Partiel MM031, Master 1
Master Mathématique; UPMC
Les documents sont autorisés

F. Hecht/ I. Danaila

Vendredi 4 Mars 2011

1 Partie théorique (1 heure)

Nous nous proposons d'écrire une méthode numérique pour résoudre le problème non-linéaire suivant :
Trouver u une fonction de $]a, b[\mapsto \mathbb{R}$, avec $0 < a < b$, qui vérifie les conditions aux limites (CL)

$$u(a) = \alpha, \qquad u(b) = \beta, \quad \text{(avec } \alpha \text{ et } \beta \text{ des valeurs données)},$$

et qui réalise **le minimum** de la fonctionnelle J suivante :

$$J(u) = 2\pi \int_a^b r\,\sqrt{1 + (u'(r))^2}\,dr. \tag{P}$$

On supposera que ce problème est bien posé dans $H^1(]a, b[)$, c'est à dire, il existe une unique solution u et qui ne dépend continûment des 4 données a, b, α, β.

Q0 Montrer que J est la surface (de révolution) définie (balayée) par la rotation de la courbe $z = u(r)$, pour $a < r < b$, autour de l'axe z, où (r, z) sont les coordonnées polaires, avec $r = \sqrt{x^2 + y^2}$.

Q1 Calculer la différentielle de Fréchet $DJ(u)$ de J en u ; $DJ(u)$ est une application linéaire de $H^1_0(]a, b[)$ dans $\mathbb{R}$.

Q2 Justifier pourquoi u est solution du problème suivant :

$$\forall v \in H^1_0((]a, b[), \quad 2\pi \int_a^b r \frac{u'v'}{\sqrt{1 + (u'(r))^2}} dr = 0$$

Q3 Montrer que la caténoïde $u(r) = acosh(r)$ (i.e. $cosh(u) = r$) est la solution du problème précédent pour $a = 1$, $\alpha = 0$, et $\beta = cosh(b)$; on rappelle que $acosh'(r) = 1/\sqrt{1 - r^2}$.

Q4 On veut discrétiser le problème (P) avec des éléments finis P_1 Lagrange : soit N un entier strictement positif, on définit $h = (b - a)/N$, $q_i = a + hi$, pour $i = 0, .., N$, les points du maillage et l'espace V_N des fonctions de $\mathcal{C}^0(]a, b[)$ affines par morceaux sur $]q_i, q_{i+1}[$ pour $i = 0, .., N - 1$.
Soit w_i la base de V_N telle que $w_i(q_j) = \delta_{ij}$, où δ_{ij} est le symbole de Kroneker.
Soit une fonction v_N définie par ses valeurs aux nœuds du maillage

$$v_N = \sum_{i=0}^{N} v_i w_i, \quad \text{avec} \quad (v_i)_{i=0}^N \in \mathbb{R}^{N+1}.$$

Ecrire un algorithme formel qui calcule la fonctionnelle $J(v_N)$ en fonction des valeurs v_i.

Q5 Ecrire une base de l'espace vectoriel $V_N \cap H^1_0(]a, b[)$.

Q6 On veut utiliser l'algorithme de minimisation à pas fixe qui s'écrit comme suit :
Soit u^0_N un élément de V_N donné, vérifiant les conditions aux limites (CL) ;
on calcule récursivement

$$u^{n+1}_N = u^n_N - \rho \nabla J(u^n_N),$$

où ρ est une constante définie par l'utilisateur, et le gradient $\nabla J(u)$ est défini à partir d'un produit scalaire $(,)_N$ de V_N par :

$$(\nabla J(u), v)_N = DJ(u)(v).$$

Pour simplifier, on prendra comme produit scalaire

$$(u,v)_N = \sum_{i=0}^{N} u_i v_i, \quad \text{où} \quad v = \sum_{i=0}^{N} v_i w_i, \quad \text{et} \quad u = \sum_{i=0}^{N} u_i w_i.$$

Evaluer les composantes g_i^n du gradient, définies par

$$\nabla J(u_N^n) = \sum_{i=0}^{N} g_i^n w_i,$$

comparer les par rapport $\partial J(u_N^n)/\partial u_i$, avec $u_N^n = \sum_{i=0}^{N} u_i w_i$. et écrire complètement l'algorithme composante par composante.

2 Partie pratique C++ (1 heure)

Le but de cette partie est de programmer l'algorithme précédent.

On supposera que cette définition est active : `typedef double R;`

Q7 Ecrire une fonction `C++` de prototype `AireTroncDeCone(R ri, R ri1, R vi, R vi1) ;` qui calcule l'aire d'un tronc de cône $c(r)$ défini pour $\text{ri} < r < \text{ri1}$; $c(r)$ la fonction affine telle que $c(\text{ri}) = \text{vi}$ et $c(\text{ri1}) = \text{vi1}$.

Q8 Ecrire une fonction `C++` de prototype `R J(R a, R b, int N, R *v) ;` qui évalue J pour la fonction $v = \sum_{i=0}^{N} v_i w_i$, représentée par le vecteur `v` de ses valeurs.

Q9 Ecrire une fonction `C++` de prototype `R * GJ(R a, R b,int N,R *v,R *g) ;` qui évalue ∇J pour la fonction $v = \sum_{i=0}^{N} v_i w_i$ et qui retourne le pointeur g qui est un argument qui va stocker le vecteur ∇J.

Q10 Supposons que l'algorithme de la question Q6 est programmé avc la fonction suivante :

```
bool AlgoQ6(R a,R b,R alpha, R beta, VN & un,R eps);
{
   int N= un.N;
   VN gn(N),
   un(0) = alpha;
   un(N) = beta;
   for(int n=0;n<100;++n)
     {
        un -= rho*GJ(a,b,N,un,gn);
        if(gn.norme_infini() <eps) return true;
     }
     else return false;
}
```

Décrire une classe tableau `VN` ainsi que les prototypes des méthodes et des fonctions associées qui seront nécessaires pour la fonction `AlgoQ6` fonctionne.

Rappel : les règles de conversion (« cast » en Anglais) d'un type T en A sont générées, soit à partir d'un constructeur `A(T)` dans la classe A, soit en utilisant l'opérateur de conversion `operator (A) ()` dans la classe T (ou `operator (A) (T)` hors de la classe).

Q11 Quel est le code (opérateurs, constructeurs,...) qu'il faut ajouter dans la classe `VN` pour éliminer la possibilité de perte de mémoire (toutes les allocations faites avec un `new` ont un `delete` correspondant) ?

Q12 Ecrire une programme qui teste et valide votre algorithme numérique.