REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITÉ ABDELHAMID IBN BADIS-MOSTAGANEM
FACULTE DES SCIENCES EXACTES ET INFORMATIQUE
DEPARTEMENT DE MATHEMATIQUES

MEMOIRE DE* MASTER *EN MATHEMATIQUES

OPTION* : *Modélisation, Contrôle et Optimisation

Sujet

MÉTHODE DE BROYDEN, FLETCHER, GOLDFARB ET SHANNO POUR L'OPTIMISATION SANS CONTRAINTES

Présenté par BOUZID Nawel

Soutenu le 26 /06/2012

Devant le Jury			
Omar Belhamiti	**Président**	**MCA**	**U. MOSTAGANEM**
Hocine ABLAOUI	**Examinateur**	**MCA**	**U. MOSTAGANEM**
Abdessamad AMIR	**Encadreur**	**MCA**	**U. MOSTAGANEM**

Table des matières

Remerciements

Tout d'abord je remercie DIEU qui m'a donné la force, le courage et la volonté pour effectuer ce modeste travail. Je veux aussi remercier mes parents qui ont consacré leur temps pour m'encourager pendant mes études.

Nous devons des remerciments particulièrs à :

Mr A. AMIR d'avoir guidé mes pas et faciliter les taches de ce travail par ses conseils précieux et son amour du travail qui m'a beaucoup aidé durant la réalisation de ce mémoire.

A Mr O. BELHAMITI et Mr H. ABLAOUI qui nous feront le grand honneur de juger ce mémoire.

Je remercie également l'ensemble de professeurs du département de mathématiques qui ont veillé à notre bonne formation et toutes les personnes qui ont participé de loin ou de prés à la réalisation de ce travail.

Introduction

Au milieu des années cinquante du siècle dérnier, au Laboratoire national d'Argonne ; un des plus importants laboratoires de recherche des États-Unis, W.C. Davidon, un physicien chercheur utilisait une méthode de descente pour résoudre un problème d'optimisation. A cette époque, les ordinateurs n'étaient pas très stable, le système informatique se plantait souvent avant la fin des calculs. Frustré, Davidon, a décidé de trouver un moyen d'accélérer les itérations de son algorithme. Ainsi, Il a développé le premier algorithme de quasi- Newton. Plus tard, Fletcher et Powell deux mathématiciens ont démontré que ce nouvel algorithme (DFP) était beaucoup plus rapide que les autres existants, il s'est avéré être l'une des idées les plus créatives de l'optimisation non linéaire. Au cours des années suivantes, de nombreuses variantes ont été proposées et des centaines d'articles ont été consacrés à son étude. l'ironie de l'histoire des sciences veut que le papier de Davidon ne sera pas accepté pour publication, il est resté comme un rapport technique pour plus de trente ans jusqu'à son apparition dans le premier numéro de la grand revue " SIAM journal of optimization " en 1991.

Le but de ce travail est d'étudié une des plus populaire des méthodes Quasi-Newtonienne, la méthode BFGS (Broyden, Fletcher, Goldfar et Shannon). Le mémoire est répartie en quatre chapitres. Le premier est consacré à l'analyse mathématique d'un problème d'optimisation sans contraintes, le schéma général des méthodes de descentes sera présenté avec deux approches de recherche linéaire ; exacte et inexacte. Au second chapitre, L'accent sera mis sur les limites de la méthode de Newton pour l'optimisation sans contraintes. Comme introduction aux méthodes quasi-Newtonienne, nous étudierons la méthode de rang un au troisième chapitre. Nous verrons en dernier lieu, que la méthode BFGS remédiée aux inconvénients des autres méthodes de quasi-Newton telles que la méthode de rang un et la méthode DFP (Davidon, , Fletcher et Powell). Toute les méthodes de ce mémoire ont été implémenté sous Matlab, des tests numérique ont étém réalisés pour justifier les inconvénients et les avantages de chaque méthode.

Chapitre 1

Optimisation sans contraintes

Un problème d'optimisation sans contraintes peut être formulé sous la forme :

$$\begin{cases} \min f(u) \\ u \in \mathbb{R}^n \end{cases}, \tag{1.1}$$

où $f : \mathbb{R}^n \to \mathbb{R}$ est supposée régulière non linéaire. Résoudre le problème (1.1) revient généralement à en chercher des solutions locales parce que notre connaissance de f est habituellement seulement locale de plus le minimum global peut être difficile à trouver.

On dira que u^* est un minimum local de f sur $U \subset \mathbb{R}^n$ s'il existe une boule $B(u^*, r)$ avec $r > 0$, telle que $f(u^*) \leq f(u), \forall u \in B(u^*, r) \cap U$, et on dira que u^* est un minimum global de f si $f(u^*) \leq f(u)$, $\forall u \in U$. Etant donnée que $\max f = -\min -f$, tous les algorithmes présentes ici recherchent un minimum de la fonction objectif.

1.1 Conditions d'optimalité

On va donc se restreindre au cas des fonctions différentiables, nous donnons d'abord une condition nécessaire d'optimalité ensuite une condition suffisante.

Théorème 1.1 *[4]* ***Condition nécessaire d'optimalité***
Soit $f : \mathbb{R}^n \to \mathbb{R}$ différentiable et u^* un minimum de f. Alors on a nécessairement

$$\nabla f(u^*) = 0, \tag{1.2}$$

où

$$\nabla f\,(u^*) = \left(\frac{\partial f}{\partial u_i}(u^*)\right)_{1 \leq i \leq n}$$

désigne le gradient de f en u.

Preuve. Il existe $\overline{\alpha} > 0$ / $\forall\ \alpha \in]0, \alpha[$, $\forall h \in \mathbb{R}^n : u^* + \alpha h \in B(u^*, r)$, d'où

$$f(u^*) \leq f(u^* + \alpha h),\ \forall h \in \mathbb{R}^n.$$

On a donc

$$\lim_{t \to 0^+} \frac{f(u^*) - f(u^* + \alpha h)}{\alpha} = \nabla f(u^*)^T h \leq 0, \forall h \in \mathbb{R}^n,$$

et

$$-h \in \mathbb{R}^n \Longrightarrow -\nabla f(u^*)^T h \leq 0 \Longrightarrow \nabla f(u^*)^T h \geq 0.$$

Donc

$$\nabla f(u^*)^T h = 0, \forall h \in R^n \Longrightarrow \nabla f(u^*) = 0.$$

D'où la condition nécessaire d'optimalité. ■

Théorème 1.2 *[4]* ***Condition suffisante d'optimalité***
Soit $f : \mathbb{R}^n \to \mathbb{R}$ convexe et différentiable. Si u^* vérifie

$$\nabla f(u^*) = 0.$$

Alors u^* est un minimum global de f.

Preuve. Soient $u \in \mathbb{R}^n$ et $\lambda \in [0, 1]$. Puisque f est convexe on a

$$f(u^* + \lambda(u - u^*)) \leq \lambda f(u) + (1 - \lambda) f(u^*).$$

En retranchant $f(u^*)$ de chaque côté de l'inégalité et en divisant par λ, on obtient

$$\frac{f(u^* + \lambda(u - u^*)) - f(u^*)}{\lambda} \leq f(u) - f(u^*).$$

On faisant tendre λ vers 0, on obteint

$$\nabla f(u^*)^T (u - u^*) \leq f(u) - f(u^*),$$

donc

$$0 \leq f(u) - f(u^*), \ \forall u \in \mathbb{R}^n$$

ce qui montre bien que u^* est un minimum global. ■

1.2 Méthodes de descente

1.2.1 Principe des méthodes de descente :

Ces méthodes sont fondées sur la notion de direction de descente.

Définition 1.1 *Soit $f : \mathbb{R}^n \to \mathbb{R}$. On dira que le vecteur d est une direction de descente en u, s'il existe $\overline{\alpha}$ tel que :*

$$f(u + \alpha d) < f(u), \ \forall \alpha \in]0, \overline{\alpha}[\, .$$

Le scalaire α est appelé le pas de déplacement ou le pas de l'algorithme.

De telles directions sont intéressantes en optimisation car, pour faire décroitre f, il suffit de faire un déplacement le long de d. Les méthodes à directions de descente utilisent cette idée pour minimiser une fonction. Elles génèrent la suite des itérés $\{u_k\}_{k \geq 1}$ approchant une solution u^* de (1.1) par la récurrence

$$u_{k+1} = u_k + \alpha_k d_k, \qquad \alpha_k > 0$$

tout en assurant la propriété

$$f(u_{k+1}) < f(u_k).$$

Le vecteur d_k est la direction de descente en u_k et α_k est le pas de la méthode à l'itération k. On peut caractériser les directions de descente en u_k à l'aide du gradient :

Proposition 1.1 *Soit $d \in \mathbb{R}^n$ vérifiant*

$$\nabla f(u)^T d < 0.$$

Alors d est une direction de descente en u.

Preuve. Le développement de Taylor donne :

$$f(u + \alpha d) = f(u) + \alpha \nabla f(u)^T d + \alpha \varepsilon(\alpha),$$

donc si on écrit

$$\frac{f(u + \alpha d) - f(u)}{\alpha} = \nabla f(u)^T d + \varepsilon(\alpha).$$

On voit bien que pour α suffisamment petit on aura

$$f(u + \alpha d) - f(u) < 0.$$

d'où d est une direction de descente. ■

Une fois le problème de direction de descente résolu, l'autre étape des méthodes de descente est la recherche du pas α_k. Cette procédure est appelée " La recherche linéaire".

1.2.2 Recherche linéaire

Le choix d'un α_k acceptable a un impact majeur sur l'efficacité des méthodes d'optimisation non linéaires. Dans cette section, nous allons décrire les différentes manières pour déterminer un pas α le long d'une direction de descente d. On suppose que d soit une direction de descente pour limiter la recherche à des valeurs de α positifs.

Recherche linéaire exacte

Comme on cherche à minimiser f, il semble naturel de chercher à minimiser le critère le long de d et donc de déterminer le pas α comme solution du problème unidimensionnel suivant

$$\begin{cases} \min \psi(\alpha) = f(u + \alpha d) \\ \alpha \geq 0 \end{cases}.$$

Rappelons que si f est différentiable, une condition nécessaire pour que α^* soit optimale est :

$$\psi'(\alpha^*) = 0. \tag{1.3}$$

Si f est une fonction quadratique (i.e ; $f(u) = \frac{1}{2} u^T H u + bu + c$ ($H = \nabla^2 f(u)$)), alors α^* vérifiant la relation (1.3) est donnée par

$$\alpha^* = -\frac{\nabla f(u)^T d}{d^T H d}.$$

Pour une fonction non quadratique, on va utiliser une méthode itérative pour calculer α. Nous allons présenter ici une méthode d'optimisation unidimentionnelle appelée " Méthode de la Sécante".

Méthode de la sécante

Dans cette méthode, on recherche successivement les zéros d'une approximation linéaire de ψ. Pour ce faire, il faut disposer de deux points de départ α_0 et α_1 et des valeurs de ψ correspondantes. L'itération principale de cette méthode est donnée par la formule suivante :

$$\alpha_{k+1} = \alpha_k - \frac{\alpha_k - \alpha_{k-1}}{\psi'(\alpha_k) - \psi'(\alpha_{k-1})}\psi'(\alpha_k).$$

avec

$$\psi'(\alpha_k) = \nabla f(u_k + \alpha d_k)^T d_k.$$

Pour plus de détail sur cette méthode voir [3].

Recherche linéaire inexactec

Dans la plupart des algorithmes d'optimisation modernes, on évite la recherche linéaire exacte, car trouver α^* signifie qu'il va falloir calculer un grand nombre de fois la fonction ψ, et cela peut être dissuasif du point de vue temps de calcul. Des stratégies plus pratiques utilisent une recherche inexacte qui assure une décroissance suffisante de f.

La recherche linéaire se fait en deux étapes : '"bracting phase " qui trouve un intervalle $[a, b]$ qui contient des α_k acceptable, et une phase d'interpolation qui calcule une bonne longueur de pas dans cet intervalle. Cela conduit à la notion d'intervalle de sécurité.

Définition 1.2 *On dit que $[a, b]$ est un intervalle de sécurité s'il permet de classer les valeurs de α de la façon suivante :*

- Si $\alpha < a$ alors α est considéré trop petit.

-Si $a \leq \alpha \leq b$ alors α est satisfaisant.

-Si $\alpha > b$ alors α est considéré trop grand.

Il faut maintenant préciser quelles sont les règles sur ψ qui vont nous permettre de caractériser les valeurs de α convenables. En littérature, il existe plusieurs règles. Nous introduisons ici une des plusieurs règles les plus utilisées.

la règle de wolfe :

On définit comme précédemment la fonction

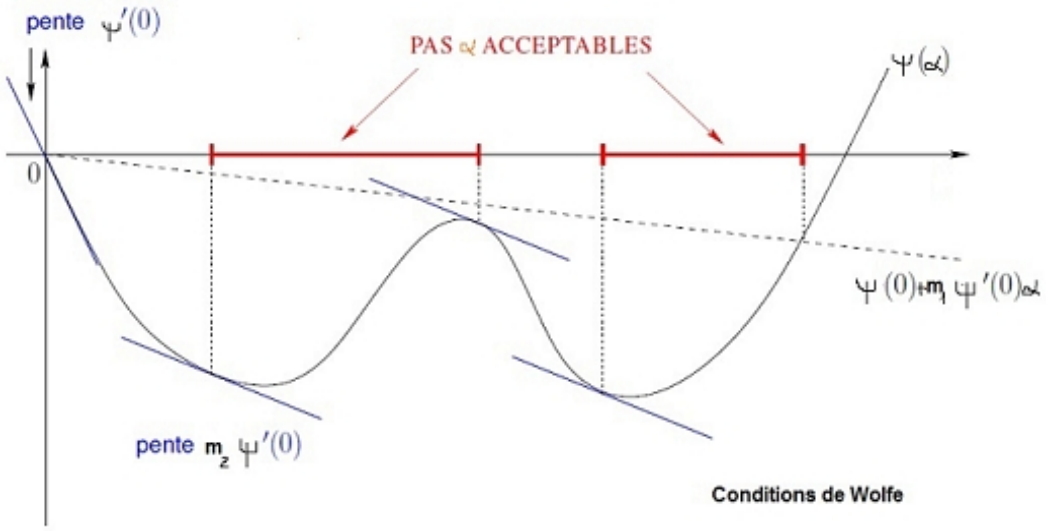
$$\psi_k(\alpha) = f(u_k + \alpha_k d_k). \tag{1.4}$$

Dans la règle de Wolfe, le pas est déterminée de manière à satisfaire les deux inégalités suivantes, appelées conditions de Wolfe :

$$\psi(\alpha) \leq \psi(0) + m_1\alpha\psi'(0) \tag{1.5}$$
$$\psi'(\alpha) \geq m_2\psi'(0). \tag{1.6}$$

où les constantes m_1 et m_2 sont choisies telles que

$$0 < m_1 < m_2 < 1.$$



La première inégalité n'est autre que la condition de décroissance linéaire, tandis que le rôle de (1.6) est d'empêcher le pas d'être trop petit. Cette règle de recherche linéaire est bien adaptée aux algorithmes de quasi-Newton. Le pas déterminé par cette règle est appelé pas de Wolfe.

Proposition 1.2 *[6] Si d est une direction de descente de f en u, si $\psi : \mathbb{R}_+ \longmapsto \mathbb{R}$ est définie par (1.4) est dérivable est bornée inférieurement et si $0 < m_1 < m_2 < 1$, alors il existe un pas α_k vérifiant les deux conditions de Wolfe .*

Algorithme de recherche pour les conditions de Wolfe :

Dans cette section on veut décrire une procédure qui nous permet de trouver le pas pour les conditions fortes de Wolfe

$$\begin{aligned} \psi(\alpha) &\leq \psi(0) + m_1\alpha\psi'(0) \\ |\psi'(\alpha)| &\leq |m_2\psi'(0)| . \end{aligned}$$

D'abord on doit supposer que d est une direction de descente et que f est bornée le long de la direction d. l'algorithme qui nous permet de trouver, est divisé en deux phase. Dans la première phase on estime α_1, et on l'incrémente jusqu'à l'obtention une longueur acceptable ou intervalle qui contient la longueur désirée. La deuxième phase est caractérisée par la fonction *zoom*, qui diminue la taille de l'intervalle jusqu'à convergence vers un α_k qui vérifie les conditions de Wolfe.

Algorithm 1.1 *(algorithme de recherche linéaire inexacte)*
posons $\alpha_0 \longleftarrow 0$, *choix de* $\alpha_{\max}$ *et* $\alpha_1 \in]\, 0, \alpha_{\max}[\, \alpha_1 > 0$
$i \longleftarrow 1$
répéter
evaluer $\psi(\alpha_i)$
si $\psi(\alpha_i) > \psi(0) + m_1\alpha_i\psi'(0)$ *ou* $[\psi(\alpha_i) \geq \psi(\alpha_{i-1}$ *et* $i > 1)]$
$\alpha_* \longleftarrow zoom(\alpha_{i-1}, \alpha_i)$ *et stop*
evaluer $\psi'(\alpha_i)$

si $|\psi'(\alpha_i)| \leq -m_2\psi'(0)$
  posons $\alpha_* \longleftarrow \alpha_i$ *et stop*
si $\psi'(\alpha_i) \geq 0$
  posons $\alpha_* \longleftarrow zoom(\alpha_i, \alpha_{i-1})$ *et stop*
choix de $\alpha_i \in (\alpha_i, \alpha_{\max})$
  $i \longleftarrow i + 1$
fin de répéter

Cette algorithme utilise le fait que l'intervalle $[\alpha_{i-1}, \alpha_i]$ contient des pas qui satisfaisont les conditions fortes de Wolfe si une des trois conditions est satisfaite :

1. α_i ne satisfait pas la première condition de Wolfe.

2. $\psi(\alpha_i) \geq \psi(\alpha_{i-1})$.

3. $\psi'(\alpha_i) \geq 0$.

Le dernier passage de l'algorithme nous permet de trouver la prochaine valeur α_{i+1}. La seule chose importante est que la suite des pas $\{\alpha_i\}$ augmente suffisamment vite pour arriver à $\alpha_{\max}$ en un nombre raisonnable d'opérations.

Maintenant on veut voir plus en détail la fonction *zoom* telle que chaque évaluation possède la forme $zoom\,(\alpha_{lo}, \alpha_{hi})$ où :

1. $[\alpha_{lo}, \alpha_{hi}]$ ou $[\alpha_{hi}, \alpha_{lo}]$ contient une longueur du pas qui satisfait la condition forte de Wolfe.
2. $\psi(\alpha_{lo}) < \psi(\alpha_{hi})$.
3. α_{hi}est choisi tel que $\psi'(\alpha_{lo})(\alpha_{hi} - \alpha_{lo}) < 0$.

Chaque itération de la forme *zoom* produit un α_j entre α_{hi} et α_{lo} et après il remplace un des deux points extrêmes de l'intervalle par α_j : si α_j satisfait la condition de diminution suffisante et possède une valeur de la fonction inférieur à α_{lo}, alors on remplace α_{lo} par α_j.

Algorithm 1.2 *(zoom*$(\alpha_{hi}, \alpha_{lo})$ *)*
chiisir α_j *entre* α_{lo} *et* α_{hi}
evaluer $\psi(\alpha_j)$
si $\psi(\alpha_j) > \psi(0) + m_1\alpha_j\psi'(0)$ *ou* $\psi(\alpha_j) \geq \psi(\alpha_{lo})$
  $\alpha_{hi} \longleftarrow \alpha_j$
sinon
evaluer $\psi'(\alpha_j)$
si $|\psi'(\alpha_j)| \leq -m_2\psi'(0)$
  $\alpha_* \longleftarrow \alpha_j$ *et stop*
si $\psi'(\alpha_j)(\alpha_{hi} - \alpha_{lo}) \geq 0$
  $\alpha_{hi} \longleftarrow \alpha_{lo}$
  $\alpha_{lo} \longleftarrow \alpha_j$
fin de répéter

Chapitre 2

Méthode de Newton pour l'optimisation

La méthode de Newton n'est pas une méthode d'optimisation à proprement parler. C'est en réalité une méthode utilisée pour résoudre des équations non linéaires de la forme $f(u) = 0$ où f est une fonction de $\mathbb{R}^n$ dans $\mathbb{R}$. Cependant, nous avons vu dans la section (1.1) (Théorème 1.1) qu'une condition nécessaire d'optimalité est $\nabla f(u^*) = 0$. Ceci est une équation non linéaire (ou plutôt un système d'équations non linéaires) où nous allons utiliser la méthode de Newton pour la résoudre.

2.1 Description de la méthode

La méthode de Newton pour l'optimisation consiste à remplacer, au voisinage du point courant u_k, la fonction objective f par son approximation quadratique

$$q(u) = f(u_k) + \bigtriangledown f^T(u_k)(u - u_k) + \frac{1}{2}(u - u_k)^T \bigtriangledown^2 f(u_k)(u - u_k),$$

où

$$\bigtriangledown^2 f(u) = \left(\frac{\partial^2 f}{\partial u_i \partial u_j}(u)\right)_{1 \leq i \leq n,\ 1 \leq j \leq n}$$

désigne le Hessien de f en u.
Le point u_{k+1} minimum de q est défini par la condition d'optimalité

$$\bigtriangledown q(u_{k+1}) = 0,$$

ceci équivalent à

$$\bigtriangledown f(u_k)^T + (u_{k+1} - u_k)^T \bigtriangledown^2 f(u_k) = 0.$$

La suite construite par la méthode de Newton est alors définie par

$$u_{k+1} = u_k - [\bigtriangledown^2 f(u_k)]^{-1} \bigtriangledown f(u_k),$$

sous réserve que les matrices Hessiennes soient inversibles. Il s'agit donc de résoudre à chaque itération le systéme linéaire obtenu par linéarisation de f en u_k , que l'on appelle l'équation de Newton. On résume l'algorithme de Newton sous comme suit :

Algorithm 2.1 *1-**Initialisation***

$k = 0$, choix de u_0 dans un voisinage de u^*(u^*est le minimum de f) et une précision $\varepsilon > 0$.

*2) **l'itération principale k***

si $\|\bigtriangledown f(u_k)\| < \varepsilon$ stop et $u^* = u_k$ sinon

3)

$$u_{k+1} = u_k - [\bigtriangledown^2 f(u_k)]^{-1} \bigtriangledown f(u_k).$$

4) on pose $k = k + 1$ et on retourne à 2.

Théorème 2.1 (convergence locale de l'algorithme de Newton en optimisation) *[3] Soit f une fonction de classe C^3. Supposons que u^* est un minimum locale de f vérifiant la condition d' optimalité(1.2) et que $\bigtriangledown^2 f(u^*)$ est définie positive. Alors pour u_0 suffisamment poche de u, la suite de Newton est bien définie, elle converge vers u^*, et cette convergence est quadratique, c'est- à- dire*

$$\|u_{k+1} - u_k\| \leq o \|u_k - u^*\|^2 .$$

Preuve. La démonstration repose essentiellement sur l'utilisation d'un développement de Taylor pour aboutir à l'inégalité (2.1) ci- dessous, à partir de laquelle tout est déduit. L'application $u \longmapsto \det(\bigtriangledown^2 f(u))$ étant continue, $\bigtriangledown^2 f(u)$ est inversible pour tout u dans une boule ouverte U centrée en u^*. Définissons alors sur$U$ l'application

$$N : u \longmapsto u - (\bigtriangledown^2 f(u^*))^{-1} \bigtriangledown f(u^*).$$

Cette application vérifie deux propriétés importantes pour notre étude :

$$u_{k+1} = N(u_k) \qquad \text{et} \qquad u^* = N(u^*).$$

Par définitions de N, nous avons une relation entre $N(u)$ et u

$$\bigtriangledown^2 f(u)(N(u) - u) + \bigtriangledown f(u) = 0.$$

Ecrivons le développement de Taylor de la fonction $u \longmapsto \bigtriangledown^2 f(u)$ de classe C^1 entre un point $u \in U$ et u^*

$$\bigtriangledown f(u) + \bigtriangledown^2 f(u)(u - u^*) + R(u, u^*) = 0,$$

où $R(u, u_*)$ est le reste de Taylor associé. En soustrayant les deux égalités, on obtient

$$\bigtriangledown^2 f(u)(N(u) - u^*) = R(u, u^*),$$

d'où

$$\begin{aligned} \|(N(u) - u^*)\| &= \left\|(\bigtriangledown^2 f(u))^{-1} R(u, u^*)\right\| \\ &\leq \left\|(\bigtriangledown^2 f(u))^{-1}\right\| \|R(u, u^*)\| . \end{aligned}$$

Les fonctions $u \longmapsto (\bigtriangledown^2 f(u))^{-1}$ et $u \longmapsto \bigtriangledown^3 f(u)$ étant continue, elles sont bornées sur tout compact de U. fixons K une boule fermées contenant u^* et incluse dans U et notons

$$c_1 = \max_{u \in K} \left\| (\bigtriangledown^2 f(u))^{-1} \right\| \qquad \text{et} \qquad c_2 = \max_{u \in K} \left\| \bigtriangledown^3 f(u) \right\|$$

Ainsi que $c = c_1 c_2$. On obtient donc la majoration

$$\|(N(u) - u^*)\| \leq c \|u - u^*\|^2 . \tag{2.1}$$

Cette inégalité garantit tout d'abord que la suite de Newton est bien définie au voisinage de u^*. Pour voir ceci, on peut choisir un réel $\alpha > 0$ vérifiant $c\alpha < 1$, et assez petit pour que la boule fermée $B(u^*, \alpha)$ de centre u et de rayon α soit incluse dans K. Toutes ces précautions sont prises pour que la fonction N envoie la boule $B(u^*, \alpha)$ dans elle- même. En effet, nous avons

$$\|(N(u) - u^*)\| \leq c \|u - u^*\| \|u - u^*\| \leq c\alpha \|u - u^*\| .$$

Puisque $c\alpha < 1$, ceci nous garantit que si $u \in B(u^*, \alpha)$ alors $N(u) \in B(u^*, \alpha)$. Ainsi, comme $u_{k+1} = N(u_k)$, on déduit par récurrence que si $u_0 \in B(u^*, \alpha)$ alors la suite de Newton est bien définie et $u_k \in B(u^*, \alpha)$ pour tout k. Pour la fin de preuve, nous nous plaçons dans cette situation.

Convergence : l'inégalité (2.1) permet alors de montrer que $(u_k)_{k \in N}$ converge vers u^* . On a en effet

$$\|u_{k+1} - u^*\| \leq c \|u_k - u^*\|^2 ,$$

ce qui entraine

$$c \|u_{k+1} - u^*\| \leq (c \|u_k - u^*\|)^2.$$

Par récurrence, on obtient donc

$$c \|u_k - u^*\| \leq (c \|u_0 - u^*\|)^{2^k},$$

ce qui donne finalement

$$\|u_k - u^*\| \leq \frac{(c\alpha)^{2^k}}{c}.$$

Comme $c\alpha < 1$, ceci implique que $\|u_k - u^*\|$ tend (très vite) vers 0. ■

2.2 Avantages et inconvénients

Le gros avantage d'une méthode de Newton est bien connu : elle converge dans tout voisinage d'un minimum local avec une vitesse de convergence quadratique. Ainsi si la fonction objectif est quadratique, la convergence est obtenue en une seule itération (exemple 2.1).

L'inconvénient majeur de la méthode est sa sensibilité au choix du point de départ u_0 : Si ce point est mal choisi ("trop loin" de la solution) la méthode peut soit diverger, soit converger vers une autre solution (exemple 2.2). Il peut paraître surprenant de vouloir choisir le point de départ "assez prés" de u^* puisqu'on ne connaît pas u^* ! !. En pratique on essaie de s'approcher de u par une méthode de type gradient par exemple, puis on applique la méthode de Newton.

La méthode de Newton possède néanmoins plusieurs inconvénients, dus essentiellement au fait qu'elle ne possède pas la propriété de convergence globale.

Pour remédier à ces difficultés, une première modification consiste à introduire une recherche linéaire le long de la direction de recherche :

$$d_k = -[\bigtriangledown^2 f(u_k)]^{-1} \bigtriangledown f(u_k).$$

La formule itérative de Newton devient

$$u_{k+1} = u_k - \alpha_k[\bigtriangledown^2 f(u_k)]^{-1} \bigtriangledown f(u_k). \tag{2.2}$$

Si $\bigtriangledown^2 f(u_k)$ est définie positive, alors d_k est une direction de descente. En effet

$$\bigtriangledown f(u_k)^T d_k = - \bigtriangledown f(u_k)^T [\bigtriangledown^2 f(u_k)]^{-1} \bigtriangledown f(u_k) = - \|\bigtriangledown f(u_k)\|^2_{\bigtriangledown^2 f(u_k)^{-1}} < 0,$$

avec $\|u\|^2_{S_{k+1}} = u^T S_{k+1} u$.

Une deuxième difficulté peut apparaître lorsque le Hessien n'est pas défini positif. Dans ce cas, la direction de déplacement n'est pas une direction de descente et la convergence globale n'est pas assurée. Il reste cependant un problème de taille, c'est le calcul de la matrice Hessienne $\bigtriangledown^2 f(u_k)$ à chaque itération . Tout ceci, lorsque c'est possible, coûte beaucoup trop cher, d'où le développement des méthodes de quasi-Newton présentées au chapitre suivant.

Exemple 2.1 *On considère le problème sans contraintes suivant :*

$$\begin{cases} \min f(u) = 4(u^1)^2 + 4(u^2)^2 - 4u^1u^2 - 12, \\ u \in \mathbb{R}^2 \end{cases}$$

avec

$$\nabla f(u) = \begin{pmatrix} 8u^1 - 4u^2 \\ -4u^1 + 8u^2 \end{pmatrix} \quad et \quad H = \begin{pmatrix} 8 & -4 \\ -4 & 8 \end{pmatrix}$$

Sachant résoudre ce problème en utilisant la condition d'optimalité (1.2)*, la solution est :*

$$u^{*^T} = (0, 0)$$

On veut maintenant résoudre ce système en utilisant la méthode de Newton avec un point de départ $u_0^T = (-\frac{1}{2}, 1)$ *et une précision* $\epsilon = 10^{-4}$*, dans le sens où on s'arrête quand* $\|\nabla f(u_k)\| < \epsilon$.

De la première itération, on obtient :

$$\begin{aligned} u_1 &= u_0 - H^{-1}\nabla f_0 \\ &= \begin{pmatrix} -\frac{1}{2} \\ 1 \end{pmatrix} - \begin{pmatrix} 0.1667 & 0.0833 \\ 0.0833 & 0.1667 \end{pmatrix} \begin{pmatrix} -8 \\ -8 \end{pmatrix} \\ &= 10^{-15} \begin{pmatrix} 0 \\ 0.1110 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \end{aligned}$$

Et $\|\nabla f(u_1)\| = 9.9301\ 10^{-16} < \epsilon$. *donc* u_1 *est le minimum de* f.

Exemple 2.2 *On veut minimiser la fonction de Rosenbrock définie par*

$$f(u) = 100(u^2 - (u^1)^2) + (1 - u^1)^2,$$

avec

$$\nabla f(u) = \begin{pmatrix} -400u^1(u^2 - (u^1)^2) - 2(1 - u^1) \\ 200(u^2 - (u^1)^2) \end{pmatrix} \quad et \quad H = \begin{pmatrix} -400(u^2 - 3(u^1)^2 + 2) & -400u^1 \\ -400u^1 & 200 \end{pmatrix}$$

la solution optimale est $u_* = (1, 1)$*. En utilisant la méthode de Newton avec* $u_0^T = (-2, 2)$ *et* $\epsilon = 10^{-4}$*, on obteint :*

Itération 1 :

$$\begin{aligned} u_1 &= u_0^T - [\nabla^2 f(u_0)]^{-1} \nabla f(u_0) \\ &= \begin{pmatrix} -2 \\ 2 \end{pmatrix} - \begin{pmatrix} 0.0010 & -0.0040 \\ -0.0040 & 0.0210 \end{pmatrix} \begin{pmatrix} -1606 \\ -400 \end{pmatrix} = \begin{pmatrix} -1.9940 \\ 3.9760 \end{pmatrix}. \end{aligned}$$

En procédant de la même manière, nous obtenons

$$\begin{aligned} u_2 &= u_1 - [\nabla^2 f(u_1)]^{-1} \nabla f(u_1) = \begin{pmatrix} -1.9641 \\ 3.8566 \end{pmatrix} \\ u_3 &= \begin{pmatrix} -1.9345 \\ 3.7413 \end{pmatrix} \\ u_4 &= \begin{pmatrix} -1.9345 \\ 3.7413 \end{pmatrix}. \end{aligned}$$

On voit qu'à partir de la troisième itération on trouve le même résultat et il est clair que cette solution obtenue par la méthode de Newton est trop loin de la solution exacte, de plus $\|\nabla f(u_4)\| = 6.3858 > \epsilon$ *, donc* u_4 *ne peut pas être le minimum de* f*.*

Chapitre 3

Méthodes de quasi-Newton

Comme son nom l'indique, le fondement de ces méthodes est la méthode de Newton classique décrite au chapitre précédent. L'idée de ces méthodes est de proposer une alternative à la méthode coûteuse de Newton pour la recherche du minimum d'une fonction f.

3.1 Description de la méthode

Avec les méthodes Quasi-Newton, on veut généraliser la formule itérative de Newton (2.2). Le principe des méthodes de résolution de type Quasi-Newton est de générer une séquence de matrices symétriques définies positives qui soient des approximations de la matrice Hessienne réelle ou de son inverse. Désignons par S_k la suite d'approximations de l'inverse de la matrice Hessienne $[\bigtriangledown^2 f(u_k)]^{-1}$ et par B_k la suite des estimations de la matrice Hessienne $\bigtriangledown^2 f(u_k)$.

On recherche une méthode telle que, dans le cas d'un problème quadratique, la matrice converge vers la valeur exacte des dérivées secondes (constantes dans ce cas), de sorte qu'en fin de convergence, on retrouve une convergence de type Newton. Si l'on applique la méthode à une fonction quelconque, peut être considéré, à chaque instant, comme une approximation (symétrique définie positive) du Hessien. L'itération principale est donnée comme suit :

$$u_{k+1} = u_k - \alpha_k S_k \bigtriangledown f(u_k).$$

Le pas de déplacement α_k est choisi afin d'avoir $f(u_{k+1}) < f(u_k)$, c'est-à-dire

$$f(u_{k+1}) = \min_{\alpha \in \mathbb{R}} f(u_k - \alpha_k S_k \bigtriangledown f(u_k)).$$

S_k est une matrice symétrique définie positive, raisonnablement proche de de $[\bigtriangledown^2 f(u_k)]^{-1}$ qu'on peut calculer à laide d'une formule simple du type :

$$S_{k+1} = S_k + C_k, \tag{3.1}$$

dans cette formule, C_k est une matrice de correction qui sera choisie de sorte que S_{k+1} satisfasse les conditions de quasi-Newton :

Définition 3.1 ***(conditions de quasi-Newton)***

(1) *On appelle première condition de quasi Newton la condition :*

$$\nabla f(u_k)^T d_k < 0.$$

qui assure que d_k est une direction de descente.

(2) *En posant $\delta_k = u_{k+1} - u_k$ et $y_k = \nabla f(u_{k+1}) - \nabla f(u_k)$, la deuxième condition de quasi-Newton pour S_{k+1} s'écrit :*

$$y_k = B_{k+1}\delta_k,$$

ou de manière équivalente

$$S_{k+1}y_k = \delta_k. \tag{3.2}$$

la deuxième condition de quasi-Newton s'obtient par un développement de $\bigtriangledown f(u)$ au voisinage du point u (prenons $f \in C^2$) :

$$\bigtriangledown f(u) \simeq \bigtriangledown f(u_k) + \bigtriangledown^2 f(u_k)\,(u - u_k)\,.$$

ou encore

$$\bigtriangledown^2 f(u_k)^{-1}\,(\bigtriangledown f(u) - \bigtriangledown f(u_k)) \simeq u - u_k.$$

Ses approximations sont évidement exactes si f est quadratique. En particulier, avec $u = u_{k+1}$ et si S_k était une bonne approximation de $\bigtriangledown^2 f(u_k)^{-1}$, on devrait avoir

$$S_k\,(\bigtriangledown f(u_{k+1}) - \bigtriangledown f(u_k)) = u_{k+1} - u_k.$$

Comme u_{k+1} est calculé après S_k, il est peu probable que cette équation soit satisfaite, même approximativement. En revanche, on peut toujours imposer que S_{k+1} satisfasse cette condition exactement, d'où la deuxième condition de quasi-Newton.

Proposition 3.1 *Si la matrice S_{k+1} est symétrique définie positive, alors la première condition de quasi-Newton est réalisée par*

$$d_k = -S_{k+1} \bigtriangledown f(u_k). \tag{3.3}$$

Preuve. On peut facilement démontrer que d_k est une direction de descente. En effet,

$$\nabla f(u_k)^T d_k = -\nabla f(u_k)^T S_{k+1} \bigtriangledown f(u_k) = -\left\|\bigtriangledown f(u_k)\right\|^2_{S_{k+1}} < 0.$$

■

Remarque 3.1 *Les méthodes de quasi-Newton, comme la méthode de Newton d'ailleurs, ne sont pas toujours des méthodes de descente. Aussi faudra-t-il réinitialiser S_k régulièrement.*

Toutes les méthodes d'adaptation de S_k devront respecter les conditions de quasi-Newton.

Différentes méthodes permettent d'effectuer une mise à jour de type Quasi-Newton (3.1). Parmis toutes celles-ci, les formules les plus connues sont :

- Mise à jour de rang un.
- Mise à jour de Davidon-Fletcher-Powell (DFP).
- Mise à jour de Broyden-Fletcher-Goldfard-Shanno (BFGS)

3.2 Méthode de rang un

La méthode de rang un doit son nom du fait que la correction de la matrice C_k dans l'équation (3.1) a de rang un. Cette correction a été proposé indépendamment par Broyden, Davidon, Fiacco et McCormick, Murtagh et Sargent et Wolfe. C'est La manière la plus simple pour mettre à jour S_k.

D'abord on énonce le résultat suivant :

Proposition 3.2 *Soit $v \in R^n$,$v \neq 0$. Alors la matrice $V = vv^T$ est symétrique définie positive de plus elle est de rang un.*

Preuve. Soit $v \in R^n$,$v \neq 0$. La matrice $V = vv^T$es symétrique car

$$V^T = (vv^T)^T = (v^T)^T v^T = vv^T = V.$$

De plus, V est définie positive, en effet
soit $x \in R^n_*$, on a

$$\begin{aligned} x^T V x &= x^T vv^T x = x^T v (x^T v)^T \\ &= \left\| x^T v \right\|^2 > 0. \end{aligned}$$

Montrons maintenant que V est de rang un.
On a

$$V = vv^T = \begin{pmatrix} v_1 \\ v_2 \\ . \\ . \\ . \\ v_n \end{pmatrix} \begin{pmatrix} v_1 & v_2 & ... & v_n \end{pmatrix} = \begin{pmatrix} v_1^2 & v_1 v_2 & & v_1 v_2 \\ v_2 v_1 & & & v_2 v_n \\ . & & & . \\ . & & . & . \\ . & & . & . \\ v_n v_1 & & & v_{n^2} \end{pmatrix}$$

$$= \begin{pmatrix} v_1 \begin{pmatrix} v_1 \\ v_2 \\ . \\ . \\ . \\ v_n \end{pmatrix} v_2 \begin{pmatrix} v_1 \\ v_2 \\ . \\ . \\ . \\ v_n \end{pmatrix} ... v_n \begin{pmatrix} v_1 \\ v_2 \\ . \\ . \\ . \\ v_n \end{pmatrix} \end{pmatrix}$$

On remarque que toutes les colonnes de la matrice V s'expriment comme combinaison linéaire d'un seul vecteur $\begin{pmatrix} v_1 & v_2 & ... & v_n \end{pmatrix}^T$donc la matrice V est de rang un. ■

3.2.1 Description de la méthode

Supposons que la condition de quasi-Newton (3.2) est Vérifiée et soit la formule de mise à jour de rang un suivante

$$S_{k+1} = S_k + \beta_k v_k v_k^T, \tag{3.4}$$

avec v_k une matrice une matrice unicolonne et β_k une constante, qui préserve la symétrie de la suite (S_k). Le terme $\beta_k v_k v_k^t$ est une matrice symétrique de rang un.

D'aprés les équations (3.2) et (3.4), on obtient

$$\delta_k = S_k y_k + \beta_k v_k v_k^T y_k, \tag{3.5}$$

et donc

$$\begin{aligned} y_k^T(\delta_k - S_k y_k) &= \beta_k y_k v_k v_k^T y_k \\ &= \beta_k (y_k^T v_k)^2 \end{aligned} \tag{3.6}$$

Alternativement, à partir de l'équation (3.5)

$$\begin{aligned} (\delta_k - S_k y_k) &= \beta_k v_k v_k^T y_k = \beta_{k(} v_k^T y_k) v_k \\ (\delta_k - S_k y_k)^T &= \beta_k y_k^T v_k v_k^T = \beta_{k(} v_k^T y_k) v_k^T, \end{aligned}$$

puisque $v_k^t y_k$ est un scalaire

$$(\delta_k - S_k y_k)(\delta_k - S_k y_k)^T = \beta_{k(} v_k^T y_k)^2 \beta_k v_k v_k^T, \tag{3.7}$$

et d'aprés les équations (3.6) et (3.7), on a :

$$\begin{aligned} \beta_k v_k v_k^T &= \frac{(\delta_k - S_k y_k)(\delta_k - S_k y_k)^T}{\beta_{k(} v_k^T y_k)^2} \\ &= \frac{(\delta_k - S_k y_k)(\delta_k - S_k y_k)^T}{y_k^T(\delta_k - S_k y_k)} \end{aligned}$$

En remplaçant cette formule dans l'équation (3.4), on obtient la mise à jour pour S_{k+1} de rang un :

$$S_{k+1} = S_k + \frac{(\delta_k - S_k y_k)(\delta_k - S_k y_k)^T}{y_k^T(\delta_k - S_k y_k)}. \tag{3.8}$$

Pour un problème quadratique, la méthode de rang un converge en n itérations.

Théorème 3.1 *[1] Si H est le Hessien d'un problème convexe quadratique tel que*

$$y_i = H\delta_i \quad \textit{pour } 0 \leq i \leq k \tag{3.9}$$

et si $\delta_0, \delta_1, ..., \delta_k$ sont des vecteurs indépendants, alors pour toute matrice initiale S_0

$$\delta_i = S_{k+1} y_i, \quad \textit{pour } 0 \leq i \leq k \tag{3.10}$$

où

$$S_{i+1} = S_i + \frac{(\delta_i - S_i y_i)(\delta_i - S_i y_i)^T}{y_i^T(\delta_i - S_i y_i)}.$$

Preuve. La preuve se faite par récurrence. L'équation (3.10) est vérifiée pour $k = 1$

$$\delta_i = S_1 y_i, \quad \textit{pour} 0 \leq i \leq k-1$$

Supposons

$$\delta_i = S_k y_i, \quad pour 0 \leq i \leq k-1 \tag{3.11}$$

et montrons que

$$\delta_i = S_{k+1} y_i, \quad pour 0 \leq i \leq k$$

si $0 \leq i \leq k-1$, l'équation (3.8) devient

$$S_{k+1} y_i = S_k y_i + \zeta_k (\delta_k - S_k y_k)^T y_i$$

où

$$\zeta_k = S_k - \frac{S_k y_k}{y_k^T (\delta_k - S_k y_k)}$$

comme S_k est symétrique, on peut écrire

$$S_{k+1} y_i = S_k y_i + \zeta_k (\delta_k^T S_k y_i - y_k^T S_k y_i)$$

et si (3.11) est vérifiée, ensuite

$$S_{k+1} y_i = \delta_i + \zeta_k (\delta_k^T y_i - y_k^T \delta_i) \tag{3.12}$$

pour $0 \leq i \leq k$

$$y_i = H \delta_i$$

et

$$y_k^T = \delta_k^T H$$

donc, pour $0 \leq i < k-1$, ona

$$\delta_k^T y_i - y_k^T \delta_i = \delta_k^T H \delta_i - \delta_k^T H \delta_i = 0,$$

à partir de l'équation (3.12)

$$\delta_i = S_{k+1} y_i, \quad \text{pour } 0 \leq i \leq k-1 \tag{3.13}$$

et avec condition de quasi-Newton

$$\delta_k = S_{k+1} y_k, \quad \text{pour } 0 \leq i \leq k$$

on établit le résultat voulu au rang $k+1$. ■
De l'équation (3.8), on peut écrire

$$\begin{aligned} y_i^T S_{k+1} y_i &= y_i^T S_k y_i + \frac{y_i^T (\delta_k - S_k y_k)(\delta_k - S_k y_k)^T y_i}{y_k^T (\delta_k - S_k y_k)} \\ &= y_i^T S_k y_i + \frac{(y_i^T \delta_k - y_k^T S_k y_i)(\delta_k^T y_i - y_k^T S_k y_i)}{y_k^T (\delta_k - S_k y_k)} \\ &= y_i^T S_k y_i + \frac{(y_i^T \delta_k - y_k^T S_k y_i)^2}{y_k^T (\delta_k - S_k y_k)}, \end{aligned}$$

donc, si $S_k$est définie positive, la condition suffisante pour que S_{k+1} soit définie positive est

$$y_k^T (\delta_k - S_k y_k) > 0.$$

Algorithm 3.1 (Algorithme de rang un)

1-Choisir u_0, S_0 symétrique définie positif quelconque et une précision $\varepsilon > 0$.
*2- **l'itération principale** :*
*a) **Critère d'arrêt :** si $\|\bigtriangledown f(u_k)\| < \varepsilon$ stop*

$$u^* = u_k$$

sinon
*b) **l'itération k :***
-Calculer la direction de déplacement

$$d_k = -S_k \bigtriangledown f(u_k).$$

-Déterminer le pas optimal α_k par recherche linéaire exacte et poser

$$u_{k+1} = u_k + \alpha_k d_k,$$

- Poser $\delta_k = \alpha_k d_k$ et $y_k = \bigtriangledown f(u_{k+1}) - \bigtriangledown f(u_k)$ puis calculer

$$S_{k+1} = S_k + \frac{(\delta_k - S_k y_k)(\delta_k - S_k y_k)^T}{y_k^T(\delta_k - S_k y_k)} \ .$$

- Faire $k = k + 1$ et retourner en 2.

3.2.2 Inconvénients

L' inconvénient majeur de cette méthode est la non positivité de S_{k+1} même si S_k l'est. Toutefois, si ce problème se produit (généralement pour des fonctions non quadratiques) fréquemment la possibilité existe que S_{k+1} ne peut pas converger vers H^{-1}. De plus, le dénominateur $y_k^t(\delta_k - S_k y_k)$ de la formule de correction (3.8) peut devenir négatif ou trop petit, rendant la procédure instable.

Cependant cette méthode a été introduite pour la bonne compréhension de la méthode BFGS du chapitre suivant où S_k est garanti d'être définie positive pour tout k, à condition que la recherche linéaire soit exacte.

Exemple 3.1 *Soit*

$$f(u) = \frac{1}{2}u^T \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} u + 3.$$

le minimum exacte est $u^* = (0,0)^T$. Rappelons qu'on n'a pas obtenue le minimum de f par la méthode de Newton.

On va minimiser cette fonction en utilisant la méthode de rang un avec les données suivantes :

$$u_0 = (1,2), \ \ S_0 = I_2 \ \text{et} \ \epsilon = 0.0001.$$

Comme la fonction objective est quadratique

$$\alpha_k = \arg\min f(u_k + \alpha d_k) = -\frac{\nabla f(u_k)^T d_k}{d_k H d_k}.$$

Donc

$$\begin{aligned} u_1 &= u_0 + \alpha_0 d_0 \\ &= \begin{pmatrix} 1 & 2 \end{pmatrix}^T + \frac{2}{3}\begin{pmatrix} -2 & 2 \end{pmatrix}^T = \begin{pmatrix} -\frac{1}{3} & \frac{2}{3} \end{pmatrix}^T \end{aligned}$$

et ainsi

$$u_2 = \begin{pmatrix} -\frac{1}{3} & \frac{2}{3} \end{pmatrix}^T + 1.\begin{pmatrix} \frac{1}{3} & \frac{-2}{3} \end{pmatrix} = \begin{pmatrix} 0 & 0 \end{pmatrix}^T$$

Notons que $\|\nabla f(u_2)\| = 0 < \epsilon$, *et donc* $u_2 = u^*$.

Exemple 3.2 *On veut maintenant minimiser la fonction de Rosembrock vue dans l'exemple* (2.2) *en utilisant la méthode de rang un où le pas* α_k *est déterminé par la méthode de la sécante (car* f *n'est pas quadratique). On prend toujours* $u_0 = (-2, 2)^T$, $\epsilon = 10^{-4}$, *et on chosit* $S_0 = I_2$ *(* I_2 *:matrice identité* 2×2*). Rappelons que*

$$f(u) = 100(u_2 - u_1^2)^2 + (1 - u_1)^2,$$

avec

$$\nabla f(u) = \begin{pmatrix} -400u_1(u_2 - u_1^2) - 2(1 - u_1) \\ 200(u_2 - u_1^2) \end{pmatrix} \quad .$$

En ce qui concerne la méthode de la sécante, on prend $u_0 = 0$ *et* $u_1 = 10^{-3}$. *Comme* $S_0 = I_2$

$$d_0 = - \bigtriangledown f(u_0).$$

Nous calculons ensuite

$$\begin{aligned} u_1 &= u_0 + \alpha_0 d_0 \\ &= \begin{pmatrix} -2 \\ 2 \end{pmatrix} + (0.0013)\begin{pmatrix} 1606 \\ 400 \end{pmatrix} = \begin{pmatrix} 0.1326 \\ 0.1326 \end{pmatrix}. \end{aligned}$$

Comme $\|\bigtriangledown f(u_1)\| > \epsilon$, *on passe à la deuxième itération*

$$\begin{aligned} u_2 &= u_1 + \alpha_1 d_1 = u_1 - \alpha_1 S_1 \bigtriangledown f(u_1) \\ &= \begin{pmatrix} 0.1326 \\ 2.5311 \end{pmatrix} + (0.0010)\begin{pmatrix} 0.2748 & -0.4454 \\ -0.4454 & 0.7264 \end{pmatrix}\begin{pmatrix} 1606 \\ 400 \end{pmatrix} = \begin{pmatrix} 0.4004 \\ 2.0948 \end{pmatrix}. \end{aligned}$$

On continue la même procédure et à l'itération 7, *on obtient* $u_7 = (1, 1)^T = u^*$.

l'exemple suivant montre que si $S_k > 0$ et $y_k^T(\delta_k - S_k y_k) < 0$, alors S_{k+1} ne peut pas être définie positive.

Exemple 3.3

$$f(u) = \frac{(u^1)^4}{4} + \frac{(u^2)^2}{2} - u^1 u^2 + u^1 - u^2.$$

avec un point initial $u_0 = (0.59607, 0.59607)^T$ *et une matrice initiale définie positive*

$$S_0 = \begin{pmatrix} 0.94913 & 0.14318 \\ 0.14318 & 0.59702 \end{pmatrix}$$

.

Rappelons que $\delta_k = u_{k+1} - u_k$ *et* $y_k = \nabla f(u_{k+1}) - \nabla f(u_k)$ *, nous avons donc*

$$y_0^T(\delta_0 - S_0 y_0) = -0.03276,$$

et

$$S_1 = \begin{pmatrix} 0.94481 & 0.23324 \\ 0.23324 & -1.2788 \end{pmatrix}.$$

Il est facile de vérifier que S_1 *n'est pas définie positive (les valeurs propres sont* $\lambda_1 = 0,96901$ *et* $\lambda_2 = -1,3030$*).*

Chapitre 4

Méthode de Broyden-Fletcher-Goldfarb-Shanno (BFGS)

Elle tient son nom des initiales des mathématiciens C.G Broyden, R.Fletcher, D.Goldfard et D.F. shanno, qui l'ont découvert indépendamment à la fin des années 60. Nous abordons ce chapitre par la méthode DFP (Davidon-Fletcher-Powell) qui a été historiquement trouvée avant.

4.1 Méthode de Davidon-Fletcher-Powell

Cette méthode a été proposée par le physicien Davidon en 1959 et plus tard développée par les mathématiciens Fletcher et Powell en 1963. Tout d'abord elle est appliquée à une fonctionnelle quadratique, non seulement elle construit l'inverse du Hessien, mais en plus elle engendre des directions conjuguées. la mise à jour de S_{k+1} en fonction de S_k se fait en ajoutant cette fois deux matrice de rang 1. La procédure est la suivante :

4.1.1 Description de la méthode DFP

Soit la formule de mise à jour de rang deux suivante

$$S_{k+1} = S_k + av_k v_k^T + bw_k w_k^T.$$

En prenant $v_k = \delta_k$ et $w_k = Hy_k$, et a et b tels que $av_k^T y_k = 1$ et $bw_k^T y_k = -1$, on obtient la mise à jour DFP

$$S_{k+1} = S_k + \frac{\delta_k \delta_k^T}{\delta_k^T y_k} - \frac{S_k y_k^T y_k S_k}{y_k^T S_k y_k} \tag{4.1}$$

On peut démontrer la validité de cette formule, en multipliant les deux membre de l'équation par y_k

$$S_{k+1} y_k = S_k y_k + \frac{\delta_k \delta_k^T y_k}{\delta_k^T y_k} - \frac{S_k y_k^T y_k^T S_k y_k}{\delta_k^T y_k}.$$

En simplifiant les termes $\delta_k^T y_k$ et $y_k^T S_k y_k$, on obtient donc la deuxième condition de quasi-Newton

$$S_{k+1} y_k = \delta_k \tag{4.2}$$

Le résultat suivant explicite une propriété importante de cette méthode :

Théorème 4.1 *[1] A l'étape k, si S^k est symétrique définie positive et si la recherche linéaire est exacte (ou bien si $\delta_k^T y_k > 0$), alors la matrice S_{k+1} (4.1) est symétrique définie positive.*

Preuve. appliquons un vecteur x non nul à gauche et à droite de S_{k+1} dans l'équation (4.1)

$$x^T S_{k+1} x = x^T S_k x + \frac{x^T \delta_k \delta_k^T x}{\delta_k^T y_k} - \frac{x^T S_k y_k^T y_k S_k x}{y_k^T S_k \delta_k}. \tag{4.3}$$

Pour une matrice réelle symétrique S_k, on peut écrire

$$w^T S_k w = \Lambda,$$

où w est une matrice unitaire telle que

$$w^T w = w w^T = I_n,$$

et Λ est une matrice diagonale dont les éléments sont les valeurs propres de S_k. Nous pouvons donc écrire

$$\begin{aligned} S_k &= w \Lambda w^T = w \Lambda^{\frac{1}{2}} \Lambda^{\frac{1}{2}} w^T \\ &= (w \Lambda^{\frac{1}{2}} w^T)(w \Lambda^{\frac{1}{2}} w^T) \\ &= S_k^{\frac{1}{2}} S_k^{\frac{1}{2}}. \end{aligned}$$

Si nous posons

$$U = S_k^{\frac{1}{2}} x \quad \text{et} \quad V = S_k^{\frac{1}{2}} y_k,$$

alors l'équation (4.3) peut être exprimé comme

$$x^T S_{k+1} x = \frac{(U^T U)(V^T V) - (U^T V)^2}{V^T V} + \frac{(x^T \delta_k)^2}{\delta_k^T y_k}. \tag{4.4}$$

D'une part

$$\delta_k = u_{k+1} - u_k = \alpha_k d_k = \alpha_k S_k \nabla f_k, \tag{4.5}$$

où α_k est la valeur de α qui minimise $f(u_k + \alpha d_k)$ au point $u = u_{k+1}$. Depuis $d_k = -S_k \bigtriangledown f(u_k)$ est une direction de descente, nous avons $\alpha_k > 0$.
En plus,

$$\frac{f(u_k + \alpha d_k)}{d\alpha} \mid_{\alpha = \alpha_k} = \nabla f(u_k + \alpha_k d_k)^T d_k = \nabla f_{k+1}^T d_k = 0,$$

ainsi

$$\alpha_k \nabla f_{k+1}^T d_k = \nabla f_{k+1}^T \alpha_k d_k = \nabla f_{k+1}^T \delta_k = \delta_k^T \nabla f_{k+1} = 0.$$

D'autre part, On peut écrire

$$\delta_k^T y_k = \delta_k^T \nabla f_{k+1} - \delta_k^T \nabla f_k = -\delta_k^T \nabla f_k.$$

Maintenant à partir de l'équation (4.5), nous obtenons

$$\delta_k^T y_k - \delta_k^T \nabla f_k = -(-\alpha_k S_k \nabla f_k)^T \nabla f_k = \alpha_k \nabla f_k^T S_k \nabla f_k, \tag{4.6}$$

et l'équation (4.4) peut être exprimé comme

$$x^T S_{k+1} x = \frac{(U^T U)(V^T V) - (U^T V)^2}{V^T V} + \frac{(x^T \delta_k)^2}{\alpha_k \nabla f_k^T S_k \nabla f_k}, \tag{4.7}$$

comme

$$U^T U = \|U\|^2, \; V^t V = \|V\|^2 \quad \text{et} \quad U^T V = \|U\| \, \|V\| \cos\theta,$$

où θ est l'angle formé entre U et V, donc l'équation (4.7) devient

$$x^T S_{k+1} x = \frac{\|U\|^2 \|V\|^2 - (\|U\| \, \|V\| \cos\theta)^2}{\|V\|^2} + \frac{(x^T \delta_k)^2}{\alpha_k \nabla f_k^T S_k \nabla f_k}$$

La valeur minimale de la partie droite de l'équation ci-dessus est atteinte lorsque $\theta = 0$. Dans ce cas, nous avons

$$x^T S_{k+1} x = \frac{(x^T \delta_k)^2}{\alpha_k \nabla f_k^T S_k \nabla f_k}. \tag{4.8}$$

Comme les vecteurs U et V ont la même direction, on peut écrire

$$U = S_k^{\frac{1}{2}} x = \beta V = \beta S_k^{\frac{1}{2}} y_k = S_k^{\frac{1}{2}} \beta y_k,$$

et ainsi

$$x = \beta y_k$$

où β est une constante positive. En remplaçant la valeur de x dans (4.8) et en utilisant l'équation (4.6), nous obtenons

$$x^T S_{k+1} x = \frac{(\beta y_k^T \delta_k)^2}{\alpha_k \nabla f_k^T S_k \nabla f_k} = \frac{(\beta \alpha_k \nabla f_k^T S_k \nabla f_k)^2}{\alpha_k \nabla f_k^T S_k \nabla f_k} = \alpha_k \beta^2 \nabla f_k^T S_k \nabla f_k$$

Maintenant pour tout $\theta \geq 0$, nous avons

$$x^T S_{k+1} x \geqslant \alpha_k \beta^2 \nabla f_k^T S_k \nabla f_k, \tag{4.9}$$

Par conséquent, si $u = u_k$ n'est pas le minimum u^* (i.e $\bigtriangledown f(u_k) \neq 0$) , nous avons

$$x^T S_{k+1} x > 0 \;\; \text{si } x \neq 0$$

d'où $S_{k+1} > 0$. ■

Il est important de noter que le résultat ci-dessus est valable pour tout $\alpha_k > 0$ pour lesquels

$$\delta_k^T y_k = \delta_k^T \bigtriangledown f(u_{k+1}) - \delta_k^T \bigtriangledown f(u_k) > 0, \tag{4.10}$$

même si $f(u)$ n'est pas minimisée au point u_{k+1}, comme on peut le vérifier en éliminant u dans l'équation (4.4), puis en utilisant l'inégalité dans l'équation (4.10).Par conséquent, si $\delta_k^T \bigtriangledown f(u_{k+1}) > \delta_k^T \bigtriangledown f(u_k)$, la définie positive de S_{k+1} peut être assuré, même dans le cas où la minimisation de $f(u_k + \alpha d_k)$ est inexacte.

Remarque 4.1 *La propriété $\delta^T y^k > 0$ est vérifiée également par des méthodes de recherche linéaire inexacte comme par exemple la règle de Wolfe.*

Cette méthode a un comportement remarquable dans le cas où f est une forme quadratique :

Théorème 4.2 *[1] a) Si la recherche linéaire utilisée dans l'algorithme DFP est exacte et f est une fonction convexe et quadratique de Hessien H, alors les vecteurs $\delta_0, \delta_1, ..., \delta_k$ forment une famille de vecteurs conjuguées par rapport à H, i. e,*

$$\delta_i^T H \delta_j = 0\,; \quad 0 \leq i < j \leq k. \tag{4.11}$$

b) Si

$$y_i = H\delta_i, \quad 0 \leq i \leq k \tag{4.12}$$

alors

$$\delta_i = S_{k+1} y_i, \quad 0 \leq i \leq k, \tag{4.13}$$

de plus

$$S_n = H^{-1}.$$

.

Preuve. La preuve se faite par récurrence. Nous supposons que

$$\delta_i^T H \delta_j = 0\,; \quad 0 \leq i < j \leq k-1, \tag{4.14}$$

$$\delta_i = S_k y_i, \quad 0 \leq i \leq k-1, \tag{4.15}$$

et montrer les équations (4.11) et (**??**) .

a) Elle est tout d'abort vraie pour $k = 0$. En effet ,

$$\begin{aligned} \delta_0^T \nabla f_1 &= (S_1 y_0)^T \nabla f_1 = (S_1 H \delta_0)^T \nabla f_1 = \delta_0^T H S_1 \nabla f_1 \\ &= -\frac{1}{\alpha_1} \delta_0^T H \delta_1, \end{aligned}$$

et si $f(x)$ minimisée exactement au point u_1,on obtient $\delta_0^T \nabla f_1 = 0$ et

$$\delta_i^T H \delta_j = 0, \quad 0 \leq i < j \leq 1.$$

D'aprés la deuxième condition de quasi-Newton (3.2), ona

$$\begin{aligned} \nabla f_k &= \nabla f_{k-1} + H\delta_{k-1}, \\ &= \nabla f_{k-2} + H\delta_{k-2} + H\delta_{k-1}, \\ &\;\;\vdots \\ &= \nabla f_{i+1} + H(\delta_{i+1} + \delta_{i+2} + ... + \delta_{k-1}), \end{aligned}$$

ainsi, pour $0 \leq i \leq k-1$

$$\delta_i^T \nabla f_k = \delta_i^T \nabla f_{i+1} + \delta_i^T H(\delta_{i+1} + \delta_{i+2} + ... + \delta_{k-1}). \tag{4.16}$$

Si la recherche utilisée est exacte pour détermine α_k dans l'algorithme de DFP et $f(u)$ est minimisée exactement au point u_{i+1},alors

$$\delta_i^T \nabla f_{i+1} = 0. \tag{4.17}$$

Maintenant, pour $0 \leq i \leq k-1$ l'équation (4.14) donne

$$\delta_i^T H(\delta_{i+1} + \delta_{i+2} + ... + \delta_{k-1}) = 0, \tag{4.18}$$

et d'aprés les équations (4.16)−(4.18), on obtient

$$\delta_i^T \nabla f_k = 0.$$

Alternativement, à partir de (4.15) et (4.12), on peut écrire

$$\delta_i^T \nabla f_k = (S_k y_i)^T \nabla f_k = (S_k H \delta_i)^T \nabla f_k = \delta_i^T H S_k \nabla f_k = 0.$$

On a

$$\delta_k = \alpha_k d_k = -\alpha_k S_k \nabla f_k,$$

donc

$$\delta_i^T \nabla f_k = -\frac{1}{\alpha_k} \delta_i^T H \delta_k = 0,$$

et comme $\alpha_k > 0$, on obtient

$$\delta_i^T H \delta_k = 0, \quad 0 \leq i \leq k-1 \tag{4.19}$$

on combine les équations (4.14) et (4.19),on trouve

$$\delta_i^T H \delta_j = 0\,; \quad 0 \leq i < j \leq k. \tag{4.20}$$

b) l'équation (**??**) est en particulier vérifiée pour $k = 1$. En utilisant la formule (4.15)

$$\delta_i = S_k y_i, \quad 0 \leq i \leq k-1,$$

On peut écrire d'une part

$$y_k^T \delta_i = y_k^T S_k y_i, \quad 0 \leq i \leq k-1, \tag{4.21}$$

D'autre part (4.12)

$$y_k^T \delta_i = \delta_k^T H \delta_i, \quad 0 \leq i \leq k-1, \tag{4.22}$$

les équations (4.21) et (4.22) devient

$$y_k^T \delta_i = y_k^T S_k y_i = \delta_i^T H \delta_i = 0, \qquad 0 \leq i \leq k-1. \tag{4.23}$$

On note

$$\delta_k^T = y_k^T S_{k+1} \text{ et } H\delta_i = y_i$$

l'équation (4.23) peut s'exprimer comme suit

$$y_k^T \delta_i = y_k^T S_k y_i = y_k^T S_{k+1} y_i = 0\,, \quad 0 \leq i \leq k-1,$$

donc

$$\delta_i = S_k y_i = S_{k+1} y_i, \quad 0 \leq i \leq k-1, \tag{4.24}$$

or, d'aprés l'équation (4.2)

$$\delta_k = S_{k+1} y_k, \tag{4.25}$$

on combine les équations (4.24) et (4.25), on obtient

$$\delta_i = S_{k+1} y_i, \quad 0 \leq i \leq k$$

ce qui démontre la propriété (**??**) au rang k.
Maintenant, posons $k = n-1$ dans l'équation (**??**). Donc

$$S_n H \delta_i = \delta_i pour i = 0, 1, ..., n-1.$$

On suppose S la matrice dont les colonnes sont $\delta_0, \delta_1, ..., \delta_{n-1}$, puis $S_n H S_i = S$. Puisque S est inversible, $S_n H = I$, ce qui est possible seulement si $S_n = H$. ■

Remarque 4.2 *Si S_0 est égale à l'identité, on obtient alors la méthode du gradient conjugué.*

Bien que similaire à la méthode de rang un, Comme l'indique le théorème (4.1), la méthode DFP à l'avantage de conserver la définie positivité de S_k lorsque S_0 est définie positive . Mais cette méthode est assez sensible à la précision de la recherche linéaire, elle ne sera généralement pas mise en œuvre sans un certain nombre de procédures Supplémentaire.

4.2 Description de la méthode BFGS

Pour obtenir la mise à jour BFGS, nous utilisons le concept de la dualité, ou de complémentarité. Plus précisément, la mise à jour BFGS pour B_k correspond à la mise à jour DFP pour S_k. Les formules liées de cette manière sont dits duaux ou complémentaires.

Rappelons que la mise à jour DFP

$$S_{k+1} = S_k + \frac{\delta_k \delta_k^T}{\delta_k^T y_k} - \frac{S_k y_k^T y_k S_k}{\delta_k^T S_k y_k}.$$

En utilisant le concept de complémentarité. On pose $S = B$ et $\delta = y$ et $y = \delta$, on peut facilement obtenir une équation de mise à jour pour l'approximation B_k du Hessien :

$$B_{k+1} = B_k + \frac{y_k y_k^T}{y_k^T \delta_k} - \frac{B_k \delta_k \delta_k^{Tt} B_k}{\delta_k^T B_k \delta_k}.$$

Maintenant, pour obtenir la mise à jour BFGS pour l'approximation de l'inverse de Hessien, on prend l'inverse de B_{k+1}

$$\begin{aligned} S_{k+1}^{BFGS} &= (B_{k+1})^{-1} \\ &= \left(B_k + \frac{y_k y_k^T}{y_k^T \delta_k} - \frac{B_k \delta_k \delta_k^T B_k}{\delta_k^T B_k \delta_k} \right)^{-1}. \end{aligned}$$

Pour calculer S_{k+1}^{BFGS} en inversant le côté droit de l'équation ci-dessus, nous appliquons la formule suivante pour une matrice inverse, connue sous le nom de la formule de Sherman-Morrison.

Lemme 4.1 ***(formule de Sherman-Morrison)*** *Soit A une matrice inversible. Soit u et v sont des vecteurs colonnes tel que $1 + v^T A^{-1} u \neq 0$. Donc, $A + uv^T$ est inversible, et*

$$(A + uv^T) = A^{-1} - \frac{(A^{-1}u)(v^T A^{-1})}{1 + v^T A^{-1} u}.$$

En appliquant le lemme ci-dessus à deux reprises pour B_{k+1} , on obtient la mise à jour BFGS :

$$S_{k+1} = S_k + (1 + \frac{y_k^T S_k y_k}{\delta_k^T y_k})\frac{\delta_k \delta_k^T}{\delta_k^T y_k} - \frac{\delta_k y_k S_k + S_k y_k \delta_k^T}{\delta_k^T y_k}. \tag{4.26}$$

Ou encore

$$S_{k+1} = (I - \frac{\delta_k y_k^T}{\delta_k^T y_k})\ S_k\ (I - \frac{y_k \delta_k^T}{\delta_k^T y_k}) + \frac{\delta_k \delta_k^T}{\delta_k^T y_k}. \tag{4.27}$$

Algorithm 4.1 (Algorithme de BFGS)

1-Choisir u_0, S_0 symétrique définie positif quelconque et une précision $\varepsilon > 0$.
2- ***l'itération principale*** *:*
a) ***Critère d'arrêt :*** *si $\|\bigtriangledown f(u_k)\| < \varepsilon$ stop*

$$u^* = u_k$$

sinon
b) ***l'itération k :***
-Calculer la direction de déplacement

$$d_k = -S_k \bigtriangledown f(u_k).$$

-Déterminer le pas optimal α_k par recherche linéaire exacte ou inexacte et poser

$$u_{k+1} = u_k + \alpha_k d_k,$$

- Poser $\delta_k = \alpha_k d_k$ et $y_k = \bigtriangledown f(u_{k+1}) - \bigtriangledown f(u_k)$ puis calculer

$$S_{k+1} = S_k + (1 + \frac{y_k^T S_k y_k}{\delta_k^T y_k})\frac{\delta_k \delta_k^T}{\delta_k^T y_k} - \frac{\delta_k y_k S_k + S_k y_k \delta_k^T}{\delta_k^T y_k}\ .$$

- Faire $k = k + 1$ et retourner en 2.

4.3 BFGS avec recherche linéaire exacte

La méthode BFGS possède les mêmes propriétés que la méthode DFP si la recherche linéaire utilisée est exacte, dans le cas quadratique, les directions engendrées sont conjuguées et si S_k est définie positive alors S_{k+1} l'est aussi. De plus, aprés n itérations on aura exactement $S_n = H^{-1}$.

Théorème 4.3 *A l'étape k, si S_k est symétrique définie positive et si la recherche linéaire est exacte (ou bien si $\delta^T y^k > 0$), alors la matrice S_{k+1} (4.6) est symétrique définie positive*

Preuve. Appliquons un vecteur x non nul à gauche et à droite de S_{k+1} dans l'équation (4.27).

$$\begin{aligned} x^T S_{k+1} x &= x^T (I - \frac{\delta_k y_k^T}{\delta_k^T y_k})\ S_k\ (I - \frac{y_k \delta_k^T}{\delta_k^T y_k}) x + \frac{x_k^T \delta_k \delta_k^T x}{\delta_k^T y_k} \\ &= w^T\ S_k\ w + \frac{\left\| x^T \delta_k \right\|^2}{\delta_k^T y_k} \geq 0 \end{aligned}$$

avec $w = (I - \frac{y_k \delta_k^T}{\delta_k^T y_k}) x$. Puisque $w^T\ S_k\ w \geq 0$ (car S_k esr définie positive) et $\delta_k^T y_k^k > 0$, donc $x^T S_{k+1} x \geq 0$. De plus, Si $w^T\ S_k\ w$ est nul, donc

$$0 \neq x = \frac{y_k \delta_k^T x}{\delta_k^T y_k} = y_k \frac{\delta_k^T x}{\delta_k^T y_k}.$$

on prend $x = \beta y_k$, $\beta \neq 0$. On déduit que $\delta_k^T x = \beta \delta_k^T y_k \neq 0$, ce qui prouve que S_{k+1} définie positive. ■

Exemple 4.1 *En utilisant la méthode de BFGS pour minimiser la fonction quadratique suivante*

$$f(u) = \frac{1}{2} u^T H u - u^T b + \log \pi.$$

$$H = \begin{pmatrix} 5 & -3 \\ -3 & 2 \end{pmatrix}, \qquad b = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

On choisit $S_0 = I_2$, $u = (0,0)^T$ *et* $\epsilon = 10^{-4}$. *on va vérifier que* $H_2 = S^{-1}$.

La fonction objective est quadratique, donc on peut utiliser la formule suivante pour calculer le pas α_k *:*

$$\alpha_k = -\frac{\nabla f(u_k)^T d_k}{d_k H d_k}.$$

Alors

$$\begin{aligned} u_1 &= u_0 + \alpha_0 d_0 = \begin{pmatrix} 0 \\ \frac{1}{2} \end{pmatrix}. \\ u_2 &= u_1 + \alpha_1 d_1 = \begin{pmatrix} 3 \\ 5 \end{pmatrix}. \end{aligned}$$

Comme la fonction f *est quadratique,* u_2 *est le minimum. Notons que* $\nabla f(u_2) = 0_{\mathbb{R}^2}$.

On a

$$\begin{aligned} H_2 &= S_1 + (1 + \frac{y_1^T S_1 y_1}{\delta_1^T y_1}) \frac{\delta_1 \delta_1^T}{\delta_1^T y_1} - \frac{\delta_1 y_1 S_1 + S_1 y_1 \delta_1^T}{\delta_1^T y_1} \\ &= \begin{pmatrix} 2 & 3 \\ 3 & 5 \end{pmatrix}. \end{aligned}$$

Notons que $S_2 H = H S_2 = I_2$, *Donc* $S_2 = H^{-1}$. *de plus* $\delta_i^T H \delta_2 = \delta_1 H \delta_0 = 0$ *i.e,* δ_0 *et* δ_1 *sont conjugués par rapport à* H.

Exemple 4.2 *Le minimum de la fonction de Rosembrock obtenue par la méthode BFGS avec une recherche linéare exacte (les même données de l'exemple* (3.2)*) est*

$$u_7 = u^* = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

4.4 BFGS avec recherche linéaire inexacte

Nous finalisons cette étude on présentons les théorèmes si dessous donnant les conditions suffisantes pour assurer la convergence de la méthode BFGS.

Convergence gobale

Nous étudions la convergence globale de la méthode BFGS avec une recherche linéaire de Wolfe, lorsqu'il est appliqué à une fonction convexe à partir de n'importe quel point de départ et de toute approximation initiale symétrique définie positive.

Théorème 4.4 *[5] Soit S_0 matrice quelconque symétrique définie positive, et soit u_0 un point de départ pour lequel*

(i) La fonction objectif f est deux fois continûment différentiable.

(ii) L'ensemble $L = \{u \in \mathbb{R}^n / \ f(u) \leq f(u_0)\}$ est convexe, et il existe deux constantes positives m et M tels que

$$m \|z\|^2 \leq z^T H(u) z \leq M \|z\|^2$$

pour tout $z \in \mathbb{R}^n$ et $u \in \mathbb{R}^n$. Alors la suite $\{u_k\}$ générée par l'algorithme de BFGS (avec recherche linéaire de Wolfe) converge vers le minimum u^* de f.

Remarque 4.3 *l' hypothèse (ii) indique que $H(u)$ est définie positive sur L et que f a un unique minimum u^* dans L.*

Convergence locale

Pour le résultat qui suit, nous devons faire des hypothèses supplémentaires.

Théorème 4.5 [5] *Supposons que $f : \mathbb{R}^n \longmapsto \mathbb{R}$ deux fois continument différentiable et que l'itérée générée par l'algorithme BFGS converge vers un minimum u^*. Avec les mêmes hypothèses du théorème (4.4). Si de plus la matrice Hessienne H est lipchitzienne en u^* telle que*

$$\|H(u) - H(u^*)\| \leq l \|u - u^*\|,$$

où l est une constante positive et supposons aussi que

$$\sum_{k=1}^{\infty} \|u_k - u^*\| < \infty.$$

Alors u_k converge vers u^* superlinéairement.

Conclusion

La méthode BFGS fonctionne bien sur des fonctions convexes différentiables. La question reste toujours ouverte pour leur efficacité sur des problèmes à objectif convexe non différentiable ou à objectif non convexe et différentiable.

Chapitre 5

Annexe

Nous donnons ci-dessous les programmes des méthodes utilisées dans le manuscrit, réalisés dans le langage de programmation Matlab. Ces programmes ont été testés sur les exemples précédents dans ce mémoire.

```
%*****************************La méthode de Newton*************************%
function [u] = newton_opt(df,H,u,epsi,itermax)
% Les données sont :
% u : un point initial.
% df : le gradient de la fonction objectif.
% H : le Hessien de f.
% epsi>0 : la précision.
% itermax : nombre max d'itérations
k = 0;
a = df(u);
b = inv(H(u)) % pour une fonction convexe.
% b = inv(H) pour une fonction non convexe.
while (norm(a) >epsi & k<=itermax)
  v = u-b*a;
  u = v;
  a = df(u);
  b = inv(H);
  k = k+1;
end
end
```

Voici maintenant le sous programme auquel les programmes précédente ont fait appel :

```
%*****************************La métode de la sécante*************************%
function [alpha] = linesearth_secant(df,u,d)
% d : la direction de descente.
beta = 0;gama=0.001;% beta et gama les points de départs
while (abs(beta-gama)>0.0001)
    a = d'*df(u+beta*d)      ;    b = d'*df(u+gama*d);
    alpha = (b*beta-a*gama)/(b-a);
    beta = gama; gama = alpha;
end
```

```
%*****************************La méthode de rang un*****************************%
function [u] = rang1(u,df,epsi,itermax)
n = length(u) ;
S = eye(n,n) ;
k = 0 ;
while (norm(df(u))>epsi & k<=itermax)
  d = -S*df(u) ; % d : la direction de quasi Newton.
   alpha = linesearth_secant(df,u,d) % pour une fonction non quadratique.
   %alpha = -(df(u)'*d)/(d'*Q*d) pour une fonction quadratique.
   v = u + alpha*d ;
   p = v-u ;
   q = df(v) - df(u) ;
   S = S+((p-S*q)*(S-H*S)')/(S'*(p-S*q))
   u = v ;
   k = k+1 ;
end
end
%*****************************La méthode de BFGS*****************************%
function [u] = BFGS(u,df,epsi,itermax)
n = length(u) ;
df(u) ;
S = eye(n,n) ;
k = 0 ;
while norm(df(u))  >= epsi & k<=itermax
   d = -S*df(u) ;
   alpha = linesearth_secant(df,u,d) ;
   d = -S*df(u) ;
   v = u + alpha*d ;
   p = v-u ;
   q = df(v) - df(u) ;
   S =S+(1+(q'*S*q)/(q'*p))*(((p*p')/(p'*q)))-((S*q*p')+(S*q*p')')/(q'*p) ;
   u = v ;
   k =k+1 ;
end
end
```

Bibliographie

[1] Andreas Antoniou and Wu-Sheng Lu, Practical Optimization, Algorithms and Engineering Applications, Springer(2007).

[2] Jean-Christophe Culioli, Introduction A L'Optimisation, ellipses(1994).

[3] Edwin K. P. Chong and Stanislaw, An introduction to optimization, Second Edition Wiley(2001).

[4] Stéphane Mottelet, Optimization Non linéaire (2003).

[5] Jorge Nocedal Stephen J. Wright, Numerical Optimization, Second Edition Springer (2006).

[6] J.-B. Hiriar-Urruty and C. Lemarechal, Convex analysis and Minimization Algorithms, Springer-Verlag, Berlin, New York, 1993.