Mini-projet d'analyse numérique (MAP431)
Calcul de fréquences propres de vibration

Dimitri Lozeve Philippe Cherabier

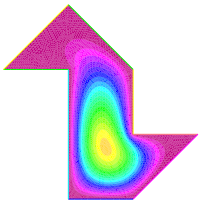
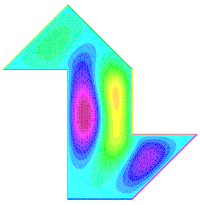
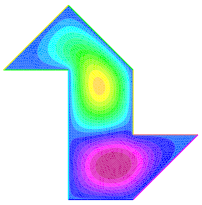
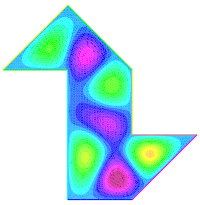
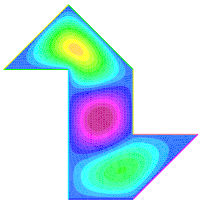
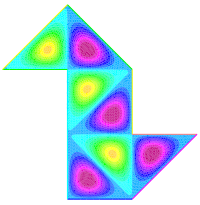
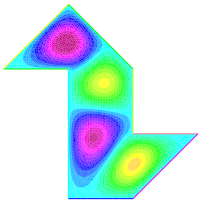
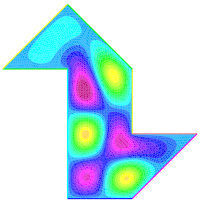
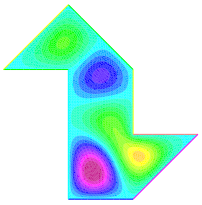
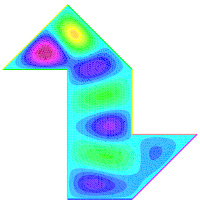
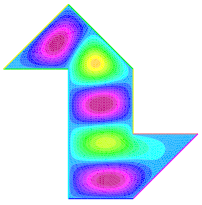
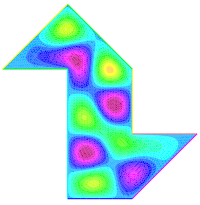
Vendredi 25 mars 2016

1 Code source

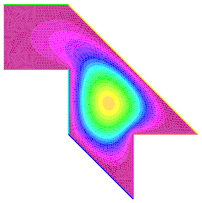
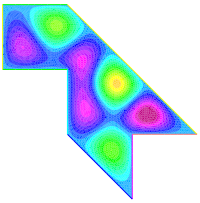
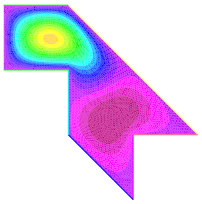
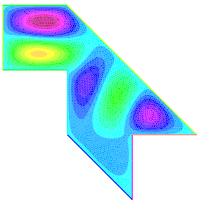
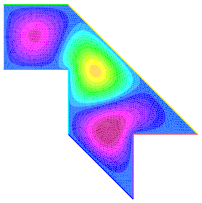
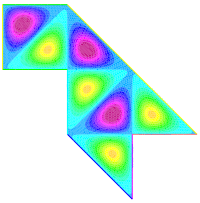
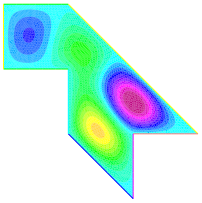
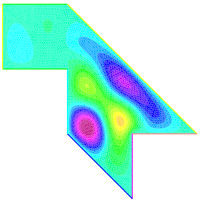
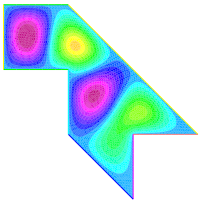
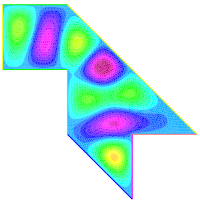
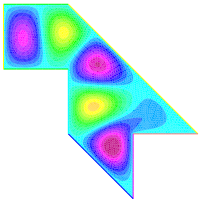
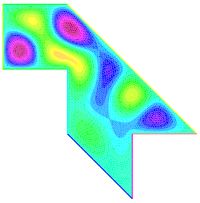
```
/* Mini-projet d'analyse numérique (MAP431)
   Calcul de fréquences propres de vibration
   Dimitri Lozeve, Philippe Cherabier
*/

load "Element_P3";

// Mesh generation

// number of vertices for each edge
int n = 10;
/*
int n;
cout << "Number of vertices = ? ";
cin >> n;
*/

//*
// Cocotte
border c01(t=0,1){ x=t; y=2+t; }
border c02(t=0,1){ x=t; y=2; }
border c03(t=0,1){ x=1; y=2*t; }
border c04(t=0,1){ x=1+t; y=0; }
border c05(t=0,1){ x=2+t; y=t; }
border c06(t=0,1){ x=2+t; y=1; }
border c07(t=0,1){ x=2; y=1+t; }
border c08(t=0,1){ x=2-t; y=2+t; }

plot(c01(-n)+c02(n)+c03(-2*n)+c04(n)+c05(n)+c06(-n)+c07(n)+c08(n),wait=1);
mesh Th = buildmesh(c01(-n)+c02(n)+c03(-2*n)+c04(n)+c05(n)+c06(-n)+c07(n)+c08(n));
//*/
/*
// Flèche
border c01(t=0,1){ x=0; y=2+t; }
```

```
35 border c02(t=0,1){ x=t; y=2; }
36 border c03(t=0,1){ x=1; y=1+t; }
37 border c04(t=0,1){ x=1+t; y=1-t; }
38 border c05(t=0,1){ x=2; y=t; }
39 border c06(t=0,1){ x=2+t; y=1; }
40 border c07(t=0,1){ x=1+2*t; y=3-2*t; }
41 border c08(t=0,1){ x=t; y=3; }
42
43 plot(c01(-n)+c02(n)+c03(-n)+c04(n)+c05(n)+c06(n)+c07(-2*n)+c08(-n),wait=1);
44 mesh Th = buildmesh(c01(-n)+c02(n)+c03(-n)+c04(n)+c05(n)+c06(n)+c07(-2*n)+c08(-n));
45 //*/
46
47 plot(Th,wait=1,ps="mesh.eps");
48
49 // Variational space definition
50 fespace Vh(Th,P3);
51
52 // Variational problem definition
53 varf a(u,v) = int2d(Th)(dx(u)*dx(v) + dy(u)*dy(v))
54   + on(c01,c02,c03,c04,c05,c06,c07,c08,u=0);
55
56 varf b(u,v) = int2d(Th)(u*v);
57
58 // Matrices computation
59 matrix A = a(Vh,Vh,solver=sparsesolver);
60 matrix B = b(Vh,Vh,solver=sparsesolver);
61
62 // Number of eigenvalues desired
63 int nev = 12;
64 // Arrays to keep eigenvalues and eigenvectors
65 real[int] lambda(nev);
66 Vh[int] u(nev);
67
68 // Resolution of the problem
69 int k = EigenValue(A,B,sym=true,sigma=0,
70                    value=lambda,vector=u,tol=1.e-10,ncv=0,maxit=0);
71
72 /*
73 // Saving eigenvalues
74 {
75   ofstream f("eigenvalues.txt",append);
76   for (int i=0; i<nev; i++) {
77     f << lambda[i] << "\t";
78   }
79 };
80 */
81
82 // Plotting eigenvectors
83 for (int i=0; i<nev; i++) {
84   plot(u[i],wait=1,fill=1,ps="cocotte"+(i+1)+".eps");
85        //       cmm="Eigenvalue: "+lambda(i));
86  }
```

2 Résultats pour la cocotte

Valeur propre	Mode propre	Valeur propre	Mode propre
10.1521		42.3883	
14.6223		46.1657	
20.7028		49.348	
26.1503		52.2155	
28.9929		57.2557	
36.8374		63.4864	

3 Résultats pour la flèche

Valeur propre	Mode propre	Valeur propre	Mode propre
10.1520		42.3882	
14.6223		46.1657	
20.7027		49.3480	
26.1503		52.2154	
28.9928		57.2557	
36.8374		63.4863	

4 Convergence des valeurs propres

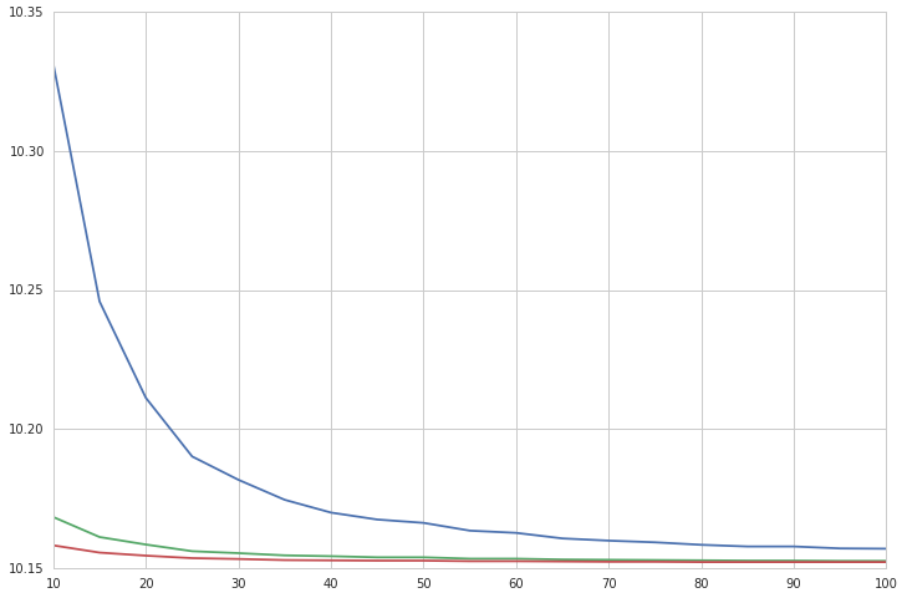


FIG. 1 : Convergence de la première valeur propre de la cocotte, pour les éléments finis $\mathbb{P}_1$, $\mathbb{P}_2$ et $\mathbb{P}_3$, par ordre croissant de vitesse de convergence

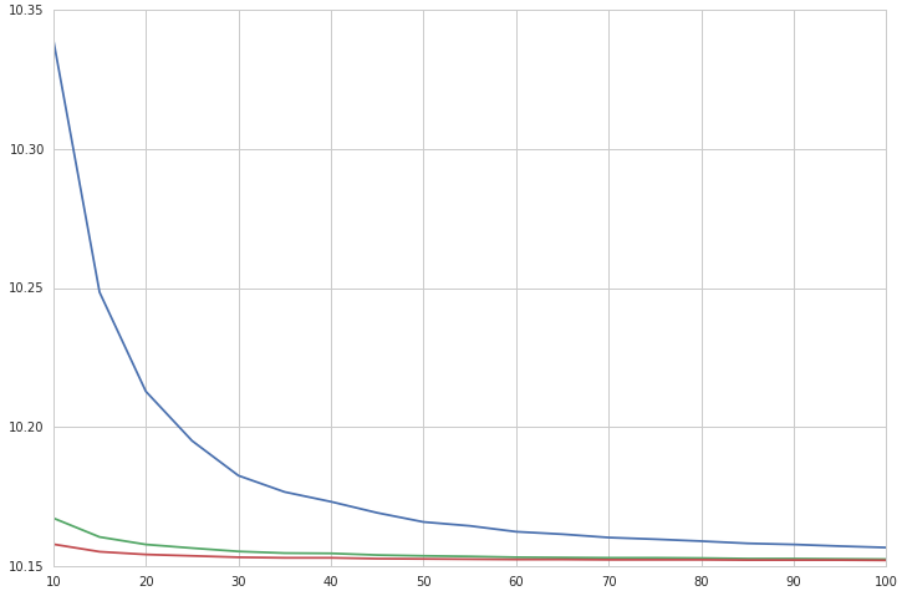


FIG. 2 : Convergence de la première valeur propre de la flèche, pour les éléments finis $\mathbb{P}_1$, $\mathbb{P}_2$ et $\mathbb{P}_3$, par ordre croissant de vitesse de convergence