question : comment retourner un tuple simuler déplacement pour faire évolution maillage ?
// question : définition de la normale n ? surface 2d ok ?
// quesiton : càd moitié de la structure ?
// question : pont pas variable globale
// question : comment définir la région  sur laquelle intégrer ?
// rédaction ?
// todo commenter code

real sqrt2=sqrt(2.);
macro epsilon(u1,u2,u3)  [dx(u1),dy(u2),dz(u3),(dz(u2)+dy(u3))/sqrt2,(dz(u1)+dx(u3))/sqrt2,(dy(u1)+dx(u2))/sqrt2] // EOM
macro div(u1,u2,u3) ( dx(u1)+dy(u2)+dz(u3) ) // EOM

// Géométrie
real r = 0.35;
real h = 0.4;
real l = 1.7-r*2;
real p = 0.5;

// Grandeurs mécaniques
real fz = -5e8;
real E = 2.1e11;
real nu = 0.3;
real g = -9.81;
real rho = 7800;

// Grandeurs calculées
real gravity = rho*g;
real mu = E/(2*(1+nu));
real lambda = E*nu/((1+nu)*(1-2*nu));

mesh3 pont = cube(2,2,2); // default

func mesh build2d(bool show, bool half, real e, int n)
{
    border b1(t=1, -1){x=t*l/2; y=0; label=1;};
    border b2(t=-1, 1){x=t*l/2; y=-e; label=2;};

    border b3(t=0, pi/2){x=r*cos(t)+l/2; y=r*sin(t)-r; label=3;};
    border b4(t=pi/2, 0){x=(r-e)*cos(t)+l/2; y=(r-e)*sin(t)-r; label=4;};

    border b5(t=0, 1){x=l/2+r-e; y=-t*h-r; label=5;};
    border b6(t=1, 0){x=l/2+r; y=-t*h-r; label=6;};
    border b7(t=1, 0){x=l/2+r-e*t; y=-h-r; label=7;};


    border b8(t=pi/2, 0){x=-(r*cos(t)+l/2); y=r*sin(t)-r; label=8;};
```

```
50      border b9(t=0, pi/2){x=-((r-e)*cos(t)+l/2); y=(r-e)*sin(t)-r; label=9;};
51
52      border b10(t=1, 0){x=-(l/2+r-e); y=-t*h-r; label=10;};
53      border b11(t=0, 1){x=-(l/2+r); y=-t*h-r; label=11;};
54      border b12(t=0, 1){x=-(l/2+r-e*t); y=-h-r; label=12;};
55
56      border b13(t=1,0){x=0; y=t*e-e;label=18;};
57
58
59      border b14(t=1, 0){x=t*l/2; y=0; label=1;};
60      border b15(t=0, 1){x=t*l/2; y=-e; label=2;};
61      border b16(t=1, 0){x=0; y=t*e-e; label=8;};
62
63      mesh pont2d = buildmesh(b1(n) + b2(n) + b3(n) + b4(n) + b5(n) + b6(n) + b7
            (n) + b8(n) + b9(n) + b10(n) + b11(n) + b12(n));
64      if (half)
65      {
66          pont2d = buildmesh(b3(n) + b4(n) + b5(n) + b6(n) + b7(n) + b14(n) +
                b15(n) + b16(n));
67      }
68      if (show)
69      {
70          plot(pont2d, wait=true);
71      }
72
73      return pont2d;
74  }
75
76  func mesh3 build3d(mesh pont2d, bool show, int m)
77  {
78      mesh3 pontretourne = buildlayers(pont2d, m, zbound=[-p/2,p/2]);
79      pont = movemesh3(pontretourne, transfo=[x, z, y], orientation=-1);
80
81      if (show)
82      {
83          plot(pont, wait=true);
84      }
85
86      return pont;
87  }
88
89  func real simulerdeplacement(bool show, real e)
90  {
91      fespace Vh(pont,[P1,P1,P1]); // P1 ou P2 ?
92      Vh [u1,u2,u3], [v1,v2,v3];
93
94      solve Lame([u1,u2,u3],[v1,v2,v3])=
95      int3d(pont)(
96              lambda*div(u1,u2,u3)*div(v1,v2,v3)
97              +2.*mu*( epsilon(u1,u2,u3)'*epsilon(v1,v2,v3) )
98              )
99      - int3d(pont) (gravity*v3)
100     + on(7,u1=0,u2=0,u3=0) // condition bord bas (Dirichlet)
101     + on(12,u1=0,u2=0,u3=0) // condition bord bas (Dirichlet)
```

```
102     - int2d(pont, 1)(fz*v3) // condition force surfacique (Neumann)
103     ;
104
105     real dmax= u3[].min; // min ou max ?
106
107     if (show)
108     {
109         real coef= 0.1/abs(dmax);
110         int[int] ref2=[1,0,2,0];
111         mesh3 pontdeforme=movemesh3(pont,transfo=[x+u1*coef,y+u2*coef,z+u3*
                coef],label=ref2);
112         //pont=change(pontdeforme,label=ref2);
113         plot(pontdeforme, wait=1,cmm="e="+e+"␣|␣coef␣␣amplification="+coef+"␣|
                ␣dep=" + dmax );
114     }
115
116     return dmax;
117 }
118
119 func real deplacementmax(bool show, bool half, real e, int n, int m)
120 {
121     mesh pont2d = build2d(show, half, e, n);
122     pont = build3d(pont2d, show, m);
123     real dmax = simulerdeplacement(show, e);
124     return dmax;
125 }
126
127 string s;
128 cout << "Tapez␣s␣pour␣voir␣la␣configuration␣classique␣g␣pour␣tracer␣le␣graphe␣
       m␣pour␣simuler␣avec␣la␣moitie␣de␣la␣maille␣a␣pour␣simuler␣avec␣ladaptation
       ␣du␣mesh␣d␣pour␣la␣dichotomie" << endl;
129 cin >> s;
130
131 int n = 10;
132 int m = n/4;
133
134 if (s == "g")
135 {
136     real e = 0.1;//0.001; pas trop petit pour intéressant
137     real inc = 0.006;
138     real rmax = 0.3; //r;
139     int nb = (rmax-e)/inc + 1;
140     real[int] xx(nb), yy(nb);
141     int i = 0;
142
143     while (e < rmax)
144     {
145         real dmax = deplacementmax(false, false, e, n, m);
146         xx[i] = e;
147         yy[i] = dmax;
148         cout << "e=" << e << endl;
149         e += inc;
150         i += 1;
151     }
```

```
152
153     {// file for gnuplot
154         ofstream gnu("plot.txt");
155         for (int i = 0; i < nb; i++)
156            gnu << xx[i] << " " << yy[i] << endl;
157     }
158     plot([xx, yy], wait=true);
159 }
160 else if (s == "d")
161 {
162     real emax = 0.13;
163     real emin = 0.17;
164     real dmax = 0.1;
165     real d = dmax*100;
166     real e = 0;
167     real eps = 0.005;
168
169     while (abs(d-dmax) > eps)
170     {
171         e = (emax+emin)/2;
172         d = abs(deplacementmax(false, false, e, n, m));
173
174         cout << "emin=" << emin << endl;
175         cout << "emax=" << emax << endl;
176         cout << "e=" << e << endl;
177         cout << "d=" << d << endl;
178
179         if (d > dmax)
180         {
181             emax = e;
182         }
183         else
184         {
185             emin = e;
186         }
187     }
188 }
189 else if (s == "a") // todo adaptation manuelle
190 {
191     real e = 0.1;
192     bool show = true;
193
194     //mesh pont2d = build2d(show, half, e, n);
195     //mesh3 pont = build3d(pont2d, show, m);
196     //simulerdeplacement(pont, show);
197
198     //while (true)
199     //{
200         //Vh[int] u = simulerdeplacement(pont, show);
201
202         //pont2d = adaptmesh(pont2d, [u[0], u[1]]);
203         //pont = build3d(pont2d, show, m);
204     //}
205 }
```

```
206 else if (s == "m") // todo adaptation manuelle
207 {
208     real e = 0.1;
209     bool show = true;
210     deplacementmax(show, true, e, n, m);
211 }
212 else if (s == "s") // todo adaptation manuelle
213 {
214     real e = 0.155;
215     bool show = true;
216     deplacementmax(show, false, e, n, m);
217 }
218 // grep "mmg3d(" */*.edp
219 //adapter les paramètres du mesh à la main sinon car contrôle la création
```