REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE BADJI MOKHTAR-ANNABA
FACULTE DES SCIENCES
DEPARTEMENT DE MATHEMATIQUES

MEMOIRE

Présenté
Pour obtenir

Le Diplôme de **MAGISTER** En Mathématiques

Option: Mathématiques Appliquées
Optimisation

Par
Mlle HEBHOUB FAHIMA

SUR QUELQUES METHODES DE RESOLUTION DES PROBLEMES D'OPTIMISATION NON LINEAIRE SANS CONTRAINTES DE PETITE ET MOYENNE TAILLE

Soutenu publiquement le …./…/2001 Devant le jury:

Dr MAKHLOUF A.	**Prof**	**Univ. B. Mokhtar-Annaba**	**Président**
Dr BENZINE R.	**M.C**	**Univ. B. Mokhtar-Annaba**	**Rapporteur**
Dr DJOUDI A.	**M.C**	**Univ. B. Mokhtar-Annaba**	**Examinateur**
Dr MOUMENI A.	**C.C**	**Univ. B. Mokhtar-Annaba**	**Examinateur**

*** DEDICACE ***

Ce travail est dédié à mes parents qui m'ont soutenue tout au long de mes études, et qui ont su me donner confiance et espoir.

A mes frères Hakim et Mohamed

A ma cousine Yassa, son marie Hamma et sa petite famille.

A toute ma famille et en particulier mes cousines.

A la mémoire de mon grand père.

A tous mes amis, en particulier ceux de ma promotion de post-graduation

A tous ceux et celles qui m'ont encourager à aller au bout de ma tâche.

****

REMERCIEMENTS*

je remercie Dieu avant tout.

Ce travail n'aurait pu être mené à bien, sans l'aide et le soutien de Monsieur Benzine Rachid qui m'a inspirée le sujet de ce mémoire et prodiguer avec bienveillance ses conseils et ses encouragements, me faisant ainsi bénéficier de son savoir et son expérience.

Qu'il veuille bien trouver ici l'expression de ma haute gratitude et le témoignage de mon respectueux attachement

Je suis heureuse d'exprimer mes remerciements au Docteur Makhlouf A., professeur à l'université Badji Mokhtar-Annaba, pour l'honneur qu'il m'a fait de bien vouloir présider le jury.

De même je remercie vivement le Docteur, Djoudi A. maître de conférences à l'université Badji Mokhtar-Annaba et le Docteur, Moumeni.A. chargé de cours à l'université Badji Mokhtar-Annaba pour leur consentement à faire partie du jury.

Que toute personne ayant contribue à la réalisation de ce travail soit persuadée de ma totale reconnaissance.

****

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE BADJI MOKHTAR-ANNABA
FACULTE DES SCIENCES
DEPARTEMENT DE MATHEMATIQUES

MEMOIRE

Présenté
Pour obtenir

Le Diplôme de **MAGISTER** En Mathématiques

Option: Mathématiques Appliquées
Optimisation

Par
Mlle HEBHOUB FAHIMA

SUR QUELQUES METHODES DE RESOLUTION DES PROBLEMES D'OPTIMISATION NON LINEAIRE SANS CONTRAINTES DE PETITE ET MOYENNE TAILLE

Soutenu publiquement le …./…/2001 Devant le jury:

Dr MAKHLOUF A.	**Prof**	**Univ. B. Mokhtar-Annaba**	**Président**
Dr BENZINE R.	**M.C**	**Univ. B. Mokhtar-Annaba**	**Rapporteur**
Dr DJOUDI A.	**M.C**	**Univ. B. Mokhtar-Annaba**	**Examinateur**
Dr MOUMENI A.	**C.C**	**Univ. B. Mokhtar-Annaba**	**Examinateur**

Table des matières

0.1 Introduction

Soit $f : \mathbb{R}^n \to \mathbb{R}$ et (P) le problème de minimisation non linéaire sans contraintes suivant :

$$Minimiser \ \{f(x) : x \in \mathbb{R}^n\} \qquad (P)$$

L'optimisation non linéaire sans contraintes est souvent rencontrée dans des domaines variés, et l'étude des algorithmes et méthodes qui traitent ces problèmes, est importante pour plusieurs raisons:

1. Parfois pour résoudre un problème avec contraintes, on le remplace par une suite de problèmes sans contraintes, comme c'est le cas des méthodes des pénalités.

2. Plusieurs techniques d'optimisation sans contraintes peuvent être prolongés d'une façon naturelle pour fournir et motiver des procédures pour résoudre des problème avec contraintes.

Parmi les plus anciennes méthodes utilisées pour résoudre les problèmes du type (P), on peut citer la méthode de Newton (voir [22] pour la méthode de Newton et la méthode de Newton-Raphson). Cette méthode présente malgré sa célébrité les inconvénients suivants:

(i) La méthode n'est pas globalement convergente.

(ii) Elle ne distingue pas entre un minimum et un maximum.

(iii) La direction de Neuton peut ne pas être définie dans le cas où la matrice hetienne est singulière.

(iv) Elle exige à chaque itération la résolution d'un système d'équations linéaires.

Pour y remedier à ce type d'inconvenients, d'autres métodes appelées quasi-Newton ont vu le jour après 1950. Cette appelation est due au fait que ces méthodes se basent sur la méthode de Newton ou la corrigent dans diverses orientations que nous verrons plus tard.

Les méthodes quasi-Newton sont considerées parmi les meilleures techniques pour

résoudre les Problèmes d'optimisation sans contraintes de petite et moyenne taille.

Ce mémoire comporte deux parties essentielles. Dans la première partie (Chapitres 1,2,3,4,5) on établit une synthèse et une étude numérique des méthodes quasi-Newton. La seconde partie (Chapitre 6) constitue l'apport plus ou moins original de ce mémoire. On présente un algorithme qui à notre connaissance est nouveau et qui accélère la convergence de la méthode de la plus forte pente (steepest descent).

Les méthodes quasi-Newton génèrent une suite $\{x_k\}$ de la manière suivante:

$$x_{k+1} = x_k + \lambda_k d_k, \quad k = 0, 1, ..., \quad x_0 \in \mathbb{R}^n, \ \textit{un point initial donné.} \tag{0.1}$$

λ_k est la solution d'une certaine recherche linéaire qui dépend de la méthode et d_k est en général une direction de descente de la forme suivante:

$$d_k = -D_k \nabla f(x_k) \tag{0.2}$$

où D_k est une matrice définie positive qui approxime l'inverse de la matrice hetienne de la fonction f à minimiser.

C'est certainement Davidon ([18]), qui en 1959 était le premier à avoir introduit les méthodes quasi-Newton sous la forme (0.1) et (0.2). En 1963 Fletcher et Powell ([24]) les ont simplifiées et reformulées et leur ont donné une nouvelle appellation: Metric Variable Method ou encore méthode DFP. Une autre généralisation assez utile de la méthode DFP a été proposée par Broyden en 1967 ([8]). En 1970 et en se basant sur ([8]), Broyden ([10]), Fletcher ([25]), Goldfarb ([27]) et Shanno ([40]) ont proposé de façon indépendente la meilleure et la plus performante méthode rentrant dans la classe des méthodes quasi-Newton pour les problèmes d'optimisation sans contraintes de petite et moyenne taille. C'est la méthode BFGS, qui porte le nom des quatres auteurs qui l'ont trouvée.

Après 1970 les autres travaux et développements des méthodes quasi-Newton se sont surtout orientés vers les propriétés de convergence en affaiblissant les propriétés de la

fonction à minimiser et en changeant de recherches linéaires (exacte, non exacte,...). Par exemple Powell a démontré en 1972 ([36]), en utilisant une recherche linéaire exacte, que la méthode DFP converge vers la solution optimale si la fonction objective est convexe et deux fois continuement différentiable. Pour d'autres résultats de convergence on peut consulter [37], [38] et [9].

Le mémoire est divisé en six chapitres. Le premièr chapitre intitulé "Notions générales" introduit les outils de base pour la suite. On y trouve particulièrement les conditions d'optimalité des problèmes d'optimisation sans contraintes, les notions de fonctions multivoques liées aux problèmes de convergence.

Le chapitre2 est consacré à l'étude de deux recherches linéaires célèbres (Armijo et Goldstein-Wolfe). Ces méthodes d'optimisation dans $\mathbb{R}$ seront utiles pour les chapitres 4, 5 et 6.

La méthode de Newton est développée au chapitre 3. Nous lui avons consacré un chapitre entier car les méthodes qui nous intéressent par suite(Quasi-Newton), s'obtiennent à partir de la méthode de Newton, c'est à dire qu'elles essayent de la généraliser en gardant ses avantages tout en se débarrassant de ses inconvénients.

Le chapitre 4 est consacré aux méthodes dites Quasi-Newton pour la résolution des problèmes d'optimisation non linéaire sans contraintes. Parmi ces méthodes, on s'étalera particulièrement sur les deux plus importantes, la méthode DFP (Davidon, Fletcher, Powell), et la méthode $BFGS$ (Broyden, Fletcher, Goldfarb, Shanno).

Au chapitre 5, on présente une modification de la méthode $BFGS$ qui évite le calcul des dérivées. Le gradient est approximé à l'aide des différences des valeurs de la fonction. Cette approximation est calculée avec le souci d'assurer la convergence. On donne tout d'abord l'algorithme qui n'éxige pas de recherche linéaire éxacte, ensuite on montre que si la fonction objective est convexe et vérifie quelques conditions, l'algorithme converge vers une solution optimale dans un certain sens qui sera défini après. A la fin on montre que l'ordre de convergence est superlinéaire si le hessien de la fonction objective est lipshititzien au voisinage de la solution optimale.

Le chapitre 6 contient la partie plus ou moins originale de ce mémoire. Après avoir effectué dans les chapitres précédents, un tour d'horizon théorique et numérique sur les méthodes quasi-Newton, qui rappelons le, sont des modifications de la méthode de Newton, on essaye dans ce chapitre d'améliorer la méthode de la plus grande pente (steepest descent).

Cette méthode qui fut découverte par Cauchy en 1847 ([17]), et malgré sa simplicité, présente l'nconvénient d'être trés lente lorsqu'on s'approche du point optimal. L'idée de ce travail est d'essayer d'accélérer la convergence de cette méthode pour y remédier à ce défaut. L'outil utilisé pour arriver à ce but est l'$\varepsilon-$Algorithme qui est un puissant outil d'accélération de la convergence, découvert par Peter Wynn ([43]). C'est pour cette raison qu'on a appelé notre modeste algorithme: l'epsilon steepest descent algorithme.

Bien sur, on ne donne dans ce mémoire que les grands axes de ce travail. Les tests numériques et l'étude de la convergence feront l'objet d'un travail de plus grande envergure (thèse de Doctorat).

Chapitre 1

Notions générales

1.1 Généralisations sur les problèmes de minimisation sans contraintes et leurs algorithmes

1.1.1 Introduction

Soit $f : \mathbb{R}^n \to \mathbb{R}$. On appelle problème de minimisation sans contraintes le problème suivant:

$$\text{Minimiser} \{ f(x) : x \in \mathbb{R}^n \}. \tag{1.1}$$

L'étude de ces problèmes est importante pour des raisons diverses. Beaucoup de problèmes d'optimisation avec contraintes sont transformés en des suites de problèmes d'optimisation sans contraintes (multiplicateur de Lagrange, méthodes des pénalités, ...). L'études des problèmes d'optimisation sans contraintes trouve aussi des applications dans la résolutions des systèmes non linéaires.

Les méthodes numériques de résolution de divers problèmes de minimisation sans contraintes ont pris ces dernièrs années un bel essor si bien que la bibliographie correspondante contient des centaines d'ouvrages et d'articles. Cet intérêt n'est nullement fortuit, il reflète le role de premier plan que les problèmes d'optimisation jouent dans les

applications.

La construction d'algorithmes consacrés à la recherche efficace de minimum d'une fonction sans contraintes est un problème complex, car il ne suffit pas d'élaborer un algorithme, il faut montrer de plus qu'il emporte sur ceux connus.

On compare des algorithmes en se basant sur des plusieurs critères, par exemple, la précision du résultat, le nombre d'itérations, le nombre d'évalutions fonctionnelles et celui d'évaluations du gradient, la vitesse de la convergence, le temps de calcul, l'occupation de la mémoire nécessaire, ...

Même en se fixant des critères de comparaison, on ne peut pas classer les algorithmes ni en indiquant lequel est le meilleur ou le pire. Le fait est qu'on obtient les estimations de l'un des critères précédents pour des classes des problèmes, et un algorithme mauvais pour une vaste classe peut s'averer efficace pour une autre, plus restreinte. L'utilisateur doit posseder tout un arsenal d'algorithmes pour être en mesure de faire face à chaque problème posé.

1.1.2 Aspect général des algorithmes

Pour construire des algorithmes de minimisation sans contraintes on fait appel à des procesuss itératifs du type

$$x_{k+1} = x_k + \lambda_k d_k \tag{1.2}$$

où d_k détermine la direction de déplacement à partir du point x_k et λ_k est un facteur numérique dont le grandeur donne la longueur du pas dans la direction d_k.

Le type d'algorithme permettant de résoudre le problème (1.1) sera déterminé dés qu'on définit les procédés de construction du vecteur d_k et de calcul de λ_k à chaque itération.

La façon avec laquelle on construit les vecteurs d_k et les scalaires λ_k détermine directement les propriétés du procesuss et spécialement en ce qui concerne la convergence de la suite $\{x_k\}$, la vitesse de la convergence,....

Pour s'approcher de la solution optimale du problème (1.1) (dans le cas général, c'est un point en lequel ont lieu peut être avec une certaine précision les conditions nécessaires d'optimalité de f), on se déplace naturellement à partir du point x_k dans la direction de la décroissance de la fonction f.

1.2 Conditions d'optimalité des problèmes d'optimisation sans contraintes

Définition 1.1

Considérons le problème de minimisation sans contraintes (1.1).

a) $x_* \in \mathbb{R}^n$ *s'appelle minimum global du problème* (1.1) *si* $f(x_*) \leq f(x)$, $\forall x \in \mathbb{R}^n$.

b) x_* *est un minimum local de* (1.1) *s'il existe un voisinage* $V_\varepsilon(x_*)$ *de* x_* *tel que* $f(x_*) \leq f(x)$, $\forall x \in V_\varepsilon(x_*)$.

c) x_* *est minimum local strict s'il existe un voisinage* $V_\varepsilon(x_*)$ *de* x_* *tel que* $f(x_*) < f(x)$, $\forall x \in V_\varepsilon(x_*)$ *et* $x \neq x_*$.

1.2.1 Conditions nécessaires d'optimalité

Théorème 1.1

Soit $f : \mathbb{R}^n \to \mathbb{R}$ *telle que* f *soit différentiable au point* $\overline{x} \in \mathbb{R}^n$. *Soit* $d \in \mathbb{R}^n$ *telle que* $\nabla f(\overline{x})^t d < 0$. *Alors il existe* $\delta > 0$ *tel que* $f(\overline{x} + \lambda d) < f(\overline{x})$ *pour tout* $\lambda \in]0, \delta[$. *La direction* d *s'appelle dans ce cas direction de descente.*

Preuve

Comme f est différentiable en $\overline{x}$ alors

$$f(\overline{x} + \lambda d) = f(\overline{x}) + \lambda \nabla f(\overline{x})^t d + \lambda \|d\| \alpha(\overline{x}; \lambda d)$$

où $\alpha(\overline{x};\lambda d) \to 0$ pour $\lambda \to 0$. Ceci implique

$$\frac{f(\overline{x}+\lambda d)-f(\overline{x})}{\lambda} = \nabla f(\overline{x})^t d + \|d\|\,\alpha(\overline{x};\lambda d),\ \lambda \neq 0$$

et comme $\nabla f(\overline{x})^t d < 0$ et $\alpha(\overline{x};\lambda d) \to 0$ pour $\lambda \to 0$, il existe $\delta > 0$ tel que

$$\nabla f(\overline{x})^t d + \|d\|\,\alpha(\overline{x};\lambda d) < 0 \text{ pour tout } \lambda \in]0,\delta[$$

et par conséquent on obtient

$$f(\overline{x}+\lambda d) < f(\overline{x}) \text{ pour tout } \lambda \in]0,\delta[.\square$$

Corollaire 1.1

Soit $f : \mathbb{R}^n \to \mathbb{R}$ différentiable en $\overline{x}$, si $\overline{x}$ est un minmum local alors $\nabla f(\overline{x}) = 0$.

Preuve

On démontre ce théorème par l'absurde, alors on suppose que $\nabla f(\overline{x}) \neq 0$. Si on pose $d = -\nabla f(\overline{x})$, on obtient

$$\nabla f(\overline{x})^t d = -\|\nabla f(\overline{x})\|^2 < 0$$

et par le théorème précédent, il existe $\delta > 0$ tel que

$$f(\overline{x}+\lambda d) < f(\overline{x}) \text{ pour tout } \lambda \in]0,\delta[$$

ce qui donne une contradiction avec le fait que $\overline{x}$ est un minimum local, d'où $\nabla f(\overline{x}) = 0.\square$

Définition 1.2

Une matrice symétrique A est dite définie positive (semi définie positive) respectivement, si

$$\forall d \in \mathbb{R},\ d \neq 0, d^t A d > 0, (d^t A d \geq 0).$$

Théorème 1.2

Soit $f : \mathbb{R}^n \to \mathbb{R}$ deux fois différentiable en $\overline{x}$. Supposons que $\overline{x}$ soit minimum local. Alors $\nabla f(\overline{x}) = 0$ et $H(\overline{x})$ est semi définie positive.

Preuve

Le corollaire ci-dessus montre la pemière proposition, pour la deuxième proposition on a

$$f(\overline{x} + \lambda d) = f(\overline{x}) + \lambda \nabla f(\overline{x})^t d + \frac{1}{2}\lambda^2 d^t H(\overline{x}) d + \lambda^2 \|d\|^2 \alpha(\overline{x}; \lambda d)$$

où $\alpha(\overline{x}; \lambda d) \to 0$ pour $\lambda \to 0$. Ceci implique

$$\frac{f(\overline{x} + \lambda d) - f(\overline{x})}{\lambda^2} = \frac{1}{2} d^t H(\overline{x}) d + \|d\|^2 \alpha(\overline{x}; \lambda d), \ \lambda \neq 0.$$

Comme $\overline{x}$ est un minimum local alors $f(\overline{x} + \lambda d) \geq f(\overline{x})$ pour λ suffisamment petit, d'où

$$\frac{1}{2} d^t H(\overline{x}) d + \|d\|^2 \alpha(\overline{x}; \lambda d) \geq 0 \text{ pour } \lambda \text{ petit.}$$

En passant à la limite quand $\lambda \to 0$, on obtient $d^t H(\overline{x}) d \geq 0$, d'où $H(\overline{x})$ est semi définie positive.□

1.2.2 Conditions suffisantes d'optimalité

Théorème 1.3

Soit $f : \mathbb{R}^n \to \mathbb{R}$ deux fois différentiable en $\overline{x}$. Si $\nabla f(\overline{x}) = 0$ et $H(\overline{x})$ est définie positive, alors $\overline{x}$ est minmum local strict.

Preuve

f est deux fois différentiable au point $\overline{x}$. Alors $\forall x \in \mathbb{R}^n$, on obtient

$$f(x) = f(\overline{x}) + \nabla f(\overline{x})^t (x - \overline{x}) + \frac{1}{2}(x - \overline{x})^t H(\overline{x})(x - \overline{x}) + \|x - \overline{x}\|^2 \lambda(\overline{x}; x - \overline{x}) \quad (1.3)$$

où $\lambda(\overline{x}; x-\overline{x}) \to 0$ pour $x \to \overline{x}$. Supposons que $\overline{x}$ n'est pas un minimum local strict. Donc il existe une suite $\{x_k\}$ convergente vers $\overline{x}$ telle que $f(x_k) \leq f(\overline{x})$, $x_k \neq \overline{x}$, $\forall k$. Posons $d_k = (x_k - \overline{x})/\|x_k - \overline{x}\|$. Donc $\|d_k\| = 1$ et on obtient à partir de (1.3)

$$\frac{f(x_k) - f(\overline{x})}{\|x_k - \overline{x}\|^2} = \frac{1}{2} d_k^t H(\overline{x}) d_k + \alpha(\overline{x}; x_k - \overline{x}) \leq 0, \quad \forall k \tag{1.4}$$

et comme $\|d_k\| = 1$, $\forall k$ alors $\exists \{d_k\}_{k \in N_1 \subset \mathbb{N}}$ telle que $d_k \to d$ pour $k \to \infty$ et $k \in N_1$. On a bein sûr $\|d\| = 1$. Considérons donc $\{d_k\}_{k \in N_1}$ et le fait que $\alpha(\overline{x}; x - \overline{x}) \to 0$ pour $k \to \infty$ et $k \in N_1$. Alors (1.4) donne: $d^t H(\overline{x}) d \leq 0$, ce qui conterdit le fait que $H(\overline{x})$ est définie positive car $\|d\| = 1$ (donc $d \neq 0$). Donc $\overline{x}$ est un minimum local strict.□

1.2.3 Cas convexe

Définition 1.3

Soit $f : \mathbb{R}^n \to \mathbb{R}$.

a) *f est dite convexe sur $\mathbb{R}^n$ si*

$$f(\mu x_1 + (1-\mu)x_2) \leq \mu f(x_1) + (1-\mu) f(x_2)$$

pour touts point x_1 et $x_2 \in \mathbb{R}^n$ et pour tout $\mu \in [0,1]$.

b) Si l'inégalité précédente est stricte pour touts points x_1 et x_2 distincts et pour tout $\mu \in]0,1[$, alors f est dite strictement convexe.

c) Supposons que f soit différentiable. f est dite pseudo convexe si ,

Pour tout x_1 et x_2 vérifiant $\nabla f(x_1)^t (x_2 - x_1) \geq 0$, on a: $f(x_2) \geq f(x_1)$.

Théorème 1.4 ([5] p.103)

Soit $f : \mathbb{R}^n \to \mathbb{R}$ telle que f est convexe et différentiable. Alors x_* est un minimum global de f si et seulement si $\nabla f(x_*) = 0$.

Théorème 1.5 ([5] *p.*114)

Soit $f : \mathbb{R}^n \to \mathbb{R}$ *telle que est pseudo convexe. Soit* $x_* \in \mathbb{R}^n$ *tel que* $\nabla f(x_*) = 0$, *alors* x_* *est un minimum global de* f.

Remarque 1.1

Les théorèmes 1.4 et 1.5 demeurent vrais si on remplace $\mathbb{R}^n$ par un ouvert S de $\mathbb{R}^n$.

Remarque 1.2

Dans le cas où f est convexe, alors tout minimum local est aussi global. De plus si f est strictement convexe, alors tout minimum local devient non seulement global mais aussi unique ([5] *p.*101).

1.3 Fonctions multivoques et algorithmes

Une application multivoque est une application A qui à $x \in \mathbb{R}^n$ fait correspondre un sous ensemble $A(x)$ de $\mathbb{R}^n$.

Etant donné un point x_k. En appliquant les instructions d'un certain algorithme, on obtient un nouveau point x_{k+1}. Cette procédure peut être décrite par une application multivoque A applée application algorithmique. Donc étant donné un point $x_1 \in \mathbb{R}^n$, l'application algorithmique génère une suite $x_1, x_2, ...$, où $x_{k+1} \in \mathbb{R}^n$ pour tout k.

La notion d'application algorithmique fermé est directement liée à la convergence des algorithmes

Définition 1.4

Soient X *et* Y *deux sous ensembles fermés non vides de* $\mathbb{R}^n$*et* $\mathbb{R}^m$ *respectivement et* $A : X \to Y$ *une application multivoque.* A *est dite fermée au point* $x \in \mathbb{R}^n$ *si*

$$\left.\begin{array}{ll} x_k \in X, & x_k \to x \\ y_k \in A(x_k), & y_k \to y \end{array}\right\} \Rightarrow y \in A(x).$$

- A *est dite fermée sur* $Z \subset \mathbb{R}^n$ *si elle est fermée en chaque point de* Z.

Donnons maintenant deux théorème de convergence qui utilisent les fonctions multivoques et qui nous seront utile par la suite. Avant celà notons qu'à cause de la non convexité, de la taille du problème et d'autres difficultés on peut arrêter le procesuss itératif si on trouve un point appartient à un ensemble spécifique qu'on appelle ensemble des solutions Ω. Voici ci-dessous quelques exemples typiques de cet ensemble.

$\Omega = \{x_* : \nabla f(x_*) = 0\}$

$\Omega = \{x_* : x_*$ est une solution optimale locale du problème (1.1)$\}$

$\Omega = \{x_* : f(x_*) < \nu_* + \varepsilon\}$ où $\varepsilon > 0$ est une tolérance définie à l'avance et ν_* est la valeur minimale de la fonction objective.

Nous dirons que l'application algorithmique $A : X \to X$ converge dans $Y \subseteq X$ si commençant par n'importe quel point initial $x_1 \in Y$, la limite de toute sous suite convergente, extraite de la suite $x_1, x_2, ...$, générée par l'algorithme, appartient à Ω.

Théorème 1.6 ([5] *p.253*)

Soient X un ensemble non vide fermé dans $\mathbb{R}^n$ et $\Omega \subseteq X$ un ensemble des solutions non vide. Soit $\alpha : \mathbb{R}^n \to \mathbb{R}$ une fonction continue, on considère que l'application algorithmique $C : X \to X$ vérifie la propriété suivante: "étant donné $x \in X$ alors $\alpha(y) \leq \alpha(x)$ pour $y \in C(x)$". Soit $B : X \to X$ une application algorithmique fermée sur le complémentaire de Ω qui vérifie: $\alpha(y) < \alpha(x)$ pour $y \in B(x)$ et $x \notin \Omega$. Maintenant considérons l'application algorithmique composé $A = CB$. Soit $x_1 \in X$, la suite $\{x_k\}$ est générée comme suit:

Si $x_k \in \Omega$, stop; sinon poser $x_{k+1} \in A(x_k)$, remplacer k par $k+1$ et repeter.

Supposons que $\Lambda = \{x \in X,\ \alpha(x) \leq \alpha(x_1)\}$ est compact. Alors, ou bien l'algorithme s'arrête à un nombre fini d'itérations par un point dans Ω, ou tous les points d'accumilation de $\{x_k\}$ appartenants à Ω.

Théorème 1.7 [5, *p.* 249]

Soit X un ensemble non vide fermé dans $\mathbb{R}^n$, et Ω l'ensemble des solutions non vide. Soit $A : X \to X$ une *application algorithmique. Donner un point $x \in X$, la suite $\{x_k\}$ est générée itérativement comme suit*

si $x_k \in \Omega$ arrêter; sinon, soit $x_{k+1} \in A(x_k)$ remplacer k par $k+1$, et répéter.

Supposons que la suite $x_1, x_2, ...$ générée est contenue dans un sous ensemble compact de X ,et qu'il existe une fonction continue α, dite fonction de descente, telque $\alpha(x) < \alpha(y)$ si $x \notin \Omega$ et $y \in A(x)$.

Si A est fermé sur le complémentaire de Ω, alors soit l'algorithme termine en un nombre fini de pas avec un point dans Ω, soit il génère une suite infinie $\{x_k\}$ telque

1. chaque sous suite de $\{x_k\}$ admet une limite dans Ω, c-.à.d chaque point d'accumulation de $\{x_k\}$ appartient à Ω.

2. $\alpha(x_k) \to \alpha(x)$ pour $x \in \Omega$.

Corollaire1.2 [5, *p.*250]

Considérent les mêmes hypothèses que dans le théoreme précédent, si $\Omega = \{\overline{x}\}$, alors la suite $\{x_k\}$ converge vers $\overline{x}$.

1.4 Modes de convergence

Définition 1.5

Soit $\{x_k\}$ une suite de vecteurs convergente vers $\overline{x}$ telle que $x_k \neq \overline{x}$ pour tout k. L'ordre de la convergence de la suite $\{x_k\}$ est le supremum des nombres p non négative tel que:

$$\overline{\lim_{k\to\infty}} \frac{\|x_{k+1} - \overline{x}\|}{\|x_k - \overline{x}\|^p} = \beta < \infty.$$

Si $p = 1$ et $\beta < 1$ alors la convergence est dite linéaire. Si $p > 1$ ou $p = 1$ et $\beta = 0$ alors la convergence est dite superlinéaire. En particulier, si $p = 2$ et $\beta < \infty$ alors la convergence est dite quadratique.

Chapitre 2

Optimisation dans $\mathbb{R}$ ou Recherche linéaire

Introduction:

On donne dans cette partie un aperçu sur les recherches linéaires qu'on utilisera plus tard. Il s'agit des recherches linéaires de Goldstein-Wolfe et celle d'Armijo.

2.1 Position du Problème

Soit: $f : \mathbb{R}^n \to \mathbb{R}$ et (P) le problème d'optimisation sans contraintes suivant:

$$Minimiser \ \{f(x) : x \in \mathbb{R}^n\} \qquad (P)$$

Beaucoup d'algorithmes d'optimisation non linéaire procèdent comme suit. Etant donné un point x_k, on cherche une certaine direction d_k et un pas λ_k. Ceci nous permet de trouver le successeur x_{k+1} de x_k par la formule: $x_{k+1} = x_k + \lambda_k d_k$ et le procesuss est ainsi répété.

Le pas λ_k est la solution optimale d'un problème de minimisation unidimentionnelle c'est à dire la minimisation d'une fonction d'une variable réelle et à valeurs réelles de la

forme suivante:

$$R.L. \quad \{Minimiser\ \theta(\lambda) = f(x_k + \lambda d_k),\ \lambda \in I,\ \ I \subset \mathbb{R}^+,\ \ I\ \ ferm\acute{e},\ born\acute{e}\ ou\ non\}$$

Donc pour les problèmes de minimisation sans contraintes, on résout à chaque itération des sous problèmes de minimisation de fonctions d'une variable réelle. C'est ce qu'on appelle recherche linéaire

2.1.1 Hypothèses

Nous avons vu au théorème1.1 que si pour le point x_k et la direction d_k, la fonction f vérifie:

$$\nabla f(x_k)^t.d_k < 0$$

alors d_k est une direction de déscente. Or si on prend: $\theta(\lambda) = f(x_k + \lambda d_k)$, on a:

$$\theta'(0) = \nabla f(x_k)^t.d_k = f'(x_k, d_k) = \lim_{\lambda \to 0_+} \frac{f(x_k + \lambda d_k) - f(x_k)}{\lambda}$$

On imposera donc dans nos recherches linéaires les deux conditions suivantes:
a) $\theta^{'}(0) < 0$ (condition de descente), qui est équivalente à:

$$\theta(\lambda_k) = f(x_k + \lambda_k d_k) < f(x_k) = \theta(0) \tag{2.1}$$

b)L'optimum λ^* de la $(R.L)$ existe. Dans le cas contraire, nous avons l'une des deux alternatives suivantes:

1. $\inf\limits_{I} \theta(\lambda) = -\infty \Rightarrow (P)$ non borné: $\inf\limits_{\mathbb{R}^n} f(x) = -\infty$
2. $\inf\limits_{I} \theta(\lambda)$ fini mais non atteint: on se limite à $\overline{\lambda}$ telque $\theta(\overline{\lambda}) \cong \inf\limits_{I} \theta(\lambda)$

2.1.2 Recherche linéaire exacte ou inexacte

Si l'on determine la valeur λ^* de façon exacte, on dit dans ce cas que la recherche linéaire est exacte, chose difficile à réaliser en pratique. Il faut donc adopter une recherche linéaire qui garantit un degré de précision convenable ou une descente suffisante de $f(x)$, cappable d'induire la convergence des algorithmes en question. Une telle recherche linéaire est dite inexacte .

2.2 Deux Méthodes de recherche linéaire inexacte

2.2.1 Méthodes de Goldstein et Wolfe

Nous présentons essetiellement deux type de procédures:

La règle de Goldstein ([26]), qui s'applique lorsque le gradient de la fonction ne peut être évalué (ou trop coûteux à obtenir).

La procédure suggerée par Wolfe ([42]), qui nécessite l'évaluation du gradient chaque fois que la fonction est calculée.

Le principe commun à toutes ces méthodes est que:

i) λ ne doit pas être choisi trop grand (sinon l'algorithme risque d'avoir un comportement oscillatoire).

ii) λ ne doit pas être choisi trop petit (sinon l'algorithme risque de converger prématurément).

Dans la règle de goldstein la condition **i)** est vérifiée si on a

$$\theta(\lambda) \leq \theta(0) + c_1\theta^{'}(0) \tag{2.2}$$

où c_1 est un coéfficient choisi dans $]0,1[$; et l'on s'assure de la condition **ii)** si la relation suivante est vérifiée

$$\theta(\lambda) \geq \theta(0) + c_2\theta^{'}(0) \tag{2.3}$$

où c_2 est un coéfficient choisi dans $]c_1, 1[$.

Notons que la condition (2.2) assure que le nouveau point $x_{k+1} = x_k + \lambda d_k$ obtenu vérifie $f(x_{k+1}) < f(x_k)$ (propriété de descente).

Dans la règle de Wolfe, c'est encore la relation (2.2) qui permet de s'assurer que λ n'est pas choisi trop grand. Mais comme on suppose que le calcul du gradient de la fonction f ne nécessite pas beaucoup plus de calcul que l'évaluation de la fonction elle-même, la relation (2.3) peut être remplacée par la condition :

$$\theta^{'}(\lambda) \geq c_3\theta^{'}(0) \tag{2.4}$$

où c_3 est un coéfficient choisi dans $]c_1, 1[$. Donc la procédure est la suivante

2.2.2 Algorithme

Etape initiale:

On dispose de $\lambda_g = 0$, $\lambda_d =$ une valeur maximale quelquonque

Poser $c_1 = 0.1$, $c_2 = 0.7$.

Etapes principales:

Etape 1: Calculer $\theta(\lambda) = f(x_k + \lambda d_k)$

Si $\theta(\lambda) \leq \theta(0) + m\lambda\theta^{'}(0)$ alors aller à l'étape 2

Sinon prendre $\lambda_d = \lambda$ et aller à l'étape 4.

Etape 2: (G) Si $\theta(\lambda) \geq \theta(0) + c_2\lambda\theta^{'}(0)$ stop

Sinon aller à l'étape 3.

(W) Calculer $\theta^{'}(\lambda) = \nabla f(x_k + \lambda d_k)^t d_k$

Si $\theta^{'}(\lambda) \geq c_2\theta^{'}(0)$ stop

Sinon aller à l'étape 3.

Etape 3: Poser $\lambda_g = \lambda$

Etape 4: Rechercher un nouveau $\lambda \in]\lambda_g, \lambda_d[$ et retourner à l'étape 2.

Programme en Fortran 90 de la recherche linéaire de Goldstein

```
program Goldstein
! ------------------------------------------------------------------
! programme qui donne une longueur du pas en utilisant la recherche linéaire
! de goldstein.
! ------------------------------------------------------------------
! ---------- partie déclaration ----------
implicit none
real l0,lg,ld ,w,b
b=0.7; w=0.1
lg=0; ld=100; l0=1
do
if(f(l0)<=f(0.)+w*l0*df(0.))then
if(f(l0)>=f(0.)+b*l0*df(0.))then
exit
else
lg=l0; l0=(lg+ld)/2.
endif
else
ld=l0; l0=(lg+ld)/2.
endif
enddo
print*,'la longueur du pas l0=',l0
contains
! ---------- sous programme qui donne la valeur de f au point x ----------
function f(x)
real f,x
f=...
```

```
endfunction
! ———————— sous programme qui donne la valeur du gradient au point x ————
function df(x)
real df, x
df=...
endfunction
end
```

Remarque 2.1:

Le programme précédent est celui de Goldstein, si on veut utiliser la condition de Wolfe il suffit de remplacer la condition if(f(10)>=f(0.)+b*10*df(0.)), par if(df(10)>=b*10*df(0.)).

2.2.3 Méthode d'Armijo

Cette méthode consiste à délimiter un intervalle convenable pour λ et à choisir le plus grand λ_k possible dans cet intervalle. Pour motiver le test d'Armijo supposons qu'on veut minimiser $f : \mathbb{R}^n \to \mathbb{R}$ telle que f est une fonction différentiable en un point $x_k \in \mathbb{R}^n$ avec $\nabla f(x_k)^t d_k < 0$ (d_k est une direction de descente).

Soit $\theta : \mathbb{R} \to \mathbb{R}$ définie comme suit

$$\theta(\lambda) = f(x_k + \lambda d_k) \qquad \text{pour } \lambda \geq 0$$

D'aprés la 1^{ere} approximatin de θ en $\lambda = 0$ on trouve

$$\theta(\lambda) = f(x_k + \lambda d_k) = \theta(0) + \lambda\theta'(0) = f(x_k) + \lambda \nabla f(x_k)^t d_k.$$

Notons

$$\widehat{\theta}(\lambda) = \theta(0) + \lambda\varepsilon\theta'(0), \quad \lambda \geq 0.$$

$\overline{\lambda}$ peut être considérée comme longueur du pas acceptable si $\theta(\overline{\lambda}) \leq \widehat{\theta}(\overline{\lambda})$.

L'inégalité $\theta(\overline{\lambda}) \leq \widehat{\theta}(\overline{\lambda})$ est dite condition d'Armijo.

2.2.4 Algorithme

Etape initiale: choisir un point initlial λ_1,et un $\varepsilon \in (0,1)$

Etapes principales:

Etape1: Si $\theta(\lambda_1) \leq \theta(0) + \lambda_1 \varepsilon \theta'(0)$,aller à l'étape 2 sinon aller à l'étape 3

Etape2: Soit t le plus petit entier positif avec $\lambda = \lambda_1 2^t$ qui vérifie

$$\theta(\lambda_1 2^t) \leq \theta(0) + \varepsilon \lambda_1 2^t \theta'(0) \tag{2.5}$$

Etape3: Soit t le plus petit entier positif avec $\lambda = \lambda_1 / 2^t$ qui vérifie

$$\theta(\lambda_1 / 2^t) \leq \theta(0) + \varepsilon (\lambda_1 / 2^t) \theta'(0) \tag{2.6}$$

Programme en Fortran 90 de la recherche linéaire d'Armijo

```
program armijo
! ---------------------------------------------------------------------
! programme donne une longueur du pas en utilisant la recherche linéaire
! d'armijo.
! ---------------------------------------------------------------------
! ---------- partie déclaration ---------------------------------------
implicit none
real eps,l,f0,g
integer t
print*,'entrer eps'
read*,eps
print*,'entrer l'
read*,l
t=1; f0=f(0.); g=df(0.)
```

```
if(f(l)<=f0+eps*l*g)then
do
if(f(l*2**t)>f0+eps*(l*2**t)*g)exit
t=t+1
enddo
l=l*2**(t-1)
else
do
if(f(l/2**t)<=f0+eps*(l/2**t)*g)exit
t=t+1
enddo
l=l/2**t
endif
print*,'la longueur du pas l=',l
contains
! ———— sous programme qui donne la valeur de f au point x ————
function f(x)
real f,x
f=...
endfunction
! ———— sous programme qui donne la valeur du gradient au point x ——
function df(x)
real df, x
df=...
endfunction
end
```

Chapitre 3

Optimisation sans contraintes: Méthode de Newton

3.1 Introduction

Soit $f : \mathbb{R}^n \to \mathbb{R}$. On supposera dans tout ce chapitre que $f \in C^2(\mathbb{R}^n)$. Soit maintenant (P) le problème d'optimisation sans contraintes que nous avons déja rencontré dans le chapitre1:

$$\text{Minimiser} \{ f(x) : x \in \mathbb{R}^n \}.$$

La méthode de Newton est attribuée au mathématicien, physicien et astronome anglais Issac Newton (1642-1727). Toutefois, comme le dit Durand ([19]) c'est Raphson qui publiait, en 1690, la formule itérative utilisée actuellement. C'est la raison pour laquelle certains auteurs l'appellent méthode de Raphson-Newton. L'algorithme de Newton est la généralisation multi-dimentionnelle de la méthode de Newton Raphson (voir [41], p.p.28,86,90 et 150), appliquée à la recherche des racines de $\nabla f(x)$.

C'est certainement, avec la méthode de la plus forte pente (Cauchy 1847, [17]) l'une des plus anciennes méthodes utilisées pour resoudre le problème (P). Nous lui avons consacré un chapitre entier car les méthodes qui nous intéressent par la suite (Quasi-

Newton), s'obtiennent à partir de la méthode de Newton, c'est à dire qu'elles essayent de la généraliser en gardant ses avantages tout en se débarrassant de ses inconvénients.

Le principe de la méthode de Newton consiste à minimiser successivement les approximations du second-ordre d'une fonction f , plus précisement si

$$f(x) \simeq f(x_k) + \nabla f(x_k)(x - x_k) + \frac{1}{2}(x - x_k)^t H(x_k)(x - x_k) = q(x), \quad \forall x \in V(x_k),$$

$H(x_k)$ étant la matrice hessienne de f au point x_k, alors une condition nécessaire de minimum pour q est que $\nabla q(x) = 0$, ou

$$\nabla f(x_k) + H(x_k)(x - x_k) = 0. \tag{3.1}$$

Supposons que H est inversible, alors le successeur de x_k est donné par

$$x_{k+1} = x_k - H^{-1}(x_k)\nabla f(x_k) \tag{3.2}$$

Cette équation donne la forme générale des points générés par l'algorithme de Newton. Supposons que $\nabla f(\overline{x}) = 0$ et que $H(\overline{x})$ est définie positive ($\overline{x}$ un minimum local), alors $H(x_k)$ reste définie positive en tout point voisin de $\overline{x}$. Ceci assure que le seccesseur de x_k est bien défini.

3.1.1 Algorithme

Etape initiale: Soit ε un paramètre qui détermine le critère d'arrêt. Choisir un point initial. Poser k=1 et aller à l'étape principale .

Etape principale:

Si $\|\nabla f(x_k)\| < \varepsilon$, stop; sinon, poser $x_{k+1} = x_k - H^{-1}(x_k)\nabla f(x_k)$, remplacer k par $k + 1$et répéter l'étape principale.

Exemple 3.1

Nous allons appliquer maintenant cet algorithme sur un problème test trés classique dû à Rosenbrock. La fonction de rosenbrock à minimiser dans $\mathbb{R}^n$est la suivante:

$$f(x_1, x_2) = (x_1 - 1)^2 + p(x_1^2 - x_2)^2,$$

où p est un paramètre positif, ici égal à 100. Dans les lignes de niveau de cette fonction il y'a une vallée très étroite, en forme de banane, d'où le surnom "Rosenbrock banana", conduisant au minimum global $x^* = (1, 1)$, avec $f(x^*) = 0$. Donc seule une méthode efficace permettra de trouver ce minimum (voir [31], p.152)

Les résultats obtenus par un programme en fortran 90 sont illustrés dans le tableau 3.1, le point initial est (-1.2,1) et $\varepsilon = 10^{-5}$.

tableau 3.1

k	$f(x)$	$\|\nabla f(x)\|$
1	.242000E+02	.232868E+03
2	.473188E+01	.463943E+01
3	.141185E+04	.137079E+04
4	.559655E-01	.473110E+00
5	.313189E+00	.250274E+02
	.185274E-10	.860863E-05

==========

Pour atteindre la précision ε donnée, il nous a fallu 6 itérations.

la solution approchée est $x_6 = (x_1, x_2)$;

$x_1 = 9.999956956536786E\text{-}001$, $x_2 = 9.999913913257368E\text{-}001$

avec $f(x_6) = f(x_1, x_2) = 1.852739725430225E\text{-}011$

==========

programme en Fortran 90 de la méthode de Newton

```
program newton
! ---------------------------------------------------------------
! Ce programme minimise une fonction deux fois continuement differentiable
! à plusieurs variables en utilisant la méthode de Newton.
! ---------------------------------------------------------------
! ------- partie déclaration ------------------------------------
implicit none
integer,parameter:: n=2
integer i,j,k,s,v
doubleprecision x(n),H(n,n),HI(n,n),g(n),eps,t,d(n)
print*,'entrer le point initial x0'
print*,'==============='
read*,(x(i),i=1,n)
print*,'entrer eps'
print*,'======='
read*,eps
s=1
g=gradf(x)
do
if(norm(g)<=eps.or.v==2)exit
print '(i4,2e20.6)',s,f(x),sqrt(dot_product(g,g))
H=Hessf(x)
! ---- appel au sous programme qui calcule l'inverse du hessien ----
call Hinvers(H,HI,t)
if(t==1)then
print*,'la matrice n est pas inverssible'
v=2
```

```
endif
! ---------- calcul de la direction ----------
d=matmul(HI,g)
! ---------- calcul du point seccesseur ----------
x=x-d
s=s+1
g=gradf(x)
enddo
! ------Impression de resultats------
print*,'=========================================
print*,'le nombre d iteration est s=',s
print*,'la solution est x=',x
print*,'la valeur de f(x) est ',f(x)
print*,'=========================================
contains
function f(u)
doubleprecision f,u(n)
f=1.0e2*(u(2)-u(1)**2)**2+(1.0e0-u(1))**2
endfunction
function gradf(u)
doubleprecision gradf(n),u(n)
gradf(1)=-4.0e2*u(1)*(u(2)-u(1)**2)-2.0e0*(1.0e0-u(1))
gradf(2)=2.0e2*(u(2)-u(1)**2)
endfunction
! ----------
! sous programme qui donne la matrice hessienne
! ----------
function Hessf(u)
```

```
doubleprecision u(n),hessf(n,n)
hessf(1,1)=-400*u(2)+1200*u(1)**2+2
hessf(1,2)=-400*u(1)
hessf(2,1)=hessf(1,2)
hessf(2,2)=200
endfunction
function norm(v)
doubleprecision norm,v(n)
norm=0
do i=1,n
norm=norm+v(i)**2
enddo
norm=sqrt(norm)
endfunction
! ------------------------------------------------------------------
! sous programme qui donne l'inverse de la matrice hessienne
! ------------------------------------------------------------------
subroutine Hinvers(A,AI,r)
doubleprecision B(n,2*n),AI(n,n),r
doubleprecision A(n,n),h,pivot
integer l,l1,l2
do i=1,n
do j=1,n
if(i==j)then
A(i,j+n)=1
else
A(i,j+n)=0
endif
```

```
enddo
enddo
do k=1,n
pivot=A(k,k); l=k
do l2=k+1,n
if(abs(A(l2,k))>pivot)then
pivot=A(l2,k)
l=l2
endif
enddo
if(pivot==0)then
r=1
goto 11
endif
do l1=1,2*n
h=A(l,l1); A(l,l1)=A(k,l1); A(k,l1)=h
enddo
do j=1,2*n
B(k,j)=A(k,j)/pivot
enddo
do i=1,n
if(i/=k)then
do j=1,2*n
B(i,j)=A(i,j)-A(i,k)*B(k,j)
enddo
endif
enddo
do i=1,n
```

```
do j=1,2*n
A(i,j)=B(i,j)
enddo
enddo
enddo
do i=1,n
do j=1,n
AI(i,j)=A(i,j+n)
enddo
enddo
11 endsubroutine
end
```

3.1.2 Etude de la convergence

En général la méthode de Newton ne converge pas à cause de la possibilité que $H(x_k)$ soit singulière et donc que le point x_{k+1} ne soit pas bien défini. Et même si $H^{-1}(x_k)$ existe, la décroissance de $f(x)$ n'est pas toujours assurée. Cependent ces problèmes peuvent être évités si le point initial est assez voisin du point $\overline{x}$, tel que $\nabla f(\overline{x}) = 0$ et $H^{-1}(\overline{x})$ existe. Dans ce cas la méthode de Newton est bien définie et elle converge vers le point $\overline{x}$. C'est ce que nous prouverons dans ce théorème.

Théorème 3.1

Soit $f : \mathbb{R}^n \to \mathbb{R}$ trois fois continuement différentiable. Considérons l'algorithme de Newton défini par l'application $A(x) = x - H^{-1}(x)\nabla f(x)$. Soit $\overline{x}$ tel que $\nabla f(\overline{x}) = 0$ et supposons que $H(\overline{x})^{-1}$ existe. Soit x_1 un point initial assez proche de $\overline{x}$ de sorte que cette proximité implique l'existence de k_1, $k_2 > 0$, avec $k_1 k_2 \|x_1 - \overline{x}\| < 1$ et vérifiant les deux propriétés suivantes:

1. $\|H(x)^{-1}\| \leq k_1$
2. $\|\nabla f(\overline{x}) - \nabla f(x) - H(x)(x - \overline{x})\| \leq k_2 \|\overline{x} - x\|^2$

pour tout x satisfaisant

$$\|x - \overline{x}\| \leq \|x_1 - \overline{x}\|.$$

Alors, l'algorithme converge de façon superlinéaire vers $\overline{x}$, avec une vitesse de convergence quadratique.

Preuve

Soit l'ensemble de solutions $\Omega = \{\overline{x}\}$ et l'ensemble

$$X = \{x : \|x - \overline{x}\| \leq \|x_1 - \overline{x}\|\}.$$

Pour prouver la convergence, on utilise le corollaire1.2 du théorème 1.7. Notons que X est compact et que l'application A définie par:

$$y \in A(x) \Leftrightarrow y = x - H^{-1}(x)\nabla f(x),$$

est fermée sur X

Montrons maintenant que $\alpha(x) = \|x - \overline{x}\|$ est une fonction de descente. Soit $x \in X$,et supposons que $x \neq \overline{x}$. Soit $y \in A(x)$, donc

$$\begin{aligned} y - \overline{x} &= (x - \overline{x}) - H^{-1}(x)\left[\nabla f(x) - \nabla f(\overline{x})\right], \\ &= H^{-1}(x)\left[\nabla f(\overline{x}) - \nabla f(x) - H(x)(\overline{x} - x)\right]. \end{aligned}$$

D'aprés (1),(2),on a

$$\begin{aligned} \|y - \overline{x}\| &= \left\|H^{-1}(x)\left[\nabla f(\overline{x}) - \nabla f(x) - H(x)(\overline{x} - x)\right]\right\|, \\ &\leq \left\|H^{-1}(x)\right\| \left\|\nabla f(\overline{x}) - \nabla f(x) - H(x)(\overline{x} - x)\right\|, \end{aligned}$$

$$\begin{aligned} &\leq \left\|H^{-1}(x)\right\| \left\|\nabla f(\overline{x}) - \nabla f(x) - H(x)(\overline{x} - x)\right\|, \\ &< \left\|x - \overline{x}\right\| . \end{aligned}$$

Donc on a prouvé que $\alpha(y) < \alpha(x)$, $\forall y \in A(x)$ et par suite α est une fonction de descente. D'aprés le corollaire 1.1, on conclut que l'algorithme converge vers $\overline{x}$. De plus, pour tout $x_k \in X$, son seccesseur x_{k+1} satisfait:

$$\begin{aligned} \left\|x_{k+1} - \overline{x}\right\| &\leq k_1 k_2 \left\|x - \overline{x}\right\|^2 \Rightarrow \frac{\left\|x_{k+1} - \overline{x}\right\|}{\left\|x_k - \overline{x}\right\|^2} \leq k_1 k_2, \\ &\Rightarrow \limsup_{k \to +\infty} \frac{\left\|x_{k+1} - \overline{x}\right\|}{\left\|x_k - \overline{x}\right\|^2} \leq k_1 k_2. \end{aligned}$$

Puisque $\{x_k\} \rightarrow \overline{x}$, donc la vitesse de convergence est quadratique.

3.1.3 Avantages et Inconvénients de la méthode de Newton

Avantages

Comme nous l'avons vu au théorème précédent, la méthode de *Newton* bénéfice d'une convergence quadratique, et c'est son principal avantage

Inconvénients

1. Cette méthode fonctionne trés bien pour les problème de petite dimension, lorsqu'on peut calculer facilement la matrice hessienne et son inverse. Cette procédure nécessite des itérations plus nombreuses et couteuses dans les problèmes de grand taille.

2. Comme

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

on voit que le successeur x_{k+1} de x_k n'est pas toujours bien défini.

3. Même si $H^{-1}(x_k)$ est inversible, la direction $d_k = -H^{-1}(x_k)\nabla f(x_k)$ n'est pas toujours une direction de descente. (Si $H(x_k)$ est définie positive, alors d_k est une direction de

descente) .

4. En plus, et d'après le théorème 1.1, on voit, puisque le pas de la progression est égal à $+1$, alors on peut bien avoir

$$f(x_{k+1}) \geq f(x_k).$$

Ceci implique que la méthode de Newton peut converger vers un point stationnaire qui n'est pas un minimum (Maximum ou point selle).

Conclusion

Ceci nous ramène à conclure que la méthode de Newton ne génère pas en général une suite qui converge vers le minimum de la fonction. Mais sous certaines conditions elle devient tres intéressante (Hessien défini positif, point initial assez proche de la solution optimale,...)

Exemple 3.2 $n = 4$,

$$\begin{aligned} f(x) &= 100.(x_2 - x_1^2)^2 + (1. - x_1)^2 + 90.(x_4 - x_3^2)^2 + (1. - x_3)^2 + 10.1[(x_2 - 1.)^2 + \\ &\quad (x_4 - 1.)^2] + 19.8(x_2 - 1.)(x_4 - 1.) \end{aligned}$$

cette fonction est la fonction de Wood, elle admet le point $(1, 1, 1, 1)$ comme un minimum et le point $(-1, 1, -1, 1)$ comme un point selle.

Nous allons appliquer la méthode de Newton pour minimiser cette fonction en prenant $x_0 = (-3, -1, -3, -1)$ et $\varepsilon = 10^{-4}$.

Les resultats obtenues sont illustrés au tableau 3.2, et on remarque bien que la méthode de Newton à convergé vers le point $(-1, 1, -1, 1)$ qui est le point selle de cette fonction.

! ==========

le nombre d'iterations est s= 15

la solution est x= (-9.679741E-01, 9.471393E-01, -9.695163E-01, 9.512478E-01)

la valeur de f(x) est 7.876967

tableau 3.2

itération	$f(x_k)$	$\|\nabla f(x_k)\|$
1	1291.438	16397.13
2	295.9513	1679.932
3	67.68565	1003.095
4	17.33662	285.0620
5	8.689081	106.9116
6	7.892798	26.45835
7	7.876516	3.768905
8	7.877190	0.150338
9	7.876882	0.650068
10	7.876977	0.198463
11	7.876966	0.186190
12	7.876967	0.015349
13	7.876967	0.007221
14	7.876967	0.000112

A cause de ces inconvénients, il existe des modifications qui permettent d'assurer la convergence globale de la méthode de *Newton*, alliant les avantages de la vitesse de convergence quadratique prés de la solution, et ceux d'une méthode de descente. Parmi ces méthodes, on notera par exemple les méthodes dites quasi-Newton discutées au chapitre suivant.

Chapitre 4

Méthodes Quasi-Newton

Introduction:

Ce chapitre est consacré aux méthodes dites Quasi-Newton pour la résolution des problèmes d'optimisation non linéaires sans contraintes. Parmi ces méthodes, on s'étalera particulièrement sur les trois plus importantes, la méthode de correction de rang un, la méthode DFP (Davidon, Fletcher, Powell) et la méthode $BFGS$ (Broyden, Fletcher, Goldfarb, Shano).

4.1 Dérivation des méthodes

Ces méthodes s'inspirent de l'algorithme de *Newton*, mais sans calculer la matrice hessienne de f, ni son inverse. L'itération de la méthode de Newton étant définie par

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

l'idée est de remplacer cette itération par

$$x_{k+1} = x_k - \lambda_k S_k \nabla f(x_k)\,,$$

où λ_k est un paramètre fourni par une recherche linéaire le long de la directon $d_k = -S_k \nabla f(x_k)$, S_k est une approximation symétrique, définie positive, de $H^{-1}(x_k)$. Bien sur, plus S_k sera proche de $H^{-1}(x_k)$, plus l'algorithme convergera rapidement. L'objectif, est donc de trouver une bonne suite de matrices S_k, faciles à construire, c'est à dire utilisant des informations seulement sur le gradient et qui converge vite vers des approximations de plus en plus précises, de l'inverse du hessien de f.
Pour réaliser celà, prenons $f \in C^2(\mathbb{R}^n)$, et faisons un dévelopement de $\nabla f(x)$ au voisinage de x_k,

$$\nabla f(x) = \nabla f(x_k) + H(x_k)(x - x_k) + o(\|x - x_k\|),$$

c.à.d

$$\nabla f(x) \simeq \nabla f(x_k) + H(x_k)(x - x_k), \quad x \in V(x_k),$$

ou encore

$$H^{-1}(x_k)\,[\nabla f(x) - \nabla f(x_k)] \simeq (x - x_k).$$

Les approximations sont exactes si f est quadratique. En particulier, avec $x = x_{k+1}$ et si S_k était une bonne aproximation de $H^{-1}(x_k)$, on devrait avoir

$$S_k\,[\nabla f(x_{k+1}) - \nabla f(x_k)] \simeq (x_{k+1} - x_k), \tag{4.1}$$

mais comme x_{k+1} est calculé aprés 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 équation exactement, d'où

$$S_{k+1}\,[\nabla f(x_{k+1}) - \nabla f(x_k)] = (x_{k+1} - x_k) \tag{4.2}$$

cette équation est dite" équation de la sécante" ou "condition de $quasi - Newton$".
A l'étape k, la remise à jour de la matrice S_k se fait avec une formule simple,

$$S_{k+1} = S_k + C_k,$$

C_k étant une matrice de correction qui intègre au mieux la nouvelle information fournie par x_{k+1} et $\nabla f(x_{k+1})$ de telle manière que S_{k+1} satisfasse la condition (4.2) . Basées sur ce principe, les méthodes $quasi - Newton$ diffèrent l'une par rapport à l'autre, d'aprés la définition de la matrice C_k

Commençons par la méthode de correction de rang un qui est considérée comme une introduction élémentaire aux deux autres méthodes (DFP et $BFGS$) décrites par la suite.

4.2 Méthode de Correction de rang un

Etand donné que $H^{-1}(x_k)$ est symétrique, il est naturel de construire des approximations successives S_k symétriques. Par exemple, on peut exprimer S_{k+1} en fonction de S_k de façon trés simple, en rajoutant à cette dernière une matrice de rang un de la forme suivante: $a_k u_k u_k^t$, où u_k est un vecteur de $\mathbb{R}^n$ et a_k est une constante. On obtiendra alors

$$S_{k+1} = S_k + a_k u_k u_k^t.$$

Montrons qu'il est facile de calculer a_k et u_k de telle sorte que la condition (4.2) soit satisfaite. Posons

$$\begin{aligned} p_k &= x_{k+1} - x_k, \\ q_k &= \nabla f(x_{k+1}) - \nabla f(x_k), \end{aligned}$$

la condition (4.2) s'écrit donc

$$S_{k+1} q_k = p_k.$$

Soit en remplaçant par l'expression de S_{k+1},

$$(S_k + a_k u_k u_k^t) q_k = p_k \,,$$

ou encore

$$a_k u_k (u_k^t q_k) = p_k - S_k q_k,$$

d'ou l'on déduit que u_k est proportionnel à $p_k - S_k q_k$, avec un facteur qui peut être pris en compte dans a_k. En particulier en prenant $u_k = p_k - S_k q_k$ et a_k tel que $a_k(u_k^t q_k) = 1$, on obtient

$$S_{k+1} = S_k + \frac{(p_k - S_k q_k)(p_k - S_k q_k)^t}{(p_k - S_k q_k)^t q_k}.$$

4.2.1 Algorithme

Etape initiale: Soit ε>0, determinant le critère d'arrêt. Choisir un point initial x_1 et une matrice symétrique définie positive S_1. Poser $y_1 = x_1, k = 1$, et aller aux étapes principales.

Etapes principales:

Etape1: Si $\|\nabla f(y_k)\| < \varepsilon$. stop; sinon, poser $d_k = -S_k \nabla f(x_k)$ et soit λ_k la solution optimale du problème $\min f(y_k + \lambda d_k)$, $\lambda \geq 0$.

Etape2: Construire S_{k+1} comme suit

$$S_{k+1} = S_k + \frac{(p_k - S_k q_k)(p_k - S_k q_k)^t}{(p_k - S_k q_k)^t q_k},$$

avec

$$\begin{aligned} p_k &= \lambda_k d_k \equiv y_{k+1} - y_k, \\ q_k &= \nabla f(y_{k+1}) - \nabla f(y_k), \end{aligned}$$

remplacer k par $k+1$, et répéter l'étape 1.

Il est utile de citer le théorème suivant, essentiellement dû à Fiacco et Mc Cromick ([21]), qui montre que lorsque H est constante, la condition (4.2) est non seulement vérifiée pour $i = k$, mais aussi pour tout $i < k$.

Théorème 4.1

Si f est quadratique, de matrice hessienne H et si $p_1, ..., p_n$ sont des vecteurs indépendants, alors la méthode de correction de rang un converge au plus dans $(n+1)$ itérations et $(S_{n+1})^{-1} = H$.

Preuve

La preuve se fait par recurrence. Montrons tout d'abord que

$$p_i = S_{k+1} q_i, \qquad \text{pour } i \leq k \tag{4.3}$$

pour $k = 1$ elle est évidente d'aprés la définition de S_2.

Supposons qu'elle est vraie pour $k > 1$ et montrons qu'elle vraie pour $k + 1$. On a donc

$$S_k q_i = p_i \quad \forall i \leq k - 1.$$

Pour éxploiter la définition de S_{k+1}, calculons

$$(p_k - S_k q_k)^t q_i = p_k^t q_i - q_k^t S_k q_i = p_k^t q_i - q_k^t p_i. \tag{4.4}$$

En remarquant que $q_k = H p_k$, on obtient

$$(p_k - S_k q_k)^t q_i = p_k^t q_i - p_k^t H p_i = p_k^t q_i - p_k^t q_i = 0. \tag{4.5}$$

Donc

$$S_{k+1} q_i = S_k q_i + 0 = p_i, \quad \forall i \leq k - 1.$$

Reste le cas $i = k$. La relation est vraie d'aprés la condition (4.2). Ce qui établit le résultat voulu au rang $k + 1$.

Pour terminer la preuve du théorème et puisque on a,

$$p_i = S_{n+1} q_i = S_{n+1} H p_i, \qquad i = 1, ..., n \tag{4.6}$$

et comme les vecteurs p_i sont indépendents, alors,

$S_{n+1}H = I$ ou encore, $S_{n+1} = H^{-1}$

Exemple 4.1

Considérons le problème suivant

$$(P1) \begin{cases} min\ \frac{1}{2}x^tQx \\ x \in \mathbb{R}^n \end{cases}$$

où

$$Q = \begin{pmatrix} 1 & 0 & 1 & 2 \\ 0 & 3 & 1 & 1 \\ 1 & 1 & 2 & 0 \\ 2 & 1 & 0 & 1 \end{pmatrix}.$$

Appliquons la méthode de correction de rang un sur cet exemple, en prenant $\epsilon = 10^{-5}$ et $y_0 = (0, 1, 2, 3)$. Les résultats obtenues sont illustrés au tableau 4.1, et la recherche linéaire considérée est exacte.

tableau 4.1

k	x_k	$f(x_k)$	$\|\nabla f(x_k)\|$	λ_k
1	(0.000, 1.000, 2.000, 3.000)	15.000	13.000	0.250
2	(−2.005, −1.005, 0.746, 1.997)	6.187	4.357	0.842
3	(0.665, −0.548, 0.004, 0.113)	0.766	0.196	0.444
4	(0.000, −0.001, 0.004, −0.002)	0.002	0.087	0.003
5	(0.000, 0.000, 0.000, 0.000)	0.000	0.000	–

Les matrices S_k obtenues à chaque itération sont

$$S_1 = \begin{pmatrix} 1.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{pmatrix}, \quad S_2 = \begin{pmatrix} 0.927 & 0.140 & 0.117 & 0.134 \\ -0.140 & 0.730 & 0.226 & 0.259 \\ -0.117 & 0.226 & 0.809 & -0.217 \\ -0.134 & -0.259 & 0.217 & 0.751 \end{pmatrix},$$

$$S_3 = \begin{pmatrix} 0.228 & -0.495 & 0.129 & 0.578 \\ -0.495 & 0.549 & -0.101 & 0.103 \\ 0.129 & -0.101 & 0.722 & -0.469 \\ 0.578 & 0.103 & -0.469 & 0.002 \end{pmatrix}, \quad S_4 = \begin{pmatrix} -0.129 & -0.213 & 0.160 & 0.485 \\ -0.213 & 0.328 & -0.126 & 0.176 \\ 0.160 & -0.126 & 0.719 & -0.461 \\ 0.485 & 0.176 & -0.461 & 0.002 \end{pmatrix}$$

D'aprés le tableau, on voit bien que le minimum de f est atteint en 4 itérations.

4.2.2 Avantages

Cette méthode présente l'avantage que le point x_{k+1} n'a pas besoin d'être choisi comme le minimum exact, c'est à dire qu'on n'a pas besoin d'effectuer des recherches linéaires exactes.

4.2.3 Inconvénients

Les inconvénients de cette méthode sont:

1. Même si la fonction est quadratique, et même si son hessien est défini positif, il se peut que la matrice S_k ne soit pas définie positive.

2. Le dénominateur $(p_k - S_k q_k)^t p_k$ peut trés bien devenir nul ou trés petit, rendant le procédé instable.

Dans ce qui suit, on présente les méthodes quasi-Newton, basées sur des formules de correction de rang 2 (DFP, BFGS), qui evitent ces inconvénients.

4.3 Méthode de Davidon-Fletcher-Powell (D.F.P)

Cette méthode a été proposée par Davidon en 1959 ([18]) et développée plus tard en 1963 par Fletcher & Powell ([24]). C'est une méthode assez remarquable pour plusieurs raisons. Tout d'abord, quand elle est appliquée à une fonction quadratique, non seulement les différentes itérations aboutissent à l'inverse du hessien, mais en plus elle engendrent des directions conjuguées. La mise à jour de S_{k+1} en fonction de S_k se fait en ajoutant cette fois deux matrices de rang un, d'où le nom "correction de rang deux". De façon plus précise construisons S_{k+1} en fonction de S_k, de la forme

$$S_{k+1} = S_k + A_k + B_k,$$

avec A_k et B_k deux matrices de rang un chacune et de la forme suivante

$$A_k = a_k u_k u_k^t, \quad B_k = b_k v_k v_k^t,$$

a_k, b_k sont des constantes, u_k, v_k sont deux vecteurs de $\mathbb{R}^n$. S_{k+1} doit satisfaire la condition (4.2), c'est à dire

$$\begin{aligned} (S_k + a_k u_k u_k^t + b_k v_k v_k^t) q_k &= p_k \\ S_k q_k + a_k u_k u_k^t q_k + b_k v_k v_k^t q_k &= p_k, \\ a_k u_k u_k^t q_k + b_k v_k v_k^t q_k &= p_k - S_k q_k, \end{aligned}$$

un choix évident pour satisfaire l'équation est de prendre $u_k = p_k$, $v_k = S_k q_k$, $a_k u_k^t q_k = 1$ et $b_k v_k^t q_k = -1$, d'où

$$S_{k+1}^{DFP} = S_k + \frac{p_k p_k^t}{p_k q_k} - \frac{S_k q_k q_k^t S_k}{q_k^t S_k q_k}. \tag{4.7}$$

Remarque 4.1

Il y'a aussi une formule DFP, pour approcher le hessien lui même, au lieu de son inverse.

Cette formule est donnée par

$$B_{k+1}^{DFP} = B_k + \left[1 + \frac{p_k^t B_k p_k}{p_k^t q_k}\right] \frac{q_k q_k^t}{p_k^t q_k} - \left[\frac{q_k p_k^t B_k + B_k p_k q_k^t}{p_k^t q_k}\right].$$

La procédure est la même que celle de correction de rang un, mais en général il est recommandable d'implémenter les méthodes quasi-Newton avec une reinitialisation periodique pour des raisons de convergence.

L'algorithme de DFP avec reinitialisation est donné sous cette forme.

4.3.1 Algorithme DFP

Etape initiale: Soit ε un scalaire d'arrêt, choisir un point initial x_1 et une matrice symétrique définie positive S_1. Poser $y_1 = x_1, k = j = 1$, et aller aux étapes principales.

Etapes principales:

Etape1: Si $\|\nabla f(y_j)\| < \varepsilon$. stop; sinon, poser $d_j = -S_j \nabla f(y_j)$ et soit λ_j la solution optimale du problème $\min f(y_j + \lambda d_j)$ telque $\lambda \geq 0$. Soit $y_{j+1} = y_j + \lambda_j d_j$.

.Si $j < n$ aller à l'étape 2

.Si $j = n$, poser $y_1 = x_{k+1} = y_{n+1}$, remplacer k par $k + 1$, poser $j = 1$ et aller à l'étape 1.

Etape2: Construire S_{j+1} comme suit:

$$S_{j+1} = S_j + \frac{p_j p_j^t}{p_j q_j^t} - \frac{S_j q_j q_j^t S_j}{q_j^t S_j q_j}$$

avec

$$\begin{aligned} p_j &= \lambda_j d_j \equiv y_{j+1} - y_j, \\ q_j &= \nabla f(y_{j+1}) - \nabla f(y_j), \end{aligned}$$

remplacer j par $j + 1$, et répéter l'étape 1.

Définition 4.1

Soit H une matrice $(n \times n)$ symétrique. Les vecteurs $d_1, ..., d_n$ sont dits $H-$conjugués s'ils sont linéairement indépendents et si $d_i^t H d_j = 0$ pour $i \neq j$

Lemme 4.1 ([15, p.62])

Si H est symétrique définie positive, et si les vecteurs non nuls $d_1, ..., d_n$ sont $H-$conjuguées, alors ils forment une famille libre.

Théorème 4.2

Considérons la méthode DFP décrite par la relation

$$S_{k+1}^{DFP} = S_k + \frac{p_k p_k^t}{p_k q_k^t} - \frac{S_k q_k q_k^t S_k}{q_k^t S_k q_k}.$$

1. A l'étape k, si S_k est symétrique définie positive et si la recherche linéaire est exacte (ou bien si $p_k q_k^t > 0$), alors la matrice S_{k+1} est symétrique définie positive aussi.

2. Si f est quadratique de hessien H, avec en plus H définie positive, alors la méthode DFP vérifie:

(i) $p_i^t H p_j = 0 \qquad 1 \leq i < j \leq k$

(ii) $S_{k+1} H p_i = p_i \qquad 1 \leq i \leq k$

3. Ces propriétés impliquent, dans le cas quadratique, la convergence de la méthode en n itérations et la propriété $S_{n+1} = H^{-1}$.

Preuve

1. Il est évident d'aprés la définition de S_{k+1} qu'elle est symétrique, montrons qu'elle est définie positive. Soit x un vecteur de $\mathbb{R}^n$ non nul,

$$\begin{aligned} x^t S_{k+1} x &= x^t S_k x + x^t \frac{p_k p_k^t}{p_k^t q_k} x - x^t \frac{S_k q_k q_k^t S_k}{q_k^t S_k q_k} x, \\ &= x^t S_k x + \frac{(x^t p_k)^2}{p_k^t q_k} - \frac{(x^t S_k q_k)^2}{q_k^t S_k q_k}, \\ &= x^t S_k x + \frac{(x^t p_k)^2}{p_k^t q_k} - \frac{(x^t S_k q_k)^2}{q_k^t S_k q_k}. \end{aligned}$$

Le terme $(x^t S_k x)(q_k^t S_k q_k) - (x^t S_k q_k)^2$ est positif d'aprés Cauchy Shwartz. En effet, comme S_k est définie positive, alors il existe une matrice $S_k^{\frac{1}{2}}$ définie positive telle que $S_k = S_k^{\frac{1}{2}}.S_k^{\frac{1}{2}}$. Posons $a = S_k^{\frac{1}{2}} x$ et $b = S_k^{\frac{1}{2}} q_k$, d'où

$$x^t S_k x = a^t a,\ q_k^t S_k q_k = b^t b, et x^t S_k q_k = a^t b,$$

alors

$$(x^t S_k x)(q_k^t S_k q_k) - (x^t S_k q_k)^2 = (a^t a)(b^t b) - (a^t b)^2 \geq 0.$$

Donc pour prouver que $x^t S_{k+1} x > 0$, il suffit de montrer que $p_k^t q_k > 0$ et que $q_k^t S_k q_k > 0$.

$$p_k^t q_k = \lambda_k d_k^t [\nabla f(y_{k+1}) - \nabla f(y_k)] = -\lambda_k d_k^t \nabla f(y_k).$$

Puisque y_{k+1} est le minimum le long de la direction d_k et $\nabla f(y_k) \neq 0$, (d'aprés l'hypothèse) donc,

$$p_k^t q_k = \alpha_k \nabla f(y_k) S_k \nabla f(y_k) > 0.$$

De plus $q_k \neq 0$, d'où $x^t S_k x > 0$.

Supposons que

$$x^t S_k x = 0 \Leftrightarrow \begin{cases} (a^t a)(b^t b) = (a^t b)^2 & ...(*) \\ x^t p_k = 0 & ...(**) \end{cases}$$

$$\begin{aligned} (*) &\Leftrightarrow a = \lambda \\ &\Leftrightarrow S_k^{\frac{1}{2}} x = \lambda b \\ &\Rightarrow x = \lambda q_k \end{aligned}$$

$$\begin{aligned} x &\neq 0 \Rightarrow \lambda \neq 0 \\ (**) &\Leftrightarrow p_k^t x = 0 \end{aligned}$$

$$\Rightarrow \quad p_k^t q_k = 0$$

contradiction avec le fait que $p_k^t q_k > 0$.

2. On démontre ce point par récurence pour tout $1 \leq k \leq n$

Pour $k = 1$:

(ii) Notons tout d'abord que pour tout i

$$Hp_i = H(y_{i+1} - y_i) = \nabla f(y_{i+1}) - \nabla f(y_i) = q_i,$$

en particulier $Hp_1 = q_1$

$$S_2 H p_1 = S_1 q_1 + \left[\frac{p_1 p_1^t}{p_1 q_1^t} - \frac{S_1 q_1 q_1^t S_1}{q_1^t S_1 q_1}\right] q_1 = S_1 q_1 - S_1 q_1 + p_1 = p_1.$$

Supposons maintenant que (i), (ii) sont vraies pour k-1, et prouvons qu'elles le sont pour k. On a

$$\nabla f(y_k) = \nabla f(y_{i+1}) + H(y_k - y_{i+1}) = \nabla f(y_{i+1}) + H(p_{i+1} + ... + p_{k-1}).$$

Multiplions par p_i^t,

$$p_i^t \nabla f(y_k) = p_i^t \nabla f(y_{i+1}) + p_i^t H(p_{i+1} + ... + p_{k-1}),$$

d'aprés l'hypothèse de reccurence pour (i) et le fait que y_{i+1} est le minimum de f suivant la direction p_i^t, on a

$$p_i^t \nabla f(y_k) = 0, \qquad \forall 1 \leq i < k,$$

d'aprés l'hypothèse de reccurence pour (ii), $p_i = S_k H p_i, \quad 0 \leq i \leq k$. D'où

$$p_i^t H S_k \nabla f(y_k) = 0,$$

et comme

$$p = -\lambda_k S_k \nabla f(y_k), \quad \text{et} \quad \lambda_k \neq 0,$$

on obtient

$$p_i^t H p_k = 0, \qquad \text{pour } i < k, \tag{4.8}$$

d'ou (i) est vraie pour k.

Maintenant d'aprés (ii) pour $k-1$, $Hp_k = q_k$, et d'après (4.8) on trouve

$$\begin{aligned} q_k^t S_k H p_i &= q_k^t p_i = p_k^t H p_i = 0, \quad 1 \leq i < k \\ S_{k+1} H p_i &= S_k H p_i = p_i, \qquad 1 \leq i < k, \end{aligned}$$

pour $i = k$ on a, $S_{k+1} H p_k = S_k q_k = p_k$, d'où (ii) est vraie pour k.

3. La méthode engendre des directions conjuguées, d'aprés le point 2. De plus, les directions $p_1, ..., p_k$ sont des vecteurs propres (associés à la valeur propre 1) de la matrice $S_{n+1}H$. Elles forment une famille libre, puisque ces vecteurs sont des directions conjuguées. Notons P la matrice telle que ses colonnes sont ces directions, alors P est inversible et d'après (ii)

$$S_{n+1} H P = P \Rightarrow S_{n+1} H = I \Rightarrow S_{n+1} = H^{-1}.$$

Exemple 4.2

Considérons le même problème que l'exemple 4.1 et appliquant la méthode DFP, on trouve les résultats suivants

tableau 4.2

k	x_k	$f(x_k)$	$\|\nabla f(x_k)\|$	λ_k
1	$(0.000, 1.000, 2.000, 3.000)$	15.000	13.000	0.250
2	$(-2.005, -1.005, 0.746, 1.997)$	6.187	4.357	0.812
3	$(0.665, -0.548, 0.004, 0.113)$	0.766	1.969	0.483
4	$(0.000, -0.001, 0.004, -0.002)$	0.002	0.087	0.602
5	$(0.000, 0.000, 0.000, 0.000)$	0.000	0.000	-

Les matrices S_k obtenues à chaque itération sont

$$S_1 = \begin{pmatrix} 1.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{pmatrix}, \quad S_2 = \begin{pmatrix} 0.947 & -0.136 & 0.117 & 0.134 \\ -0.140 & 0.730 & 0.227 & 0.261 \\ -0.123 & 0.227 & 0.811 & -0.213 \\ -0.149 & -0.261 & -0.213 & 0.761 \end{pmatrix},$$

$$S_3 = \begin{pmatrix} 0.176 & -0.495 & 0.129 & 0.567 \\ -0.453 & 0.515 & -0.101 & 0.112 \\ 0.129 & -0.101 & 0.722 & -0.469 \\ 0.567 & 0.112 & -0.469 & 0.002 \end{pmatrix}, \quad S_4 = \begin{pmatrix} -0.129 & -0.213 & 0.160 & 0.485 \\ -0.213 & 0.328 & -0.126 & 0.176 \\ 0.160 & -0.126 & 0.720 & -0.462 \\ 0.485 & 0.176 & -0.462 & 0.002 \end{pmatrix}$$

4.3.2 Avantages et inconvénients de la méthode DFP

Avantages

1 .Pour des fonctions quadratique (avec une recherche linéaire exacte)

(i) elle termine au plus en n étapes avec $S_{n+1} = H^{-1}$

(ii) Elle engendre des directions conjuguées.

2. Pour des fonctions quelconques

(iii) Les matrices S_k restent définies positives. Ceci entraine que les propriétes de descente de la fonction sont vérifiées.

(iv) une convergence superlinéaire

Inconvénients

Cette méthode est assez sensible à la précision de la recherche linéaire (la convergence dépend du fait que la recherche linéaire est exacte ou non).

4.4 Méthode BFGS

Cette méthode a été développée de façon indépendente par Broyden ([10],1970), Fletcher ([25],1970), Goldfarb ([27],1970) et Shanno ([40],1969). Pour construire une approximation de l'inverse du hessien, elle utilise une formule de correction de rang deux qui dérive directement de la formule (4.7). Plus précisement:

$$S_{k+1} = S_k + \frac{p_k p_k^t}{p_k^t q_k} + \frac{q_k^t S_k q_k p_k p_k^t}{(p_k q_k^t)^2} - \frac{p_k q_k^t S_k + S_k q_k p_k^t}{p_k^t q_k}. \tag{4.9}$$

Cette formule est moins sensible aux erreurs dans les recherches linéaire, et elle est considérée comme la meilleure actuellement. On peut se demander ce qui justifie l'introduction d'une telle formule. Une des réponses est que si l'on note $B_k = (S_k)^{-1}$, c.à.d, B_k approxime H_k et non pas son inverse.
la condition quasi-Newton (voir formule 4.2) sera dans ce cas

$$B_{k+1} p_k = q_k, \tag{4.10}$$

et

$$B_{k+1} = B_k + \frac{q_k q_k^t}{q_k^t p_k} - \frac{B_k p_k p_k^t B}{p_k^t B p_k}. \tag{4.11}$$

Comme l'a remarqué Fletcher ([25]), si l'on interverti les rôles de p_k et de q_k dans (4.7), on obtient les matrices verifiant (4.10), qui est la relation inverse de (4.9). On voit alors

que la formule (4.11) permet de construire une approximation de la matrice hessienne elle même (et non pas son inverse). Posons

$$C_k = \frac{q_k q_k^t}{q_k^t p_k} - \frac{B_k p_k p_k^t B}{p_k^t B p_k},$$

d'aprés la relation

$$S_{k+1} = S_k + C_k = B_{k+1}^{-1} = (B_k + C_k)^{-1}, \tag{4.12}$$

et par application de la formule de Sherman – Morrison – woodbury, ([21, p.55]) suivante

$$(A + a^t b)^{-1} = A^{-1} - \frac{A^{-1} a b^t A^{-1}}{1 + b^t A^{-1} a}, \tag{4.13}$$

où A est une matrice $(n \times n)$, b est un vecteur de $\mathbb{R}^n$, et en supposant que

$$(1 + b^t A^{-1} a) \neq 0,$$

on obtient

$$[B_{k+1}]^{-1} = [B_k]^{-1} + \left[1 + \frac{q_k^t [B_k]^{-1} q_k}{p_k^t q_k}\right] \frac{p_k p_k^t}{p_k^t q_k} - \frac{p_k q_k^t [B_k]^{-1} + [B_k]^{-1} q_k p_k^t}{p_k^t q_k}.$$

Ce qui montre que $[B_{k+1}]^{-1}$ est exiprimé directement en fonction de $[B_k]^{-1}$ et d'où la formule de correction de rang (4.9).

Remarques importantes

1. La formule (4.9) à des propriétés analogues à celles de la formule (4.7). En particulier :

(i) Si $p_k^t q_k > 0$, alors la définie positivité des matrices S_k est préservée.

(ii) Appliquée à une fonction quadratique (matrice hessienne H définie positive), cette formule permet d'obtenir en au plus n itérations, l'inverse H^{-1} du hessien de f. De plus les directions p_k engendrées par l'algorithme BFGS sont mutuellement conjuguées par rapport à H.

2. Lorsque l'algorithme BFGS ou DFP est appliqué à une fonction non linéaire quelconque

(non quadratique), il faut procéder à des réinitialisations périodiques, pour assurer la convergence globale.

3. Théoriquement, la condition $p_k^t q_k > 0$ assure la définie positivité des matrices S_{k+1} ou B_{k+1}, mais sur le plan pratique rien n'est garanti, à cause des erreurs d'arrondi, S_{k+1} ou B_{k+1} peuvent devenir singulières ou non définies.

4. La majorité des auteurs affirme la superiorité de l'algorithme BFGS sur celui de DFP, soit sur le plan théorique ou pratique (il est plus stable).

Exemple 4.3

Si on considère le même exemple traité par DFP et la méthode de correction de rang un pour illustrer la remarque1(ii), on trouve les résultats si dessous

tableau 4.3

k	x_k	$f(x_k)$	$\|\nabla f(x_k)\|$	λ_k
1	$(0.000, 1.000, 2.000, 3.000)$	15.000	13.000	0.250
2	$(-2.005, -1.005, 0.746, 1.997)$	6.187	4.357	0.732
3	$(0.665, -0.548, 0.004, 0.113)$	0.766	0.196	0.393
4	$(0.000, -0.001, 0.004, -0.002)$	0.002	0.087	0.601
5	$(0.000, 0.000, 0.000, 0.000)$	0.000	0.000	–

et les matrices S_k obtenues à chaque itération sont:

$$S_1 = \begin{pmatrix} 1.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{pmatrix}, \quad S_2 = \begin{pmatrix} 1.017 & -0.124 & -0.141 & -0.196 \\ -0.124 & 0.732 & -0.230 & 0.270 \\ -0.141 & 0.230 & 0.815 & -0.200 \\ -0.198 & -0.270 & -0.200 & 0.796 \end{pmatrix},$$

$$S_3 = \begin{pmatrix} 0.309 & -0.561 & 0.129 & 0.595 \\ -0.561 & 0.603 & -0.101 & 0.008 \\ 0.129 & -0.101 & 0.722 & -0.469 \\ 0.595 & 0.008 & -0.469 & 0.002 \end{pmatrix}, \quad S_4 = \begin{pmatrix} -0.129 & -0.213 & 0.160 & 0.485 \\ -0.213 & 0.328 & -0.126 & 0.176 \\ 0.160 & -0.126 & 0.722 & -0.463 \\ 0.485 & 0.176 & -0.463 & 0.006 \end{pmatrix}.$$

Implémentation numérique

On a mis en oeuvre les méthodes quasi-Newton (DFP et BFGS) définies précédement, et comme fonction test on a pris la fonction de Rosenbrock généralisée, définie par

$$f(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$$

Comme point de départ on prend $x_0 = (-1.2, 1, ..., -1.2, 1)^t$

Notons que le minimum de cette fonction est atteint au point $x^* = (1, 1, ..., 1)^t$ avec $f(x^*) = 0$

On a implémenté l'algorithme (DFP et BFGS) dans le cas continu et dans le cas avec reinitialisation en introduisant la recherche linéaire de Wolfe (chapitre 2).

On a pris une précision $\varepsilon = 10^{-5}$ et $S_0 = I$.

Les résultats obtenus sont présentés dans les tableaux si dessous.

tableau 4.4: DFP et BFGS implémentés sans reinitialisation

n	DFP			$BFGS$		
	k	Ef	$f(x)$	k	Ef	$f(x)$
2	692	759	3.75×10^{-15}	34	88	6.73×10^{-20}
4	1781	1883	5.69×10^{-15}	42	123	1.12×10^{-15}
10	3483	3543	2.08×10^{-13}	85	183	1.09×10^{-14}
30	pas de convergence	–	–	174	447	3.30×10^{-16}

tableau 4.5: DFP et BFGS implémentés avec reinitialisation

n	DFP			$BFGS$		
	k	Ef	$f(x)$	k	Ef	$f(x)$
2	55	292	4.65×10^{-11}	45	265	1.04×10^{-11}
4	99	706	1.03×10^{-11}	69	542	7.95×10^{-14}
10	193	1669	1.18×10^{-12}	182	1403	0.39
30	572	2933	1.44×10^{-13}	586	2868	3.30×10^{-16}

k est le nombre d'itérations, Ef est le nombre d'évaluations fonctionnelles.

On a aussi implémenté (DFP et BFGS) avec la procédure d'Armijo (voir chap2, recherches linéaires inexactes). Les résultats sont illustrés dans les tableaux si dessous

tableau 4.6: DFP et BFGS implémentés sans reinitialisation

n	DFP			$BFGS$		
	k	Ef	$f(x)$	k	Ef	$f(x)$
2	39	193	7.36×10^{-14}	42	190	5.04×10^{-15}
4	68	386	1.34×10^{-15}	48	238	5.40×10^{-16}
10	121	663	3.98*	68	335	3.98*
50	pas de convergence	–	–	266	1290	9.96×10^{-16}

tableau 4.7: DFP et BFGS implémentés avec reinitialisation

n	DFP			$BFGS$		
	k	Ef	$f(x)$	k	Ef	$f(x)$
2	55	312	5.05×10^{-11}	47	271	4.75×10^{-11}
4	69	554	6.25×10^{-11}	57	507	2.31×10^{-13}
10	191	1864	3.98*	181	1823	3.98*
50	967	5673	3.98*	844	5822	4.26×10^{-14}

la valeur 3.98 correspond à un minimum local de f, $x^* = (-1, 1, ..., 1)$ souvent rencontré avec $n = 10$, $n = 50$.

D'aprés ces tests on confirme deux points essentiels notés dans les remarques précédentes

1. La superiorité et la stabilité de l'algorithme BFGS sur celui de DFP.

2. L'utilité de la reinitialisation, comme le montre le cas $n = 30$, avec Wolfe et $n = 50$ avec Armijo.

On a aussi implémenté ces méthodes avec d'autres fonctions tests et les résultats sont représentés dans l' annexe A à la fin de ce mémoire.

4.5 Convergence des méthodes quasi-newton

4.5.1 Cas où la recherche linéaire est exacte

Dans ce cas, le théorème (4.2) montre que si f est une fonction quadratique, de hessien défini positif, la convergence se fait dans un nombre fini ditérations.

Dans le cas général et en 1972 Powell ([36]) a montré que si f était convexe, deux fois continuement différentiable, alors la méthode DFP converge globalement vers la solution optimale. Sous des conditions plus fortes que celles que nous venons de citer, le même auteur ([37]) démontra que la méthode DFP convergeait vers la solution optimale de façon superlinéaire.

Maintenant on va prouver la convergence des méthodes quasi-Newton en utilisant le théorème1.6 (Voir Chap.1, Fonctions mutivoques et Algorithmes). Pour celà on note par A notre algorithme tel que $A = CB$ où A et B sont décris par la suite. Alors on doit vérifier les propriétés suivantes:

1.B est fermé sur $C\Omega = \mathbb{R}^n \backslash \Omega$

2. Si $y \in B(x) \Rightarrow f(y) \leq f(x) \qquad \forall x \notin \Omega.$

3. Si $z \in C(y) \Rightarrow f(z) \leq f(y).$

4. L'ensemble $\Lambda = \{x : f(x) \leq f(x_0)\}$ est compact, où x_0 est le point de départ.

Pour les méthodes quasi-Newton, B est définie comme suit:

donnons un point x, alors $y \in B(x)$ si y est obtenu en minimusant f à partir de x le long de la direction $d = -S_0 \nabla f(x)$, où S_0 est une matrice arbitraire définie positive. De plus en démarrant du point obtenu par B, l'application C donne le minimum de f le long des directions générées, il suit que C vérifie la troisième condition. Il nous reste donc à montrer que B est fermée sur le complémentaire de Ω, où $\Omega = \{x : \nabla f(x) = 0\}$. En effet, soit $x_k \notin \Omega$, $x_k \to x$, et soit $y_k \in B(x_k)$, $y_k \to y$, on doit démontrer que $y \in B(x)$.
Par définition de y_k, $y_k = x_k - \lambda_k S_0 \nabla f(x_k)$ pour $\lambda_k \geq 0$ tel que,

$$f(y_k) \leq f(x_k - \lambda S_0 \nabla f(x_k)) \qquad \forall \lambda \geq 0, \tag{4.14}$$

comme $\nabla f(x) \neq 0$ (car $x \notin \Omega$), alors

$$\lambda_k \to \vec{\lambda} = \frac{\|y - x\|}{\|S_0 \nabla f(x)\|} \geq 0,$$

de plus, quand $k \to \infty$, $y = x - \overline{\lambda} S_0 \nabla f(x)$. Alors

$$f(y_k) \leq f(x_k - \lambda S_0 \nabla f(x_k)) \qquad \forall \lambda \geq 0,$$

donc y est obtenu en mimimisant f le long de la direction $-S_0 \nabla f(x)$,-c.à.d- $y \in B(x)$, d'où B est fermé sur $C\Omega$.

Notons encore que -$\nabla f(x)^t S_0 \nabla f(x) < 0$ (car S_0 est définie positive), ce qui justifie que d est une direction de descente -c.à.d- $y \in B(x) \Rightarrow f(y) \leq f(x)$, $\forall x \notin \Omega$.

Supposons de plus que Λ est compact, pour que les quatre conditions soient satisfaites et par conséquent, en cosidérant le théorème 1.6, il suit que l'algorithme A converge vers un point stationnaire.

4.5.2 Cas où la recherche linéaire est inexacte

Dans ce cas, le théorème 4.2, de convergence finie, n'est plus valable pour les méthodes DFP et BFGS car sa preuve nécessite les propriétés de la recherche linéaire éxacte.

Pour les fonctions non quadratiques, sans utiliser une recherche linéaire exacte, et sous certaines conditions, Powell, en 1976 ([38]) a montré qu'une version de la méthode metric-variable (méthodes DFP et BFGS) converge vers la solution optimale si la fonction objective f était convexe. Il montra aussi dans le même contexe que si la matrice hessienne de f était définie positive au point x^* (x^* solution optimale), alors la convergence était superlinéaire.

Dans notre mémoire on utilisé les méthodes DFP et BFGS avec les recherches linéaire inexactes de Wolfe et d'Armijo (voir Chap.2). Donc il est nécessaire de donner les théorèmes qui caractérisent la convergence de ces méthodes.

Théorème 4.3

Soit l'algorithme $x_{k+1} = x_k - \lambda_k S_k \nabla f(x_k)$, *où* $\{S_k\}$ *est une suite de matrices définies positives dont les spectres sont uniformément bornés* $(0 < c_1 \leq \min_i m_i(S_k) \leq \max_i m_i(S_k) \leq c_2)$. *Si* $f \in C^1(\mathbb{R}^n)$, *minorée, et à gradient Lipschitsien de constante* L, *et si les* λ_k *sont choisis selon la méthode de sélection d'Armijo (voir recherche linéaire d'Armijo, Chap.2), alors cet algorithme converge vers un mimimum local de* f.

Preuve

L'algorithme est bien une méthode de descente, car

$$\nabla f(x_k)^t d_k = -\|\nabla f(x_k)\|^2_{S_k} \leq -c_1 \|\nabla f(x_k)\|^2,$$

par ailleurs, en appliquant la formule de la moyenne à la fonction $\lambda \mapsto f(x_k + \lambda d_k)$, on obtient

$$f(x_k + \lambda d_k) = f(x_k) + \nabla f(\xi_k)^t(\lambda d_k),\ \xi_k \in [x_k\ ,\ x_k + \lambda d_k].$$

On peut donc écrire

$$f(x_k + \lambda d_k) - f(x_k) = (\nabla f(\xi_k) - \nabla f(x_k))^t(\lambda d_k) + \nabla f(x_k)^t(\lambda d_k),$$

ce qui, en utilisant le caractère lipschitizien du gradient de f et la définition de d_k,

$$\begin{aligned} f(x_k + \lambda d_k) - f(x_k) &\leq L\|\lambda d_k\|^2 + (\nabla f(x_k))^t(\lambda d_k), \\ &\leq L\lambda^2 \|S_k \nabla f(x_k)\|^2 - \lambda \nabla f(x_k)^t S_k \nabla f(x_k). \end{aligned}$$

Le premier terme du membre de droite étant majoré par $L\lambda^2 c_2 \|\nabla f(x_k)\|^2_{S_k}$, on peut conclure que

$$f(x_k + \lambda d_k) - f(x_k) \leq \lambda(L\lambda c_2 - 1)\|\nabla f(x_k)\|^2_{S_k}. \tag{4.15}$$

Rappelons que la méthode de sélection d'armijo, dans le cas d'une direction de descente

$d_k = -S_k \nabla f(x_k)$ est

$$f(x_k + \lambda d_k) - f(x_k) \leq -\lambda\varepsilon \left\|\nabla f(x_k)\right\|^2_{S_k}.$$

La constante ε étant fixée, il suffit de choisir λ_k tel que $Lac_2 - 1 \leq -\varepsilon$,c.-à-d. $\lambda_k \leq \dfrac{1-\varepsilon}{c_2 L}$ dans (4.15) pour remplir la condition d'armijo. Par ailleurs, compte tenu du principe de sélection, on aura $\lambda_k \geq \dfrac{1-\varepsilon}{2c_2 L}$.
On en déduit d'une part qu'avec ce choix, on a $f(x_{k+1}) = f(x_k + \lambda_k d_k) < f(x_k)$, et donc que la suite $\{f(x_k)\}_{k\in\mathbb{N}}$ converge, puisqu'elle est minorée, et d'autre part que $\left\|\nabla f(x_k)\right\|^2_{S_k} \leq \dfrac{f(x_k) - f(x_{k+1})}{\varepsilon\lambda_k}$. Le terme de droite convergeant vers 0; donc $\|\nabla f(x_k)\| \to 0$, puisque tous les S_k ont un spectre strictement positif . La méthode proposée converge donc vers un minimum local de f.

Théorème 4.4

Supposons que f est convexe et possède des dérivées secondes continues sur l'ensemble borné $\{x : f(x) \leq f(x_1)\}$. Alors l'algorithme BFGS avec recherche linéaire de wolfe vérifie

$$\lim_{k\to+\infty} \inf \|\nabla f(x_k)\| = 0.$$

Preuve

Soit

$$\begin{aligned} p_k &= x_{k+1} - x_k, \\ q_k &= \nabla f(x_{k+1}) - \nabla f(x_k). \end{aligned}$$

Les hypothèses de ce théorème impliquent

$$\frac{\|q_k\|^2}{(p_k, q_k)} \leq M, \tag{4.16}$$

où M est une constante vérifiant

$$\|H_k\| \leq M,$$

(pour la démonstation de cette partie voir lemme 5.2, chapitre 5).

Prenons maintenant la formule BFGS approchant le Hessien. La trace et le déterminant de B_{k+1} sont donnés par

$$tr(B_{k+1}) = tr(B) + \frac{\|q_k\|^2}{p_k^t q_k} - \frac{\|B_k p_k\|^2}{p_k^t B_k p_k} \tag{4.17}$$

$$\det(B_{k+1}) = \det(B_k)\frac{p_k^t q_k}{p_k^t B_k p_k}. \tag{4.18}$$

pour la relation (4.18) voir ([35], appendice B)

L'inégalité (4.16) et la notion de trace impliquent la majoration suivante

$$\begin{aligned} tr(B_{k+1}) &< tr(B_1) + \sum_{j=1}^{k} \frac{\|q_j\|^2}{P_j^t q_j} \\ &\leq tr(B_1) + Mk \\ &\leq c_3 k, \quad k = 1, 2, 3, .. \end{aligned}$$

où $c_3 = tr(B_1) + M$.

Rapplons ici que la moyenne arithmétique est superieure à la moyenne géométrique : si $a_1, a_2, ..., a_n$ sont k nombres positifs, alors

$$\left(\frac{1}{k}\sum_{j=1}^{k} a_j\right)^k \geq \prod_{j=1}^{k} a_j \tag{4.19}$$

Appliquons ce resultat aux valeurs propres de la matrice B_{k+1}, on obtient la relation

$$\det(B_{k+1}) < \frac{(c_3 k)^n}{n}, \quad k = 1, 2, 3, ... \tag{4.20}$$

Puisque la trace de B_{k+1} est positive, on déduit aussi que

$$\begin{aligned}\sum_{j=1}^{k} \frac{\|B_j p_j\|^2}{p_j^t B_j p_j} &< tr(B_1) + \sum_{j=1}^{k} \frac{\|q_j\|^2}{p_j^t q_j}, \\ &\leq c_3 k, \quad k = 1, 2, 3, ...\end{aligned}$$

Appliquons encore une fois la relation (4.19), on trouve

$$\prod_{j=1}^{k} \frac{\|B_j p_j\|^2}{p_j^t B_j p_j} < c_3^k, \quad k = 1, 2, 3, ... \tag{4.21}$$

De plus (4.18) nous donne

$$\frac{\det(B_{k+1})}{\det(B_1)} = \prod_{j=1}^{k} \frac{p_j^t q_j}{p_j^t B_j p_j}. \tag{4.22}$$

D'après (4.20), (4.21), et (4.22), on déduit:

$$\begin{aligned}\prod_{j=1}^{k} \frac{\|B_j p_j\|^2 p_j^t q_j}{\left(p_j^t B_j p_j\right)^2} &< \frac{\left(c_3^k/n\right)^n c_3^k}{\det(B_1)}, \\ &\leq c_4^k, \quad k = 1, 2, 3, ...\end{aligned} \tag{4.23}$$

remplaçons $B_j p_j = B_j \lambda_j d_j = \lambda_j B_j(-B_j^{-1} \nabla f_j) = -\lambda_j \nabla f_j$, (où $\nabla f(x_j)$ est noté par ∇f_j) on obtient

$$\prod_{j=1}^{n} \frac{\|\nabla f_j\|^2 p_j^t q_j}{(-p_j^t \nabla f_j)^2} < c_4^k, \; k = 1, 2, 3, ...,$$

Rappelons que la recherche de Wolfe permet de donner

$$\begin{aligned} d_k^t \nabla f(x_k + \lambda_k d_k) &\geq c_2 d_k^t \nabla f_k, \\ d_k^t \left[\nabla f(x_k + \lambda_k d_k) - \nabla f(x_k)\right] &\geq d_k^t \nabla f_k (c_2 - 1), \\ p_k^t q_k &\geq (-p_k^t \nabla f_k)(1 - c_2), \end{aligned}$$

d'où,

$$\frac{p_k^t q_k}{-p_k^t \nabla f_k} \geq 1 - c_2,$$

et il suit que

$$\prod_{j=1}^{k} \frac{\|\nabla f_j\|^2}{-p_j^t \nabla f_j} < \left(\frac{c_4}{1-c_2}\right)^k,$$

où bien

$$\prod_{j=1}^{k} \frac{\|\nabla f_j\|^2}{\|p_j\| \cos\theta_j} < \left(\frac{c_4}{1-c_2}\right)^k, k = 1, 2, 3, ..., \tag{4.24}$$

où θ_j est l'angle compris entre p_j et $-\nabla f_j$.

Comme $f(x)$ est bornée inferieurement, la condition (Wolfe) nous permet de conclure que la somme ?

$$\sum_{k=1}^{\infty} \lambda_k d_k^t \nabla f_k = \sum_{k=1}^{\infty} \|p_k\| \, \|\nabla f_k\| \cos(\theta_k),$$

est convergente.

Donc si on suppose qu'il existe un $\varepsilon > 0$ telque $\|\nabla f_k\| \geqslant \varepsilon > 0$ pour tout k, alors

$$\|p_k\| \cos(\theta_k) \longrightarrow 0.$$

Dans ce cas les termes $\dfrac{\|\nabla f_j\|^2}{\|p_j\| \cos\theta_j} \longrightarrow \infty$ quand $j \longrightarrow \infty$
qui donne une contradiction avec(4.24), et par conséquent

$\lim_{k \to +\infty} \inf \|\nabla f(x_k)\| = 0.\square$

Chapitre 5

Methode BFGS sans calcul de dérivées

Introduction:

Nous avons rencontré dans le chapitre précédent la méthode $BFGS$ réputée d'être parmi les plus performantes méthodes d'optimisation sans contraintes dans sa catégorie. Sous certaines conditions ($f \in C^2(\mathbb{R}^n)$, f convexe, la matrice Hessienne de f définie positive au voisinage de la solution optimale,(voir [38] pour plus de détails), la méthode $BFGS$ converge de façon superlinéaire. Dans la pratique et malgré que f soit assez régulière ($f \in C^1(\mathbb{R}^n)$ ou $f \in C^2(\mathbb{R}^n)$, il s'avère que quelque fois il est difficile ou même impossible d'exprimer analytiquement $\nabla f(x)$. La résolution du problème se fera alors de préférence par des techniques n'éxigeant que le calcul des valeurs de la fonction. Le calcul du gradient par une formule analytique est alors remplacé par une approximation convenable. De là, des méthodes se sont développées mais elles ont considéré une classe bien particulière de fonctions et la vitesse de convergence des méthodes quasi-Newton n'a pas été sauvegardée. En 1981, Brauninger ([7]) a réussi à construire un algorithme possédant les mêmes propriétes que la méthode $BFGS$ (hypothèse sur f identiques, même vitesse de convegence). Le grand avantage de cet algorithme, c'est qu'il évite le calcul du gradient. On consacre tout ce chapitre à ce travail.

5.1 Formulation du problème, définitions et notations

Soit $f : \mathbb{R}^n \to \mathbb{R}$ et (P') le problème d'optimisation non linéaire sans contraintes suivant:

$$\textit{minimiser } \{f(x) : x \in \mathbb{R}^n\}.$$

Notations

Si f est différentiable en un point x_k, on note par $\nabla f(x_k)$ ou bien ∇f_k son gradient au point x_k, et si elle est deux fois continuement différentiable on garde la même notation que dans les chapitres précédents pour le hessien de f au point x_k,c.à.d $H(x_k)$. Pour $X \subset \mathbb{R}^n$, et $x \in \mathbb{R}^n$, notons par

$$d(x, X) = \inf \{\|x - y\| : y \in X\}$$

5.1.1 Hypothèses de départ

L'Algorithme démarre avec un point initial $x_0 \in \mathbb{R}^n$ quelconque et un $\alpha_0 > 0$ donné. Nous considererons tout le long de ce chapitre que la supposition suivante est vraie.

Supposition 5.1

(i) $X_0 = \{x / f(x) \leq f(x_0),\}$ *est borné*

(ii) $f(x)$ *est convexe et deux fois continuement différentiable sur l'ensemble* X *défini comme suit*

$$X_0 \cup \{x / d(x, X_0) < \alpha_0\} \subset X.$$

Comme ensemble X, on pourrait prendre une boule fermée de centre a et de rayon R : $B(a, R)$ telle que

$$X_0 \cup \{x / d(x, X_0) < \alpha_0\} \subset int(B(a, R))$$

Il est clair que X_0 et X sont compacts. En pratique, on aimerait que pour tout point

initial $x_0 \in \mathbb{R}^n$, l'ensemble X_0 reste borné. Ceci nous ramène à la notion d'inf compacité suivante

5.1.2 L'inf-compacité

Définition 5.1

Soit $f : \mathbb{R}^n \to \mathbb{R}$, f est dite inf compacte, si pour tout $\lambda \in \mathbb{R}$, les ensembles suivants:

$$\{y \in \mathbb{R}^n : f(y) \leq \lambda\}$$

sont compacts.

Proposition 5.1 ([32], p.349)

Si f est convexe et semi-continue inferieurement et s'il existe $\lambda_0 \in \mathbb{R}$ tel que l'ensemble

$$X_{\lambda_0} = \{y \in \mathbb{R}^n : f(y) \leq \lambda_0\}$$

soit compact et

$$\inf\{f(y) : y \in \mathbb{R}^n\} < \lambda_0.$$

Alors f est inf compacte.

Remarque importante

Les hypothèses de la proposition 5.1 sont bien vérifiées dans notre cas. En effet la première hypothèse est vérifiée en prenant $\lambda_0 = f(x_0)$. Pour la seconde proposition on a bien

$$\inf\{f(y) : y \in \mathbb{R}^n\} < f(x_0),$$

sinon x_0 serait notre solution optimale.

5.2 Approximation du Gradient

Pour éviter le calcule du gradient à chaque itération, on approxime ∇f_k par le vecteur $gr(f, x_k, \alpha)$ qui dépend d'un paramètre d'approximation $\alpha > 0$. La i^{eme} composante de ce vecteur est définie comme suit

$$g_r(f, x_k, \alpha)_i = \begin{cases} \dfrac{f(x_k + \alpha e_i) - f(x_k - \alpha e_i)}{2\alpha}, & \text{si } \alpha \leq 10^{-6} \\ \dfrac{f(x_k + \alpha^2 e_i) - f(x_k)}{\alpha^2}, & \text{si } \alpha > 10^{-6} \end{cases}$$

La valeur 10^{-6} dépend de la précision de l'ordinateur utilisé.

Le théorème suivant montre la précision de cette approximation

Théorème 5.1

Soit $x_k \in X_0$, alors,

$$\lim_{\alpha \to 0} \frac{1}{\alpha} \|g_r(f, x_k, \alpha) - \nabla f_k\| = 0.$$

Preuve

$x_k \in X_0$ donc $x_k + \alpha e_i \in X$, $x_k - \alpha e_i \in X$ pour α suffisement petit.
En utilisant la formule de taylor sur $[x_k + -\alpha e_i, x_k]$, on obtient

$$f(x_k - \alpha e_i) = f(x_k) + \alpha e_i^t \nabla f(x_k) + \frac{1}{2}\alpha^2 e_i^t H(\xi_k) e_i,$$

avec

$$\xi_k = \theta(x_k - \alpha e_i) + (1 - \theta)x_k = x_k - \theta\alpha e_i, \qquad \theta \in]0, 1[.$$

Maintenant, considérons la formule de taylor sur $[x_k, x_k + \alpha e_i]$, on obtient

$$f(x + \alpha e_i) = f(x_k) + \alpha e_i^t \nabla f(x_k) + \tfrac{1}{2}\alpha^2 e_i^t H(\xi_k) e_i,$$

avec $\eta_k = x_k + (1 - \theta')\alpha e_i, \qquad \theta' \in]0, 1[$

D'où l'on déduit que

$$f(x+\alpha e_i) - f(x-\alpha e_i) = 2\alpha e_i^t \nabla f(x_k) + \frac{1}{2}\alpha^2 e_i^t \left[H(\eta_k) - H(\xi_k)\right] e_i,$$

où bien

$$\frac{f(x+\alpha e_i) - f(x-\alpha e_i)}{2\alpha} - (\nabla f_k)_i = \frac{1}{4}\alpha e_i^t \left[H(\eta_k) - H(\xi_k)\right] e_i,$$

de sorte qu'on obtient

$$\begin{aligned} \frac{1}{\alpha}\max_i \left|(g_k)_i - (\nabla f_k)_i\right| &= \max_i \left|\tfrac{1}{4} e_i^t \left[H(\eta_k) - H(\xi_k)\right] e_i\right|, \\ &\leq \max_i \left\|H(\eta_k) - H(\xi_k)\right\|. \end{aligned}$$

Remarquant que $\eta_k, \xi_k \in X$, (X est convexe) et que $\eta_k \to x_k, \xi_k \to x_k$ quand $\alpha \to 0$, et comme

$$H \text{ continue sur un compact} \Rightarrow H \text{ uniformemernt continue},$$

alors le deuxième terme tend vers 0 quand $\alpha \to 0$, celà implique

$$\frac{1}{\alpha}\left\|g_r(f, x_k, \alpha) - \nabla f_k\right\| \to 0 \quad \text{quand} \quad \alpha \to 0.$$

5.3 Algorithme

L'algorithme débute avec un point initial arbitraire $x_0 \in \mathbb{R}^n$,un paramètre initial d'approximation $\alpha_0 > 0$, et une matrice symétrique définie positive B_0(la matrice identité pourrait convenir). La suite générée par l'algorithme est de la forme suivante $x_{k+1} = x_k - \lambda_k d_k$ où λ_k est un scalaire fourni par une recherche linéaire suivant la direction de descente -$d_k \in \mathbb{R}^n$. Dans la recherche linéaire on utilise deux paramètres de perturbation c_1, c_2 qui sont arbitraires au départ et fixés le long du procédé.

Citons maintenant les étapes de cet algorithme.

Etape Initiale:

Choisir un point initial x_0, un paramètre $\alpha_0 > 0$, et une matrice symétrique définie positive initiale B_0, poser $k = 0$ et aller aux étapes principales.

Etapes principales:

Etape1: a) Calculer $g_k = gr(f, x_k, \alpha_k)$

b) si $\|g_k\| < 0.001\alpha_k$. poser $\alpha_k = (0.5)\alpha_k$ et aller à l'étape 1a).

Etape2: Calculer la direction d_k telque $d_k = B_k^{-1} g_k$.

Etape3: a) Soit t le plus petit entier non négatif telque avec $\lambda_a = (0.5)^t$

$$f(x_k - \lambda_a d_k) < f(x_k) - c_1 \lambda_a d_k^t g_k \tag{5.1}$$

ou bien

$$\lambda_a < \alpha_k$$

si $\lambda_a < \alpha_k$, poser $\alpha_k = (0.5)\alpha_k$ et aller à l'étape 1a).

b) Soit t le plus petit entier non négatif telque avec $\lambda_b = 2^t \lambda_a$

$$f(x_k - \lambda_b d_k) > f(x_k) - c_2 \lambda_b d_k^t g_k. \tag{5.2}$$

c) Si $\lambda_b - \lambda_a < \alpha_k$. Poser $\lambda_k = (0.5)\lambda_k$.

Poser $\lambda_m = \dfrac{\lambda_a + \lambda_b}{2}$.

d) Si λ_m vérifie

$$f(x_k - \lambda_m d_k) > f(x_k) - c_1 \lambda_m d_k^t g_k,$$

poser $\lambda_b = \lambda_m$ et aller à l'étape 3c).

Si λ_m vérifie

$$d_k^t gr(f, x_k - \lambda_m d_k) > f(x_k) - c_1 \lambda_m d_k^t g_k,$$

aller à l'étape 3c). Sinon poser $\lambda_k = \lambda_m$.

Etape 4: $x_{k+1} = x_k - \lambda_k d_k$, $g_{k+1} = gr(f, x_{k+1}, \alpha_k)$, $p_k = x_{k+1} - x_k$,

$q_k = g_{k+1} - g_k$.

Si $p_k^t q_k < \alpha_k \|p_k\|$, poser $\alpha_k = (0.5)\alpha_k$ et aller à l'étape 1a). Sinon poser

$$B_{k+1} = B_k - \frac{B_k p_k p_k^t B_k}{p_k^t B_k p_k} + \frac{q_k q_k^t}{p_k^t q_k},$$

$\alpha_{k+1} = (0.5)\alpha_k$, remplacer k par $k+1$ et aller à l'étape 1b).

5.3.1 Description de l'algorithme

Le principe de cet algorithme est le même que la méthode de $BFGS$ à la différence de l'utilisation de l'approximation du gradient.

•A l'étape 1b) on décide si l'approximation est suffisante ou pas. Si $\|g_k\| < 0.001\alpha_k$, l'approximation est considérée comme insuffisante, et avec un $\alpha_k = (0.5)\alpha_k$, un autre g_k doit être défini. On montre par la suite que si on obtient un cycle infini, alors ce point est un point stationnaire .

•A l'étape 3) on introduit une recherche linéaire inexacte dans le but d'assurer la diminution de la fonction objective, et par suite la convergence de l'algorithme. Cette recherche linéaire est une combinaison entre les deux recherches linéaires inexactes de Goldstein et de Wolfe présentées au chapitre 2, et elle est présentée en trois étapes

3a) On détermine un λ_a telque (5.1) soit satisfaite, si celà n'est pas possible pour $\lambda_a \geq \alpha_k$, l'approximation est considérée encore comme insuffisante et on doit redémarrer avec une autre approximation pour $\alpha_k = \frac{\alpha_k}{2}$.

3b) On détermine un λ_b telque (5.2) soit satisfaite, l'existence de λ_b est justifiée par la suppositon 1(i) et le faite que $d_k^t g_k > 0$.

3c) et 3d): On trouve finalement un longueur du pas satisfaisante vérifiant

$$f(x_k - \lambda d_k) \leq f(x_k) - c_1 \lambda d_k^t g_k \tag{5.3}$$

$$d_k^t gr(f, x_k - \lambda d_k, \alpha_k) \leq c_2 d_k^t g_k \tag{5.4}$$

•A l'étape 4, on vérifie un test trés important dans notre théorie de convergence qui est $p_k^t q_k \geq \alpha_k \|p_k\|$.

Comme préparation à l'étude de convergence de cet algorithme on donne ce lemme qui resume quelques propriétées.

Lemme 5.1

(i) $\exists\ \tau > 0$ *tel que* $\|H(x)\| \leq \tau$ *pour tout* $x \in X$.

(ii) $\{g_k\}$ *est bornée.*

(iii) B_k *et* B_k^{-1} *sont définies positives por tout* $k = 0, 1, 2, ...$

(iv) $f(x_{k+1}) < f(x_k)$ *pour tout* $k = 0, 1, 2, ...$

Preuve

(i) Par définition de la norme et la continuité de H sur le compact X on trouve

$$\begin{aligned} \|H(x)\| &= \max_{1\leq i\leq n} \Big(\sum_{j=1}^{n} \left|\frac{\partial^2 f}{\partial x_i \partial x_j}(x)\right|\Big), \\ &\leq \max_{1\leq i\leq n} \Big(\sum_{j=1}^{n} \max_{x\in X} \left|\frac{\partial^2 f}{\partial x_i \partial x_j}(x)\right|\Big), \\ &= \tau. \end{aligned}$$

(ii) D'aprés le théorème (5.1) on a

$$\|g_k\| = \max_i |(g_k)_i| = \max_i \left|(\nabla f_k)_i + \tfrac{1}{4}\alpha_k e_i^t \left[H(\eta_k) - H(\xi_k')\right] e_i\right|,$$

en utilisant *(i)* on trouve

$$\|g_k\| \leq \max_i |(\nabla f_k)_i| + \tfrac{1}{2}\alpha_k \tau,$$

d'autre part

$$\begin{aligned}\max_i |(\nabla f_k)_i| &= \max_i \left|\frac{\partial f}{\partial x_i}(x_k)\right|, \\ &\leq \max_i (\max_{x\in X_0} \left|\frac{\partial f}{\partial x_i}(x_k)\right|).\end{aligned}$$

et comme $\dfrac{\partial f}{\partial x_i}$ continue sur X_0 compact, alors le deuxième terme de l'inégalité est fini. D'où

$$\exists\ A > 0 \text{ tel que } \|g_k\| < A \ \forall\ k = 0, 1, 2, ...$$

ce qui montre que $\{g_k\}$ est bornée.

(iii) la définie positivité de B_k^{-1} découle de la définie positivité de B_k et la preuve de cette dernière est similaire à celle du théorème 4.2 (1) où la condition $p_k^t q_k > 0$ est vérifiée par $p_k^t q_k \geq \alpha_k \|p_k\| > 0$.

(iv) D'aprés (5.3), on a

$$\exists \lambda_k \quad \text{telque} \quad f(x_k - \lambda_k d_k) \leq f(x_k) - c_1 \lambda_k d_k^t g_k,$$

ou bien

$$f(x_{k+1}) \leq f(x_k) - c_1 \lambda_k d_k^t g_k,$$

et comme $d_k^t g_k = g_k B_k^{-1} g_k > 0$ (puisque B_k^{-1} est définie positive et $\|g_k\| \geq 0.001\alpha_k$). d'où

$$f(x_{k+1}) \leq f(x_k) \qquad \forall\ k = 0, 1, 2, ...\square$$

Les propositions suivantes montrent que notre recherche linéaire est bien définie.

Proposition 5.2

A chaque itération, il existe un $\lambda_a \geq \alpha_k$ *vérifiant*

$$f(x_k - \lambda_a d_k) < f(x_k) - c_1 \lambda_a d_k^t g_k$$

Preuve

Supposons que pour un $k = s$, il n'existe aucun $\lambda_a \geq \alpha_k$, et qui vérifie (5.1). Alors d'aprés le lemme 5.1 les directions d_k sont bornées pour tout k car

$$\|d_k\| = \left\|B_k^{-1} g_k\right\| \leq \left\|B_k^{-1}\right\| \|g_k\| \tag{5.5}$$

et$\{g_k\}$ est borné et B_k^{-1}est définie positive.

Prenons $x_k \in X_0$, donc il existe un $\lambda > 0$ suffisament petit telque $x_k - \lambda d_k \in X$ et par suite $[x_k - \lambda d_k, x_k] \subset X$.

En utilisant la formule de taylor sur $[x_k - \lambda d_k, x_k]$ on obtient

$$f(x + \lambda d_k) = f(x_k) + \lambda d_k^t \nabla f(x_k) + \tfrac{1}{2}\lambda^2 d_k^t H(\xi_k) d_k,$$

avec $\xi_k \in]x_k - \lambda d_k, x_k[$,

$$f(x_k) - f(x + \lambda d_k) = \lambda d_k^t \nabla f_k - \tfrac{1}{2}\lambda^2 d_k^t H(\xi_k) d_k + \lambda d_k^t g_k - \lambda d_k^t g_k.$$

Comme $\lambda d_k^t g_k = \lambda g_k^t B_k^{-1} g_k > 0$(B_k^{-1}est définie positive) on peut réecrire la formule comme suit

$$\frac{f(x_k) - f(x + \lambda d_k)}{\lambda d_k^t g_k} = 1 + \frac{d_k^t(\nabla f_k - g_k)}{d_k^t g_k} - \frac{1}{2}\frac{\lambda d_k^t H(\xi_k) d_k}{d_k^t g_k}.$$

En utilisant les propriétées suivantes

- $\forall\, a \subset \mathbb{R}$, $a \geq -|a|$ et $-a \geq -|a|$
- l'inégalité de schwartz,

on trouve

$$\frac{f(x_k) - f(x + \lambda d_k)}{\lambda d_k^t g_k} \geq 1 - \frac{\|d_k\| \, \|(\nabla f_k - g_k)\|}{d_k^t g_k} - \frac{\lambda}{2} \frac{\tau \|d_k\|^2}{d_k^t g_k}.$$

Si on note par m_p la plus petite valeur propre de B_k^{-1}, donc $m_p > 0$, et d'aprés la définition de la valeur propre on a

$$B_k^t g_k = \lambda_k g_k \Rightarrow g_k^t B_k^t g_k = m \|g_k\|^2 \geq m_p \|g_k\|^2,$$

d'où

$$d_k^t g_k \geq m_p \|g_k\|^2. \tag{5.6}$$

Par conséquent

$$\begin{aligned} \frac{\|d_k\| \, \|\nabla f_k - g_k\|}{d_k^t g_k} &\leq \frac{\left\|B_k^{-1}\right\| \|g_k\| \, \|\nabla f_k - g_k\|}{m_p \|g_k\|^2}, \\ &= \frac{\left\|B_k^{-1}\right\| \|\nabla f_k - g_k\|}{m_p \|g_k\|}, \\ &\leq \frac{\left\|B_k^{-1}\right\| \|\nabla f_k - g_k\|}{0.001 \alpha_k m_p}. \end{aligned}$$

D'aprés le théorème (5.1) le second membre de l'inégalité tend vers 0 quand $t \to +\infty$ d'où l'on déduit que:

$$\frac{\|d_k\| \, \|\nabla f_k - g_k\|}{d_k^t g_k} \to 0 \qquad \text{quand } t \to 0. \tag{5.7}$$

De plus

$$\frac{\tau \|d_k\|^2}{d_k^t g_k} \leq \frac{\tau \left\|B_k^{-1}\right\|^2 \|g_k\|^2}{m_p \|g_k\|^2} = \frac{\tau \left\|B_k^{-1}\right\|^2}{m_p}.$$

Comme $\dfrac{\tau \left\|B_k^{-1}\right\|^2}{m_p}$ est bornée, donc il existe un $c > 0$ telque

$$\frac{\tau \|d_k\|^2}{d_k^t g_k} \leq c. \tag{5.8}$$

Puisque on suppose que λ est petit, et d'aprés (5.6) et(5.7) on peut aussi déduire qu'il existe un $\varepsilon > 0$ tel que

$$\frac{f(x_k) - f(x + \lambda d_k)}{\lambda d_k^t g_k} \geq 1 - \varepsilon.$$

En prenant $c_1 = 1 - \varepsilon$ on trouve une contradiction à notre supposition. Donc aprés un nombre fini de pas on peut trouver un $\lambda_a \geq \alpha_k$ telque (5.1) soit vérifiée.□

Proposition 5.3

A chaque itération il existe un λ_m vérifiant

$$f(x_k - \lambda d_k) \leq f(x_k) - c_1 \lambda d_k^t g_k$$

et

$$d_k^t gr(f, x_k - \lambda d_k, \alpha_k) \leq c_2 d_k^t g_k$$

Preuve

Supposons que l'algorithme fait un cycle infini de l'étape 3d) à l'étape 3c), alors $\alpha_k \to 0$. D'aprés (5.1) et la supposition1 il existe un $\overline{\lambda}$ telque $x_k - \overline{\lambda} d_k \in X$ qui vérifie

$$f(x_k - \overline{\lambda} d_k) = f(x_k) - c_1 \overline{\lambda} d_k^t g_k$$

X étant convexe, choisissons $\overline{\lambda}$ comme étant la première intersection du graphe de la fonction $f(x_k - \lambda d_k)$ (de la variable λ), et la droite $f(x_k) - c_1 \lambda d_k^t g_k$. Du fait que f est convexe sur X, on déduit que $\lambda_a < \overline{\lambda}$, et que (5.3) reste vraie pour tout $0 < \lambda < \overline{\lambda}$.
D'autre part, $d_k^t \nabla f(x_k - \lambda d_k)$ est une fonction monotone décroissante de $\lambda > 0$ pour tout $x_k - \lambda d_k \in X$. En effet :
Soit $\lambda_1 > \lambda_2$ tels que $x_k - \lambda_1 d_k \in X$, et $x_k - \lambda_2 d_k \in X$, on a

$$\begin{aligned} d_k^t \nabla f(x_k - \lambda_1 d_k) - d_k^t \nabla f(x_k - \lambda_2 d_k) &= d_k^t \left[\nabla f(x_k - \lambda_1 d_k) - \nabla f(x_k - \lambda_2 d_k)\right], \\ &= (\lambda_2 - \lambda_1) d_k^t H(\xi_k) d_k \leq 0, \end{aligned}$$

puisque $\xi_k \in X$ et H est semi définie positive pour tout $x \in X$.

Si $x_k - \lambda_b d_k \in X$ et d'aprés une relation due à la convexité de f on trouve

$$f(x_k - \lambda_b d_k) \leq f(x_k) - \lambda_b d_k^t \nabla f(x_k - \lambda_b d_k).$$

Aprés l'étape 3b) on peut écrire

$$d_k^t \nabla f(x_k - \lambda_b d_k) < c_2 d_k^t g_k. \tag{5.9}$$

En utilisant la formule de taylor, on peut trouver $\eta_k \in]x_k, x_k - \overline{\lambda} d_k[$ tel que

$$f(x_k) - f(x_k - \overline{\lambda} d_k) = \overline{\lambda} d_k^t \nabla f(\eta_k),$$

$\eta_k \in]x_k, x_k - \overline{\lambda} d_k[\Rightarrow \exists \theta \in]0,1[$ tel que $\eta_k = x_k - (1-\theta)\overline{\lambda} d_k$.

Donc, il existe un $\hat{\lambda} = (1-\theta)\overline{\lambda} \in]0, \overline{\lambda}[$ avec

$$d_k^t \nabla f(x_k - \hat{\lambda} d_k) = \frac{1}{\hat{\lambda}} \left[f(x_k) - f(x_k - \overline{\lambda} d_k) \right] = c_1 d_k^t g_k < c_2 d_k^t g_k.$$

Par conséquent de la décroissance de $d_k^t \nabla f(x_k - \lambda d_k)$, $]\lambda_a, \overline{\lambda}[$ contient un intervalle de longueur positive dans lequel (5.3) et (5.9) restent vraies.

Aprés l'étape 3b) on distingue deux cas

$\underline{1^{er} cas : \lambda_a = \lambda_b}$

Dans ce cas, $x_k - \lambda_b d_k \in X$, et (5.3) et (5.9) sont vérifiées pour $\lambda = \lambda_a = \lambda_b$ et λ_a et λ_b ne changent pas dans l'étape 3d).

$\underline{2^{ème} cas : \lambda_a < \lambda_b}$

Dans ce cas, soit on a $\lambda_a < \overline{\lambda} \leqslant \lambda_b$, soit $\lambda_a < \lambda_b < \overline{\lambda}$

- Si $\lambda_a < \overline{\lambda} \leqslant \lambda_b$, cela signifie que λ_b où λ_a a été changé en étape 3d).
- Si $\lambda_a < \lambda_b < \overline{\lambda}$, cela implique que λ_b ne change pas à l'étape 3d).

d'où l'on conclut que $]\lambda_a, \lambda_b[$ contient toujours un intervalle de longueur positive dans lequel (5.3) et (5.4) sont vérifiées.

D'aprés le théorème 5.1,

$$d_k^t gr(f, x_k - \lambda d_k, \alpha_k) \rightarrow d_k^t \nabla f(x_k - \lambda d_k),$$

donc aprés un nombre fini de pas (5.3) et (5.4) sont vérifiées pour $\lambda = \lambda_m$, ce qui donne une contradiction à notre supposition.□

5.4 Convergence de l'algorithme

Dans cette partie on étudie la convergence de l'algorithme quand il génère une suite finie ou bien une suite infinie.

Pour le premier cas, il est possible que pour un k fixé on obtienne un cycle infinie qui ne permet pas de trouver un seccesseur pour x_k.

Le théorème suivant montre que si on tombe sur ce cas, alors le point x_k est un point stationnaire.

Théorème 5.2

Si l'Algorithme génère seulement une suite finie $\{x_0, x_1, ..., x_s\}$, *alors* $\nabla f(x_s) = 0$

Preuve

(i) Si pour $k = s$, l'algorithme saute indéfiniment de l'étape 1b) à l'étape 1a), alors

$$\begin{aligned} \|\nabla f(x_s)\| &\leq \|\nabla f_s - g_s + g_s\| \leq \|g_s\| + \|\nabla f_s - g_s\|, \\ &\leq 0.001\alpha_s + \|\nabla f_s - g_s\| \rightarrow 0, \end{aligned}$$

d'où

$$\|\nabla f(x_s)\| = 0 \Rightarrow \nabla f(x_s) = 0.$$

(ii) Si pour $k = s$ l'algorithme saute indéfiniment de l'étape 4 à l'étape 1a), alors $\alpha_k \rightarrow 0$. Puisque $d_k^t q_k = d_k^t g_{k+1} - d_k^t g_k \leq c_2 d_k^t g_k - d_k^t g_k \leq (c_2 - 1) d_k^t g_k < 0$, et comme $\{p_k\}$ est

bornée(X_0 est borné), on obtient:

$$(1-c_2)\lambda_k d_k^t g_k \leq -\lambda_k d_k^t q_k = p_k^t q_k < \alpha_k \|p_k\|,$$

et

$$(1-c_2)\lambda_k d_k^t q_k > (1-c_2)\lambda_k m_p \|q_k\|^2,$$

d'où

$$0 < (1-c_2)\lambda_k m_p \|q_k\|^2 < \alpha_k \|p_k\| \to 0,$$

puisque m_p reste fixé pour $k = s$ et d'aprés la proposition1, $\lambda_k \geq \lambda_a > cte$, alors, $q_k \to 0$. D'autre part

$$\|\nabla f(x_s)\| \leq \|g_s\| + \|\nabla f_s - g_s\| \to 0,$$

donc

$$\nabla f(x_s) = 0.\square$$

Prenant maintenant le deuxième cas, et suposons que la suite $\{x_k\}$ générée par l'algorithme, est infinie. Pour montrer la convergence de cette suite on donne tout d'abord quelques théorèmes et lemmes qui nous serons utiles.

Lemme 5.2

$\exists\, M > 0$ telle que

$$\frac{q_k^t q_k}{p_k^t q_k} \leq M,$$

pour tout k.

Preuve

Par définition on a

$$\begin{aligned} q_k &= g_{k+1} - g_k = \nabla f_{k+1} - \nabla f_{k+1} + \nabla f_k - \nabla f_k + g_{k+1} - g_k, \\ &= \nabla f_{k+1} - \nabla f_k + (g_{k+1} - \nabla f_{k+1}) - (g_k - \nabla f_k). \end{aligned}$$

En posant

$$\overline{H}_k = \int_0^1 H(x_k + \theta p_k) d\theta,$$

et

$$y_k = \frac{1}{\alpha_k} \left[(g_{k+1} - \nabla f_{k+1}) - (g_k - \nabla f_k) \right],$$

on peut réecrire q_k comme suit

$$q_k = \overline{H}_k p_k + \alpha_k y_k, \tag{5.10}$$

avec $y_k \to 0$ quand $\alpha_k \to 0$, et $\left\| \overline{H}_k \right\| \leq \tau$.

D'aprés l'étape 4, on a $\alpha_k \|p_k\| \leq p_k^t q_k \leq \|p_k^t\| \|q_k\|$, et en utilisant (5.10), on obtient

$$\begin{aligned} q_k^t q_k &= (\overline{H}_k p_k + \alpha_k y_k)^t (\overline{H}_k p_k + \alpha_k y_k), \\ &= p_k^t \overline{H}_k \overline{H}_k p_k + 2\alpha_k p_k \overline{H}_k y_k + \alpha_k^2 \|y_k\|^2, \\ &\leq p_k^t \overline{H}_k \overline{H}_k p_k + 2\alpha_k p_k \tau \|y_k\| + \|q_k\|^2 \|y_k\|^2. \end{aligned}$$

D'autre part,

$$\begin{aligned} p_k^t \overline{H}_k p_k &= p_k^t (q_k - \alpha_k y_k) = p_k^t q_k - \alpha_k p_k^t y_k, \\ &\leq p_k^t q_k + p_k^t q_k \|y_k\| = p_k^t q_k (1 + \|y_k\|). \end{aligned}$$

Comme $\|y_k\| \to 0$ quand $\alpha_k \to 0$, alors $\exists c > 0$ telque

$$p_k^t \overline{H}_k p_k \leq c p_k^t q_k, \tag{5.11}$$

H_k étant semi définie positive pour tout $x_k \in X$, alors $\overline{H}_k$ est semi définie positive $\forall x_k \in X$, ceci impliquequ'il existe une matrice semi définie positive $\overline{H}_k^{\frac{1}{2}}$ telque $\overline{H}_k = \overline{H}_k^{\frac{1}{2}} \overline{H}_k^{\frac{1}{2}}$.

Soit en remplaçant on obtient

$$\begin{aligned} \frac{q_k^t q_k}{p_k^t q_k} &\leq \frac{p_k^t \overline{H}_k \overline{H}_k p_k}{p_k^t q_k} + 2\tau \|y_k\| + \frac{\|q_k\|^2 \|y_k\|^2}{p_k^t q_k}, \\ \frac{q_k^t q_k}{p_k^t q_k}(1 - \|y_k\|^2) &\leq \frac{p_k^t \overline{H}_k \overline{H}_k p_k}{p_k^t q_k} + 2\tau \|y_k\| \end{aligned}$$

D'aprés (4.11) on a

$$\frac{1}{p_k^t q_k} \leq \frac{c}{p_k^t \overline{H}_k p_k} = \frac{c}{p_k^t \overline{H}_k^{\frac{1}{2}} \overline{H}_k^{\frac{1}{2}} p_k}.$$

D'autre part on a

$$\frac{p_k^t \overline{H}_k \overline{H}_k p_k}{p_k^t \overline{H}_k \overline{H}_k p_k} = \frac{p_k^t \overline{H}_k^{\frac{1}{2}} \overline{H}_k \overline{H}_k^{\frac{1}{2}} p_k}{\left\| p_k \overline{H}_k^{\frac{1}{2}} \right\|^2} \leq \frac{\left\| \overline{H}_k \right\| \left\| p_k \overline{H}_k^{\frac{1}{2}} \right\|}{\left\| p_k \overline{H}_k^{\frac{1}{2}} \right\|^2} \leq \tau,$$

d'où

$$\begin{aligned} (1 - \|y_k\|^2) \frac{q_k^t q_k}{p_k^t q_k} &\leq \frac{c p_k^t \overline{H}_k \overline{H}_k p_k}{p_k^t \overline{H}_k^{\frac{1}{2}} \overline{H}_k^{\frac{1}{2}} p_k} + 2\tau \|y_k\|, \\ (1 - \|y_k\|^2) \frac{q_k^t q_k}{p_k^t q_k} &\leq c\tau + 2\tau \|y_k\|. \end{aligned}$$

Comme $\|p_k\| \to 0$, pour k suffisament grand, alors il existe un $M = c\tau$ telque

$$\frac{q_k^t q_k}{p_k^t q_k} \leq M. \square$$

Lemme 5.3

(i) $\lim_{k\to+\infty} \inf \|g_k\| = 0$

(ii) $\lim_{k\to+\infty} \inf \|\nabla f_k\| = 0$

Preuve

(i) On va tout d'abord prouver que

$$\frac{p_j^t q_j}{-p_j^t g_j} \geq 1 - c_2. \tag{5.12}$$

En effet:

$-p_j^t g_j = \lambda_j g_j^t B_j^{-1} g_j > 0$ $(B_j^{-1}$est définie positive).

et on a

$$p_j^t q_j = p_j^t g_{j+1} - p_j^t g_j \geq c_2(-\lambda_j) d_j^t g_j - p_j^t g_j = c_2 p_j^t g_j - p_j^t g_j = (c_2 - 1) p_j^t g_j.$$

Dans le théorème 4.4 , on a prouvé que pour un $c_3 > 0$, on avait

$$\prod_{j=0}^{k} \frac{\|g_j\|}{-p_j^t g_j} < c_3^k, \qquad \forall k \tag{5.13}$$

et comme (5.3) est vraie pour $\lambda = \lambda_k$ et f est bornée sur X_0, alors $\sum_{j=1}^{\infty} -p_j^t q_j$ est convergante. En effet

$$f(x_{j+1}) \leq f(x_j) - c_1 \lambda_j d_j^t g_j \Rightarrow \frac{f(x_j) - f(x_{j+1})}{c_1} \geq \lambda_j d_j^t g_j = -p_j^t g_j > 0,$$

alors,

$$-p_j^t g_j \leq \frac{f(x_j) - f(x_{j+1})}{c_1},$$

donc il suffit de montrer que la suite des sommes partielles $\sum_{i=1}^{n} f(x_j) - f(x_{j+1})$ est convergente. ceci est immédiat d'aprés la décroissance de f et le fait q'elle est minorée.

Une conséquence de cette convergence est que $-p_j^t g_j \to 0$ quand $j \to \infty$, et d'aprés l'inégalité (5.13) on conclut que

$$\lim_{k\to+\infty} \inf \|g_k\| = 0.$$

(ii) résulte immédiatement de (i). En effet:

$$\begin{aligned} \|\nabla f_k\| &\leq \|\nabla f_k - g_k\| + \|g_k\| \Rightarrow \lim_{k\to+\infty} \inf \|\nabla f_k\| \leq \lim_{k\to+\infty} \inf \|\nabla f_k - g_k\| + \lim_{k\to+\infty} \inf \|g_k\|, \\ &\Rightarrow \lim_{k\to+\infty} \inf \|\nabla f_k\| = 0. \end{aligned}$$

Théorème 5.5

tout point d'adhérence $\overline{x}$ de la suite $\{x_k\}$ est une solution du problème (P').

Preuve

D'aprés le lemme précédent on a

$$\lim_{k\to+\infty} \inf \|\nabla f_k\| = 0 \Rightarrow \exists \{x_k\}, k \in J \text{ telque } \nabla f_k \to 0 \text{ quand } k \to \infty, k \in J.$$

Notons $\hat{x}$ une solution du problème (P') (l'existence de $\hat{x}$ est evidente d'après la supposition (1). Donc $f(x_k) \geq f(\hat{x})\ \forall k \geq 0$, et d'aprés la convexité de f on a

$$f(\hat{x}) \geq f(x_k) + \nabla f_k^t(\hat{x} - x_k),$$

comme, $\hat{x}, x_k \in X_0$, donc $\{\hat{x} - x_k\}$ est bornée. D'où

$$f(x_k) \to f(\hat{x}) \text{ pour } k \to \infty, k \in J.$$

et comme$\{f(x_k)\}$ est convergente, alors

$f(x_k) \to f(\hat{x})$ pour $k \to \infty$.□

5.5 Convergence superlinéaire

Pour prouver la convergence superlinéaire, nous avons besoin d'autres suppositions sur la fonction f.

Supposition 5.2

Il existe un $\mu > 0$ telque

$$\mu \|z\|^2 \leq z^t H(x) z \leq \tau \|z\|^2 \text{ pour } z \in \mathbb{R}^n \text{ et } x \in X_0.$$

Supposition 5.3

la matrice hessienne $H(x)$ vérifie la condition de lipshitz suivante au voisinage de $\overline{x}$

$$\|H(x) - H(\overline{x})\| \leq L \|x - \overline{x}\| \qquad \forall x \in U(\overline{x}).$$

Avant de donner le théorème 4 qui montre que l'ordre de convergence est superlinéaire, nous montrons tout d'abord deux lemmes importants.

Lemme 5.4

Pour k suffisament grand, on a

$$c_4(-p_k^t g_k) \leq \|p_k\|^2 \leq c_5(-p_k^t g_k),$$

avec

$$c_4 = \frac{1 - c_2}{2\tau} \quad , \quad c_5 = \frac{4(1 - c_1)}{\mu}.$$

Preuve

D'aprés (5.10), on a

$$p_k^t q_k = p_k^t \overline{H} p_k + \alpha_k p_k^t y_k \leq \tau \|p_k\|^2 + p_k^t q_k \|y_k\|,$$

ou encore

$$(1 - \|y_k\|) p_k^t q_k \leq \tau \|p_k\|^2,$$

puisque $\|p_k\| \to 0$, quand $k \to +\infty$, et d'aprés (5.12)

$$(1-c_2)(-p_k^t g_k) \leq p_k^t q_k \leq 2\tau \|p_k\|^2 . \tag{5.14}$$

Pour prouver l'autre borne, on observe en appliquant la formule de taylor avec $\xi_k \in [x_k, x_{k+1}]$ que

$$\begin{aligned} f(x_{k+1}) &= f(x_k) + p_k^t \nabla f_k + \tfrac{1}{2} p_k^t H(\xi_k) p_k + p_k^t g_k - p_k^t g_k, \\ &= f(x_k) + p_k^t g_k + + p_k^t (\nabla f_k - g_k) + \tfrac{1}{2} p_k^t H(\xi_k) p_k. \end{aligned}$$

Comme (5.3) est vérifiée pour $\lambda = \lambda_k$, et en utilisant la supposition 2 on trouve

$$c_1 p_k^t g_k \geq f(x_{k+1}) - f(x_k) \geq p_k^t g_k + p_k^t (\nabla f_k - g_k) + \frac{\mu}{2} \|p_k\|^2 ,$$

et d'aprés le théorème 1 et (5.14) pour k suffisament grand, on obtient

$$\begin{aligned} \left| p_k^t (\nabla f_k - g_k) \right| &\leq \left\| p_k^t \right\| \left\| (\nabla f_k - g_k) \right\| , \\ &\leq \frac{\mu}{8\tau} \alpha_k \|p_k\| , \\ &\leq \frac{\mu}{8\tau} p_k^t q_k , \\ &\leq \frac{\mu}{4} \|p_k\|^2 , \end{aligned}$$

d'où

$$\begin{aligned} (c_1 - 1) p_k^t g_k &\geq \frac{\mu}{2} \|p_k\|^2 + p_k^t (\nabla f_k - g_k), \\ &\geq \frac{\mu}{2} \|p_k\|^2 - \left| p_k^t (\nabla f_k - g_k) \right| , \\ &\geq \frac{\mu}{4} \|p_k\|^2 , \end{aligned}$$

donc

$$\|p_k\|^2 \leq \frac{4(c_1 - 1)}{\mu} p_k^t g_k \Rightarrow \|p_k\|^2 \leq -c_5 p_k^t g_k,$$

avec $c_5 = \dfrac{4(c_1-1)}{\mu}$.□

Lemme 5.5

Pour tout k, le nombre d'entiers j qui satisfait l'inégalité

$$\frac{-p_j^t q_j}{\|p_j\|\,\|q_j\|} \geq \frac{1}{c_3 c_5}, \qquad 0 \leq j \leq k$$

est plus grand que $\dfrac{k}{2}$.

Preuve

Soit $i(k)$ le nombre des termes $\dfrac{-p_j^t q_j}{\|p_j\|\,\|q_j\|}$, $j = 0, ..., k$ qui sont $\leq \dfrac{1}{c_3 c_5}$

D'aprés (5.13) on a

$$\forall k \prod_{j=0}^{k} \frac{\|g_j\|}{-p_j^t g_j} < c_3^k \quad \Rightarrow \prod_{j=0}^{k} \frac{-p_j^t g_j}{\|g_j\|} > \frac{1}{c_3^k},$$
$$\Rightarrow \left(\prod_{j=0}^{k} \frac{-p_j^t g_j}{\|g_j\|}\right)^2 > \prod_{j=0}^{k} \frac{-p_j^t g_j}{\|g_j\|} \frac{1}{c_3^k}.$$

En prenant en considération le lemme 5.4, on a

$$\|p_j\|^2 \leq c_5(-p_j^t g_j) \Rightarrow \frac{1}{\|p_j\|^2} \geq \frac{1}{c_5(-p_j^t g_j)},$$

d'où

$$\prod_{j=0}^{k} \frac{(-p_j^t g_j)^2}{\|p_j\|^2 \|g_j\|^2} > \prod_{j=0}^{k} \frac{-p_j^t g_j}{c_5(-p_j^t g_j)} \frac{1}{c_3^k} = \frac{1}{(c_3 c_5)^k}.$$

D'aprés la définition de $i(k)$ et le fait que $\dfrac{-p_j^t q_j}{\|p_j\|\,\|q_j\|} \leq 1$, on obtient

$$\prod_{j=0}^{k} \frac{(-p_j^t g_j)^2}{\|p_j\|^2 \|g_j\|^2} = \left(\prod_{j=0}^{k} \frac{(-p_j^t g_j)}{\|p_j\|\,\|g_j\|}\right)^2 \leq \left(\left(\frac{1}{(c_3 c_5)}\right)^{i(k)}\right)^2,$$

par conséquent

$$\left(\frac{1}{(c_3 c_5)}\right)^k < \prod_{j=0}^{k} \frac{(-p_j^t g_j)^2}{\|p_j\|^2 \|g_j\|^2} < \left(\frac{1}{(c_3 c_5)}\right)^{2i(k)},$$

comme $\frac{1}{(c_3c_5)} < 1$ on conclut que, $2i(k) < \frac{k}{2}$.□

Lemme 5.6

$$\sum_{k=0}^{\infty} \|x_k - \overline{x}\| < \infty.$$

Preuve

D'aprés (5.4) (avec $\lambda = \lambda_k$), ona

$$f(x_{k+1}) \leq f(x_k) - c_1\lambda_k(-p_k^t g_k),$$

et d'aprés le lemme 5.4 on a

$$c_4(\frac{-p_k^t g_k}{\|p_k\|^2}) \leq 1,$$

d'où

$$\begin{aligned} f(x_{k+1}) \leq f(x_k) - c_1\lambda_k(-p_k^t g_k) &\leq f(x_k) - c_1\lambda_k(-p_k^t g_k)c_4(\frac{-p_k^t g_k}{\|p_k\|^2}), \\ &= f(x_k) - c_1c_4(\frac{-p_k^t g_k}{\|p_k\|^2}), \end{aligned}$$

Maintenant on va montrer que

$$\|\nabla f_k\|^2 \geq \mu\,[f(x_k) - f(\overline{x})].$$

En utilisant la formule de taylor sur $[x, \overline{x}]$, on trouve

$$f(x_k) = f(\overline{x}) + (x_k - \overline{x})^t \nabla f(\overline{x}) + (x_k - \overline{x})^t H(\xi_k)(x_k - \overline{x}),$$

avec $\xi_k \in [x_k, \overline{x}] \subset X_0$.

D'aprés la supposition 2, et le fait que $\overline{x}$ est un minimum de f,on obtient

$$f(x_k) - f(\overline{x}) \geq \mu\,\|x_k - \overline{x}\|^2, \tag{5.15}$$

et d'aprés la convexité de f on a l'inégalité

$$f(\overline{x}) - f(x_k) \geq (\overline{x} - x_k)^t \nabla f(x_k) \Rightarrow f(x_k) - f(\overline{x}) \leq -(\overline{x} - x_k)^t \nabla f(x_k),$$

donc

$$f(x_k) - f(\overline{x}) \leq \|(\overline{x} - x_k)\| \, \|\nabla f(x_k)\| . \tag{5.16}$$

Par conséquent de (5.16) et (5.15),on a

$$\begin{aligned} \|(\overline{x} - x_k)\| \, \|\nabla f(x_k)\| \geq \mu \|x_k - \overline{x}\|^2 \quad &\Rightarrow \|\nabla f(x_k)\| \geq \mu \|x_k - \overline{x}\| , \\ &\Rightarrow \|x_k - \overline{x}\| \leq \frac{\|\nabla f(x_k)\|}{\mu}. \end{aligned}$$

En substituant dans (5.15), on trouve

$$f(x_k) - f(\overline{x}) \leq \frac{\|\nabla f(x_k)\|^2}{\mu},$$

d'où le résultat

$$4 \|g_k\|^2 \geq \mu [f(x_k) - f(\overline{x})] .$$

Puisque pour k suffisament grand on a,

$$\|\nabla f_k\| \leq \|\nabla f_k - g_k\| + \|g_k\| \leq 0.001\alpha_k + \|g_k\| \leq 2 \|g_k\| ,$$

ceci implique donc pour k suffisament grand que

$$4 \|g_k\|^2 \geq \mu [f(x_k) - f(\overline{x})] .$$

Maintenant d'aprés (15) on a

$$\begin{aligned} f(x_{k+1}) - f(\overline{x}) &\leq f(x_k) - f(\overline{x}) - c_1 c_4 \frac{(-p_k^t g_k)^2}{\|p_k\|^2} \frac{4 \|g_k\|^2}{4 \|g_k\|^2}, \\ &\leq f(x_k) - f(\overline{x}) - c_1 c_4 \frac{(-p_k^t g_k)^2}{4 \|g_k\|^2 \|p_k\|^2} \mu [f(x_k) - f(\overline{x})] , \end{aligned}$$

$$= \left(1 - \frac{c_1 c_4 (p_k^t g_k)^2}{4 \|g_k\|^2 \|p_k\|^2}\right) [f(x_k) - f(\overline{x})],$$

et d'aprés la relation $f(x_{k+1}) - f(\overline{x}) \leq f(x_k) - f(\overline{x})$, pour tout k, on peut écrire

$$f(x_{k+1}) - f(\overline{x}) \leq \left(1 - \frac{c_1 c_4 (p_k^t g_k)^2}{4 \|g_k\|^2 \|p_k\|^2}\right) ... \left(1 - \frac{c_1 c_4 (p_0^t g_0)^2}{4 \|g0\|^2 \|p_0\|^2}\right) |f(x_k) - f(\overline{x})|.$$

Finalement en considérant le lemme 5.5, on conclut

$$f(x_{k+1}) - f(\overline{x}) \leq \left(1 - \frac{c_1 c_{4\mu}}{4(c_3 c_5)^2}\right)^{\frac{k+1}{2}} [f(x_0) - f(\overline{x})]. \tag{5.17}$$

Posons $c_6 = \left(1 - \frac{c_1 c_{4\mu}}{4(c_3 c_5)^2}\right)^{\frac{1}{2}}$. Remarquons que $\frac{c_1 c_{4\mu}}{4(c_3 c_5)^2} < 1$.
D'aprés la formule de taylor sur $[x_{k+1}, x_k]$ on obtient

$$f(x_{k+1}) = f(\overline{x}) + (x_{k+1} - \overline{x})^t \nabla f_k + \tfrac{1}{2}(x_{k+1} - \overline{x})^t H(\xi_k)(x_{k+1} - \overline{x}),$$

Comme $\nabla f(\overline{x}) = 0$ ($\overline{x}$ est un point minimum de f), et d'aprés la supposition 5.2, on a

$$f(x_{k+1}) = f(\overline{x}) + \frac{\mu}{2} \|x_{k+1} - \overline{x}\|^2, \tag{5.18}$$

de (5.17) et (5.18) on obtient

$$\frac{\mu}{2} \|x_{k+1} - \overline{x}\|^2 \leq c_6^{k+1} [f(x_0) - f(\overline{x})],$$

et comme $c_6 < 1$, alors la série $\sum_{k=0}^{\infty} c_6^{k+1}$ est convergente, ce qui assure la convergence de la série $\sum_{k=0}^{\infty} \|x_{k+1} - \overline{x}\|$.□

Théorème 5.4

$x_k \longrightarrow \overline{x}$ et l'ordre de convergence est superlinéaire.

Avant de montrer ce théorème, rappelons quelques résultats obtenus par Dennis et Moré.[20] qui nous seront utiles pour la preuve.

On rappelle aussi que pour une matrice $A \in L(\mathbb{R}^n)$ (espace vectoriel des applications linéaires de $\mathbb{R}^n$ dans $\mathbb{R}^n$), la norme de Frobenius est définie par

$$\|A\|_F^2 = \sum_{i,j=1}^{n} |a_{ij}|^2, \text{ avec } A = (a_{ij})$$

Lemme 5.7 [20, *p.*554]

Soit $M \in L(\mathbb{R}^n)$ une matrice symétrique non singulière telle que

$$\left\|Mp - M^{-1}q\right\| \leq \beta \left\|M^{-1}q\right\|, \tag{5.19}$$

avec $\beta \in \left[0, \frac{1}{3}\right]$, $q, p \in \mathbb{R}^n$ et $p \neq 0$. Alors $p^t q > 0$ et $\overline{S} \in L(\mathbb{R}^n)$ peut être définie par

$$\overline{S} = S + \frac{pp^t}{p^t q} + \frac{q^t S q p p^t}{(pq^t)^2} - \frac{pq^t S + Sqp^t}{p^t q},$$

où $S \in L(\mathbb{R}^n)$ est symétrique. De plus, si $\|.\|_M$ est la norme matricielle définie par

$$\|Q\|_M = \|MQM\|_F.$$

Alors, il existent σ, σ_1 et σ_2 des constantes ne dépendent que de M et n vérifiant

$$\left\|\overline{S} - A\right\|_M \leq \left[(1 - \sigma\theta^2)^{\frac{1}{2}} + \sigma_1 \frac{\|Mp - M^{-1}q\|}{\|M^{-1}q\|}\right] \|S - A\|_M + \sigma_2 \frac{\|p - Aq\|}{\|M^{-1}q\|},$$

où

$$\theta = \begin{cases} \dfrac{\|M(S-A)q\|}{\|S - A\|_M \|M^{-1}q\|} & \text{si } S \neq A \\ 0 & \text{ailleurs} \end{cases}$$

le lemme suivant montre que si $H(\overline{x})$, ($\overline{x}$ est le minimum de f) est symétrique et définie positive, alors M peut prendre la valeur de $H^{\frac{1}{2}}(\overline{x})$

Lemme 5.8 [20, *p.*555]

soit $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$ *différentiable sur un voisinage ouvert* D *covexe de* x^* *pour lequel* $H(x^*)$ *est symétrique et définie positive, et supposons que* H *vérifie la condition de lipschitz dans un voisinage de* x^*. *Si pour* $\{x_k\} \subset D$ *qui converge vers* $\overline{x}$, *on définie* $\{p_k\}$ *et* $\{q_k\}$ *par* $p_k = x_{k+1} - x_k$, $q_k = \nabla f(x_{k+1}) - \nabla f(x_k)$, *alors* $p_k^t q_k > 0$ *pour* $k \geq k_0$. *De plus, pour tout* $S_{k_0} \in L(\mathbb{R}^n)$ *la suite* $\{S_k\}$ *est bien définie par (4.9) pour* $k \geq k_0$, *et ils existent* σ, σ_3 *et* σ_4 *des constantes vérifient:*

$$\left\|S_{k+1} - H^{-1}(\overline{x})\right\|_M \leq \left[\left(1 - \sigma\theta^2\right)^{\frac{1}{2}} + \sigma_3\delta_k\right] \left\|S_k - H^{-1}(\overline{x})\right\|_M + \sigma_4\delta_k,$$

où $\delta_k = \max\left\{\|x_{k+1} - \overline{x}\|, \|x_k - \overline{x}\|\right\}$, $\sigma \in]0, 1]$,

$$\theta = \begin{cases} \dfrac{\|M[S_k - H^{-1}(\overline{x})]q_k\|}{\|S_k - H^{-1}(\overline{x})\|_M \|M^{-1}q_k\|} & \text{si } S_k \neq H^{-1}(\overline{x}) \\ \\ 0 & \text{ailleurs} \end{cases}.$$

Théorème 5.5 [20, *p.*556]

Supposons que $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$ *vérifie les hypothèses du lemme (5.8), dans l'ensemble* D *et soit* $\{x_k\}$ *une suite dans* D *qui vérifie:* $\sum_{k=1}^{\infty} \|x_k - \overline{x}\| < +\infty$. *Alors il existe un entier positif* k_0 *telque pour toute matrice symétrique* $S_{k_0} \in L(\mathbb{R}^n)$ *les suites*$\{p_k\}$,$\{q_k\}$ *et* $\{S_k\}$ *sont bien définie pour* $k \geq k_0$. *De plus,*

(i) $\lim_{k \to +\infty} \left\|H^{-\frac{1}{2}}(\overline{x})S_kH^{-\frac{1}{2}}(\overline{x}) - I\right\|_F$ *existe, et*

(ii) $\lim_{k \to +\infty} \dfrac{\|[S_k - H^{-1}(\overline{x})]q_k\|}{\|q_k\|} = 0.$

Preuve du théorème 5.4

pour prouver ce théorème on a besoin du (ii) du théorème 5.6, vérifions donc les hypothèses du lemme 5.8, et comme le lemme 5.7 implique le lemme 5.8 ,alors il suffit de vérifier que (5.19) est vraie. En effet :

$$\begin{aligned}
\|q_k - Hp_k\| &= \left\|\overline{H}p_k + \alpha_k y_k - Hp_k\right\|, \\
&\leq \left\|(\overline{H} - H)p_k\right\| + \|\alpha_k y_k\|, \\
&\leq \left\|(\overline{H} - H)\right\| \|p_k\| + \frac{\alpha_k \|y_k\| \|p_k\|}{\|p_k\|}, \\
&\leq 2\tau \|p_k\| + p_k^t q_k, \\
&\leq 2\tau \|p_k\| + 2\tau \|p_k\| \|y_k\|, \\
&\leq 2\tau c \|p_k\|, \\
&\leq 3\tau c \|p_k\|, \\
&\leq \frac{\tau\mu}{3\tau c} \|p_k\|, \qquad \text{puisque } \frac{\mu}{\tau} \leq 1
\end{aligned}$$

alors

$$\|q_k - Hp_k\| \leq \frac{\mu}{3c} \|p_k\|, \tag{5.20}$$

d'autre part, d'aprés(5.11) on a

$$\mu \|p_k\|^2 < p_k^t \overline{H} p_k \leq c p_k^t q_k \leq c \|p_k\| \|q_k\|,$$

donc

$$\|p_k\| \leq \frac{c}{\mu} \|q_k\|,$$

soit en substituant dans (5.19), on trouve

$$\|q_k - Hp_k\| \leq \frac{1}{3} \|p_k\|,$$

d'où l'hypothèse du théorème est satisfaite, et par conséquent

$$\lim_{k\to+\infty} \frac{\|[S_k - H^{-1}(\overline{x})]q_k\|}{\|q_k\|} = 0.$$

Cette limite et l'égalité suivante

$$[B_k - H(\overline{x})]p_k = \left[I - B_k H^{-1}(\bar{x})\right](q_k - H(\bar{x})p_k) - B_k[S_k - H^{-1}(\overline{x})]q_k,$$

nous affirme que

$$\lim_{k\to+\infty} \frac{\|(B_k - H)p_k\|}{\|p_k\|} = 0,$$

d'autre part on a

$$\begin{aligned}\lim_{k\to\infty} \frac{\|g_k - Hd_k\|}{\|d_k\|} &= \lim_{k\to\infty} \frac{\left\|g_k - HB_k^{-1}g_k\right\|}{\left\|B_k^{-1}g_k\right\|} = \lim_{k\to\infty} \frac{\left\|(B_k - H)\lambda_k B_k^{-1}g_k\right\|}{\left\|\lambda_k B_k^{-1}g_k\right\|}, \\ &= \lim_{k\to\infty} \frac{\|(B_k - H)p_k\|}{\|p_k\|} = 0,\end{aligned}$$

d'où

$$\lim_{k\to\infty} \frac{\|g_k - Hd_k\|}{\|d_k\|} = 0. \tag{5.21}$$

On a aussi

$$\frac{d_k^t g_k}{\|d_k\|^2} = \frac{d_k^t H d_k}{\|d_k\|^2} - \frac{d_k^t(Hd_k - g_k)}{\|d_k\|^2} \geq \frac{d_k^t H d_k}{\|d_k\|^2} - \frac{\|Hd_k - g_k\|}{\|d_k\|},$$

on obtient (en faisant tendre k vers l'infini) et d'après (5.21) et la supposition 2

$$\frac{d_k^t g_k}{\|d_k\|^2} \geq \frac{\mu}{2} \Rightarrow \|d_k\| \leq \frac{2}{\mu}\|g_k\|. \tag{5.22}$$

Si on utilisant la formule de taylor sur $[x_k, x_k - d_k]$, on obtient

$$f(x_k - d_k) = f(x_k) - d_k^t \nabla f_k + \tfrac{1}{2} d_k^t H(\xi_k) d_k, \xi_k \in [x_k, x_k - d_k],$$

où encore,

$$f(x_k) - f(x_k - d_k) - \tfrac{1}{2} d_k^t g_k = d_k^t(\nabla f_k - g_k) + \tfrac{1}{2} d_k^t (H - H(\xi_k)) + \tfrac{1}{2} d_k^t (g_k - H d_k),$$

soit en divisant par $d_k^t g_k$, on trouve

$$\left| \frac{f(x_k) - f(x_k - d_k)}{d_k^t g_k} - \frac{1}{2} \right| \leq \left| \frac{d_k^t(\nabla f_k - g_k)}{d_k^t g_k} + \frac{1}{2} \frac{d_k^t (H - H(\xi_k)) d_k}{d_k^t g_k} + \frac{1}{2} \frac{d_k^t (g_k - H d_k)}{d_k^t g_k} \right|,$$

d'après (5.22), on obtient finalement

$$\left| \frac{f(x_k) - f(x_k - d_k)}{d_k^t g_k} - \frac{1}{2} \right| \leq \frac{2 \|g_k\| \, \|\nabla f_k - g_k\|}{\mu m_k \|g_k\|^2} + \frac{\|H - H(\xi_k)\|}{\mu} + \frac{\|g_k - H d_k\|}{\mu d_k}.$$

Quand $k \to \infty$ le deuxième terme de l'inégalité tend vers 0. Celà implique que le premier terme nous donne les deux inégalités suivantes

$$f(x_k - d_k) \geq f(x_k) - \tfrac{1}{2} d_k^t g_k > f(x_k) - c_2 d_k^t g_k,$$

et

$$f(x_k - d_k) \leq f(x_k) - \tfrac{1}{2} d_k^t g_k < f(x_k) - c_1 d_k^t g_k,$$

ce qui montre que (5.1) et (5.2) sont vérifiées pour $\lambda_a = \lambda_b = 1$

par conséquent

$\exists k_0 \in \mathbb{N}$ telque $\lambda_k = 1 \; \forall k \geq k_0$.

Dans ce qui suit on va garder $k \geq k_0$, on obtient alors

$$(B_k - H)p_k = -g_k - H p_k = q_k - g_{k+1} - H p_k = \overline{H} p_k + \alpha_k y_k - H p_k - g_{k+1},$$

donc

$$\frac{\|g_{k+1}\|}{\|p_k\|} \leq \frac{\|(\overline{H}-H)p_k\|}{\|p_k\|} + \frac{\alpha_k \|y_k\|}{\|p_k\|} + \frac{\|(B_k-H)p_k\|}{\|p_k\|}.$$

D'aprés (5.14), on a

$$\frac{\|g_{k+1}\|}{\|p_k\|} \leq \frac{\|(\overline{H}-H)p_k\|}{\|p_k\|} + 2\tau \|y_k\| + \frac{\|(B_k-H)p_k\|}{\|p_k\|},$$

le deuxième terme de l'inégalité tend vers 0 quand $k \to +\infty$, d'où

$$\lim_{k\to+\infty} \frac{\|g_{k+1}\|}{\|p_k\|} = 0. \tag{5.23}$$

Une conséquence du théorème de Taylor et la supposition 5.2 pour k suffisament grand donnent

$$\|g_{k+1}\| \geq \mu \|x_{k+1} - x_k\|.$$

En effet :

$$\nabla f(x_{k+1}) = \nabla f(\overline{x}) + (x_{k+1} - \overline{x})H(\xi_k) = (x_{k+1} - \overline{x})H(\xi_k),$$

donc,

$$\|\nabla f(x_{k+1})\| \geq \mu \|x_{k+1} - \overline{x}\|.$$

Comme

$$\begin{aligned} \|g_{k+1}\| &\geq \|\nabla f_{k+1}\| - \|\nabla f_{k+1} - g_{k+1}\|, \\ &\geq \|\nabla f_{k+1}\| - 10^{-4}\alpha_{k+1}, \\ &\geq \|\nabla f_{k+1}\| - \|g_{k+1}\|, \end{aligned}$$

doù

$$2\|g_{k+1}\| \geq \|\nabla f_{k+1}\| \geq \mu \|x_{k+1} - \overline{x}\|,$$

soit en divisant par $\|p_k\|$, on obtient

$$\frac{\|g_{k+1}\|}{\|p_k\|} \geq \frac{\mu}{2} \frac{\|x_{k+1} - \overline{x}\|}{\|x_{k+1} - \overline{x} + \overline{x} - x_k\|} \geq \frac{\mu}{2} \frac{\|x_{k+1} - \overline{x}\|}{\|x_{k+1} - \overline{x}\| + \|\overline{x} - x_k\|} = \frac{\mu}{2} \frac{\rho_k}{1 + \rho_k},$$

telque $\rho_k = \dfrac{\|x_{k+1} - \overline{x}\|}{\|x_k - \overline{x}\|}$, en prenant en compte (5.23), on obtient

$$\lim_{k \to +\infty} \frac{\rho_k}{1 + \rho_k} = 0 \Rightarrow \lim_{k \to +\infty} \rho_k = 0$$

d'où finalement

$$\lim_{k \to +\infty} \frac{\|x_{k+1} - \overline{x}\|}{\|x_k - \overline{x}\|} = 0.\square$$

5.6 Résultats numériques

Pour l'application numérique de cette méthode, on a choisi les valeurs $\alpha = 0.1$, $c_1 = 0.1$, $c_2 = 0.7$ et comme matrice initiale, la matrice identité.

Nous tenons compte également de la valeur de la fonction objective obtenue en un nombre d'itération fixé.

Nous donnons, ci-dessous les fonctions tests qui ont été traitée, ensuite le tableau résumant les résultats numériques obtenues.

Problème 1 (Rosenbrock): $n = 2$

$f(x) = 100.(x_2 - x_1^2)^2 + (1. - x_1)^2$

$x_0 = (-1.2, 1.)$

Problème 2 (Beale): $n = 2$

$f(x) = [1.5 - x_1(1. - x_2)]^2 + [2.25 - x_1(1. - x_2^2)]^2 + [2.625 - x_1(1. - x_2^3)]^2$

$x_0 = (1., 1.)$

Problème 3 (Powell): $n = 3$

$$f(x) = 3 - \frac{1}{1+(x_1-x_2)^2} - \sin(\frac{\pi}{2}x_2x_3) - \exp\left[-(\frac{x_1+x_3}{x_2} - 2.)\right]$$

$x_0 = (0., 1., 2.)$

Problème 4 (Singular): $n = 4$

$$f(x) = (x_1+10.x_2)^2+(1.-x_1)^2+90.(x_4-x_3^2)^2+(1.-x_3)^2+10.1\left[(x_2-1.)^2+(x_4-1.)^2\right]$$

$x_0 = (3., -1., 0., 1.)$

Problème 5, 6 (Dixon): $n = 3, 10$

Voir problème 2 (Annexe A).

Problème 7, 8 (Oren): $n = 2, 6$

Voir problème 1 (Annexe A)

Problème 9, 10 (Powell généralisé): $n = 8, 16$

Voir problème 3 (Annexe A).

Tableau 5.1: resultats numériques

Problèm. N.	Nombre de variables	Nombre d'iteration	Evaluation fonctionn.	valeur de la fonction	norme de l'app. du gradient
1	2	26	204	1.65×10^{-3}	2.1×10^{-1}
		29	228	2.86×10^{-8}	6.27×10^{-2}
2	2	14	93	9.82×10^{-18}	1.04×10^{-8}
		16	132	3.75×10^{-27}	3.16×10^{-13}
3	3	10	73	2.46×10^{-11}	2.57×10^{-5}
		12	91	4.01×10^{-18}	9.82×10^{-9}
		13	100	0.00	1.04×10^{-10}
4	4	18	208	1.60×10^{-8}	1.69×10^{-4}
		21	244	5.21×10^{-10}	9.73×10^{-7}
5	3	13	112	1.81×10^{-17}	2.97×10^{-8}
		15	130	2.27×10^{-25}	2.72×10^{-12}
6	10	17	377	2.28×10^{-9}	5.35×10^{-4}
		22	492	6.39×10^{-15}	2.84×10^{-8}
7	2	40	300	3.46×10^{-17}	1.82×10^{-12}
8	6	43	687	7.74×10^{-11}	1.82×10^{-7}
9	8	30	591	3.57×10^{-9}	4.67×10^{-6}
10	16	35	4472	2.52×10^{-7}	8.89×10^{-5}

Chapitre 6

Accélération de la convergence de la méthode de la plus forte pente (steepest descent)

Introduction:

Ce chapitre contient la partie plus ou moins originale de ce mémoire. Après avoir effectué dans les chapitres précédents, un tour d'horizon théorique et numérique sur les méthodes quasi-Newton, qui rappelons le, sont des modifications de la méthode de Newton, on essaye dans ce chapitre d'améliorer la méthode de la plus grande pente (steepest descent).

Cette méthode qui fut découverte par Cauchy en 1847 ([17]), et malgré sa simplicité, présente l'nconvénient d'être trés lente lorsqu'on s'approche du point optimal. L'idée de ce travail est d'essayer d'accélérer la convergence de cette méthode pour y remédier à ce défaut. L'outil utilisé pour arriver à ce but est l'$\varepsilon-$Algorithme qui est un puissant outil d'accélération de la convergence, découvert par Peter Wynn ([43]). C'est pour cela qu'on a appelé notre algorithme: l'epsilon steepest descent.

Bien sur, on ne donne dans ce mémoire que les grands axes de ce travail. Les tests numériques, l'étude de la convergence feront l'objet d'un travail de plus grande envergure

(thèse de Doctorat).

6.1 Méthode de la plus forte pente (steepest descent)

Il est naturel de se demander l'origine ou la justification d'une telle appelation (méthode de la plus forte pente). Considérons un point $x_k \in \mathbb{R}^n$, si $\nabla f(x_k) \neq 0$, alors la direction $d_k = -\nabla f(x_k)$ est une direction de descente (voir Théorème 1.1, Chap.1). Le lemme qui suit va nous montrer c'est en fait la meilleure direction de descente. En d'autres termes la décroissance de la fonction sera la plus forte en suivant la direction: $-\nabla f(x_k)$.

Lemme 6.1

Supposons que $f : \mathbb{R}^n \to \mathbb{R}$, soit différentiable au point x, et supposons que $\nabla f(x) \neq 0$. Considérons le problème optimal

$$\underset{\|d\| \leq 1}{Minimiser} \quad f'(x,d)$$

où $f'(x,d)$ *est la dérivée directionnelle de f au point x et dans la direction d. Alors La solution optimale de ce problème est donnée par*

$$\bar{d} = -\frac{\nabla f(x)}{\|\nabla f(x)\|}$$

Preuve

f étant différentiable au point x. Donc

$$f'(x,d) = \lim_{\lambda \to 0_+} \frac{f(x+\lambda d) - f(x)}{\lambda} = \nabla f(x)^t d.$$

Notre problème revient donc à minimiser $\nabla f(x)^t d$ dans $\{d : \|d\| \leq 1\}$.

L'inégalité de shwartz donne

$$\left|\nabla f(x)^t d\right| \leq \|\nabla f(x)\| \, \|d\| \Rightarrow \nabla f(x)^t d \geq - \|\nabla f(x)\| \, \|d\| .$$

Pour $\|d\| \leq 1$, on a

$$\|\nabla f(x)\| \, \|d\| \leq \|\nabla f(x)\| \Rightarrow - \|\nabla f(x)\| \, \|d\| \geq - \|\nabla f(x)\| .$$

Donc: $\forall d : \|d\| \leq 1$ on a

$$\nabla f(x)^t d \geq - \|\nabla f(x)\| .$$

D'autre part on a: $\left\|\tilde{d}\right\| = 1$ et $\tilde{d}$ vérifie:

$$\nabla f(x)^t \tilde{d} = \nabla f(x)^t (-\frac{\nabla f(x)}{\|\nabla f(x)\|}) = - \|\nabla f(x)\| . \square$$

6.1.1 Algorithme de la méthode de la plus forte pente

Etape initiale: Choisir un $\varepsilon > 0$. Choisir un point initial x_1. Poser $k = 1$ et aller à l'étape principale.

Etape principale: Si $\|\nabla f(x)\| < \varepsilon$ stop. Sinon poser $d_k = -\nabla f(x_k)$ et soit λ_k la solution optimale de la recherche linéaire

$$Min \ \{f(x_k + \lambda d_k); \ \lambda \geq 0\} .$$

Poser $x_{k+1} = x_k + \lambda_k d_k$. Remplacer k par $k + 1$ et répéter l'étape principale.

6.1.2 Inconvénients de la méthode de la la plus forte pente

Cette méthode travaille de façon performante dans les premières étapes de l'algorithme. Malheuresement, dès qu'on s'approche du point stationnaire, la methode devient très lente.

On peut expliquer intuitivement ce phénomène par les considérations suivantes

$$f(x_k + \lambda d) = f(x_k) + \lambda \nabla f(x_k)^t d + \lambda \|d\| \alpha(x_k; \lambda d), \tag{6.1}$$

où $\alpha(x_k; \lambda d) \to 0$ quand $\lambda d \to 0$.
Si $d = -\nabla f(x_k)$, on obtient: $x_{k+1} = x_k - \lambda \nabla f(x_k)$ et (6.1) devient

$$f(x_{k+1}) - f(x_k) = \lambda \left[- \|\nabla f(x_k)\|^2 + \|\nabla f(x_k)\| \alpha(x_k; \lambda \nabla f(x_k)) \right]. \tag{6.2}$$

D'après l'expression (6.2), on voit que lorsque x_k s'approche d'un point stationnaire, et si f est continuement différentiable, alors $\|\nabla f(x_k)\|$ est proche de zéro. Donc le terme à droite s'approche de zéro, indépendemment de λ, et par conséquent $f(x_{k+1})$ ne s'éloigne pas beaucoup de $f(x_k)$ quand on passe du point x_k au point x_{k+1}.

6.1.3 Remèdes

Pour y remédier à ces inconvénients on essayera dans les paragraphes qui suivent, d'accélérer la convergence de la méthode de la plus forte pente. Pour cela on fera appel à un algorithme trés célèbre dans le domaine de l'accélération de la convergence qui est l'epsilon Algorithme.

Dans ce mémoire on ne donnera que les grands axes et idées de cette technique. Les développements se feront dans un autre cadre plus approprié (Thèse de doctorat).

6.2 Accélération de la Convergence en Analyse numérique

Quand on utilise une méthode itérative pour résoudre un problème, on génère une suite $\{s_n\}$ qui converge vers la solution s du problème lorsque n tend vers l'infini. En pratique, la suite $\{s_n\}$ peut être une suite de nombres réels ou complexes, une suite de vecteurs

ou une suite de matrices; la suite $\{s_n\}$ peut dépendre d'une suite auxiliaire $\{x_n\}$ de nombres réels ou complexes.

Toutes les méthodes d'accélération de la convergence consistent à transformer la suite $\{s_n\}$ en une suite $\{v_n\}$ de même nature. Il faudre évidemment que $\{v_n\}$ converge vers la solution s de notre problème. Dans le cas d'une suite de nombres réels, on dit que $\{v_n\}$ converge plus vite que $\{s_n\}$ si

$$\lim_{n\to+\infty} \frac{v_n - s}{s_n - s} = 0,$$

Il faut, en d'autre termes, que l'erreur absolue sur v_n soit un infiniment petit devant l'erreur absolue sur s_n, pour n suffisamment grand. On dira dans ce cas que l'on a accéléré la convergence de la suite $\{s_n\}$. La méthode qui transforme la suite $\{s_n\}$ en la suite $\{v_n\}$, sera une méthode d'accéléretion de la convergence.

Les possibilités d'accélération de la convergence d'une suite $\{s_n\}$ dépendront de la vitesse avec laquelle la suite $\{s_n\}$ converge. Si $\{s_n\}$ converge "vite", elle sera difficile à accélérer et cela ne présentera d'ailleurs pas beaucoup d'intérêt. La notion fondamentale qui permet de mesurer la vitesse de convergence d'une suite est la notion d'ordre.

Voyons maintenant un algorithme assez puissant d'accélération de la convergence. Il se présente sous plusieurs formes selon la nature de la suite à accélérer: suite scalaire, suite vectorielle, suite matricielle,...

6.2.1 L'ε−algorithme scalaire

L'ε-algorithme est dû à P. Wynn ([43]). C'est certainement l'une des méthodes d'accélération de la convergence les plus puissantes connues à l'heure actuelle. Cet algorithme repose sur de solides bases théoriques et ses applications sont extrêmement importantes. L'ε-algorithme est une généralisation d'une méthode célèbre d'accélération de la convergence: le procédé Δ^2 d'Aitken ([1]).

Dans cet algorithme, on calcule également des quantités avec deux indices $\varepsilon_k^{(n)}$. On

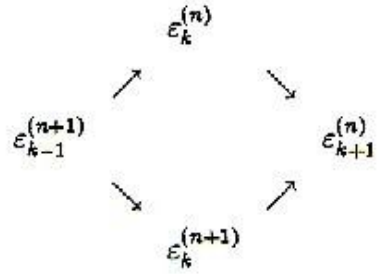
utilise pour cela les relations

$$\begin{aligned} \varepsilon_{-1}^{(n)} &= 0 \qquad \varepsilon_0^{(n)} = s_n \qquad\qquad n = 0, 1, \ldots \\ \varepsilon_{k+1}^{(n)} &= \varepsilon_{k-1}^{(n+1)} + \frac{1}{\varepsilon_k^{(n+1)} - \varepsilon_k^{(n)}} \qquad n, k = 0, 1, \ldots \end{aligned} \tag{6.3}$$

On place ces quantités dans un tableau à double entrée: le tableau ε. L'indice inférieur reste constant dans une colonne tandis que l'indice superieur reste constant dans une diagonale descente:

$$\begin{array}{lllllll} \varepsilon_{-1}^{(0)} = 0 & & & & & & \\ & \varepsilon_0^{(0)} = s_0 & & & & & \\ \varepsilon_{-1}^{(1)} = 0 & & \varepsilon_1^{(0)} & & & & \\ & \varepsilon_0^{(1)} = s_1 & & \varepsilon_2^{(0)} & & & \\ \varepsilon_{-1}^{(2)} = 0 & & \varepsilon_1^{(1)} & & \varepsilon_3^{(0)} & & \\ & \varepsilon_0^{(2)} = s_2 & & \varepsilon_2^{(1)} & & \varepsilon_4^{(0)} & \\ \varepsilon_{-1}^{(3)} = 0 & & \varepsilon_1^{(2)} & & \varepsilon_3^{(1)} & & . \\ & \varepsilon_0^{(3)} = s_3 & & \varepsilon_2^{(2)} & & \varepsilon_4^{(1)} & . \\ \varepsilon_{-1}^{(4)} = 0 & & \varepsilon_1^{(3)} & & \varepsilon_3^{(2)} & & . \\ . & \varepsilon_0^{(4)} = s_4 & . & \varepsilon_2^{(3)} & . & \varepsilon_4^{(2)} & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \quad . \end{array}$$

La relation (6.3) relie des quantités situées aux quatre sommets d'un losange:



(6.4)

Si l'on connaît $\varepsilon_{k-1}^{(n+1)}$, $\varepsilon_k^{(n)}$ et $\varepsilon_k^{(n+1)}$, la relation (6.3) permet de calculer $\varepsilon_{k+1}^{(n)}$; c'est ce

que signifient les flèches du tableau (6.4). On dit que l'ε-algorithme est un algorithme de losange.

La relation (6.3) permet donc de progresser de la gauche vers la droite et de haut en bas dans le tableau ε, à partir des conditions initiales $\varepsilon_{-1}^{(n)} = 0$ et $\varepsilon_0^{(n)} = s_n$ pour $n = 0, 1, ...$; si l'on connaît s_0 et s_1 on peut calculer $\varepsilon_1^{(0)}$, si l'on connaît s_1 et s_2 on peut calculer $\varepsilon_1^{(1)}$; à partir de $\varepsilon_1^{(0)}$ et de $\varepsilon_1^{(1)}$ on obtiendra ensuite $\varepsilon_2^{(0)}$ et ainsi de suite.

6.2.2 L'ε-algorithme vectoriel

La forme vectorielle de l'ε-algorithme a également été étudiée par Wynn. Soit $\{s_n\}$ une suite de vecteurs de $\mathbb{C}^p$. Les règles de l'ε-algorithme vectoriel sont les suivantes

$$\begin{aligned} \varepsilon_{-1}^{(n)} &= 0 \in \mathbb{C}^p \qquad \varepsilon_0^{(n)} = s_n \in \mathbb{C}^p \qquad && n = 0, 1, ... \\ \varepsilon_{k+1}^{(n)} &= \varepsilon_{k-1}^{(n+1)} + \left[\varepsilon_k^{(n+1)} - \varepsilon_k^{(n)}\right]^{-1} && n, k = 0, 1, ... \end{aligned} \tag{6.5}$$

On voit que, pour pouvoir appliquer cet algorithme, il faut définir ce qu'on appelle l'inverse d'un vecteur. Etand donné $y \in \mathbb{C}^p$, on définit son inverse $y^{-1} \in \mathbb{C}^p$ par la relation

$$y^{-1} = \frac{\bar{y}}{(y, y)}, \tag{6.6}$$

$\bar{y}$ est le vecteur dont les composantes sont les nombres complexes conjugués des composantes du vecteur y, (y, y) désigne le produit scalaire dans $\mathbb{C}^p$, du vecteur par lui même. Si l'on désigne par $y_i (i = 1, ..., p)$, les composantes du vecteur y, alors

$$(y, y) = \sum_{i=1}^{p} y_i \bar{y}_i = \sum_{i=1}^{p} |y_i|^2 .$$

Du point de vue géométrique, y^{-1} est l'inverse de y par rapport à la sphère de centre 0 et de rayon 1.

En partant de cette remarque, Cordellier a donné une interprétation géométrique d'une

étape de l'ε-algorithme vectoriel. Les propriétés remarquables de l'ε-algorithme vectoriel tiennent au fait que l'inverse d'un vecteur tel que nous venons de le définir est tel que

$$\begin{aligned}(y^{-1})^{-1} &= y,\\ (y, y^{-1}) &= 1,\\ (-y)^{-1} &= -(y^{-1}).\end{aligned}$$

Au lieu d'utiliser la définition (6.7) dans la relation (6.6), on peut utiliser comme définition de l'inverse d'un vecteur

$$y^{-1} = \frac{y}{(y, y)}. \tag{6.7}$$

Les vecteurs $\varepsilon_{2k+1}^{(n)}$ ainsi obtenus sont les complexes conjugués de ceux obtenus avec (6.7). Les vecteurs $\varepsilon_{2k}^{(n)}$ obtenus avec (6.8) sont identiques à ceux obtenus à l'aide de la définition (6.7) de l'inverse d'un vecteur. On utilisera donc indifféremment la définition (6.7) ou la définition (6.8) pour l'inverse d'un vecteur.

6.2.3 L'$\varepsilon-$algorithme vectoriel normé

Soit $\{s_n\}$ une suite de vecteurs de $\mathbb{C}^p$. Les règles de l'ε-algorithme vectoriel normé sont identiques à celles de l'ε-algorithme vectoriel du paragraphe (6.2.2); seule est changée la définition de l'inverse d'un vecteur. Nous définirons maintenant l'inverse y^{-1} de $y \in \mathbb{C}^p$ par

$$y^{-1} = \frac{\bar{y}}{\|y\|^2}, \tag{6.8}$$

où $\|y\|$ désigne n'importe quelle norme vectorielle de y. On peut, en particulier, prendre la plus simple d'entre elles à calculer

$$\|y\| = \max_{1 \le i \le p} |y|,$$

L'inverse défini par la relation (6.9) est tel que

$$(y^{-1})^{-1} = y,$$

Par contre le produit scalaire de y par y^{-1} n'est pas toujours égal à 1 et l'ε-algorithme vectoriel normé ne vérifiera pas toutes les propriétés de l'ε-algorithme vectoriel. Cependant, en ce qui concerne l'accélération de la convergence et certaines applications, on pourra utiliser indifféremmant l'ε-algorithme vectoriel du paragraphe (6.2.2) ou l'ε-algorithme vectoriel normé de ce paragraphe.

6.3 l'epsilon steepest descent algorithme

Pour construire notre Algorithme, on utilise l'$\varepsilon-$Algorithme scalaire d'ordre deux, c'est à dire $\varepsilon_2^{(n)}$.

De façon plus précise, étant donnée une suite $\{s_n\}$, on montre dans ([14], p.13), que les quantités $\varepsilon_2^{(n)}$ peuvent être calculées de la façon suivante

$$\varepsilon_2^{(n)} = \frac{s_n s_{n+2} - (s_{n+1})^2}{s_{n+2} - 2s_{n+1} + s_n}, \quad n = 0, 1, 2, ...$$

On considère donc ici la deuxième colonne paire du tableau du tableau de l'$\varepsilon-$Algorithme scalaire, associé à la suite $\{s_n\}$.

Remarque importante

Pour calculer la quantité $\varepsilon_2^{(n)}$, il suffit d'avoir les élements: s_n, s_{n+1} et s_{n+2} de la suite $\{s_n\}$.

L' ALGORITHME

Initialisation: L'Algorithme commence par un point initial $x_0 \in \mathbb{R}^n$, et $\varepsilon > 0$, qui définit le critère d'arrêt, les composantes de x_0 seront notées comme suit

$$x_0 = (x_0^1, x_0^2, ..., x_0^i, ..., x_0^n),$$

poser $k = 0$ et aller à étape principale.

Etape principale: Supposons qu'à l'étape k on ait le point x_k,

$$x_k = (x_k^1, x_k^2, ..., x_k^i, ..., x_k^n),$$

si $\|\nabla f(x_k)\| < \varepsilon$, stop; sinon poser

$$\begin{aligned} y_k &= x_k, \\ y_k &= (y_k^1, y_k^2, ..., y_k^i, ..., y_k^n), \end{aligned}$$

Calculer par la méthode steepest descent les successeurs y_{k+1} et y_{k+2} de y_k, c'est à dire

$$y_{k+1} = (y_{k+1}^1, y_{k+1}^2, ..., y_{k+1}^i, ..., y_{k+1}^n),$$

$$y_{k+2} = (y_{k+2}^1, y_{k+2}^2, ..., y_{k+2}^i, ..., y_{k+2}^n),$$

$$y_{k+1} = y_k - \lambda_k \nabla f(y_k),$$

λ_k solution optimale de la recherche linéaire exacte

$$\underset{\lambda \geq 0}{Minimiser} \quad f(y_k - \lambda \nabla f(y_k)$$

$$y_{k+2} = y_{k+1} - \lambda_{k+1} \nabla f(y_{k+1}),$$

λ_{k+1} solution optimale de la recherche linéaire exacte

$$\underset{\lambda \geq 0}{Minimiser} \quad f(y_{k+1} - \lambda \nabla f(y_{k+1}).$$

Calculer

$$\varepsilon_2^k = \left(\varepsilon_2^{k,1}, \varepsilon_2^{k,2}, ...\varepsilon_2^{k,i}, ...\varepsilon_2^{k,n}\right),$$

avec

$$\varepsilon_2^{k,i} = \frac{y_k^i y_{k+2}^i - \left(y_{k+1}^i\right)^2}{y_{k+2}^i - 2y_{k+1}^i + y_k^i}, \quad i = 1, 2, ...n.$$

Poser

$$x_k = \varepsilon_2^k.$$

Remplacer k par $k+1$ et aller à l'étape principale.

Annexe A

Tableaux des résultats numérique

A.1 Fonctions tests

Avant de donner les tableaux des résultats numérique obtenue en implémentant l'algorithme de (DFP et BFGS) avec d'autres fonctions tests, on donne ces fonctions.

Problème 1: (la fonction de Oren)

$f(x) = \left[\sum_{i=1}^{n} i x_i^2\right]^2$

$x_0 = (1, 1, ..., 1)^t$

$x^* = (0, 0, ..., 0)^t$

$f(x^*) = 0$

Problème 2: (la fonction de Dixon)

$f(x) = (x_1 - 1)^2 + \sum_{i=2}^{n} i(2x_i^2 - x_{i-1})^2$

$x_0 = (1, 1, ..., 1)^t$

$f(x^*) = 0$

Problème 3: (la fonction de Powell généralisée)

$f(x) = \sum_{i=1}^{n/4} [(x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} + x_{4i})^2 + (x_{4i-2} + 2x_{4i-1})^4 + 10(x_{4i-3} + x_{4i})^4]$

$x_0 = (3, -1, 0, 1, ..., 3, -1, 0, 1)^t$

$x^* = (0, 0, ..., 0)^t$

$f(x^*) = 0$

Problème 4: (la fonction de Box)

$f(x) = \sum_{i=1}^{10} [e^{-t_i x_1} - x_3(e^{-t_i} - e^{-10t_i})]$

$t_i = 0.1i$

$x_0 = (0, 10, 20)^t$

$f(x^*) = 0$ pour $(1, 10, 1)^t$, $(10, 1, -1)^t$ et pour $x_1 = x_2, x_3 = 0$

Problème 5: (la fonction de wood)

voir (exemple 3.2, chapitre 3)

A.2 Résultats numérique

DFP et BFGS avec la recherche linéaire de Wolfe sans reinitialisation

Tableau A.1: Problème 1

n	DFP			$BFGS$		
	k	Ef	$f(x)$	k	Ef	$f(x)$
2	30	35	2.27×10^{-8}	26	31	1.59×10^{-8}
10	327	388	8.52×10^{-8}	53	136	2.69×10^{-8}
30	758	954	1.10×10^{-7}	104	393	4.06×10^{-8}
80	2913	3633	3.13×10^{-5}	194	1362	2.25×10^{-5}

Tabteau A.2: problème 2.

n	DFP			$BFGS$		
	k	Ef	$f(x)$	k	Ef	$f(x)$
2	9	21	1.05×10^{-14}	9	21	1.25×10^{-16}
4	85	125	5.57×10^{-13}	19	50	7.43×10^{-14}
30	65	154	0.67	63	111	0.66
50	pas de convergence			149	454	0.66

Tabteau A.3: problème 3.

n	DFP			$BFGS$		
	k	Ef	$f(x)$	k	Ef	$f(x)$
4	81	177	3.14×10^{-10}	25	47	2.48×10^{-9}
8	326	556	2.11×10^{-10}	27	58	6.15×10^{-9}
16	889	946	1.00×10^{-9}	33	85	4.82×10^{-9}
32	1572	1652	3.81×10^{-9}	42	130	3.70×10^{-9}

Tabteau A.4: problème 4.

n	DFP			$BFGS$		
	k	Ef	$f(x)$	k	Ef	$f(x)$
3	101	140	1.24×10^{-11}	30	41	5.21×10^{-13}

Tabteau A.5: problème 5.

n	DFP			$BFGS$		
	k	Ef	$f(x)$	k	Ef	$f(x)$
4	pas de convergence	–	–	55	243	6.88×10^{-13}

DFP et BFGS avec la recherche linéaire de Wolfe avec reinitialisation

Tabteau A.6: problème 1.

n	DFP			$BFGS$		
	k	Ef	$f(x)$	k	Ef	$f(x)$
2	13	23	1.29×10^{-8}	13	23	1.29×10^{-8}
10	43	183	1.24×10^{-8}	38	156	1.37×10^{-8}
30	92	484	1.81×10^{-8}	88	467	1.22×10^{-8}
80	244	1254	1.03×10^{-8}	191	1208	5.30×10^{-9}

Tabteau A.7: problème 2.

n	DFP			$BFGS$		
	k	Ef	$f(x)$	k	Ef	$f(x)$
2	24	78	4.64×10^{-12}	16	58	6.91×10^{-12}
4	53	298	4.29×10^{-12}	45	201	2.01×10^{-12}
30	87	300	0.66	90	343	0.67
50	245	1747	0.66	225	1249	0.66

Tabteau A.8: problème 3.

n	DFP			$BFGS$		
	k	Ef	$f(x)$	k	Ef	$f(x)$
4	260	1113	2.68×10^{-8}	60	252	8.71×10^{-9}
8	212	689	3.99×10^{-8}	51	211	9.21×10^{-9}
16	161	432	1.28×10^{-7}	78	291	1.34×10^{-8}
32	132	338	2.34×10^{-8}	50	165	1.11×10^{-9}

Tabteau A.9: problème 4.

n	DFP			$BFGS$		
	k	Ef	$f(x)$	k	Ef	$f(x)$
3	1115	2846	4.05×10^{-8}	137	331	1.21×10^{-8}

Tabteau A.10: problème 5.

n	DFP			$BFGS$		
	k	Ef	$f(x)$	k	Ef	$f(x)$
4	74	485	8.57×10^{-12}	41	292	2.89×10^{-12}

DFP et BFGS avec la recherche linéaire d'Armijo avec reinitialisation

Tabteau A.11: problème 1.

n	DFP			$BFGS$		
	k	Ef	$f(x)$	k	Ef	$f(x)$
2	7	35	2.91×10^{-9}	7	35	2.79×10^{-9}
4	11	59	1.11×10^{-8}	11	59	7.81×10^{-9}
30	92	525	7.16×10^{-9}	69	558	9.55×10^{-9}
80	226	2659	1.21×10^{-9}	191	2023	5.53×10^{-9}

Tabteau A.12: problème 2.

n	DEP			$BFGS$		
	k	Ef	$f(x)$	k	Ef	$f(x)$
2	14	63	2.01×10^{-11}	10	49	4.45×10^{-12}
10	72	401	9.05×10^{-12}	53	344	1.60×10^{-12}
30	102	920	4.17×10^{-13}	113	912	5.53×10^{-13}
50	160	1323	2.36×10^{-13}	151	917	3.71×10^{-12}

Tabteau A.13: problème 3.

n	DEP			$BFGS$		
	k	Ef	$f(x)$	k	Ef	$f(x)$
8	58	412	1.81×10^{-12}	63	402	2.64×10^{-12}
16	82	575	3.14×10^{-8}	82	577	8.11×10^{-13}
40	328	2915	0.66	299	2057	0.66
60	241	1596	0.66	187	1588	1.35×10^{-13}

Tabteau A.14: problème 4.

n	DEP			$BFGS$		
	k	Ef	$f(x)$	k	Ef	$f(x)$
3	525	2743	4.03×10^{-8}	157	719	1.19×10^{-9}

Tabteau A.15: problème 4.

n	DFP			$BFGS$		
	k	Ef	$f(x)$	k	Ef	$f(x)$
4	89	646	3.62×10^{-13}	86	608	2.84×10^{-13}

DFP et BFGS avec la recherche linéaire d'Armijo sans reinitialisation

Tabteau A.16: problème 1.

n	DEP			$BFGS$		
	k	Ef	$f(x)$	k	Ef	$f(x)$
2	10	54	2.39×10^{-8}	35	54	2.37×10^{-9}
4	37	200	2.49×10^{-8}	50	87	1.66×10^{-8}
30	563	3284	2.60×10^{-9}	67	614	4.46×10^{-9}
80	2061	12529	1.67×10^{-9}	93	1894	1.13×10^{-9}

Tabteau A.17: problème 2.

n	DEP			$BFGS$		
	k	Ef	$f(x)$	k	Ef	$f(x)$
2	10	61	1.18×10^{-14}	10	53	6.92×10^{-15}
10	24	128	3.42×10^{-15}	22	126	2.05×10^{-14}
30	46	339	1.03×10^{-15}	44	306	8.36×10^{-15}
50	68	596	1.25×10^{-13}	64	525	5.21×10^{-14}

Tabteau A.18: problème 3.

n	DEP			$BFGS$		
	k	Ef	$f(x)$	k	Ef	$f(x)$
8	93	486	1.02×10^{-11}	40	194	6.81×10^{-10}
16	150	791	5.98×10^{-10}	46	208	6.91×10^{-10}
40	134	694	4.67×10^{-9}	67	332	5.74×10^{-10}
60	130	626	7.68×10^{-9}	88	439	7.45×10^{-9}

TabteauA.19: problème 4

n	DEP			$BFGS$		
	k	Ef	$f(x)$	k	Ef	$f(x)$
3	64	333	2.58×10^{-12}	28	128	2.72×10^{-13}

TabteauA.20: problème 5

n	DEP DEP			$BFGS$ BFGS		
	k	Ef	$f(x)$	k	Ef	$f(x)$
4	317	1784	4.74×10^{-13}	55	243	6.88×10^{-13}

Annexe B

Programmes

B.1 Programme de la méthode de Quasi-newton

Comme la seule différence entre DFP et BFGS est la formule de correction, donc nous avons écrit un programme commun dans le but de faciliter la comparaison entre ces deux méthodes.

Comme fonction test on prend la fonction de rosenbrock généralisée, et comme recherche linéaire on établit la procedure de wolfe.

```
program quasin
! ------------------------------------------------------------
! Programme de DFP ou BFGS avec reinitialisation et la recherche linéaire
! de wolfe appliqué sur la fonction de rosenbrock généralisée.
! ------------------------------------------------------------
! ---------- partie déclaration ----------
implicit none
integer,parameter:: n=50
double precision d(n,n)
double precision p(n),q(n),dr(n),f0,f1,gn(n)
double precision eps,y(n),g(n),l0,ld,lg,w,b,dg
```

```
integer i1,i2,z,j,ct,e,choix,cont
print*,'donner eps '
print*,'=========='
read*,eps
print*,'donner votre choix'
print*,'================'
print*,'1 pour bfgs'
print*,'=========='
print*,'2 pour dfp'
print*,'========'
read*,choix
! ---------- point initial ----------
do i1=1,n
if(mod(i1,2)==0)then
y(i1)=1.0d0
else
y(i1)=-1.2
endif
enddo
! ------------------------------------------------------------
!       z compteur d'itérations
!       cont compteur d'évaluations fonctionelles
! ------------------------------------------------------------
z=1; w=1.d-1; b=7.d-1
f0=f(y); g=df(y); j=1; cont=1
do i1=1,n
do i2=1,n
if(i1==i2)then
```

```
d(i1,i2)=1.0d0
else
d(i1,i2)=0.0d0
endif
enddo
enddo
do
if(sqrt(dot_product(g,g))<=eps.or.e==2)exit
! ---------- calcule de la direction ----------
dr=-matmul(d,g)
! ---------- procédure de powel pour la recherche linéaire ----------
lg=0.d0; ld=100.; l0=1.d0; ct=0.d0
dg=dot_product(g,dr)
do
ct=ct+1
f1=f(y+l0*dr)
if(f1<=f0+w*l0*dg)then
gn=df(y+l0*dr)
if(dot_product(gn,dr)>=b*dg)exit
lg=l0; l0=(lg+ld)/2.
else
ld=l0; l0=(lg+ld)/2.
endif
if(ct==10000)exit
enddo
if(ct==10000)then
print*,'failure'
e=2
```

```
endif
y=y+l0*dr
if(j<n)then
gn=df(y)
p=l0*dr; q=gn-g
if(choix==1)then
call newmatrix1(p,q,d)
endif
if(choix==2)then
call newmatrix2(p,q,d)
endif
j=j+1
else
j=1
do i1=1,n
do i2=1,n
if(i1==i2)then
d(i1,i2)=1.0d0
else
d(i1,i2)=0.0d0
endif
enddo
enddo
endif
f0=f1; g=gn; z=z+1; cont=cont+ct
enddo
! ———————— écriture des resultats ————————
print*,'=========================================='
```

```
print*,'la solution est'
print*,'-----------'
print '(6f8.3)',y
print*,'le nombre d iteration est '
print*,'-----------------------'
print '(i10)',z
print*,'la valeur de la fonction est'
print*,'-----------------------'
print '(d20.6)',f(y)
print*,'le nombre d evaluation fonctionelles est '
print*,'-----------------------'
print '(i10)',cont
print*,'=========================================='
contains
! ---------------------------------------------------
! Sous programme qui calcule la valeur de la fonction objective
! ---------------------------------------------------
function f(u)
double precision f,u(n)
integer i1
f=0
do i1=1,n-1
f=f+100*(u(i1+1)-u(i1)**2)**2+(1-u(i1))**2
enddo
endfunction
! ---------------------------------------------------
! sous programme qui calcule la valeur du gradient
! ---------------------------------------------------
```

```
function df(u)
double precision df(n),u(n)
integer i1
df(1)= 400*u(1)*(u(2)-u(1)**2)-2*(1-u(1))
do i1=2,n-1
df(i1)=-400*u(i1)*(u(i1+1)-u(i1)**2)-2*(1-u(i1))+200*(u(i1)-u(i1-1)**2)
enddo
df(n)=200*(u(n)-u(n-1)**2)
endfunction
! ---------------------------------------------------------------------
! sous programmes qui calculent la valeur de la remise à jour de la
! matrice B qui approxime l'inverse de la hessienne soit par la
! formule de BFGS (newmatrix1), soit par formule de DFP (newmatrix2).
! ---------------------------------------------------------------------
subroutine newmatrix1(p1,q1,d1)
double precision p1(n),q1(n),d1(n,n),a(n,n)
integer j1,j2
double precision s1,s2
s1=dot_product(p1,q1)
s2=dot_product(q1,matmul(d1,q1))
do j1=1,n
do j2=1,n
a(j1,j2)=p(j1)*q(j2)
enddo
enddo
a=matmul(a,d1)
q1=matmul(d1,q1)
do j1=1,n
```

```
do j2=1,n
d1(j1,j2)= d1(j1,j2)+(1/s1)*(1+s2/s1)*p1(j1)*p1(j2)-(1/s1)*(q1(j1)*p1(j2)+a(j1,j2))
enddo
enddo
endsubroutine
subroutine newmatrix2(p1,q1,d1)
double precision p1(n),q1(n),d1(n,n)
integer j1,j2
double precision s1,s2
s1=dot_product(p1,q1)
s2=dot_product(q1,matmul(d1,q1))
q1=matmul(d1,q1)
do j1=1,n
do j2=1,n
d1(j1,j2)= d1(j1,j2)+(1/s1)*p1(j1)*p1(j2)-(1/s2)*q1(j1)*q1(j2)
enddo
enddo
endsubroutine
end
```

B.2 Programme de la méthode de BFGS sans calcul des dérivées

Dans le but de faciliter la programmation ce programme a été divisé en deux sous-programmes:

1. Un sous-programme qui calcule la remise à jour de la matrice B .
2. Un sous-programme qui calcule la valeur de l'approximation du gradient.

Ce programme a été appliqué sur la fonction de powell généralisée (problème10, chapitre 5)

```
program principal
! ------------------------------------------------------------------
! Programme de la méthode de BFGS sans calcule des dérivées appliqué
! sur la fonction de powell généralisée.
! ------------------------------------------------------------------
! ---------- partie déclaration ----------
implicit none
integer,parameter:: n=16
double precision c1,c2,a,b,m,l,lpha,f0
double precision p(n),q(n),f1
double precision h(n,n),d(n),g(n),g1(n),x(n)
integer i,j,k,t,s,w,v,ctf,ctf1
integer (2) hour,minut,second,hsedt
print*,'entrer lpha'
print*,'=============='
read*,lpha
! ---------- calcul de tempt avans l'éxecution ----------
call gettim(hour,minut,second,hsedt)
print*,hour,minut,second,hsedt
data c1,c2/1.0d-1,7.0d-1/
! ---------- point initial ----------
do i=1,n/4
x(4*i-3)=3.d0
x(4*i-2)=-1.d0
x(4*i-1)=0.d0
x(4*i)=1.d0
```

```
enddo
! ---------- k compteur d'itérations ----------
k=0
! ----------la matrice initiale (identite) ----------
do i=1,n
do j=1,n
if(i==j)then
h(i,j)=1.0d0
else
h(i,j)=0.0d0
endif
enddo
enddo
! ---------- w et w1 sont des compteurs ----------
w=0; v=0
f0=f(x)
ctf=1
1 call gradf(x,lpha,ctf,f0,g,ctf1)
2 do
if(norm(g)>=0.001*lpha)exit
w=w+1
if(w==100)exit
lpha=0.5*lpha
ctf=ctf1
call gradf(x,lpha,ctf,f0,g,ctf1)
enddo
if(w==100) goto 100
d=matmul(h,g)
```

```
! ------------------------------------------------------------
! la recherche linéaire
! ------------------------------------------------------------
t=0
do
a=(0.5)**t
ctf1=ctf1+1
if(f(x-a*d)<f0-c1*a*dot_product(d,g).or.a<lpha)exit
t=t+1
enddo
if(a<lpha)then
lpha=(0.5)*lpha; goto 1
endif
s=0
do
b=a*(2**s)
ctf1=ctf1+1
if(f(x-b*d)>f0-c2*b*dot_product(d,g))exit
s=s+1
enddo
3 if((b-a)<lpha)then
lpha=lpha*(0.5)
endif
m=(a+b)*0.5
f1=f(x-m*d)
ctf1=ctf1+1
if(f1>f0-c1*m*dot_product(d,g))then
b=m; goto 3
```

```
else
a=m
endif
ctf=ctf1
call gradf(x-m*d,lpha,ctf,f1,g1,ctf1)
if(dot_product(d,g1)>c2*dot_product(d,g))then
goto 3
else
l=m
endif
x=x-l*d; p=-l*d
q=g1-g
if(dot_product(p,q)<lpha*norm(p))then
v=v+1
if(v==100)then
print*,'point stationnaire'
goto 100
endif
lpha=lpha*(0.5); goto 1
else
call hupdate(p,q,h)
endif
! ---------- nouvelle iteration ----------
g=g1; f0=f(x)
lpha=lpha*(0.5); k=k+1
if(k==100)then
goto 100
else
```

```
goto 2
endif
! ---------------------------------------------------------
! appel au sous programme qui calcule le tempt après l'execution
! ---------------------------------------------------------
100 call gettim(hour,minut,second,hsedt)
! ---------- écriture des resultats ----------
print*,hour,minut,second,hsedt
print*,'=========================================='
print*,'la solution est x='
print '(2d18.6)',x
print*,'la valeur de f(x)='
print '(2d30.8)',f(x)
print*,'la valeur de la norme de g1(x)='
print '(2d30.8)',sqrt(dot_product(g1,g1)),norm(g1)
print*,'le nombre d iteration est=',k
print*,'le nombre d ivaluation fonctionnelle est=',ctf1
print*,'=========================================='
contains
! ---------------------------------------------------------
! Sous programme qui donne la valeur de la fonction objective
! ---------------------------------------------------------
function f(u)
double precision f,u(n)
integer i1
f=0.d0
do i1=1,n/4
f=f+(u(4*i1-3)+10*u(4*i1-2))**2+5*(u(4*i1-1)-u(4*i1))**2+&
```

```
(u(4*i1-2)-2*u(4*i1-1))**4+10*(u(4*i1-3)-u(4*i1))**4
enddo
endfunction
function norm(u)
double precision norm,u(n)
norm=0.0d0
do i=1,n
norm=norm+u(i)**2
enddo
norm=sqrt(norm)
endfunction
! ------------------------------------------------------------
! sous programme qui calcule la remise à jour de la matrice B par
! formule de BFGS
! ------------------------------------------------------------
subroutine hupdate(p1,q1,h1)
double precision p1(n),q1(n),h1(n,n),a1(n,n)
integer j1,j2
double precision r1,r2
r1=dot_product(p1,q1)
r2=dot_product(q1,matmul(h1,q1))
do j1=1,n
do j2=1,n
a1(j1,j2)=p(j1)*q(j2)
enddo
enddo
a1=matmul(a1,h1)
q1=matmul(h1,q1)
```

```
do j1=1,n
do j2=1,n
h1(j1,j2)= h1(j1,j2)+1/r1*(1+r2/r1)*p1(j1)*p1(j2)-1/r1*(q1(j1)*p1(j2)+a1(j1,j2))
enddo
enddo
endsubroutine
! ---------------------------------------------------------------
! sous programme qui calcule la valeur du gradient par les
! différences fini
! ---------------------------------------------------------------
subroutine gradf(y,lpha1,cot,ff0,gr3,cont)
implicit none
integer,parameter:: m1=16
double precision y(m1),gr3(m1),z(m1)
double precision lpha1,ff0
integer i1,i2,cont,cot
cont=cot
do i1=1,m1
do i2=1,m1
if(i1==i2)then
z(i2)=1.0d0
else
z(i2)=0.0d0
endif
enddo
if(lpha1<=1.0d-6)then
gr3(i1)=(f(y+lpha1*z)-f(y-lpha1*z))/(2*lpha1)
cont=cont+2
```

```
else
gr3(i1)=(f(y+(lpha1**2)*z)-ff0)/(lpha1**2)
cont=cont+1
endif
enddo
endsubroutine
end
```

Bibliographie

[1] Aitken, A.C. *"On Bernoulli's numerical solution of algebraic equations."*, Proc. Roy. Soc. Edinburgh, 46, pp. 289-305, 1926.

[2] L. Armijo. "*Minimisation of functions having Lipshitz continus first partial derivatives.*" Pacific journal of Mathematics, 16, pp. 1-3, 1966.

[3] J. Baptiste & H. Urruty. *Optimisation et Analyse Convexe.* Presse Universitaire de France 1998.

[4] J.F. Bonnans, J.C Gilbert, C.Lemarechal, C.Sagastizbal. 78153, le Chesnay, France1995.

[5] M.S. Bazaraa, H .D. Sherali & C .M . Shetty. *Nonlinear Programming.* John Wiley & Sons , New york , 1993

[6] E.M.L. Beale. *Introduction to Optimisation.* Jhon Wiley & sons, Inc. 1987.

[7] J. Brauninger. "*A Variable Metric Algorithm for Unconstrained Minimization Without Evaluation of Derivatives*". Numer. Math.Vol 36, pp, 359-373, 1981.

[8] C.G. Broyden. "*Quasi-Newton Methods and their application to Function Minimization*", Mathematics of Computation, 21 pp.368-381, 1967

[9] C.G.Broyden, J.E.,Dennis,and J.J. Moré. " *On the local and superlinear convergence of quasi-Newton Methods* ", J. Institute of Mathematics and its Applications, 12, pp. 223-246, 1973.

[10] C.G. Broyden. " *The convergence of a class of double rank minimization algorithm 2. The new algorithm,"* J. Institute of mathematics and its applications, 6, pp. 222-231, 1970.

[11] M. J. Box, D. Davies, W. H. Swann. *Techniques d 'optimisation non lièaire.* Entreprise Moderne D'Edition , France ,1971.

[12] S. Boukaroura. *Etude du Charachtère dans L'algorithme de Karmakar.* Thèse de Magister, Université de Setif, 1997.

[13] D.P.Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods.* Academic Press 1982.

[14] Brezinski, C. *Algorithmes d'accélération de la convergence: Etude numérique,* Editions Technip, Paris 1978.

[15] J.C Culioli. *Introduction à l'Optimisation.* Ellipses, France, 1994.

[16] P. G. Ciarlet. *Introduction à l'analyse Numèrique Matricielle et à l'Optimisation.* Massan, France, 1985.

[17] Cauchy, A.L. *"Méthode générale pour la résolution des systèmes d'équations simultanées"*, C. R. Acad. Sci. Paris, 25, 1847.

[18] W.C. Davidon. "*Variable Metric Method for Minimization*", AEC research Development, Report ANL-5990, 1959

[19] Durand, E. *Solutions numériques des équations algébriques.* Tomes 1 et 2, Masson, 1961.

[20] J.E. Dennis, & J. J. Moré. *"A Characterization of Superlinear Convergence and Its Application to Quasi-Neuton Methods"*. Mathematics of computing, vol, 28, number 126, pp, 549-560, april 1974.

[34] B. Pchenitchny & Y . Daniline. *Mèthodes Numèriques dans Les Problèmes d Extrèmum* .1965.

[35] J. D. Pearson. "*Variable Metric Methods of Minimization*". Computer journal, Vol 12, pp 171-178, 1969.

[36] M.J.D. Powell. "*Quadratic temination Properties of Minimization Algorithms I,II,*" J. Institute of Mathematics and its Applications, 10, pp. 333-342, 343-357, 1972.

[37] M. J. D. Powell. "*On the Convergence of the Variable Metric Algorithm*" J. Inst. Maths Applics, vol 7, pp 21-36, 1971.

[38] M. J. D. Powell. "*Some Global Convergence Properties of a Variable Metric Algorithm for Minimisation Without Exact Line Searches*". Non linear Programming, SIAM, AMS Proceedings, Vol. IX, R. W. Cottle and C.E. Lemke (Eds), New York, pp. 53-72, 1976.

[39] M. J. D. Powell. *"Updating conjugate directions by the BFGS formula"*. Mathematical Programming, vol,38, pp, 29-46, 1987.

[40] D.F. Shanno. "*Conditionning of quasi-Newton Methods for function minimization* ", Mathematics of Computation, 24, pp. 641-656, 1970.

[41] Vignes, J. *Algorithmes numériques: Analyse et mise en oeuvre*. Vol.2, Editions Technip, Paris 1980.

[42] P. Wolfe. "Convergence conditions for ascent methods." Siam Review, 11, pp. 226-235, 1969.

[43] Wynn, P. "*On a device for computing the $e_m(S_n)$ transformation*", M.T.A.C., 10, pp. 91-96. 1956.