TD 5

J.Ph. Chancelier*

15 décembre 2003

Table des matières

1 Recherche de zéros

On cherche ici à trouver numériquement $u^\star \in \mathbb{R}^d$ tel que $\Phi(u^\star) = \alpha$ où Φ est une fonction strictement croissante donnée et α un réel donné. On suppose par ailleurs que la fonction Φ est donnée par :

$$\Phi(u) = \mathbb{E}\left[f(u, \xi)\right]$$

où ξ est une variable aléatoire que l'on sait simuler et f une fonction que l'on sait évaluer. Un exemple typique est $f(u, \xi) = \Phi(u) + \xi$. Pour u donné on ne connaît $\Phi(u)$ qu'a travers une mesure $f(u, \xi)$ qui est entachée d'une erreur de mesure ξ de loi $\mathcal{N}(0, 1)$.

Un algorithme stochastique pour trouver un zéro de $\Phi(u) - \alpha$ sera le suivant. On se donne U_0 et une suite de variables aléatoire $\xi_1, \ldots, \xi_n$ indépendantes et de même loi que ξ. On met à jour U_n l'estimation de $u^\star$ par :

$$U_{n+1} = U_n - \gamma_n(f(U_n, \xi_n) - \alpha)$$

où γ_n est une suite déterministe décroissant vers 0 et telle que :

$$\sum \gamma_n = \infty \quad \text{et} \quad \sum \gamma_n^2 < \infty$$

La suite U_n converge alors p.s (voir le cours) vers $u^\star$ si l'on suppose par exemple :

*Cermics, École Nationale des Ponts et Chaussées, 6 et 8 avenue Blaise Pascal, 77455, Marne la Vallée, Cedex 2

– $f(u, \xi)$ est bornée.
– $\Phi(u).(u - u^\star) \geq c\|u - u^\star\|^2$, $c > 0$

Écrire une fonction Scilab qui mets en oeuvre cet algorithme. La fonction dont on cherche à calculer le zéro sera passée en argument. Faire tourner l'algorithme sur une fonction de votre choix ($d = 1$, ou 2, ...).

```
function y=dosage(x)
  // on cherche un zéro de dosage(x) + un bruit gaussien
  y= exp(x)
  y= y + rand(1,1,'n');
endfunction

function x=rob_monro(x0,f,a,n,stop)
  x= 0*ones(1,n);
  x(1)=x0;
  for i=2:n
    x(i) = x(i-1) - (1/n)*( f(x(i-1)) -a  )
    if (abs(x(i)-x(i-1))) < stop*abs(x(i)) then
      x(i+1:$)=[];
      return
    end
  end
endfunction

n= 200000;
x=rob_monro(0,dosage,7,n,1.e-8) ;
nr=size(x,'*')
plot2d((1:nr)',x')
plot2d((1:nr)',log(7)*ones(nr,1))
```

2 Un calcul de quantiles

On regarde dans cette section le cas particulier où on cherche à évaluer un quantile. Soit $\Phi(u)$ la fonction de répartition d'une loi $\mathcal{L}$. On suppose ici $\Phi(u)$ continue. Calculer un quantile d'ordre α de $\mathcal{L}$ consiste à calculer u_α vérifiant :

$$\mathbb{P}\left[X \leq u_\alpha\right] = \alpha$$

où X est une variable aléatoire de loi $\mathcal{L}$. Soit $f(u, x) = \mathbb{I}_{\{x \leq u\}}$ cela s'écrit encore :

$$\mathbb{E}\left[f(u_\alpha, X)\right] = \alpha$$

On peut donc utiliser l'algorithme décrit dans la section précédente. une autre méthode est aussi envisageable. Soient $X_1, \ldots, X_n$ n tirages indépendants de loi $\mathcal{L}$. On peut les réordonner en ordre croissant $Y_1, \ldots, Y_n$ et estimer le quantile d'ordre α par Y_j avec

$$j/n \leq \alpha \leq (j+1)/n$$

Cela revient à calculer le quantile d'ordre α de la fonction de répartition empirique :

$$\Phi_n(u) = \frac{1}{n}\sum_{k=1}^{n} \mathbb{I}_{Y_k \leq u}$$

Mettre en oeuvre les deux algorithmes par exemple pour la loi normale ou pour d'autres loi de votre choix (`grand`). Pour l'algorithme sur la fonction de répartition empirique on pourra utiliser `dsearch` pour rajouter un élément dans un tableau déjà trié.

Une estimation d'un quantile dans scilab On estime le quantile a partir de la distribution empirique a l'etape k xk contiendra le vecteur des xi tries auquel on rajoute -l'utilisation de `dsearch`.

```
alpha = 0.5
xk=[-%inf,+%inf];

n=10000;
q=0*ones(1,n);

// En utilisant le quantile empirique

for i=1:n
  xn = grand(1,1,'nor',0,1);
  in=dsearch(xn,xk);
  xk=[xk(1:in),xn,xk(in+1:i+1)];
  // estimation du quantile
  j=max(0,floor(alpha*i));
  q(i)= xk(1+j); // On rajoute 1 a cause du -%inf
end

plot2d((1:n)',q);

// En utilisant un algorithme stochastique
// que l'on fait tourner par block
n=10000;
qrm(1)=1;
p= 0

function [qrm]=quantile_os(qrm,n,nb)
 ns=1
 while %t
  qrm(1)=qrm($) ;
  for i=2:n
    xn = grand(1,1,'nor',0,1);
    qrm(i)= qrm(i-1) - (1/(i+p))* ( sign(max(0,qrm(i-1) -xn)) -alpha );
  end
  xbasc();
  plot2d((1:n)',qrm);
```

```
  xclick();
  ns = ns+1;
  p= p+n;
  if ns == nb then return ; end
 end
endfunction

// on itere 10 fois par block de 1000 points avec
// un graphique tous les 1000 points
qrm=quantile_os(qrm,1000,10);
```

Faire une animation graphique montrant l'évolution des deux algorithme aux cours des itérations. Pour certaines lois donc la loi normale on peut calculer les quantiles dans Scilab (`cdfnor`). On pourra utiliser cela pour superpose sur les courbes la solution u_α.

```
driver('X11')
xset('pixmap',1) // on utilise un mémoire graphique pour dessiner
for i=1:n
   .......
   xset('wwpc')  // efface la mémoire graphique
   // commande graphiques à introduire ici
   xset('wshow') // affiche le contenu de la mémoire graphique à l'écran.
end
```

3 Recherche d'un maximum (Kiefer-Wolfowitz)

On suppose cette fois que l'on veut maximiser une fonction concave $\Phi(u) = \mathbb{E}[f(u,\xi)]$. On pourrait utiliser des méthodes d'optimisation déjà vue, par exemple une méthode de gradient sur la fonction Φu mais cela demanderait à chaque itération l'évaluation d'une espérance $\mathbb{E}\left[\frac{\partial}{\partial_u} f(u,\xi)\right]$. On va donc plutôt utiliser un algorithme stochastique pour trouver un zéro de $\mathbb{E}\left[\frac{\partial}{\partial_u} f(u,\xi)\right]$ et plutôt que de calculer explicitement le gradient on va l'évaluer par :

$$\frac{1}{2}(\Phi(u+c) - \Phi(u-c))$$

et c va tendre vers 0 au cours de l'algorithme itératif. Soient $\xi_1^1, \xi_1^2, \ldots, \xi_n^1, \xi_n^2$ une suite de variables i.i.d de même loi que ξ la remise à jour de U_n se fait par :

$$U_{n+1} = U_n + (\gamma_n/c_n)(f(U_n + c_n, \xi_n^1) - f(U_n - c_n, \xi_n^2))$$

où γ_n est une suite déterministe décroissant vers 0 et telle que :

$$\sum \gamma_n = \infty \quad , \sum \gamma_n c_n < \infty \quad \text{et} \quad \sum (\gamma_n/c_n)^2 < \infty$$

U_n converge alors p.s vers $u^\star$ pour Φ de classe $\mathcal{C}^2$ strictement concave et telle que $|\Phi''(u)| \leq K(1+|u|)$ et $\mathbb{E}\left[f^2(u,\xi)\right] \leq K(1+u^2)$.

Mettre en oeuvre cet algorithme dans Scilab.

```
function z=f(x,y)
  z= - x^2 + y
endfunction

function z=df(x,y)
  z= -2*x
endfunction

function x=kiwolf(x0,f,n)
  x= 0*ones(1,n);
  x(1)=x0;
  alpha=1/4;
  for i=2:n
    eps1= grand(1,1,'nor',0,1)
    eps2= grand(1,1,'nor',0,1)
    gn=(1/n)
    cn=n^(-alpha)
    x(i) = x(i-1) + (gn/cn)*(f(x(i-1)+cn,eps1) - f(x(i-1)-cn,eps2))
  end
endfunction

function x=gstoch(x0,f,n)
  x= 0*ones(1,n);
  x(1)=x0;
  alpha=1/4;
  for i=2:n
    eps1= grand(1,1,'nor',0,1)
    gn=(1/n)
    cn=n^(-alpha)
    x(i) = x(i-1) + (gn/cn)*( df(x(i-1),eps1))
  end
endfunction

n= 2000;
x= kiwolf(1,f,n) ;
y= gstoch(1,f,n) ;

plot2d((1:n)',x')
```

4 Simulation du Brownien

On se propose d'implémenter en Scilab un simulateur de trajectoire d'un mouvement brownien sur $\mathbb{R}$. On pourra se reporter sur les pages du cours cours (p39) pour retrouver la méthodologie proposée. Il est proposé dans le polycopié une implémentation récursive par contre dans Scilab on utilisera une implémentation itérative.

– On dispose à l'étape `k` de l'algorithme de la valeur du Brownien que l'on a simulé aux

date `tk`.
- A l'etape `k1+` on decoupe chaque intervalle de temps en 2 et on calcule pour ces nouveaux instants la valeur du brownien en utilisant la formule (2.1) page 40.
- On initialise l'algorithme avec la valeurs du Brownien aux dates 0 et 1
- Au cours des itérations on dessinera sur une fenetre la trajectoire du Brownien et sur une autre l'histogramme des acroissements du Brownien.
- On cherchera aussi à verifier numériquement la Proposition 2.4.1.

```
// construction du brownien sur [0,1]

vk=[ 0, grand(1,1,'nor',0,1)];
tk=[ 0, 1];
dk= 1;
nk = 1;
// On rajoute des points entre chaque points de tk
n=20;
eq=0*ones(1,n);

for i=1:10 //
  nkp1 = 2*nk;
  dkp1 = dk/2;
  vkp1 = ones(1,nkp1+1);
  tkp1 = ones(1,nkp1+1);
  vkp1(1:2:$) = vk;
  vkp1(2:2:$) = ( vk(1:$-1)+vk(2:$))/2 + grand(1,nk,'nor',0,1)*(1/2)*sqrt(dk);
  eqk= vkp1(2:$)-vkp1(1:$-1); eq(i) = norm(eqk)^2;
  tkp1(1:2:$) = tk;
  tkp1(2:2:$) = ( tk(1:$-1)+tk(2:$))/2 ;
  xbasc();
  plot2d(tk,vk)
  xclick();
  nk=nkp1;
  tk=tkp1;
  dk=dkp1;
  vk=vkp1;
  tk=tkp1;
end
```