Chapitre V

METHODES ITERATIVES d'OPTIMISATION SANS CONTRAINTES

Ce chapitre introduit une classe importante d'algorithmes de résolution des problèmes d'optimisation sans contrainte. Le concept central est celui de la direction de descente.

Après avoir décrit le fonctionnement d'un algorithme a directions de descente, nous donnons quelques exemples d'algorithmes de ce type, nous énonçons des critères qui permettent d'estimer la qualité de la direction de descente proche d'une solution : celui de l'admissibilité, du pas unité et celui de la convergence super-linéaire.

V.1. Notion d'algorithme

Considérons le problème consistant à minimiser une fonction *f(x)* où x de R^n, un algorithme itératif permettant de résoudre ce problème est un processus itératif générant une suite de vecteur $(x_0,x_1,...,x_n)$ de R^n en fonction d'une séquence d'instructions et d'une condition d'arrêt, c à d qu'a partir d'un point initial x_0, on engendre (construit) une suite (x_n) qui converge vers la solution du problème

$$(P):\left\{\min\ f(x),\ \ x\in R^n\right\}$$

Un algorithme est définie par l'application *h* de R^n dans R^n, qui à chaque point x_k fait correspondant à $x_{k+1}=h(x_k)$. L'étude de la convergence revient à l'étude des propriétés de l'application h.

V.2.Modes de Convergence

Nous dirons qu'un algorithme décrit par une application *h* est globalement convergeant si : quelque soit le point x_0 choisi, la suite (x_k) engendrée par : $x_{k+1}=h(x_k)$ converge vers un point $\hat{x}$ satisfaisant les conditions d'optimalité $(\nabla f(\hat{x})=0\ \ et\ \ H(\hat{x})\ \ est\ \ s.d.p)$. On a plusieurs modes de convergence :

a. La suite (x_k) converge vers $\hat{x}$ linéairement avec un taux $\alpha<1$ si : $\lim\limits_{k\to+\infty}\sup\dfrac{\|x_{k+1}-\hat{x}\|}{\|x_k-\hat{x}\|}=\alpha$

b. La convergence est dite asymptotique si $\|x_{K+1}-\hat{x}\|\cong\alpha\|x_k-\hat{x}\|$

c. La convergence est dite super-linéaire si : $\lim\limits_{k\to+\infty}\dfrac{\|x_{k+1}-\hat{x}\|}{\|x_k-\hat{x}\|}=0$.

d. La convergence est dite super-linéaire d'ordre $\gamma>1$ si : $\lim\limits_{k\to+\infty}\dfrac{\|x_{k+1}-\hat{x}\|}{\|x_k-\hat{x}\|^{\gamma}}<+\infty$.

Si $\gamma=2$ la convergence de l'algorithme est dite quadratique

V.3. Schémas général des algorithmes d'optimalité

Il convient de souligne que la plupart des algorithmes d'optimisation, avec contrainte ou non, fonctionnent selon un schéma général consistant, à chaque itération, à se rapprocher du minimum par la résolution d'un sous problème de minimisation.

Les méthodes (ou les algorithmes) itératives d'optimalité fait partie d'une classe plus grande de méthodes numériques appelées *méthodes de descente.*

Un algorithme à directions de descente est un algorithme d'optimisation différentiable, destiné à minimiser une fonction réelle différentiable définie sur un espace euclidien (par exemple R^n, muni d'un produit scalaire) ou, plus généralement, sur un espace hilbertien. L'algorithme est itératif et procède donc par améliorations successives. Au point courant, un déplacement est effectué le long d'une direction de descente, de manière à faire décroître la fonction.

Principe de cette méthode : On veut minimiser une fonction *f*. Pour cela on se donne un point de départ arbitraire x_0, pour construire l'itéré suivant x_1 il faut penser qu'on veut se rapprocher du min de *f*, on veut donc que $f(x_1) < f(x_0)$. On cherche alors x_1 sous la forme : $x_1 = x_0 + \rho_0 d_0$ où d_0 est un vecteur non nul de R^n et ρ_0 un réel strictement positif, donc on cherche ρ_0 *et* d_0 pour que $f(x_0 + \rho_0 d_0) < f(x_0)$. On ne peut pas toujours trouver d_0. Quand d_0 existe on dit que c'est une direction de descente et ρ_0 est le pas de descente, cette opération de détermination du pas s'appelle la recherche linéaire. La direction et le pas de descente peuvent être fixes ou changer à chaque itération. Le schéma général d'une méthode de descente est le suivant :

$$\begin{cases} x_0 \in R^n \ \ donné \\ x_{k+1} = x_k + \rho_k d_k, \ \ d_k \in R^n / \{0\}, \ \ \rho_k \in R_+^* \end{cases}$$

où ρ_k *et* d_k sont choisis de telle sorte que : $f(x_k + \rho_k d_k) < f(x_k)$.

Une idée pour trouver une direction descente est de faire un développement de Taylor à l'ordre 2 de la fonction *f* entre deux itérés x_k *et* $x_{k+1} = x_k + \rho_k d_k$

$$f(x_k + \rho_k d_k) = f(x_k) + \rho_k (\nabla f(x_k), d_k) + O(\rho_k d_k)$$

Comme on veut $f(x_k + \rho_k d_k) < f(x_k)$, on peut choisir, par exemple :

1- $d_k = -\nabla f(x_k)$: la méthode obtenue s'appelle L'**algorithme du Gradient.**

2- $d_k = -[H(x_k)]^{-1} \nabla f(x_k)$: la méthode obtenue s'appelle L'**algorithme de Newton**

V. 4. Méthode de Gradient

L'algorithme du gradient désigne un algorithme d'optimisation différentiable. L'algorithme est itératif et procède donc par améliorations successives. Au point courant, un déplacement est effectué dans la direction opposée au gradient, de manière à faire décroître la fonction. Cette description montre que l'algorithme fait partie de la famille des algorithmes à directions de descente.

L'algorithme du gradient est également connu sous le nom d'algorithme de la plus forte pente ou de la plus profonde descente (steepest descent, en anglais) parce que le gradient est la pente de la fonction linéarisée au point courant et est donc, localement, sa plus forte pente (une notion qui dépend du produit scalaire).

La direction de descente choisis sera à chaque itération

$$d_k = -\nabla f(x_k),$$

les points sont ainsi successivement générés par cette méthode de la manière suivante :

$$\begin{cases} x_0 \in R^n \ \ donné \\ x_{k+1} = x_k - \rho_k \nabla f(x_k), \ \ \rho_k > 0 \end{cases}$$

L'algorithme de Gradient est donné pas :

1. **Initialisation** K=0

   choix de x_0 , $\rho_0 > 0$ *et* ε

2. **Itération k**

   $x_{k+1} = x_k - \rho_k \nabla f(x_k)$

3. **Critère d'arrêt**

   $si \ \|x_{k+1} - x_k\| < \varepsilon$ ou $si \ \|\nabla f(x_k)\| < \varepsilon$ stop

   Sinon, on pose *k=k+1*, et on retourne à 2.

avec ε est un réel positif (petit) donné qui représente la précision désirée (la valeur que nous devons fixer pour ε dépend du problème considéré).

V.4.1. Méthode de Gradient à pas optimal

Cette méthode consiste à faire les itérations suivantes :

$$\begin{cases} d_k = -\nabla f(x_k) \\ x_{k+1} = x_k + \rho_k d_k, \ \ \rho_k > 0 \end{cases}$$

où ρ_k est choisi par le règle de minimisation, il consiste à choisir, à chaque itération ρ_k comme étant la solution optimal du problème de minimisation monodimensionnelle de f le long de la demi-droite définie par le point x_k et la direction d_k. Donc, ρ_k est choisi de manière à ce que :

$$f(x_k + \rho_k d_k) < f(x_k + \rho d_k), \quad \forall \rho > 0,$$

dans ce cas les directions de descente d_k générées vérifient : $d_{k+1}^t . d_k = 0$ car si on introduit la fonction

$$g(\rho) = f(x_k + \rho d_k),$$

on a

$$g'(\rho) = \nabla f(x_k + \rho d_k)^t . d_k,$$

et puisque g est dérivable on a nécessairement $g'(\rho_k) = 0$ donc :

$$\nabla f(x_k + \rho_k d_k)^t . d_k = \nabla f(x_{k+1})^t . d_k = -d_{K+1} . d_k = 0.$$

Cette opération de détermination du pas s'appelle **la recherche linéaire.**

Remarque :

1- La méthode de gradient à pas optimal propose un choix du pas qui rend la fonction de coût minimale le long de la direction de descente choisie, Plus précisément le calcul de $x_{k+1} = x_k - \rho_k \nabla f(x_k)$.

2- Deux directions de descente successives calculées par l'algorithme de plus profonde descente (gradient) sont orthogonales ce qui traduisant les **zig-zags** des itérés.

Exemple :

Soit la fonction quadratique suivante: $f(x) = \frac{1}{2} x^t A x - b^t x$ avec A>0 (c'est-à-dire A est une matrice définie positive), on note $g(\rho) = f(x_k + \rho d_k)$, où le optimal ρ_k est caractérisé par $g'(\rho_k) = 0$, on a donc

$$\nabla f(x_k + \rho_k d_k)^t . d_k = (A(x_k + \rho_k d_k) - b)^t . d_k = 0$$

Soit $$\nabla f(x_k + \rho_k A d_k)^t . d_k = 0 \Rightarrow \rho_k = -\frac{(\nabla f(x))^t . d_k}{d_k^t . A d_k} > 0$$

car d_k est une direction de descente et $d_k^t . A d_k > 0$.

La méthode du gradient à pas optimal peut s'écrire : $x_{k+1} = x_k + \rho_k d_k$ avec $\begin{cases} d_k = b - Ax_k \\ \rho_k = \dfrac{d_k^t . d_k}{d_k^t . A} \end{cases}$

Théorème de Convergence :

Soit *f* une fonction $C^1(R^n,R)$, coercive et strictement convexe. On suppose qu'il existe une constante M>0 tel que

$$\forall (x,y) \in R^n \times R^n, \ \|\nabla f(x) - \nabla f(y)\| \le M\|x - y\|$$

Alors, si on choisit le pas ρ_k dans un intervalle $\lfloor \beta_1, \ \beta_2 \rfloor$ tel que $0 < \beta_1 < \beta_2 < \dfrac{2}{M}$, la méthode du gradient converge vers le minimum de *f*.

Preuve :

La fonction *f* admet un minimum $\hat{x}$ unique sur R^n caractérisé par $\nabla f(\hat{x}) = 0$ puisque *f* est strictement convexe. Montrons que la suite (x_k) engendrée par l'algorithme converge vers $\hat{x}$,

on a :

$$f(y) = f(x) + (\nabla f(x), y - x) + \int_0^1 (\nabla f(x + t(y - x)) - \nabla f(x), y - x) dt$$

On applique cette relation à $y = x_{k+1}, \ x = x_k$

$$f(x_{k+1}) = f(x_k) + (\nabla f(x_k), x_{k+1} - x_k) + \int_0^1 (\nabla f(x_k + t(x_{k+1} - x_k)) - \nabla f(x_k), x_{k+1} - x_k) dt$$

Comme $x_{k+1} = x_k - \rho_k \nabla f(x_k)$, on obtient

$$f(x_{k+1}) - f(x_k) \le -\frac{1}{\rho_k}\|x_{k+1} - x_k\|^2 + \int_0^1 \|\nabla f(x_k + t(x_{k+1} - x_k)) - \nabla f(x_k)\| . \|x_{k+1} - x_k\| dt$$

$$\le -\frac{1}{\rho_k}\|x_{k+1} - x_k\|^2 + \frac{M}{2}\|x_{k+1} - x_k\|^2 = \left(\frac{M}{2} - \frac{1}{\rho_k}\right) . \|x_{k+1} - x_k\|^2 .$$

Si on choisit ρ_k dans un intervalle $[\beta_1, \ \beta_2]$ tq $0 < \beta_1 < \beta_2 < \dfrac{2}{M}$, nous obtenons alors

$$f(x_{k+1}) - f(x_k) \le \left(\frac{M}{2} - \frac{1}{\beta_2}\right) . \|x_{k+1} - x_k\|^2$$

La suite $f(x_k)$ est alors strictement décroissante, comme elle est minorée car

$$f(x_k) \ge f(\hat{x}), \ \ \forall k$$

elle est converge. Cela entraîne d'une part que $\left(f(x_{k+1})-f(x_k)\right)$ tends vers « 0 » et d'autre part, la suite (x_k) est bornée (car *f* est coercive). On peut donc extraire une sous suite converge vers $\bar{x}$. De plus, comme

$$\|x_{k+1}-x_k\|^2 \le \left(\frac{1}{\beta_2}-\frac{M}{2}\right)^{-1}.\, f(x_{k+1})-f(x_k)$$

La suite $\left(x_{k+1}-x_k\right)$ tend également vers 0.
Par conséquent

$$\nabla f(x_k)=\frac{x_{k+1}-x_k}{\rho_k}\to 0$$

Par continuité de $\nabla f(\bar{x})=0$, donc $\bar{x}$ est l'unique minimum $\hat{x}$ de *f*. Ceci étant vrai pour toute valeur d'adhérence de la suite (x_k) cela prouve que toute la suite (x_k) converge vers $\hat{x}$.

V.4.2. Méthode de Gradient à pas fixe

On peut utiliser un pas fixé a priori $\rho>0,\ \forall k$ on obtient alors la méthode de gradient simple :

$$\begin{cases} d_k=-\nabla f(x_k) \\ x_{k+1}=x_k+\rho\ d_k \end{cases}$$

Pour $f\in C^1$, cette méthode converge si ρ est choisi assez petit.

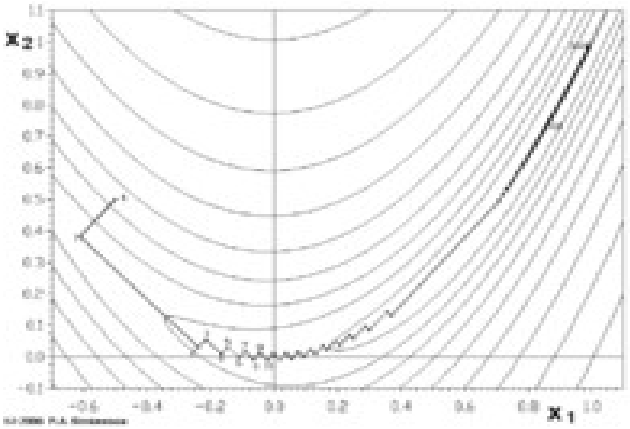
Le choix de pas :

- Un pas « bien choisi » donne des résultats à ceux obtenus par la plus profonde descente.
- Un pas plus petit atténue les zigzags des itérés mais augmente significativement le nombre d'itérations.
- Un pas trop grand fait diverger la méthode.

Inconvénient de la méthode de Gradient :

L'algorithme du gradient peut rencontrer un certain nombre de problèmes, en particulier celui de la convergence lorsque le minimum de la fonction se trouve au fond d'une vallée étroite (plus précisément lorsque le conditionnement de la matrice hessienne est élevée). Dans un tel cas, la suite des $\{x_k\}$ oscille de part et d'autre de la vallée et progresse laborieusement, même lorsque les ρ_k sont choisis de sorte à minimiser *f*(*x*).

La figure ci-dessous illustre ce type de comportement pour une fonction à 2 dimensions.



Les points faibles de l'algorithme du gradient sont :

- L'algorithme peut nécessiter de nombreuses itérations pour converger vers un minimum local, notamment si la courbure est très différente dans des directions différentes.
- La recherche du pas ρ optimal, généralement effectuée par une recherche linéaire, peut se révéler très longue. Inversement, utiliser un pas ρ fixe peut conduire à de mauvais résultats.
- L'inconvénient majeur de cette méthode survient lorsque la direction du gradient y est presque orthogonale à la direction menant au minimum. Celle-ci adopter alors le comportement bien connu du « zig-zag » et progresse extrêmement lentement.
- Cette méthode a pour avantage d'être très facile à mettre en œuvre. Malheureusement les conditions de convergence sont assez lourdes (c'est essentiellement de la stricte convexité) et la méthode est en général assez lente.
- Pour construire les méthodes de gradient, nous avons remplacé *f* par son approximation linéaire au voisinage de l'itéré courant. Nous avons vu que ces méthodes ne sont pas très performantes, en partie parce qu'elles ne tiennent pas compte de la courbure (ou de la Hessienne) qui est une information de second ordre.

Exemple :

Comparaison de la vitesse de l'algorithme de gradient à pas fixe et à pas optimal pour la résolution du Problème :

$$\min_{x\in R^n} f(x) = \min_{x\in R^n} \frac{1}{2}(Ax,x) - (b,x)$$

avec A est une matrice symétrique carré $(n.n)$ et b est un vecteur de R^n. Ce problème équivaut a la résolution du système $Ax=b$

On a choisi A = diag([1 : 1 : 100]), xsol = ones(N, 1) et x0 = rand(N, 1).

//Gradient a pas optimal pour problème quadratique

```
clear
clf
N=100;
//A=2*diag(ones(N,1),0)-diag(ones(N-1,1),-1)-diag(ones(N-1,1),1);
A=diag([1:1:N]);
xsol=ones(N,1);
b=A*xsol;
x0=rand(N,1);
itermax=100;
```

//1.Gradient pas fixe

```
x=x0;
Tfixe=[norm(x-xsol,'inf')];
mu=0.01;
for k=1:itermax
d=A*x-b;
x=x-mu*d;
err=norm(x-xsol,'inf');
Tfixe=[Tfixe err];
end
```

//2.Gradient pas optimal

```
x=x0;
Topt=[norm(x-xsol,'inf')];
for k=1:itermax
d=A*x-b;
mu=d'*d/(d'*A*d);
x=x-mu*d;
err=norm(x-xsol,'inf');
Topt=[Topt err];
end

plot2d([0:1:itermax]',[Tfixe' Topt'],logflag='nl',style=[1 2]);
xtitle('','k','||xk-x||')
legends(['pas fixe';'pas optimal'],[1 2],opt='ll')
```

V. 5. Méthode des gradients conjuguées

La méthode du gradient conjugué est un algorithme pour résoudre des systèmes d'équations linéaires dont la matrice est symétrique définie positive. Cette méthode, imaginée en 1950 simultanément par Cornelius Lanczos et Magnus Hestenes, est une méthode itérative qui converge en un nombre fini d'itérations (au plus égal à la dimension du système linéaire).

Principe de la méthode

On considère *f* une fonction quadratique de R^n dans R : $f(x)=\frac{1}{2}x^tAx-b^tx$ où A est une matrice symétrique définie positive *(n.n)*, b est un vecteur de R^n

L'idée est de reprendre la méthode de plus forte pente (i.e. la méthode de Gradient) mais d'éviter les "mauvaises" directions de descente trouvées précédemment qui faisaient "rebondir" le chemin de descente sur les parois de *f*. Pour éviter les zigzags et accélérer la convergence, on cherche des directions de descente « d_i » mutuellement conjuguées par rapport « A » pour prendre en compte la géométrie de la fonction *f*.

Directions conjuguées

Définition :

1- On dit que deux directions x et y de R^n sont conjuguées par rapport à la matrice « A » si $x^t.Ay=0$.

2- Si la matrice « A » est symétrique et définie positive, on déduit un produit scalaire :

$$(x,y)_A=(Ax,y)=(x,Ay)=x^tAy.$$

3- Des vecteurs $d_1, d_2 ..., d_k$ de R^n sont dits A-conjugués si pour tous $i \neq j$, on a :

$$(d_i;d_j)_A=0$$

Remarque :

1- Les vecteurs x et y sont conjugués par r apport A s'ils sont orthogonaux pour ce produit scalaire de A, donc la notion de conjugaison équivaut à la notion d'orthogonalité pour le produit scalaire associé à la matrice A

2- La conjugaison est une relation symétrique.

3- (d_k) est une suite de « n » directions conjuguée deux à deux, forment une base orthonormée de R^n, donc

$$d_k^t A\hat{x} = d_k^t b = \sum_{i=1}^{n} \alpha_i d_k^t A d_i = \alpha_k d_k^t A d_d \Rightarrow \alpha_k = \frac{d_k^t b}{d_k^t A d_k} = \frac{(d_k, b)}{(d_k, d_k)_A} = \frac{(d_k, b)}{\|d\|_A^2}$$

$$\forall \hat{x} \in R^n, \quad \hat{x} = \sum_{i=1}^{n} \alpha_i d_i \Rightarrow A\hat{x} = b = \sum_{i=1}^{n} \alpha_i A d_i$$

Donc l'idée de la méthode de gradient conjuguée est de

a- Trouver une suite de n-directions conjuguée et

b- Calculer le coefficient α_i.

Théorème :

Soient $d_1, d_2, \ldots, d_k$ un ensemble de vecteurs non nuls de R^n, A-conjugués. Alors les vecteurs $d_1, d_2 \ldots, d_k$ sont linéairement indépendants.

Preuve :

C'est une conséquence immédiate du résultat bien connu : si des vecteurs non nuls sont orthogonaux par rapport à un produit scalaire, alors ils sont indépendants

Méthode des directions conjuguées

L'idée est que si on arrive à définir un algorithme itératif utilisant n-directions conjuguées, alors on aura parcouru tout l'espace R^n dans toutes les directions possibles (ce qui n'était pas le cas pour l'algorithme de plus profonde descente).

Étant donnés un point initial x_0 *de* R^n et *n*-directions A-conjuguées $d_1, d_2 \ldots, d_k$ on définit le -schéma suivant : $x_{k+1} = x_k + \rho_k d_k$

où ρ_k est le scalaire minimisant *f(x)* selon la direction $x_k + \rho_k d_k$ et qui est défini par

$$\rho_k = -\frac{r_k^t . d_k}{d_k^t . A d_k}$$

et $$r_k = -\nabla f(x_k) = b - A x_k$$

L'algorithme ci-dessous résout $Ax = b$, où A est une matrice réelle, symétrique, et définit positive. Le vecteur d'entré x_0 peut être une approximation de la solution initiale ou 0.

Algorithme de la méthode de Gradient Conjuguée

1. **Initialisation** k=0,

Choix de x_0 dans R^n ; $r_0 = b - Ax_0$; $p_0 = r_0$

2. **Itération k**

$$\alpha_k = \frac{r_k^t . r_k}{p_k^t A p_k}$$

$$x_{k+1} = x_k + \alpha_k p_k, \qquad r_{k+1} = r_k - \alpha_k A p_k$$

3. **Critère d'arrêt**

Si r_{k+1} est suffisamment petit, alors on sort de la boucle

$$\beta_k = \frac{r_{k+1}^t . r_{k+1}}{r_k^t r_k} \qquad p_{k+1} = r_{k+1} + \beta_k p_k$$

Sinon, on pose $k = k+1$, et on retourne à 2.

Propriétés :

1. $\forall i \neq j;\ d_i^t . A d_j = 0$: les directions de descentes sont toutes mutuellement conjuguées.
2. L'algorithme des gradients conjuguées converge en au plus 'n' itérations.
3. On va noter pour tout $k \in N,\ G_k = L(\nabla f(x_0), ...\nabla f(x_k)) \subset R^n$ avec $d_i = -\nabla f(x_i)$ .La méthode des gradients conjugués consiste à chercher $x_{k+1} \in x_k + G_k$ tel que : $f(x_{k+1}) = \min_{y \in x_k + G_k} f(y)$

(en supposant qu'un tel minimum existe). Donc, on minimise la fonction f sur un espace plus "grand" que dans la méthode de gradient à pas optimal où on minimise f sur $x_{k+1} \in x_k + G_k$ avec

$G_k = L(\nabla f(x_k))$

Remarque :

1- Si A a seulement « r » valeurs propres distinctes (avec r < n), alors l'algorithme du gradient conjugué arrive à la solution en au plus **r**-itérations.

2- Si A a des valeurs propres $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$, nous avons que

$$\left\| x_{k+1} - \hat{x} \right\|_A^2 \leq \left(\frac{\lambda_{n-k} - \lambda_1}{\lambda_{n-k} + \lambda_1} \right)^2 \left\| x_0 - \hat{x} \right\|_A^2$$

3- Une caractéristique de l'algorithme de gradients conjugués est que la matrice A ne nécessite aucune manipulation : pas besoin donc de stocker la matrice A, il suffit d'implémenter (efficacement) les produits matrice-vecteur sans jamais former la matrice.
De plus cet algorithme est surtout utilisé pour résoudre des systèmes linéaires de la forme : Ax = b (ce qui correspond à la recherche de points critiques pour le problème quadratique considéré) sans former la matrice A. Ceci est particulièrement intéressant pour les problèmes de grande taille pour lesquels la matrice A est souvent creuse. A noter également, que la méthode des gradients conjugués est une méthode dans laquelle les erreurs d'arrondis sont amplifiées au cours des itérations. Peu à peu, les relations de conjugaison sont perdues car les relations d'orthogonalité des gradients ne sont pas vérifiées exactement.

Théorème :
Si $f : R^n \to R$ est quadratique et elliptique la méthode de gradient conjugué converge en n -itérations au plus où n est l'ordre de A.

Remarque

Cette méthode est très stable même pour des matrices mal conditionnées. Elle demande $2n^3$ opérations dans le cas d'une matrice pleine et de n itérations pour une matrice creuse, le nombre d'opérations diminue beaucoup.

V.6. Méthode de Newton

La méthode de Newton est attribuée au mathématicien, physicien et astronome anglais *Issac Newton* (1642-1727). C'est l'un des méthodes les plus anciennes utilisées pour résoudre les problèmes d'optimisation.

Cette méthode permettant de trouver une solution d'un système (en général non linéaire) de n équations avec n inconnues, autrement dit de trouver les zéros d'une fonction $F : \mathbb{R}^n \to \mathbb{R}$ (dans le cas évoqué, on a $F = \nabla f$), i.e. résoudre l'équation non linéaire $F(x) = 0$ dans R^n.

Dans le cas n = 1, cette méthode de Newton s'appelle également la méthode de la tangente.

L'idée est de remplacer le point x_k obtenu à l'itération k, par le point d'intersection de la tangente en $(x_k; f(x_k))$ à la courbe représentative de f avec l'axe des abscisses.

Considérons maintenant le cas général, i.e. on dispose d'une fonction $F : \mathbb{R}^n \to \mathbb{R}$ de classe (au moins) C^1 telle que l'équation $\nabla F(x) = 0$ admette au moins une solution x et la matrice $H(x)$ est définie positive.

Principe de la méthode de Newton :

L'idée de la méthode itérative de Newton est de minimiser à chaque itération l'approximation quadratique de f (de classe C^2) au point courant x_k et donnée par le développement de Taylor d'ordre 2 :

$$f(x) \cong q(x) = f(x_k) + (\nabla f(x_k), x - x_k) + \frac{1}{2}(x - x_k)^t . H(x_k).(x - x_k)$$

On considère le problème : $\{\min \quad q(x), \quad x \in R^n\}$.

Une condition nécessaire pour que le minimum de $q(x)$ soit atteint est $\nabla q(x) = 0$ tel que

$$\nabla q(x) = \nabla f(x_k) + H(x_k)(x - x_k) = 0$$

Le vecteur généré à l'itération $(k+1)$ est le vecteur minimisant $q(x)$

$$\nabla q(x_{k+1}) = 0 \Rightarrow \nabla f(x_k) + H(x_k)(x_{k+1} - x_k) = 0$$

$$\Rightarrow x_{k+1} = x_k - (H(x_k))^{-1} \nabla f(x_k),$$

Donc, la direction de descente choisie a chaque itération est

$$d_k = -(H(x_k))^{-1} \nabla f(x_k),$$

Elle est bien une direction de descente car elle vérifie la relation

$$(d_k ; \nabla f(x_k) \prec 0, \quad \forall k \geq 0.$$

En effet,

$$d_k = -(H(x_k))^{-1} \nabla f(x_k) \Rightarrow \nabla f(x_k) = -H(x_k).d_k$$

$$(d_k ; \nabla f(x_k) = (d_k, -H(x_k).d_k) = -d_k^t.H(x_k).d_k \prec 0$$

Car la matrice Hiessien est définie positive.

d_k est appelée direction de Newton et les points sont successivement générés par cette méthode de la manière :

$$\begin{cases} x_0 \in R^n \ donné, \quad \rho = 1 \\ x_{k+1} = x_k - (H(x_k))^{-1} \nabla f(x_k) \end{cases}$$

Remarques :

- Cette méthode est bien définie à chaque itération si la matrice hessienne $H(x_k)$ est définie positive : ceci est vrai en particulier au voisinage de la solution $\hat{x}$ cherchée si on suppose que $H(\hat{x})$ est définie positive (par continuité de H).
- La méthode de Newton est un algorithme de descente à pas fixe égal à 1.
- Si la fonctionnelle f est quadratique, strictement convexe, alors l'algorithme converge en une seule itération.

Algorithme de la méthode de Newton

1. **Initialisation** K=0,

   choix de x_0 dans un voisinage de $\hat{x}$, ε

2. **Itération k**

   $x_{k+1} = x_k - \left(H(x_k)\right)^{-1} \nabla f(x_k)$

3. **Critère d'arrêt**

   $si \ \|x_{k+1} - x_k\| < \varepsilon$ stop

   Sinon, on pose $k=k+1$, et on retourne à 2.

Remarques :

-La méthode de Newton est intéressante car sa convergente est quadratique au voisinage de la solution

c-à-d : $\|x_{k+1} - \hat{x}\| \leq \gamma \|x_k - \hat{x}\|^2, \quad \gamma > 0$

mais la convergence n'est assurée que si x_0 est suffisamment proche de $\hat{x}$, ce qui en limite l'intérêt pour cela nous allons donner le théorème qui garanti la convergence.

-L'étape 2 de l'algorithme revient à résoudre le système linéaire suivant:

$$H(x_k)(x_k - x_{k+1}) = \nabla f(x_k)$$

Théorème :

Soit $f : R^n \to R$ avec $f \in C^2(R^n)$ et un point $\hat{x}$ de Rⁿ tel que $\nabla f(\hat{x}) = 0,\ et\ H^{-1}(\hat{x})$ existe. Soit x_0 un point initial assez proche de $\hat{x}$ de sorte que cette proximité s'exprime de la façon suivante :

$$\exists\ K_1, K_2 > 0 \text{ avec } K_1 K_2 \|x_0 - \hat{x}\| < 1. \text{ tel que}$$

1) $\left\|(H(x))^{-1}\right\| < K_1$.
2) $\left\|\nabla f(\hat{x}) - \nabla f(x) - H(x)(x - \hat{x})\right\| \le K_2 \|x - \hat{x}\|^2$. Pour tout x satisfaisant $\|x - \hat{x}\| \le \|x_0 - \hat{x}\|$.

Alors, l'algorithme converge d'une façon super-linéaire (quadratique) vers $\hat{x}$.

Inconvénients de la méthode de Newton :

1. Cette méthode fonctionne très bien pour les petits dimensions ($1 \le n \le 10$) lorsque on peut calculer facilement $H(x)\ et\ (H(x))^{-1}$, ce calcul nécessite des itérations plus nombreuse et coûteuse dans les problèmes des grandes tailles.
2. Comme $x_{k+1} = x_k - (H(x_k))^{-1} \nabla f(x_k)$, le point (x_{k+1}) n'est pas toujours bien définie c-à-d possible que $(H(x))^{-1}$ n'existe pas (cela intervient typiquement lorsque la méthode atteint un région où f est linéaire donc ses secondes dérivées partielles valent zéro), et même elle est existe, la direction $d_k = -(H(x_k))^{-1} \nabla f(x_k)$, n'est pas toujours une direction de descente (si $(H(x_k))$ est définie positive alors d_k est une direction de descente).
3- L'inconvénient majeur de la méthode est sa sensibilité au choix du point de départ x_0 : Si ce point est mal choisis (trop loin de la solution) la méthode peut soit diverger, soit converge vers une autre solution. Pour choisir le point de départ x_0 "assez prés" de $\hat{x}$ on essaie de s'approcher de $\hat{x}$ par une méthode de type gradient par exemple, puis on applique la méthode de Newton.

V.7. Méthode de Quasi-Newton

Les méthodes de Quasi-Newton sont élaborées pour l'optimisation et pour pallier aux inconvénients de la méthode de Newton elle :

1- garde la rapidité de la méthode de Newton,
2- évite le calcul (coûteux) de la matrice [$H(x_k)$] à chaque itération.
3- plus robustes par rapport au du point de départ. On y trouve les méthodes dites *"région de confiance"* qui s'attachent à rendre la méthode robuste (i.e. peu sensible) par rapport x_0.
4- $(H(x))^{-1}$ n'est pas nécessairement connue, peut être très chère à calculer, et $H(x_k)$ peut être très difficile à inverser. On remplace alors et $(H(x_k))^{-1}$ par une matrice D_k, éventuellement constante, qui est censée approcher $H(x_k)$ ou bien son inverse. Parfois même, on remplace $\left(H(x_k)\right)^{-1}\nabla f(x_k)$ par un vecteur y_k facile à calculer, économisant ainsi l'encombrement de la matrice en mémoire.

Donc, l'algorithme de cette méthode est donné comme suit :

Algorithme de la méthode de Quasi-Newton

1. **Initialisation** K=0,

   choix de x_0 , α_0, ε

2. **Itération k**

   $x_{k+1} = x_k - \alpha_k D_k \nabla f(x_k)$

3. **Critère d'arrêt**

   $si\ \|x_{k+1} - x_k\| < \varepsilon$ stop

   Sinon, on pose $k=k+1$, et on retourne à 2.

Remarque :

Dans cet algorithme, on va utiliser une approximation D_k *de* $(H(x_k))^{-1}$ et puis on va trouver α_k (par une recherche linéaire) qui minimise la fonction $\phi(\alpha) = f(x_k + \alpha_k d_k)$

Exercices du chapitre III

Exercice 1 : ----------

Soit $f(x)=\frac{1}{2}(Ax,x)-(b,x)$ où A est une matrice symétrique définie positive $N\times N$ de valeurs propres $0<\lambda_1<...<\lambda_N$ et $b\in R^n$. Notons $\hat{x}$ le minimum de *f* sur R^n. On considère la suite (u_k) d'éléments de R^n telle que $u_0\in R^n$ quelconque et

$$\forall k\in N,\ u_{k+1}=u_k-\rho\nabla f(u_k)$$

où ρ est un réel positif fixé.

- Montrer que $u_{k+1}-\hat{x}=(I_n-\rho A)(u-\hat{x})$, où I_n désigne la matrice identité de taille N .
- Montrer que l'algorithme de gradient à pas fixe converge pour : $0<\rho<2/\lambda_N$

Exercice 2 :----------

Soit *f(x)* une fonction C^1 sur R^n. On sait que dans un voisinage d'un point $a\in R^n$, *f* diminue le plus rapidement si l'on passe dans la direction de la pente négative de *f* à a, c'est à dire la direction $(-\nabla f(a))$

On commence avec une estimation, x_0, pour un minimum local de *f* et considère la séquence

$(x_0,x_1,.....)$ où $\forall i\in N,\ x_{i+1}=x_i-\rho\nabla f(x_i)$ tel que : $f(x_0)\geq f(x_1)\geq f(x_2)\geq .....$ et *f(x)* est définie comme suit:

$$f(x)=\frac{1}{2}x^t\begin{pmatrix}6 & -2\\ -2 & 6\end{pmatrix}x+\begin{pmatrix}-1 & -1\end{pmatrix}x$$

(i) Quel est l'unique minimum (global) $\hat{x}$ de *f*?

(ii) En commençant avec $x_0=[0\ 0]^t$ et $\rho=0.1$, calculer les deux itérés suivants x_1 et x_2.

(iii) Trouver la taille de pas maximum ρ pour que la méthode converge vers $\hat{x}$ quel que soit le point x_0.

Exercice 3 :----------

On cherche les minima locaux de la fonction : $J(x_1,x_2)=\frac{1}{2}x_1^2+x_1\cos(x_2)$

1. Déterminer l'ensemble de points de minimum locaux en utilisant les conditions nécessaires.

2. Appliquer la méthode de Newton avec le point de départ $u_0=\begin{pmatrix}1\\1\end{pmatrix}$ .Observer le comportement des itérées et le point de convergence. Expliquer les observations.

3. Déterminer le modèle quadratique de la fonctionnelle J. Y a-t-il équivalence entre l'algorithme de Newton et la méthode utilisant le modèle quadratique ? Pourquoi ?

Exercice 4 :----------

Considérons le problème d'optimisation quadratique suivant $(P)\left\{\min \ \frac{1}{2}X^tAX - b^tX\right\}$

où A est une matrice symétrique définie positive $N \times N$ de valeurs propres $0 < \lambda_1 < ... < \lambda_N$ et $b \in R^N$.

- Comment s'exprime dans ce cas la méthode de Newton ?
- Sa convergence dépend-elle du point initial ?
- En combien d'itération converge-t-elle ? Réinterpréter la méthode de Newton.

Exercice 5 :----------

On cherche à trouver le min d'une fonction *f* de R^n dans R, on utilise la méthode de gradient à pas optimal : $x_{k+1} = x_k - \rho_k \nabla f(x_k)$ où ρ_k est la solution du problème $\{\min \ f(x_k - \rho \nabla f(x_k)),\ \rho \in R\}$

1- On suppose que *f* est convexe, montrer la convergence de la méthode ?
2- Quels sont les problèmes posés par la mise en œuvre numérique de la méthode ?
3- Proposer un algorithme basé sur la méthode du gradient à pas constant, consistant à poser $x_{k+1} = x_k - \rho_k \nabla f(x_k)$ où ρ_k est choisi dans une liste donnée à l'avance : $\rho,\ \frac{\rho}{2},\ \frac{\rho}{4},...,\frac{\rho}{2^k}$

Exercice 6 :----------

Soit $A \in M_N(R)$ et *J* la fonction définie de IR^N dans IR par : $J(x) = e^{\|Ax\|^2}$, où $\|.\|$ désigne la norme euclidienne sur IR^N.

1. Montrer que *J* admet un minimum (on pourra le calculer. . .).

2. On suppose que la matrice *A* est inversible, montrer que ce minimum est unique.

3. Ecrire l'algorithme du gradient à pas optimal pour la recherche de ce minimum. [On demande de calculer le paramètre optimal ρ_k en fonction de A et de x_k]. A quelle condition suffisante cet algorithme converge-t-il ?

Exercice 7 :----------

On considère la fonction *f* définie par $f(x, y) = x^4 + y^4 + 4xy$

1- Trouver les points critiques et la nature de chacun d'entre eux.
2- On applique l'algorithme de gradient à pas fixe à la minimisation de *f* avec un pas fixé ρ. Prouver que, pour toute initialisation (x_0, y_0), l'algorithme converge, pour ρ assez petit, vers un point critique de *f*. On choisit $x_0 = y_0 = 1$, Que ce passe-t-il si $\rho = 0.25$? *si* $\rho = 0.5$?

Exercice 8 :----------

Soit la fonction de Rosenbrock définie comme suit : $f(X) = f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$

1- Calculer le gradient et la matrice Hessienne de la fonction *f*.
2- Vérifier que $\hat{x} = [1,\ 1]^t$ est un minimum local de *f*.
3- Calculer les 5 premiers itérés de la méthode de Newton pour minimiser *f* en commençant par $x_0 = [-1,\ -2]$. Calculer la norme de l'erreur $\|x - \hat{x}\|$ à chaque itération et déterminer si le taux de convergence est quadratique.