Résolution d'un problème d'optimisation différentiable

Marc Bourqui Victor Constantin Ian Schori
Floriant Simond

January 3, 2013

Énoncé du problème

Trouver (une approximation de) la solution du problème suivant en appliquant le théorème de la plus forte pente:

$$\min_{x \in \mathbb{R}^2} (x_1 - 2)^4 + (x_1 - 2)^2 x_2^2 + (x_2 + 1)^2 \tag{1}$$

Réponses aux questions

(a) Implémenter la méthode de plus forte pente (Algorithme 11.3) à l'aide du logiciel MATLAB. Déterminer la taille du pas en appliquant la recherche linéaire, Algorithme 11.2 (les deux conditions de Wolfe).

Listing 1: pfp.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                          %
% Methodes de descente pour l'optimisation non lineaire    %
% sans contraintes                                         %
%                                                          %
% BOURQUI Marc                                             %
% CONSTANTIN Victor                                        %
% SCHORI Ian                                               %
% SIMOND Floriant                                          %
%                                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function x = pfp(f, x0, alpha, useRL)


useRLInner = useRL;

%%%%%%%%%%%%%%%%
% Interface    %
%%%%%%%%%%%%%%%%

% nom de la fonction a minimiser, qui est specifiee dans le fichier 'f.m'
% et qui est declaree sous forme de string
fct = f;

% point initial
x = x0;

%%%%%%%%%%%%%%%%%%
% Parametres     %
%%%%%%%%%%%%%%%%%%

% pour le critere d'arret
epsilon = 0.001 ;
```

```
34 maxIter = 200    ;
35
36 % initialisation du nombre d'iterations
37 i=1 ;
38
39 % initialisation de la matrice qui stocke tous les iter[U+FFFD]s
40 % un iter[U+FFFD]= une colonne de cette matrice
41
42 stock(:,i) = x0;
43
44 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
45 % Boucle principale                  %
46 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
47
48
49 % Critere d'arret: x a ateint la precision demand[U+FFFD]OU nb ...
       iterations max ateint
50
51 while ( normGradient(fct,stock(:,i)) >= epsilon ) && ( i < ...
       maxIter )
52     % mise a jour du nombre d'iterations
53     i = i+1;
54
55     % calcul et stockage de la valeur du nouveau x
56     stock(:,i) = pfpInnerLoop(fct, stock(:,i-1), useRLInner);
57
58 end
59
60 % Calcul de la taille de la matrice contenant tous les x
61 taille = size(stock,2);
62
63 % Evaluation de la fonction en chaque point
64 for i=1:taille
65     valeurstock(i)=feval(fct,stock(:,i));
66 end
67
68
69 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
70 % Affichage des r[U+FFFD]sultats %
71 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
72
73 disp('Valeur de la suite des x :') ;
74 stock'
75
76 disp('****************************************')
77 disp(['Nombre d''iterations :                    ' ...
       num2str(i-1)])
78 disp(['Valeur de la fonction a l''optimum : ' ...
       num2str(feval(fct,stock(:,i)))] ) ;
79 disp('Valeur de l''optimum : ')
80 xOptim = stock(:,i)'
81 disp('****************************************')
82
```

```
83 % passage au module de visualisation de la fonction et des ...
       resultats
84
85 visual3d(fct, stock, valeurstock);
86
87 sprintf('Nombre de fois que la boucle a ete parcourue : ...
       %d',i)
88
89 clear;
90 end
```

Listing 2: pfpInnerLoop.m

```
1  function x = pfpInnerLoop(f, x0, useRL)
2      x = x0;
3      alpha = 1;
4
5      [fx, gfx] = feval(f, x);
6      d = -gfx;
7
8      if useRL
9          beta1 = 0.5;
10         beta2 = 0.75;
11         lambda = 2;
12         alpha = rl(f, x, fx, gfx, alpha, beta1, beta2, ...
               lambda);
13     else
14         %Soit on peut utiliser la fonction dans b) pour ...
               calculer le pas
15         alpha = tp(f,x);
16     end
17     x = x + alpha * d;
18 end
```

Listing 3: rl.m

```
1  function alpha = rl(f, x, fx, gfx, alpha0, beta1, beta2, ...
       lambda)
2      alpha = alpha0;
3      alphal = 0;
4      alphar = inf;
5
6      [fxad, fgxad] = feval(f, x + alpha * -gfx);
7
8      while (fxad > fx + alpha * beta1 * gfx' * -gfx) || ...
           (fgxad' * -gfx < beta2 * gfx' * -gfx)
9          if fxad > fx + alpha * beta1 * gfx' * -gfx
10             alphar = alpha;
11             alpha = (alphal + alphar)/2;
```

```
12          elseif fgxad' * -gfx < beta2 * gfx' * -gfx
13              alphal = alpha;
14              if alphar < inf
15                  alpha = (alphal + alphar)/2;
16              else
17                  alpha = lambda * alpha;
18              end
19          end
20
21          [fxad, fgxad] = feval(f, x + alpha * -gfx);
22      end
23  end
```

(b) Implémenter une fonction qui donne la taille du pas suivant:

$$\alpha_k = \frac{\bigtriangledown f(x_k)^T \bigtriangledown f(x_k)}{\bigtriangledown f(x_k)^T \bigtriangledown^2 f(x_k) \bigtriangledown f(x_k)} \tag{2}$$

Quelle est la nature de ce pas? D'où cette formule vient-elle?

(c) Comparer le comportement de l'algorithme en utilisant les pas (a) et (b).

(d) Comparer la methode de plus forte pente et la methode quasi-Newton (qui est déjà implementée – Série 3).