Master Maths 1 - Optimisation

TP5 : Méthode de newton

1 Quelques explications

Nous avons vu en cours qu'une condition nécessaire d'optimalité est :

$$\nabla f(x^*) = 0$$

Une autre idée pour minimiser une fonction f serait donc de trouver un zéro de son gradient, ce qui suggère d'utiliser la méthode de Newton sur le gradient de f. Par Taylor on peut écrire :

$$0 = \nabla f(x^*) = \nabla f(x^{(k)}) + Hf(x^{(k)})(x^* - x^{(k)}) + o(x^* - x^{(k)})$$

et si $x^{(k)}$ est assez près de l'optimum x^* alors on doit pouvoir négliger le reste. Par contre même dans ce cas on ne va pas obtenir exactement x^* mais une approximation notée $x^{(k+1)}$, ce qui nous donne l'itération de Newton :

$$x^{(k+1)} = x^{(k)} - Hf(x^{(k)})^{-1}\nabla f(x^{(k)})$$

pour laquelle la matrice Hessienne $Hf(x^{(k)})$ doit être inversible. On peut ré-interpréter cette méthode différemment. Pour cela poser $d^k = -Hf(x^{(k)})^{-1}\nabla f(x^{(k)})$ et :

Question 1 : montrer que sous la condition $Hf(x^{(k)})$ définie positive d^k est une direction de descente pour f en $x^{(k)}$.

Cette condition sera au moins vérifiée près d'un minimum local "régulier" (on entend par là que $\nabla f(x^*) = 0$ et $Hf(x^*)$ définie positive). Ceci suggère d'utiliser la stratégie suivante :

1. étant donné $x^{(k)}$, on calcule $g^k = \nabla f(x^{(k)})$ et $H_k = H_f(x^{(k)})$;
2. on calcule d^k par résolution du système linéaire $H_k d^k = -g^k$
3. si cette résolution échoue ou si d^k n'est pas une direction de descente, on fait une itération de gradient à pas optimal qui permet d'obtenir un $x^{(k+1)}$
4. sinon, on tente l'itération $x^{new} = x^{(k)} + \tau d^k$ avec au départ $\tau = 1$ (le pas de Newton), et si $f(x^{new}) \geq f(x^{(k)})$ on recommence avec un pas plus petit, par exemple $\tau \leftarrow \tau/2$ et ceci jusqu'à ce que la condition de décroissance sur f soit satisfaite ; lorsque c'est le cas, on pose $x^{(k+1)} = x^{new}$
5. aller en 1

Question 2 : comment vérifie-t-on que d^k est bien une direction de descente ?

Question 3 : écrire cet algorithme en pseudo code en rajoutant le test d'arrêt usuel sur le gradient. On admettra qu'on dispose d'une résolution de système linéaire $Ax = b$ qui "n'échoue pas". Dans les logiciels comme matlab, octave, scilab, c'est le cas car si la matrice n'est pas inversible une solution en un "certain sens" est calculée. Dans ces langages cela s'écrit : `x = A\b`.

Question 4 : coder cet algorithme en langage scilab par une fonction d'entête :

```
function [x0, f0, g0, iter] = newton(x0, f, grdf, pas, tol, itermax, verb)
// x0 : point de départ
// f : fonction à optimiser
// grdf : gradient de cette fonction
// pas : le pas à utiliser au début de la recherche unidimensionnelle
//       pour le cas ou on fait une iteration de gradient
// tol : la tolérance : on sort lorsque || grdf(x) || <= tol
// itermax : le nb max d'itérations
// verb : booléen (si vrai alors on affiche la valeur de f, de son
//        gradient et du x courant à la fin de chaque itération)
```

Plutôt que de calculer pour chaque fonction à minimiser le Hessien de manière exacte, on utilisera la fonction `derivative` de scilab à partir du gradient (dériver "numériquement" le gradient). On pourra éventuellement jouer sur le paramètre `order` de cette fonction (méthode d'ordre 1, 2 ou 4).

2 Tests

Tester cette méthode et la comparer au gradient conjugué sur les 2 exemples vu précédemment et sur deux autres exemples de fonctions dont le code scilab est disponible sur ma page web :

- la fonction `helical` de $\mathbb{R}^3$ dans $\mathbb{R}$; prendre `x0 = [-1;0;0]` comme point de départ (la solution est $xopt = [1; 0; 0]$) ;
- la fonction `expsquares` permet de définir une fonction de $\mathbb{R}^n$ dans $\mathbb{R}$ avec n "variable" ; la solution du problème de minimisation s'obtient avec la fonction `[fopt,xopt] = sol_expsquares(n)` également fournie. Etant donné n (prendre $n = 5, 10, 20$ par exemple) vous pouvez prendre un point de départ dont toutes les composantes sont égales à un.