MM031-Informatique scientifique: Examen du 15 Avril 2010

I. Danaila, F. Hecht, O. Pironneau

2 heures, les documents sont autorisés

Rendre chacune des 2 parties sur des feuilles séparées avec votre nom sur chacune et **ATTENTION les 2 parties sont notées sur 10 chacune.**

1 Partie théorique (O. Pironneau) : 1 heure

Soit Ω un ouvert borné de $\mathbb{R}^2$ de frontière $\Gamma = \Gamma_1 \cup \Gamma_2$ avec $\Gamma_1 \cap \Gamma_2$ réduit à deux points; on note $n(x)$ la normale extérieure en $x \in \Gamma$. Soit un champ de vecteurs $\boldsymbol{a}(x) \in \mathbb{R}^2$ pour tout $x \in \Omega$ tel que $\nabla \cdot \boldsymbol{a} = 0$ et $\boldsymbol{a}(x) \cdot n(x)$ est negatif si $x \in \Gamma_1$ et positif ou nul si $x \in \Gamma_2$. Soient 2 fonctions à valeurs réelles $x \in \Omega \to f(x) \in \mathbb{R}$ et $x \in \Omega \to g(x) \in \mathbb{R}$. On considère le problème

$$\boldsymbol{a} \cdot \nabla u = f \text{ dans } \Omega, \quad u|_{\Gamma_1} = g \tag{1}$$

On commence par essayer de trouver u comme solution de

$$\min_{u-g \in V} E(u) := \int_\Omega |\boldsymbol{a} \cdot \nabla u - f|^2 \text{ où } V = \{v \in H^1(\Omega) \ : \ v|_{\Gamma_1} = 0\} \tag{2}$$

Notez que même si la solution u existe, rien ne dit que $E(u)$ soit nul.

Q0 Montrer que la solution de (2) vérifie : $u - g \in V$ et

$$\int_\Omega (\boldsymbol{a} \cdot \nabla u - f)(\boldsymbol{a} \cdot \nabla w) = 0, \quad \forall w \in V \tag{3}$$

Q1 Soit $\boldsymbol{a} \otimes \boldsymbol{a}$ la matrice de coefficients $a_i a_j$. En déduire que la solution de (2) vérifie :

$$-\nabla \cdot (\boldsymbol{a} \otimes \boldsymbol{a} \nabla u) = -\nabla \cdot (\boldsymbol{a} f), \quad u|_{\Gamma_1} = g, \quad (\boldsymbol{a} \cdot n)(\boldsymbol{a} \nabla u - f)|_{\Gamma_2} = 0 \tag{4}$$

Q2 Soit V_h l'approximation par éléments finis P^1 de V. Ecrire le système linéaire à résoudre issu de (3_h), i.e. (3) avec V remplacé par V_h. On donnera la matrice, sa taille et le second membre du système linéaire en terme des fonctions chapeaux.

Q3 Ecrire l'algorithme du gradient conjugué pour résoudre (2_h), sans construire la matrice.

Q4 On peut montrer que si pour tout x_0 les $\boldsymbol{a}$-trajectoires sont conformes, i.e. les solutions de $\dot{X}(s) = \boldsymbol{a}(X(s)),\ s \in \mathbb{R},\ X(0) = x_0$ intersectent chaque Γ_i en un point unique, i=1,2, alors la solution de (1) existe et est unique. Si ce n'est pas le cas, la méthode risque de ne pas marcher.

Proposer une régularisation du problème (2_h) pour que la solution soit unique même lorsque les trajectoires ne sont pas conformes.

Q5 On change de méthode . On va chercher $u \in H^1(\Omega)$ tel que

$$\int_\Omega (\boldsymbol{a} \cdot \nabla u - f) w = 0, \quad \text{pour tout } w \in L^2(\Omega) \tag{5}$$

On discrétise ce problème en remplaçant $L^2(\Omega)$ par W_h, l'espace des fonctions constantes par morceaux sur les triangles d'une triangulation de Ω. Sachant que $\nabla \cdot (\boldsymbol{a} u) = \boldsymbol{a} \cdot \nabla u$, montrer qu'on obtient

$$\int_{\partial T} \boldsymbol{a} \cdot n u = \int_T f \text{ pour tout T triangle de la triangulation.} \tag{6}$$

Q6 On peut introduire un décentrage en cherchant $u_h \in W_h$ tel que pour tout triangle T de la triangulation

$$\int_{\partial T} (\boldsymbol{a}\cdot n)^- \tilde{u}_h + \int_{\partial T} (\boldsymbol{a}\cdot n)^+ u_h = -\int_{\partial T\cap\Gamma_1} (\boldsymbol{a}\cdot n)^- g + \int_T f \tag{7}$$

où $b = b^- + b^+,\ b^- = \min(b,0)$, $\tilde{u}_h$ est la valeur sur le triangle de l'autre coté de l'arête ∂T si il existe ou 0 sinon. Donner l'équivalent différences finies en 1D lorsque la triangulation de $\Omega = (0,L)$ est uniforme.

Remarque : cette deuxième méthode est dite par "volumes finis".

Infoéa MM031 Examen final 2010 corrigé de O. Pironneau

Q0 $E(u+\lambda w) = \int_\Omega |a\nabla(u+\lambda w) - f|^2 = \int_\Omega |a\nabla u - f|^2 + 2\int_\Omega (a\nabla u - f)a\nabla w\lambda + \lambda^2\int_\Omega |a\nabla w|^2$

$\frac{1}{2\lambda}(E(u+\lambda w) - E(u)) = \int_\Omega (a\nabla u - f)(a\nabla w) + \frac{\lambda}{2}\int_\Omega |a\nabla w|^2 \geq 0$ si $\lambda > 0$

donc $\lambda \xrightarrow{\geq} 0 \Rightarrow \int_\Omega (a\nabla u - f)(a\nabla w) \geq 0 \quad \forall w \in V$

On change $w \to -w$ (ou $\lambda \to -\lambda$) et on trouve (3)

Q1 On intègre par partie (Green ou Stokes) (3) $\Rightarrow$

$$\int_\Omega -w\nabla\cdot(a(a\nabla u - f)) + \int_\Gamma w a\cdot n(a\nabla u - f) = 0 \quad \forall w \in V$$

$w \in \mathcal{D}(\Omega) \Rightarrow \nabla\cdot(a(a\nabla u - f)) = 0$ et $w \in H^1(\Omega) \cap V \Rightarrow (a\cdot n)(a\nabla u - f)|_{\Gamma_2} = 0$

Q2 $u = \sum u_i w^i$ et $w = w^j$ dans (3) donne

$$\sum_{i=1}^N u_i \int_\Omega a\nabla w^i a\nabla w^j = \int_\Omega f a\nabla w^j \quad \forall j \text{ tq } q^j \notin \Gamma_1$$

Si on utilise la penalité TGV la matrice est de taille $N\times N$ le nombre total de point de maillage et

$A_{ij} = \int_\Omega a\nabla w^i a\nabla w^j \quad F_j = \int_\Omega f a\nabla w^j$ sauf si $j = i$ et $q^j \in \Gamma_1$

$A_{ii} = 10^{30}$ $\quad F_j = 10^{30} g(q^j)$ si $i = j$ $q^j \in \Gamma_1$.

Q3 On peut prendre l'algorithme du gradient conjugué pour résoudre $AU = F$ mais on ne stocke pas A et

Au se calcule par $(Au)_j = \int_\Omega a\nabla u\, a\nabla w^j$ où $u = \sum u_i w^i + \sum g_i w^i$ $\quad q^i \notin \Gamma_1$, $q^i \in \Gamma_1$

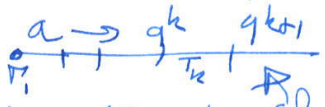
Q4 $a\otimes a$ n'est pas une matrice positive définie (ex $z \perp a \Rightarrow a\otimes a\, z = 0$) donc le GC peut buguer. si on rajoute εu ou bien $-\varepsilon\Delta u$

$\varepsilon u + a\nabla u = f$ $\quad -\varepsilon\Delta u + a\nabla u = f$

alors pour un maillage assez fin les matrices Elementaires sont positives définies

Q5 $w|_{T_k} = 1$ $w|_{T_j} = 0$ $j \neq k \Rightarrow \int_{T_k}(a\nabla u - f) = 0$ et par Green

$$\int_{\partial T_k} a\cdot n\, u = \int_{T_k} f$$



Q6 en 1D $\int_{\partial T_k} f = f(q^{k+1}) + f(q^k)$ donc (7) est (fig 1)

$$-a|_{T_{k-1}} u_k + a|_{T_k} u_{k+1} = a|_{T_{k-1}\cap(x=0)} g(0) + f_k(q^{k+1} - q^k)$$

Si a est constant $\Rightarrow \frac{u_{k+1} - u_k}{q^{k+1} - q^k} a|_{T_k} = f_k$ dans f en $k=0$

2 Partie pratique (F. Hecht) : 1 heure

On veut programmer la méthode de décentrage en dimension 2 défini par l'equation (7) en réutilisant les classes du projet définies dans `Mesh2d.hpp`. Pour cela nous avons besoin pour chaque triangle des au plus 3 triangles adjacents (de l'autre cote de l'arête). Nous utilisons la méthode GMRES (GC adapté aux matrices non symétriques) pour résoudre le système linéaire via `VirtualMatrice<double>` ce qui implique d'écrire le produit matrice vecteur. Dans le cours nous avons vu un algorithme pour trouver l'ensemble $\mathcal{K}_s$ des triangles ayant un sommet s en commun, que voici

```
 2 : #include <cmath>
 3 : #include <iostream>
 4 : #include <fstream>
 5 : #include <cassert>
 6 : #include "Mesh2d.hpp"
 7 :
 8 : void BuildListeTrianglesVertex(const Mesh2  Th, int *& head_s,int *& next_p)
 9 : {
10 :   const int end_list=1;
11 :   head_s = new int[Th.nv];
12 :   next_p = new int[Th.nt*3];
13 :   for (int i=0 ;i<Th.nv ;i++)
14 :     head_s[i] = end_list;
15 :   for (int k=0 ;k<Th.nt ;k++) // forall triangles
16 :     for(int j=0,j<3 ;j++)
17 :       {
18 :     int p = 3*k+j ,
19 :     int i = Th(k,j);
20 :     next_p[p]=head_s[i];
21 :     head_s[i]= p ;
22 :       }
23 : }
25 : #ifdef TEST
27 : int main(int argc,char ** argv)
28 : {
29 :   if (argc < 2) return 1;
30 :   Mesh2 Th(argv[1]);
31 :   int nbex = 3*Th.nt; // nombre maximal d'arete
32 :   int * head = 0, * next = 0;
33 :   BuildListeTrianglesVertex(Th,head,next);
34 :   cout << " s : les triangle contenant s et  numéro dans le triangle; ... " << endl ;
35 :   for (int s=0;s<Th.nv;s++)
36 :     {
37 :       cout <<  s << " : " ;
38 :       for(int k=head[s]; k >=0 ; k=next[k])
39 :         {
40 :       int it=k/3, iv=k%3;
41 :       assert(Th(it,iv)==s);
42 :       cout << it << "," << iv <<  ";  ";
43 :     }
44 :       cout << endl;
45 :     }
46 :   delete [] head;
47 :   delete [] next;
64 :   return 0;
65 : }
66 : #endif
```

Q7 Le programme contient quatre erreurs typographiques (caractère oublié ou changé) qui se trouvent dans la fonction `BuildListeTrianglesVertex`.
Corriger les trois erreurs de syntaxe à l'aide des messages d'erreurs du compilateur

```
brochet:EF2D hecht$ g++ -c  BLTV.cpp  -DTEST
BLTV.cpp: In function "void BuildListeTrianglesVertex(Mesh2, int*&, int*&)":
BLTV.cpp:16: error: expected initializer before "<" token
BLTV.cpp:16: error: expected ';' before ")" token
BLTV.cpp:19: error: expected unqualified-id before "int"
```

```
BLTV.cpp:20: error: name lookup of "i" changed for new ISO "for" scoping
BLTV.cpp:13: error:   using obsolete binding at "i"
Mesh2d.hpp: In function "int main(int, char**)":
Mesh2d.hpp:156: error: "Mesh2::Mesh2(const Mesh2&)" is private
BLTV.cpp:33: error: within this context
BLTV.cpp:33: error:   initializing argument 1 of
     "void BuildListeTrianglesVertex(Mesh2, int*&, int*&)"
```

```
 8 : void BuildListeTrianglesVertex(const Mesh2  & /*  manque & */  Th, int *& head_s,int *& next_p)
16 :     for(int j=0;  j<3;j++) /* , ->; */
18 :     int p = 3*k+j  ; /* , ->; */
```

Q8 Après ces trois corrections, l'exécution du programme de termine pas, trouver la raison et corriger le programme.

Effectivement la fin de liste `end_list` dans l'ensemble $[0, n[$ il faut par exemple écrire

```
10 :   const int end_list=- 1;/*  pas dans [0, 3*nt[ */
```

Q9 En utilisant le fait que l'ensemble des triangles ayant une arête (s_1, s_2) commune est exactement l'ensemble $\mathcal{K}_{s_1} \cap \mathcal{K}_{s_2}$, écrire une fonction qui construit un tableau `tadj` qui a le prototype suivant : `void BuidAdj(const Mesh2 & Th, int tadj[][3]);` où `tadj[k][i]` donne le numéro du triangle de l'autre côté de l'arête numéro $i \in \{0, 1, 2\}$ du triangle numéro k de `Th` s'il existe et donne -1 sinon.

Remarque la notion de couleur permet de définir simplement si un triangle est dans un ensemble ou non (même couleur), pour réinitailiser l'ensemble à l'ensemble vide, il suffit incrémenter la couleur (moins de 20 lignes de c++ à écrire).

Voila une correction :

```
void BuidAdj(const Mesh2 & Th, int tadj[][3])
{
  int color =0;
  KN<int> mark(Th.nt);
  int * head = 0, * next = 0;
  BuildListeTrianglesVertex(Th, head, next);
  mark= color;
  for (int k=0;k<Th.nt;++k)
      for (int e=0;e<3;++e)
       {
           color++;                                                    //    nouvelle color;
          int kt=0;
          tabj[k][e] = -1;
          for (int step=1;step<=2;++step)
           {
              int s = Th(k,(e+step)%3);          //    les sommets de l'arete oppose au sommet e
              for(int p=head[s]; p >=0;p=next[p])
               { int kk = p/3;
                 if(step==1)   mark[kk]=color;
                  else if(mark[kk]==color && kk!= k) tabj[k][e] = kk;
               }
           }
       }
    delete [] next;
    delete [] head;
}
```

Q10 Le champ $\boldsymbol{a}$ de la partie théorique sera considéré constant par triangle et il est défini par le tableau `R2 a[]` où $a[k]$ donne la valeur de $\boldsymbol{a}$ dans le triangle numéro k.

Ecrire la méthode `addMatMul` de la classe `MatVF` qui correspond à la matrice de l'équation (7) :

```
class MatVF:  public VirtualMatrice<R>
    {  public:
      const  Mesh & Th;
      const  R2 *a;                                                 //    le tableau a.
      int (*tadj)[3];                                        //    le tableau tadj[Th.nt][3]
      typedef    VirtualMatrice<R>::plusAx plusAx;
      MatVF(const  Mesh & T,R2 *aa,int (*ttadj)[3])
           : VirtualMatrice<R>(T.nt),  Th(T),a(aa),tadj(ttadj) {}
```

```
    void addMatMul(const  KN_<R>  & x, KN_<R> & Ax) const;
    plusAx operator*(const KN<R> &  x) const {return plusAx(this,x);}
  };
```

La solution :

```
void MatVF::addMatMul(const  KN_<R>  & x, KN_<R> & Ax) const
{
  for(int k=0;k<Th.nt;++k)
    {
        const Triangle & K=Th[k];
        for(int e=0;e<3;++e)
         {
           int  kk = tabj[k][e] >=0;
           R2 intn = K.Edge(e).perp();       //    normal interne de longueur la taille du cote
           double anl = - (intn,a[k]);
           if(an >0 )  Ax[k] += anl*x[k];
           else if  (kk>=0) Ax[k] += anl*x[kk];
         }
}
```

Q11 Les fonctions f et g seront considérées comme des fonctions éléments finis P_1 Lagrange classique et elles sont définies par les tableau respectif `f` et `g` des valeurs aux sommets du maillage.

Ecrire une fonction qui résout le système linéaire de l'équation (7) qui a cinq paramètres : le maillage `Th`, les trois tableaux `a` , `f`,`g`, et pour finir le tableau `uh` qui correspond à la solution calculée de type `KN<double>`.

Rappel : voila un exemple d'appel de la fonction GMRES défini dans le fichier `"gmres.hpp"` :

```
double eps=1e-10;
int nbkylov=50, nitermax=Th.nt;
KNM<R> H(nbkylov+1,nbkylov+1);
int res=GMRES(A,uh,b,MatriceIdentite<R>(Th.nt),H,nbkylov,nitermax,eps);
cout << " GMRES: nb iter : " << nitermax << "   residu : "
     << eps << " convergence : " << (res == 1) << endl;
```

`A` est la matrice de type `MatVF`, `uh` le tableau solution et `b` est le tableau définissant second le membre de type `KN<double>`

Une solution :

```
int SolveVF(Mesh & Th, R2 * a, double *f, double *g, KN<double> & uh)
{
  int (*tadj)[3] = new int[Th.nt][3];
  BuidAdj( Th, tadj);
  MatVF A(Th,a,tadj);
  KN<double> b(Th.nt);
  for(int k=0;k<Th.nt;++k)                                  //    assemblage du second membre
    {
       const Triangle & K=Th[k];
       b[k]=0;
       double ck = K.area/3.;
       for(int s=0;s<3;s++)
         b[k] += ck*f[th(k,s)];                  //    add ∫_K f= |K| ( f[i1]+f[i2]+f[i3] )/3.
       for(int e=0;e<3;++e)
        if( tabj[k][e]<0)                                                  //   arete du bord
          {
             R2 intn = K.Edge(e).perp();   //    normal interne de longueur la taille du cote
             double anl = - (intn,a[k]);
             b[k] += min(0.,anl)*(g(Th(k,(e+1)%3))+g(Th(k,(e+2)%3)))/2;
          }
    }
  double eps=1e-10;
  nt nbkylov=50, nitermax=Th.nt;
  KNM<R> H(nbkylov+1,nbkylov+1);
  int res=GMRES(A,uh,b,MatriceIdentite<R>(Th.nt),H,nbkylov,nitermax,eps);
  cout << " GMRES: nb iter : " << nitermax << "   residu : "
       << eps << " convergence : " << (res == 1) << endl;
  delete [] tadj;
  return res;   }
```