Présentée par : **MAALEM Soulef et FERHI Ibtissam**
Dirigé par : **Dr.BOUALI Tahar**
Faculté des Sciences Exactes et Sciences de Nature et de Vie
Département de Mathématiques
Université Laarbi Tébessi
Tébessa .

21 mai 2017

TABLE DES MATIÈRES

Dédicace

Je dédie ce modeste travail...

A les êtres qui mes sont les plus chers, mes parents ; mon père qui m'a aidé et m'a soutenu moralement ; ma mère qui a consacré tous ses efforts pour atteindre mes objectifs, sans eux aucune réussite n'aurait été possible.

A mes frères : Hichem, Amer.et Talal.

A mes sœur : Ahlem, Hadia et Rayen.

A ma sœur Awatef que Dieu son âme.

A mes chères amies : Meriem, Noura et Chafika.

A tous mes professeurs qui m'ont aidé et encouragé pour suivre mes études surtout :

- Mr. HAFDAHALLAH Abdelhak.

A ma binôme Ibtissam et sa familles.

A tous mes chers amis, et à toute ma famille.

Enfin, à tous ceux qui ont collaboré de près ou de loin à la réalisation de ce travail, en guise de reconnaissance.

MAALEM Soulef

Dédicace

Je dédie ce modeste travail :

A mon mère qui a consacré tous ses efforts

pour atteindre mes objectifs.

A mon père qui m'a aidé et m'a soutenu moralement.

A ma fille petite.

A mes frères :Fayçel. Fared. Mohamed. Aymen. Abd almonsef.

A mes sœurs :Saida. Abla. Rayan.

A ma trèe chère amie latifa

A ma binôme Soulef et sa familles.

A tous mes amis et tous mes collègues sans exception

FERHI Ibtissam

Remerciements

Nous remercions **Dieu** qui nous a donné le courage et la volonté de poursuivre nos études, ainsi qui nos parents qui ont sacrifié leur vie pour notre réussite

Nous tenons à adresser nos sincères remerciements et nos plus grand respect à notre Monsieur **BOUALI Tahar** pour sa compréhension, Sa disponibilité, de savoir faire, ses conseils judicieux, et toute l'aide qu'elle nous a rapporté.

Nos remerciements s'adressent égalent aux membres du jury qui ont accepté d'évaluer notre travail et de nous avoir honorés par leur présence.

Nous remercions toute la famille, tous les amis leurs encouragements.

Nous remercions tous ceux qui ont contribué de prés ou de loin à la mise en œuvre de ce travail.

Résumé Arabe

Résumé

Soit $f : \mathbb{R}^n \to \mathbb{R}$ est une fonction continûment différentiable. On cherche à résoudre le problème d'optimisation sans contraintes suivant :

$$(\mathcal{P}) : \begin{cases} \min_{x \in \mathbb{R}^n} f(x) \ . \end{cases}$$

On s'intéresse dans ce mémoire à une nouvelle classe de méthodes du gradient conjugué à trois terme introduite par **D. Songhai** et **W. Zhong**, pour résoudre les problèmes d'optimisation sans contrainte de grand taille. Pour cette classe des méthodes du gradient conjugué avec la recherche linéaires de **Wolfe** fortes la propriété de descente est assurée à chaque itération. Certains résultats généraux de convergence sont également établis pour la nouvelle famille de gradient conjugué.

On fera une synthèse des derniers travaux concernant ces méthodes. Des tests numériques sont donnés pour montrer la performance numérique du nouvel algorithme.

Mots clés : Gradient conjugué, Algorithme, Convergence globale, Recherche linéaire inexacte, Méthode de gradient conjugué à trois termes, Règle de **Wolfe**.

Abstract

Consider the following unconstrained optimization problem (where $f : \mathbb{R}^n \to \mathbb{R}$ is countinious and differentiabl) :

$$(\mathcal{P}) : \begin{cases} \min\limits_{x \in \mathbb{R}^n} f(x) \, . \end{cases}$$

In this memory we are interested in a new class of a three-term conjugate gradient algorithm introduced by **D. Songhai** and **W. Zhong**, for solving large-scale unconstrained optimization problems. It is important that the proposed method produce sufficient descent search direction at every iteration with the strong **Wolfe** line searches, and the global convergence properties of the given methods are discussed.

We will synthesize latest research on these methods. The numerical results show that the proposed methods are efficient.

Key words : Conjugate gradient, Algorithm, Global convergence, Inexact line search, Three-term conjugate gradient method, **Wolfe** line search.

Introduction

L'optimisation est une branche des mathématiques et de l'informatique en tant que disciplines, cherchant à modéliser, à analyser et à résoudre analytiquement ou numériquement les problèmes qui consistent à déterminer quelles sont la ou les solution(s) satisfaisant un objectif quantitatif tout en respectant d'éventuelles contraintes.

L'optimisation joue un rôle important en recherche opérationnelle (domaine à la frontière entre l'informatique, les mathématiques et l'économie), dans les mathématiques appliquées (fondamentales pour l'industrie et l'ingénierie), en analyse et en analyse numérique, en statistique pour l'estimation du maximum de vraisemblance d'une distribution, pour la recherche de stratégies dans le cadre de la théorie des jeux, ou encore en théorie du contrôle et de la commande.

Aujourd'hui, tous les systèmes susceptibles d'être décrits par un modèle mathématique sont optimisés. La qualité des résultats et des prédictions dépend de la pertinence du modèle, de l'efficacité de l'algorithme et des moyens pour le traitement numérique.

Les domaines d'applications sont extrêmement variés : optimisation d'un trajet, de la forme d'un objet, d'un prix de vente, d'une réaction chimique, du contrôle aérien, du rendement d'un appareil, du fonctionnement d'un moteur, de la gestion des lignes ferroviaires, du choix des investissements économiques, de la construction d'un navire, etc. L'optimisation de ces systèmes permet de trouver une configuration idéale, d'obtenir un gain d'effort, de temps, d'argent, d'énergie, de matière première, ou encore de satisfaction.

Très loin de constituer une liste exhaustive, ces quelques exemples attestent de la variété des formulations et préfigure la diversité des outils mathématiques susceptibles de résoudre ces problèmes.

Plus formellement, l'optimisation est l'étude des problèmes qui s'expriment de la manière suivante.

Étant donné une fonction $f : \mathbb{R}^n \to \mathbb{R}$, trouver un élément $\overline{x}$ de $\mathbb{R}^n$ tel que $f(\overline{x}) \leq f(x)$ pour tout $x \in \mathbb{R}$.

On dit que l'on cherche à minimiser la fonction f sur l'ensemble $\mathbb{R}$. La fonction f porte divers noms : fonction-coût ou simplement coût, fonction-objectif ou simplement objectif, critère, etc.

L'optimisation linéaire étudie le cas où la fonction objectif et les contraintes caractérisant l'ensemble A sont linéaires. C'est une méthode très employée pour établir les programmes des raffineries pétrolières, mais aussi pour déterminer la composition la plus rentable d'un mélange salé, sous contraintes, à partir des prix de marché du moment.

L'optimisation linéaire en nombres entiers étudie les problèmes d'optimisation linéaire dans lesquels certaines ou toutes les variables sont contraintes de prendre des valeurs

entières. Ces problèmes peuvent être résolus par différentes méthodes : séparation et évaluation, méthode des plans sécants.

L'optimisation quadratique étudie le cas où la fonction objective est une forme quadratique (avec contraintes linéaires)

L'optimisation non linéaire étudie le cas général dans lequel l'objectif ou les contraintes (ou les deux) contiennent des parties non linéaires, éventuellement non-convexes.

L'optimisation stochastique étudie le cas dans lequel certaines des contraintes dépendent de variables aléatoires. En optimisation robuste, les aléas sont supposés être situés dans des intervalles autour de positions nominales et on cherche à optimiser le système soumis à de tels aléas, dans le pire des cas.

La programmation dynamique utilise la propriété qu'une solution se compose nécessairement de sous-solutions optimales (attention : le contraire n'est pas vrai en général) pour décomposer le problème en évitant l'explosion combinatoire. Elle est utilisable lorsque la fonction objectif est une somme de fonctions monotones croissantes dont les arguments sont des inconnues distinctes. C'est la programmation dynamique qui permet par exemple :

– - aux avionneurs de trouver les plans de décollage optimaux de leurs engins,

- aux ingénieurs de bassin de répartir la production minière entre leurs différents puits,

- aux producteurs d'électricité de planifier la marche des usines hydroélectriques,

- aux media planners de répartir efficacement un budget de publicité entre différents supports.

Formellement on peut écrire ce problème noté $(\mathcal{P})$ de la manière suivante :

Soit $f : \mathbb{R}^n \to \mathbb{R}$ est une fonction continûment différentiable et $(\mathcal{P})$ le problème de minimisation sans contraintes suivant :

$$(\mathcal{P}) : \begin{cases} \min\limits_{x \in \mathbb{R}^n} f(x) \ . \end{cases}$$

Pour trouver (ou approcher) un point en lequel une fonction de plusieurs variables est minimum, il est classique de remplacer le problème initial par une suite (infinie en générale) qui converge vers la solution du probléme de minimisation où la fonction économique ne dépend plus que d'une variables : c'est le principe des méthodes de descente. Autrement dit ; de construire un algorithme itératif de la forme :

$$x_{k+1} = x_k + \alpha_k d_k, \tag{1}$$

telle que α_k est appelé le pas. il est déterminé par une optimisation unidimensionnelle ou recherche linéaire exacte ou inexacte. d_k est la direction de descente qui faire les différents méthodes à direction de descente.

Parmi les plus anciennes méthodes utilisées pour résoudre les problèmes d'optimisation $(\mathcal{P})$, on peut citer la méthode du Gradient conjugué. Cette méthode est surtout utilisée pour les problèmes de grand taille dont la direction d_k est calculée par la formule récurrente suivant :

$$d_k = \begin{cases} -g_0 & \text{si } k = 0 \\ -g_k + \beta_k d_{k-1} & \text{si } k \geq 1 \end{cases}, \tag{2}$$

telle que : $g_k = \nabla f(x_k)$.

Les différentes valeurs attribuées à β_k définissent les différentes formes du gradient conjugué. Si on note $y_k = g_{k+1} - g_k$, on va présenter quelques variantes suivantes :

1- Gradient conjugué variante **Hestenes-Stiefel (HS)**([15], 1952) :

$$\beta_k^{HS} = \frac{g_{k+1}^T y_k}{d_k^T y_k}.$$

2- Gradient conjugué variante **Fletcher Reeves (FR)**([13], 1964) :

$$\beta_k^{FR} = \frac{\|g_k\|^2}{\|g_{k-1}\|^2}.$$

3- Gradient conjugué variante **Polak-Ribière-Polyak (PRP)**([20-21],1969) :

$$\beta_k^{PRP} = \frac{g_k^T y_k}{\|g_{k-1}\|^2}.$$

4- Gradient conjugué variante **Dai-Yuan (DY)**([11],1999) :

$$\beta_k^{DY} = \frac{\|g_{k+1}\|^2}{d_k^T y_k}.$$

5- Gradient conjugué variante **Hager-Zhang (HZ)**([17],2005) :

$$\beta_k^{HZ} = \frac{1}{d_k^T y_k}\left(y_k - 2d_k \frac{\|y_k\|^2}{d_k^T y_k}\right) g_{k+1}.$$

Malgré que toutes ces méthodes ont été très importantes pour la résolution du problème $(\mathcal{P})$, leur étude s'est faite de façon individuelle et chacune à part par **Flécher, Powell, Wolf, El Baali, Polak-Ribière, Polyak,Y. H. Dai and Y. Yuan**,... Il est alors légitime de se poser le problème suivant :

Peut-on regrouper toutes ces méthodes et définir d'autres nouvelles et les mettre dans une même famille et étudier leurs propriétés de descente et de

convergence ensemble ?

E.M.L. Beale est répondu dans ([8]) positivement à cette question. Le but principal de ce mémoire est d'étudier de façon approfondie le travail de **D. Songhai et W. Zhong** ([23]).

La forme unifiée du gradient conjugué en question, est de la forme (2) et la direction de recherche a été déterminée par :

$$d_{k+1} = -g_{k+1} + \beta_k d_k + \gamma_k d_t, \tag{3}$$

où le paramètre β_k peut être donnée par β_k^{HS}, β_k^{FR} ou β_k^{DY}, ..., etc..., et le paramètre γ_k est donnée par :

$$\gamma_k = \begin{cases} 0 & k = t+1, \\ \frac{g_{k+1}^T y_t}{d_t^T y_t} & k > t+1. \end{cases} . \tag{4}$$

Le mémoire est divisé en trois chapitres.

Le premier chapitre contient les notions de base utiles pour la suite. On insiste surtout sur les notions de direction de descente, sur les conditions d'optimalité des problèmes de minimisation sans contraintes.

Le deuxième chapitre est consacré aux recherches linéaires inexactes célèbres (**Armijo**, **Wolf et Goldstein**). On montre comment ces recherches linéaires contribuent dans la convergence des algorithmes à directions de descente.

Le dernier chapitre est consacré à un algorithme de gradient conjugué à trois termes pour résoudre les problèmes d'optimisation sans contrainte de grand taille introduite par **D. Songhai** et **W. Zhong** ([23]). Cette classe génère des suites $\{x_k\}_{k\in\mathbb{N}}$ de la façon suivante :

$$x_{k+1} = x_k + \alpha_k d_k.$$

Le pas $\alpha_k \in \mathbb{R}$ est déterminé par une optimisation unidimensionnelle ou recherche linéaire inexacte.

Les directions d_k sont calculées de façon récurrente par les formules suivantes :

$$d_{k+1} = -g_{k+1} - \delta_k s_k - \eta_k y_k, \tag{5}$$

où :

$$\delta_k = \left(1 - \min\left\{1, \frac{\|y_k\|^2}{y_k^T s_k}\right\}\right) \frac{s_k^T g_{k+1}}{y_k^T s_k} - \frac{y_k^T g_{k+1}}{y_k^T s_k}, \tag{6}$$

et

$$\eta_k = \frac{y_k^T g_{k+1}}{y_k^T s_k}. \tag{7}$$

CHAPITRE 1

Généralités sur l'optimisation sans contraintes

1.1 Rappel du calcul différentiel

1.1.1 Différentiabilité et rappel du gradient et hessienne

Définition 1.1.1 *Soit $f : \Omega \subset \mathbb{R}^n \to \mathbb{R}$ définie sur l'ouvert Ω. f est dite différentiable en x_0 s'il existe une application $g : \mathbb{R}^n \to \mathbb{R}$ linéaire telle que :*

$$f(x_0 + h) = f(x_0) + g(h) + \|h\| \varepsilon(h),$$

avec :

$$\lim_{h \to 0} \varepsilon(h) = 0.$$

La fonction g est représentée (dans $(\mathbb{R}^n, \langle \cdot, \cdot \rangle)$ euclidien) par un unique vecteur.

Dans le cas où f est deux fois différentiable en x_0, on a le développement de Taylor-Young d'ordre 2 suivant :

$$f(x_0 + h) = f(x_0) + \langle \nabla f(x_0), h \rangle + \frac{1}{2} \left\langle \nabla^2 f(x_0) h, h \right\rangle + \|h\|^2 \varepsilon(h).$$

Définition 1.1.2 *Pour tout $x \in \Omega$ et $h \in \mathbb{R}^n$ on note (quand existe) **la dérivée directionnelle** de f en x de direction h par :*

$$\frac{\partial f}{\partial h}(x) = \lim_{t \to 0} \frac{f(x + th) - f(x)}{t} = g'(0),$$

où

$$g(t) = f(x + th).$$

Nous rappelons aussi la formule

$$\frac{\partial f}{\partial h}(x) = \langle \nabla f(x), h \rangle, \forall x \in \Omega, \forall h \in \mathbb{R}^n.$$

- Plus généralement, soit $F : \Omega \subset \mathbb{R}^n \to \mathbb{R}^m$ une fonction de plusieurs variables qui à $x \in \Omega$ associe $F(x) = \begin{pmatrix} f_1(x) \\ \vdots \\ f_m(x) \end{pmatrix}$. On dit qu'elle est différentiable en x_0 si chacune des fonctions composantes $f_1, \cdots, f_m$ l'est, et on note la matrice jacobienne de F en x par $J_F(x) \in \mathcal{M}_{m,n}(\mathbb{R})$ telle que :

$$J_F(x) = \begin{pmatrix} \nabla f_1(x) \\ \vdots \\ \nabla f_m(x) \end{pmatrix} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(x) & \cdots\cdots & \frac{\partial f_1}{\partial x_n}(x) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1}(x) & \cdots\cdots & \frac{\partial f_m}{\partial x_n}(x) \end{pmatrix}.$$

Définition 1.1.3 *On dit que f est deux fois différentiable en x lorsque le gradient est différentiable en x. Pour tout $x \in \Omega$ on note (quand existe) $\nabla^2 f(x) \in \mathcal{M}_n(\mathbb{R})$ la matrice carrée donnée par :*

$$\nabla^2 f(x) = H(x) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2}(x) & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n}(x) \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1}(x) & \cdots & \frac{\partial^2 f}{\partial x_n^2}(x) \end{pmatrix},$$

($H(x)$ est une matrice symétrique s'appelle ***la matrice hessienne*** *de f en x).*

Exemple 1.1.1 *Soit $f : \mathbb{R}^n \to \mathbb{R}$ définie par :*

$$f(x) = \langle a, x \rangle, \forall x \in \mathbb{R}^n,$$

où a est un vecteur donnée (c'est à dire une fonction linéaire). Alors on calcul facilement : $\frac{\partial f}{\partial x_k} = a_k$, donc :

$$\nabla f(x) = a,$$

le gradient est constante, ceci nous donne :

$$\nabla^2 f(x) = H(x) = 0.$$

1.2 Notions de convexité

Le cas où les données sont convexes est un cas très important car les problèmes quadratiques sont à la base de nombreux des algorithmes non linéaires.

1.2.1 Ensembles convexes

Définition 1.2.1 *On dit que l'ensemble $C \subseteq \mathbb{R}^n$ est convexe si et seulement si :*

$$\forall (x, y) \in C^2, \forall t \in [0, 1], tx + (1 - t)y \in C.$$

Autrement dit, C est convexe s'il contient tout segment reliant deux quelconque de ses points.

Exemple 1.2.1 *1- Un intervalle $[a, b]$ ou $]a, b[$ est convexe dans $\mathbb{R}$.*

2- $\mathbb{R}$ est un ensemble convexe mais $\mathbb{R}^*$ n'est pas un ensemble convexe.

3- Une réunion disjointe d'intervalles de $\mathbb{R}$ n'est pas convexe.

1.2.2 Fonctions convexes

Définition 1.2.2 (Fonction convexe) *On dit que la fonction $f : C \subseteq \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ est convexe si C est une ensemble convexe et si :*

$$\forall (x, y) \in C^2, \forall t \in [0, 1]; \quad f(tx + (1 - t)y) \leq tf(x) + (1 - t) f(y).$$

Définition 1.2.3 (Domaine d'une fonction convexe) *Soit $f : C \subseteq \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ une fonction convexe. On appelle domaine de f l'ensemble :*

$$\text{dom}(f) = \{x \in \mathbb{R}^n \mid f(x) < +\infty\}.$$

- Cette ensemble est convexe.
- Lorsque le domaine de f est non vide, f est dite fonction propre.

Définition 1.2.4 (Fonction strictement convexe) *On dit que $f : C \subseteq \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ est strictement convexe si C est convexe et si :*

$$\forall (x, y) \in C^2 \text{ avec } x \neq y, \forall t \in]0, 1[; \quad f(tx + (1 - t)y) < tf(x) + (1 - t) f(y).$$

Exemple 1.2.2 ***1-*** *$f(x) = \|x\|$ est strictement convexe.*

2- *Tout application affine est convexe. (c'est à dire de la forme :*

$$f(x) = \langle \alpha, x \rangle + \beta,$$

où α et x dans le domaine de définition et $\beta \in \mathbb{R}$.

- On dit qu'une matrice carré symétrique est semi définie positive si

$$\forall X \neq 0, X^T AX \geq 0.$$

A est définie positive si l'inégalité précédente est stricte.

Exemple 1.2.3 *Soit A une matrice carrée d'ordre n et $b \in \mathbb{R}^n$ et soit la fonction définie par :*

$$f(x) = \frac{1}{2} \langle Ax, x \rangle_n - \langle b, x \rangle_n .$$

Si A est semi définie positive alors f est convexe, sinon A est définie positive et f est strictement convexe.

1.2.3 Minimum (maximum) d'une fonction

- On appelle qu'une boule ouverte de $\mathbb{R}^n$ de centre x^* et de rayon r est l'ensemble :

$$B(x^*, r) = \{x \in \mathbb{R}^n \mid \|x - x^*\| < r\},$$

Où $\|.\|$désigne la norme de $\mathbb{R}$.

Définition 1.2.5 (Minimum et maximum local) *Soit C un ensemble non vide de $\mathbb{R}^n$ et f une fonction de C dans $\mathbb{R}$.*

On dit que $x^* \in C$ réalise ***un minimum local*** *de f sur C si on peut trouver une boule ouverte $B(x^*, r)$ de $\mathbb{R}^n$ telle que :*

$$\forall x \in B(x^*, r) \cap C \quad f(x^*) \leq f(x).$$

Définition 1.2.6 *On dit que $x^* \in C$ réalise* ***un maximum local*** *de f sur C si on peut trouver une boule ouverte $B(x^*, r)$ de $\mathbb{R}^n$ telle que :*

$$\forall x \in B(x^*, r) \cap C \quad f(x^*) \geq f(x).$$

Définition 1.2.7 (Minimum et maximum global) *On dit que $x^* \in C$ réalise **un minimum global** de f sur C si :*

$$\forall x \in C \quad f(x^*) \leq f(x).$$

*On dit que $x^* \in C$ réalise **un maximum global** de f sur C si :*

$$\forall x \in C \quad f(x^*) \geq f(x).$$

Définition 1.2.8 (Minimum et maximum strict) *Le minimum et le maximum sont dites **strict** si les inégalités dans les définitions précédentes sont $(<, >)$.*

1.2.4 Continuité des fonctions convexes

Donnons maintenant quelques propriétés (topologiques) importantes des fonctions convexes.

Théorème 1.2.1 *Soit $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ une fonction convexe, il est équivalent de dire :*

***1-** Il existe un ouvert non vide O sur lequel f est majorée par une constante réelle et ne vaut pas constamment $(-\infty)$.*

***2-** f est propre, dom (f) est d'intérieur non vide et f est continue sur l'intérieur de son domaine.*

Lemme 1.2.1 *Si au voisinage d'un point $u_0 \in \mathbb{R}$, une fonction convexe f est majorée par une constante finie,alors f est continue en u_0.*

Preuve. (Du lemme) En se ramène par translation au cas où $u_0 = 0$ et $f(0) = 0$. Soit V un voisinage de l'origine tel que $f(u) \leq a < +\infty$, pour tout $u \in V$. posons $W = V \cap -V$ et donnons nous $\varepsilon \in]0, 1[$. Si $u \in \varepsilon W$ on a par convexité :

$$\frac{u}{\varepsilon} \in W \text{ donc } f(u) \leq (1-\varepsilon)f(0) + \varepsilon f(\frac{u}{\varepsilon}) \leq \varepsilon a,$$

$$-\frac{u}{\varepsilon} \in W \text{ donc } f(u) \geq (1+\varepsilon)f(0) - \varepsilon f(-\frac{u}{\varepsilon}) \geq \varepsilon a.$$

Finalement

$$\forall u \in W \quad |f(u)| \leq \varepsilon a.$$

D'où la continuité. ■

Preuve. (Du théorème) Supposons donc **"1"** est vérifié ; O est donc inclus dans l'intérieur du domaine de f qui est en particulier non vide. Donc f est propre. Soit $u \in O$ tel que $f(u) > -\infty$. D'après **le lemme 1** f sera continue en u donc finie sur un voisinage de u. pour tout $v \in int(dom(f))$, il existe $\rho > 1$ tel que $w = u + \rho(v-u)$ qui est appartienne à l'intérieur de $dom(f)$ car l'intérieur d'un convexe est un convexe. L'homothétie h de centre w et de rapport $1 - \dfrac{1}{\rho}$ transforme u en v et O en un ouvert $h(O)$ contenant v. Pour tout v' de $h(O)$, on a par convexité :

$$f(v') \leq \frac{\rho-1}{\rho} f(h^{-1}(v')) + \frac{1}{\rho} f(w) \leq \frac{\rho-1}{\rho} a + \frac{1}{\rho} f(w).$$

Par conséquent :tout point v de l'intérieur de domaine de f possède un voisinage $h(O)$ sur lequel f est majorée par une constante finie.

D'après le **lemme 1** f est continue en v. ■

Corollaire 1.2.1 *Toute fonction convexe propre sur un espace de dimension finie est continue sur l'intérieur de son domaine.*

1.2.5 Différentiabilité des fonctions convexes

Définition 1.2.9 *Soit f est une fonction de $\mathbb{R}^n$ dans $\mathbb{R} \cup \{+\infty\}$. On dit que f est* ***Gâtaux-différentiable*** *en $u \in \mathrm{dom}(f)$ si la dérivée directionnelle :*

$$f'(u,v) = \lim_{t \to 0^+} \frac{f(u+tv) - f(u)}{t},$$

existe dans toute direction $v \in \mathbb{R}^n$, et si l'application $v \to f'(u,v)$ est linéaire. On note :

$$f'(u,v) = \langle \nabla f(u), v \rangle ,$$

où $\langle .,. \rangle$ désigne le produit scalaire de $\mathbb{R}^n$. L'élément $\nabla f(u)$ de $\mathbb{R}^n$ est le gradient de f en u.

Remarque 1.2.1 *Si f est différentiable au sens classique en u (on dit alors* ***Fréchet -différentiable****), alors f est Gâteaux-différentiable en u, la réciproque est fausse, comme le montre le contre exemple suivant :*

$$f(x,y) = \begin{cases} x & si\ x = y^2 \\ 0 & si\ non \end{cases}.$$

La fonction f est continue en $(0,0)$ et Gâteaux-différentiable en $(0,0)$ mais pas Fréchet différentiable en $(0,0)$.

Théorème 1.2.2 *Soit $f : C \subset \mathbb{R}^n \longrightarrow \mathbb{R}$ Gâteaux différentiable sur C avec C convexe. f est convexe si et seulement si :*

$$\forall (u,v) \in C \times C : f(v) \geq f(u) + \langle \nabla f(u), v-u \rangle .$$

Preuve. Montrons l'implication :

$$f \text{ est convexe} \Longrightarrow \forall (u,v) \in C \times C : f(v) \geq f(u) + \langle \nabla f(u), v-u \rangle .$$

Donc, supposons f est convexe. On a par définition :

$$\forall t \in]0,1] , \forall u, v \in C^2 \ f(tv + (1-t)u) \leq tf(v) + (1-t)f(u),$$

$$\Longrightarrow f(u + t(v-u)) \leq f(u) + t(f(v) - f(u)),$$

$$\Longrightarrow f(u + t(v-u)) - f(u) \leq t(f(v) - f(u)),$$

divisant par $t \in]0,1]$ et passant au limite quand $t \to 0^+$ on obtient :

$$\lim_{t \to 0^+} \frac{f(u + t(v-u)) - f(u)}{t} \leq (f(v) - f(u)),$$

$$\lim_{t \to 0^+} \frac{f(u + t(v-u)) - f(u)}{t} + f(u) \leq f(v),$$

ce qui donne :

$$\langle \nabla f(u), v-u \rangle + f(u) \leq f(v),$$

ce qu'il fallait démontrer. Montrons maintenant réciproquement,

$$\forall (u,v) \in C \times C : f(v) \geq f(u) + \langle \nabla f(u), v-u \rangle \Longrightarrow f \text{ est convexe.}$$

On a alors :

$$\forall (u,v) \in C \times C : f(v) \geq f(u) + \langle \nabla f(u), v-u \rangle ,$$

On applique cette inégalité à $\langle u + t(v-u), u \rangle$ donc :

$$\begin{aligned} \forall (u,v) &\in C \times C : f(u) \geq f(u + t(v-u)) + \langle \nabla f(u + t(v-u)), u - u - t(v-u) \rangle , \\ \forall (u,v) &\in C \times C : f(u) \geq f(u + t(v-u)) - t \langle \nabla f(u + t(v-u)), (v-u) \rangle . \end{aligned} \tag{1}$$

On applique aussi cette inégalité à $\langle (u + t(v-u)), v \rangle$ donc :

$$\forall (u,v) \in C \times C : f(v) \geq f(u + t(v-u)) + \langle \nabla f(u + t(v-u)), (v-u) - t(v-u) \rangle ,$$

$$\forall (u,v) \in C \times C : f(v) \geq f(u + t(v-u)) + (1-t)\left\langle \nabla f(u + t(v-u)), v-u \right\rangle . \quad (2)$$

Multipliant (1) fois $(1-t)$ et (2) fois (t) et on fait l'addition (1)+(2) :

$$\forall (u,v) \in C \times C : tf(v) + (1-t)f(u) \geq (t + 1 - t)f(u + t(v-u)),$$

Ce qui vérifié que f est convexe. D'où la démonstration du théorème. ■

Théorème 1.2.3 *Soit $f : C \subset \mathbb{R}^n \longrightarrow \mathbb{R}$ gâteaux différentiable sur C avec C est convexe. f est convexe si et seulement si ∇f est un opérateur monotone, c'est-à-dire :*

$$f \text{ est convexe} \Longleftrightarrow \forall (u,v) \in C \times C : \langle \nabla f(u) - \nabla f(v), u - v \rangle \geq 0.$$

1.3 Optimisation sans contraintes

Soit le problème :

$$(\mathcal{P}) = \left\{ \min_{x \in \mathbb{R}^n} f(x) \right. ,$$

où f est une fonction de $\mathbb{R}^n$ vers $\mathbb{R} \cup \{+\infty\}$.

Avons d'étudier les propriétés de la solution (ou les solutions) de $(\mathcal{P})$, il faut s'assurer de leur existence et donner des résultats d'unicité.

Définition 1.3.1 (Coércivité) *On dit que $f : \mathbb{R}^n \to \mathbb{R}$ est **coercive** si :*

$$\lim_{\|x\| \to +\infty} f(x) = +\infty,$$

tel que $\|.\|$ désigne la norme euclidienne de $\mathbb{R}^n$.

$$\|x\| = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2} = (\sum_{i=1}^{n} x_i^2)^{\frac{1}{2}} \ tq \ x = (x_1, \cdots, x_n)^T \in \mathbb{R}^n.$$

Exemple 1.3.1 ***1-*** *la fonction $f(x) = \|x\|$ est coécrive.*

2- *les fonctions affines définie par :*

$$f(x) = \langle \alpha, x \rangle + \beta,$$

telle que $\alpha \in \mathbb{R}^n$ et $\beta \in \mathbb{R}$ ne sont pas coércives.

1.3.1 Résultats d'existence et d'unicité

Commençons par donner un résultat d'existence.

Théorème 1.3.1 *Soit $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ une fonction **propre**, **continue** et **coércive** alors le problème $(\mathcal{P})$ admet au moins une solution.*

Preuve. Soit $d = \inf(\mathcal{P})$ alors $d < +\infty$ car f est propre. Soit $(x_p)_{p\in\mathbb{N}} \in \mathbb{R}^n$ une suite minimisant c'est à dire telle que :

$$\lim_{p\to+\infty} f(x_p) = d.$$

Montrons que (x_p) est bornée, si ce n'était pas le cas on pourrait extraire de cette suite une sous suite (encore noté (x_p)) telle que :

$$\lim_{p\to+\infty} \|x_p\| = +\infty.$$

Par coércivité de f on aurait :

$$\lim_{p\to+\infty} f(x_p) = +\infty.$$

Ce qui contredit le fait que

$$\lim_{p\to+\infty} f(x_p) = d < +\infty.$$

Comme (x_p) est bornée, on peut alors en extraire une sous suite (encore notée (x_p)) qui converge vers $\bar{x} \in \mathbb{R}^n$. Par continuité de f, on a alors :

$$d = \lim_{p\to+\infty} f(x_p) = f(\bar{x}).$$

En particulier $d > -\infty$ et $\bar{x}$ est une solution du problème $(\mathcal{P})$. ■

Théorème 1.3.2 *Sous les hypothèses du théorème précédente, si f est **strictement convexe**, alors le problème $(\mathcal{P})$ admet une solution unique.*

Preuve. Supposons que f admette au moins un minimum m et soient $x_1 \neq x_2$ (dans $\mathbb{R}^n$) réalisant ce minimum :

$$f(x_1) = f(x_2) = m.$$

Par stricte convexité de la fonction f on a alors :

$$f(\frac{x_1 + x_2}{2}) < \frac{1}{2}(f(x_1) + f(x_2)) = m.$$

Ceci contredit le fait que m est le minimum. Donc $x_1 = x_2$. ■

Théorème 1.3.3 *Soit $f : \mathbb{R}^n \to \mathbb{R}$ de classe C^1, on suppose qu'il existe $\alpha > 0$ tel que :*

$$\forall (x, y) \in \mathbb{R}^n \times \mathbb{R}^n \quad \langle \nabla f(x) - \nabla f(y), x - y \rangle \geq \alpha \|x - y\|^2 , \qquad (*)$$

*alors f est **strictement convexe** et **coércive.***

Par conséquent, f admet un minimum unique x^* sur $\mathbb{R}^n$ caractériser par :

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

Définition 1.3.2 (Fonction elliptique) *On dit que $f : \mathbb{R}^n \to \mathbb{R}$ est **elliptique** si la condition $(*)$ est vérifiée. α est appelé la constante d'ellipticité.*

Proposition. 1.3.1 *Une fonction f deux fois différentiable est elliptique si et seulemen si :*

$$\forall (x, y) \in \mathbb{R}^n \times \mathbb{R}^n \quad \left\langle D^2 f(x) y, y \right\rangle \geq \alpha \|y\|^2 .$$

1.3.2 Conditions d'optimalités

Les conditions nous allons donner sont des conditions différentielles qui porte sur la dérivée de la fonction à minimiser. On va donc restreindre au cas des fonctions gâteaux-différentiable.

Condition nécessaire d'optimalité du premier ordre

Théorème 1.3.4 *Soit f une fonction telle que $f : \mathbb{R}^n \to \mathbb{R}$ gâteau-différentiable sur $\mathbb{R}^n$. Si x^* réalise un minimum (global ou local) de f sur $\mathbb{R}^n$, alors :*

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

Définition 1.3.3 *Un point $x^* \in \mathbb{R}^n$ qui vérifie $(*)$ est appelé point **critique** ou **stationnaire.** La relation $(**)$ est aussi appelé **Equation d'Euler.***

Remarque 1.3.1 *Le théorème précédente n'a pas de sens si la fonction f n'est pas différentiable par exemple :*

$$f(x) = |x| , \forall x \in \mathbb{R}.$$

Donc le résultat du théorème est une condition nécessaire qui n'est pas en générale suffisante.

Condition nécéssaire et suffisante d'optimalité du premier ordre

Théorème 1.3.5 *Soit $f : \mathbb{R}^n \to \mathbb{R}$ gâteau-différentiable et convexe sur $\mathbb{R}^n$. Un point x^* réalise un minimum global de f sur $\mathbb{R}^n$ ssi :*

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

Preuve. Comme f est convexe on a :

$$\forall x, x^* \in \mathbb{R}^n, f(x) \geq f(x^*) + \langle \nabla f(x^*), x - x^* \rangle ,$$

et comme

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

alors : $\forall x, x^* \in \mathbb{R}^n, f(x) \geq f(x^*) \Rightarrow x^*$est réalise le minimum global. ■

Condition nécessaire d'optimalité du second ordre

Théorème 1.3.6 *On suppose que x^* est un minimum (local) de f et que f est deux fois dérivables sur $\mathbb{R}^n$ alors :*

1- $\nabla f(x^*) = 0$.

2- $\forall x \in \mathbb{R}^n \quad \langle D^2 f(x^*)x, x\rangle \geq 0$.

Remarque 1.3.2 *La deuxième assertion* (**2**) *est équivalente à dire que la matrice hessienne de f en x^* est semi définie positive et aussi symétrique.*

Condition nécessaire et suffisante d'optimalité du second ordre

Théorème 1.3.7 *Soit $f : \mathbb{R}^n \to \mathbb{R}$ deux fois dérivables sur $\mathbb{R}^n$ vérifiant :*

1- $\nabla f(x^*) = 0$.

2- $\exists \alpha > 0$ *telle que :* $\forall x \in \mathbb{R}^n \quad \langle D^2 f(x^*)x, x\rangle \geq \alpha \|x\|^2$.

Alors la fonction f admet un minimum local strict en x^*.

Remarque 1.3.3 *La condition* (**2**) *revient à dire que $D^2 f(x^*)$ est définie positive.*

1.4 Problèmes d'optimisation sans contraintes

Soit $f : \mathbb{R}^n \to \mathbb{R}$ est une fonction continûment différentiable et $(\mathcal{P})$ le problème de minimisation sans contraintes suivant :

$$(\mathcal{P}) : \begin{cases} \min\limits_{x \in \mathbb{R}^n} f(x) . \end{cases}$$

1.4.1 Aspect générale des algorithmes

Pour trouver (ou approcher) un point en lequel une fonction de plusieurs variables est minimum, il est classique de remplacer le problème initial $(\mathcal{P})$ par une suite $(x_k)_{k\in\mathbb{N}}$ (infinie en générale) qui converge vers la solution du problème de minimisation où la fonction économique ne dépend plus que d'une variables : c'est le principe des méthodes de descente. Autrement dit ; de construire un algorithme itératif de la forme :

$$x_{k+1} = x_k + \alpha_k d_k,$$

telle que α_k est appelé le pas. il est déterminé par une optimisation unidimensionnelle ou recherche linéaire exacte ou inexacte. d_k est la direction de descente qui faire les différents méthodes à direction de descente.

1.4.2 Notion de convergence globale

Un algorithme qui définie par une application multivoque $A : \mathbb{R}^n \to \mathbb{R}^n$ permettant la génération d'une suite d'élément de $\mathbb{R}^n$ par la formule suivante :

$$\begin{cases} x_0 & \text{donnée } k = 0 \\ x_{k+1} = A(x_k), k = k + 1 \end{cases}.$$

Donc, écrire un algorithme n'est ni plus ni moins que se donner une suite $(x_k)_{k\in\mathbb{N}}$ de $\mathbb{R}^n$. Étudier la convergence de l'algorithme c'est simplement étudier la convergence de la suite $(x_k)_{k\in\mathbb{N}}$.

• On dit qu'un algorithme est globalement convergent si, pour quelque soit le point de départ x_0 choisi, la suite $(x_k)_{k\in\mathbb{N}}$ générée par cet algorithme converge vers un point satisfont une condition nécessaire d'optimalité.

1.4.3 Vitesse de convergence

Nous nous intéressant maintenant à l'évaluation de son efficacité d'un point de vue pratique, l'efficacité d'un algorithme dépend du nombre d'itérations nécessaires pour obtenir une approximation à ε près (ε fixé à l'avance) de l'optimum $\hat{x}$. Si l'on compare entre eux plusieurs algorithmes, et si l'on admet que le temps de calcul par itération est sensiblement le même pour tous, le meilleur est celui qui nécessitera le plus petit nombre d'itérations. Malheureusement, il se révèle impossible de dégager des conclusions générales de ce genre de comparaison. Suivant le point de départ choisi, la nature de la fonction à optimiser, la valeur de la tolérance choisie, la hiérarchie des algorithmes peut varier consi-

dérablement. Si l'on veut dégager un critère ayant une certaine valeur d'absolu, il faut par conséquent recourir à un autre type d'analyse, c'est l'objet de l'étude de la *convergence asymptotique* c'est-à-dire du comportement de la suite $\{x_k\}$ au voisinage du point limite $\hat{x}$. Ceci conduit à attribuer à chaque algorithme un indice d'efficacité appelé sa **vitesse de convergence**.

Nous introduisons maintenant les différents types de convergence. Plaçons nous dans $\mathbb{R}^n$, où $\|.\|$ désigne la norme euclidienne et considérons une suite $\{x_k\}$ convergeant vers $\hat{x}$.

• Si $\limsup \frac{\|x_{k+1}-x_*\|}{\|x_k-x_*\|} = \lambda < 1$ quand $k \longrightarrow \infty$. On dit que la convergence est linéaire et λ est le *taux de convergence* associé.

• Si $\frac{\|x_{k+1}-x_*\|}{\|x_k-x_*\|} \longrightarrow 0$ quand $k \longrightarrow \infty$, on dit que la convergence est *super-linéaire*.

• Plus précisément si $\exists p > 1$ tel que $\limsup\limits_{k\longrightarrow\infty} \frac{\|x_{k+1}-x_*\|}{\|x_k-x_*\|^p} < +\infty$, on dit que la convergence est *super-linéaire d'ordre p*.

• En particulier si $\limsup\limits_{k\longrightarrow\infty} \frac{\left\|x^{k+1}-x^*\right\|}{\left\|x^k-x^*\right\|^2} < +\infty$, on dit que la convergence est *quadratique* (super-linéaire d'ordre 2).

CHAPITRE 2

Méthodes à direction de descente et recherche linéaire

2.1 Méthodes à direction de descente

Le schéma général d'une méthode à direction de descente est le suivant :

$$\begin{cases} x_0 \in \mathbb{R}^n \text{ donnée} \\ x_{k+1} = x_k + \alpha_k d_k \end{cases},$$

où $\alpha_k \in \mathbb{R}_+^*$ le pas et d_k la direction de descente sont choisis de telle sorte que :

$$f(x_k + \alpha_k d_k) \leq f(x_k).$$

Le pas et la direction peuvent être fixées ou changés à chaque itération.

2.1.1 Direction de descente

Soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction continûment différentiable et soit x un vecteur de $\mathbb{R}^n$.

Idée :

∘ On démarre d'un point x_0.

∘ On génère une suite des vecteurs $x_1, x_2, \cdots$ tel que la valeur de la fonction à minimiser décroit à chaque itération :

$$f(x_{k+1}) < f(x_k) \; k = 1, 2, \cdots .$$

Théorème 2.1.1 *Soit f une fonction à minimiser, on note x_k le point courant et $\nabla f(x_k)$ le gradient de f en x_k. $d_k \in \mathbb{R}^n$ est une direction de descente au point x_k si et seulement si :*

$$\exists \delta > 0 \quad tel\ que\ f(x_k + \alpha_k d_k) < f(x_k) : \forall \alpha_k \in]0, \delta[.$$

Définition 2.1.1 *On dit que d_k est une direction de descente de f en $x_k \in \mathbb{R}^n$ si :*

$$\nabla f(x_k)^T . d_k < 0. \tag{d}$$

- α_k fait avec $(-\nabla f(x_k))$ un ongle θ strictement plus petit que $90°$.

$$\theta = \arccos \frac{-\nabla^T f(x_k).d_k}{\left\|\nabla^T f(x_k)\right\| \|d_k\|} \in \left[0, \frac{\pi}{2}\right[.$$

2.1.2 Algorithme de la méthode à direction de descente

Algorithme de la méthode à direction de descente

▶ **Etape 1 : (Initialisation) :** $x_0 \in \mathbb{R}^n$ donnée $(k = 0)$.

▶ **Etape 2 :** calcul d'une direction de descente d_k.

▶ **Etape 3 :** on détermine une pas $\alpha_k > 0$ le long du d_k par une recherche linéaire.

▶ **Etape 4 :** nouvelle itérée : $x_{k+1} = x_k + \alpha_k d_k$. $(k = k + 1$ Itération $k)$.

▶ **Etape 5 : (Critère d'arrêt)** : si $\|x_{k+1} - x_k\| < \varepsilon$ stop, sinon, on pose $k = k+1$ et on retourne à l'étape 2 tel que $\varepsilon > 0$ assez petit donnée qui présente la précision désirée.

2.1.3 Exemples des directions de descente :

Si on pose :

$$\nabla f(x_k) = g_k.$$

1- Au méthode du **Gradient** la direction d_k est définie par :

$$d_k = -g_k. \text{ avec } g_k \neq 0.$$

En effet ; vérifions la condition **(d)** :

$$g_k^T.(-g_k) = -\|g_k\|^2 < 0.$$

2- Au méthode de **Gradient conjugué** la direction d_k est définie par :

$$d_k = \begin{cases} -g_0 & \text{si } k = 0 \\ -g_k + \beta_k d_{k-1} & \text{si } k \geq 1 \end{cases} .$$

3- Au méthode de **Newton** la direction d_k est définie par :

$$d_k = -\left[H(x_k)\right]^{-1}.g_k \text{ où } H \text{ est la matrice héssienne de } f.$$

2.2 Recherches linéaires (Optimisation unidimensionnelle)

Considérons le problème d'optimisation sans contraintes $(\mathcal{P})$:

$$(\mathcal{P}) : \begin{cases} \min\limits_{x\in\mathbb{R}^n} f(x) \ , \end{cases}$$

où $f : \mathbb{R}^n \longrightarrow \mathbb{R}$.

Les algorithmes qu'on étudie par la suite suivent les schémas généraux suivants. L'algorithme démarre d'un point $x_0 \in \mathbb{R}^n$. A l'itération k supposons qu'on ait le point x_k, le point x_{k+1} successeur de x_k sera construit de la façon suivante, on choisit une direction d_k (de descente en général) et déterminons x_{k+1} comme suit :

$$x_{k+1} = x_k + \alpha_k d_k,$$

où α_k est solution optimale du problème d'optimisation unidimensionnel suivant :

$$\begin{cases} \min\limits_{\alpha_k\geq 0} f(x_k + \alpha_k d_k) \ . \end{cases}$$

C'est à dire que α_k vérifie :

$$f\left(x_k + \alpha_k d_k\right) \leq f\left(x_k + \alpha d_k\right), \forall \alpha \geq 0,$$

où x_k et d_k sont fixes et la fonction à minimiser est une fonction d'une variable réelle définie comme suit :

$$\alpha \rightarrow h_k\left(\alpha\right) = f\left(x_k + \alpha d_k\right).$$

Il faut noter que dans les problèmes d'optimisation sans contraintes on a besoin de résoudre à chaque Itération x_k, un problème d'optimisation dans $\mathbb{R}$.

2.2.1 Objectifs à atteindre

Une recherche linéaire veut dire déterminer un pas α_k le long d'une direction de descente d_k. Il s'agit de réaliser deux objectifs :

▷ **Le premier objectif**

Consiste à faire décroître f suffisamment, Cela se traduit le plus souvent par réalisation d'une inégalité de la forme :

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \text{"}un\ terme\ négatif\text{"}. \qquad (**)$$

Le terme négatif joue un rôle-clé dans la convergence de l'algorithme utilisant cette recherche linéaire, et devra prendre une forme bien particulière si on veut pouvoir en tirer de l'information. En particulier, il ne suffit pas d'imposer :

$$f(x_k + \alpha_k d_k) < f(x_k).$$

▷ **Le second objectif**

consiste d'empêcher le pas $\alpha_k > 0$ d'être trop petit, trop proche de zéro.

Le premier objectif n'est en effet pas suffisant car l'inégalité $(**)$ est en général satisfaite par des pas $\alpha_k > 0$ arbitrairement petit.

Il existe deux grandes classes de méthodes qui s'intéressent a' l'optimisation unidimensionnelle :

- ○ les recherches linéaires exactes.
- ○ les recherches linéaires inexactes.

2.2.2 Recherches linéaires exactes

Comme on cherche à minimiser f, il semble naturel de chercher à minimiser le critère le long de d_k et donc de déterminer le pas α_k comme solution du problème :

$$\min_{\alpha \geq 0} h_k(\alpha) = \min_{\alpha \geq 0} f(x_k + \alpha d_k).$$

C'est ce que l'on appelle la règle de **Cauchy** et le pas déterminé par cette règle est appelé pas de **Cauchy** ou pas optimal. Dans certains cas, on préférera le plus petit point stationnaire de h_k qui fait décroître par cette fonction :

$$\alpha_k = \inf \{\alpha \geq 0 : h_k(\alpha) < h_k(0)\}.$$

On parle alors de règle de **Curry** et le pas déterminé par cette règle est appelé pas de **Curry**. De manière un peu imprécise, ces deux règles sont parfois qualifiées de recherche linéaire exacte.

Remarque 2.2.1 *Ils ne sont utilisés que dans des cas particuliers, par exemple lorsque*

f est quadratique, la solution de la recherche linéaire s'obtient de façon exacte et dans un nombre fini d'itérations.

Les inconvénients des recherches linéaires exactes

Pour une fonction non linéaire arbitraire,

♦ il ne peut pas exister de pas de **Cauchy** ou de **Curry**,

♦ la détermination de ces pas demande en général beaucoup de temps de calcul et ne peut de toutes façons pas être faite avec une précision infinie,

♦ L'efficacité supplémentaire éventuellement apportée à un algorithme par une recherche linéaire exacte ne permet pas, en général, de compenser le temps perdu à déterminer un tel pas.

♦ les résultats de convergence autorisent d'autres types de règles (recherche linéaire inexacte), moins gourmandes en temps de calcul.

Intervalle de sécurité

Dans la plupart des algorithmes d'optimisation modernes, on ne fait jamais de recherche linéaire exacte, car trouver α_k signifie qu'il va falloir calculer un grand nombre de fois la fonction h_k et cela peut être dissuasif du point de vue du temps de calcul. En pratique, on recherche plutôt une valeur de $\hat{\alpha}$ qui assure une décroissance suffisante de f. Cela conduit à la notion d'intervalle de sécurité.

Définition 2.2.1 *On dit que $[\alpha_g, \alpha_d]$ est un intervalle de sécurité s'il permet de classer les valeurs de α la façon suivante :*

♦ *Si $\alpha < \alpha_g$ alors α est considéré trop petit.*

♦ *Si $\alpha_g \leq \alpha \leq \alpha_d$ alors α est satisfaisant.*

♦ *Si $\alpha > \alpha_d$ alors α est considéré trop grand.*

Le problème est de traduire de façon numérique sur h_k les trois conditions précédentes, ainsi que de trouver un algorithme permettant de déterminer α_g et α_d. L'idée est de partir d'un intervalle suffisamment grand pour contenir $[\alpha_g, \alpha_d]$, et d'appliquer une bonne stratégie pour itérativement réduire cet intervalle.

Réduction de l'intervalle

On suppose maintenant que l'on dispose d'un intervalle $[\alpha_g, \alpha_d]$ mais que l'on n'a pas encore de α satisfaisant.

Une manière simple de procéder par exemple par dichotomie, en choisissant :

$$\alpha = \frac{\alpha_g + \alpha_d}{2},$$

puis en conservant soit $[\alpha_g, \alpha]$ ou $[\alpha, \alpha_d]$ suivant que α est trop grand ou trop petit. le problème est que cette stratégie ne réduit pas assez rapidement l'intervalle. Cependant elle n'utilise aucune informations sur h_k. On préfère en général procéder en construisant une approximation polynomiale $p(\alpha)$ de h_k et en choisissant α réalisant le minimum (s'il existe) de $p(\alpha)$ sur $[\alpha_g, \alpha_d]$.

Algorithme (Algorithme de base)

▶ **Etape 0 : (Initialisation) :** $\alpha_g = \alpha_d = 0$, choisir $\alpha_1 > 0$; poser $k = 1$ et aller à l'étape 1 ;

▶ **Etape 1 :**

- Si α_k convient, poser $\hat{\alpha} = \alpha_k$ et on s'arrête.
- Si α_k est trop petit on prend $\alpha_{g,k+1} = \alpha_k, \alpha_{d,k+1} = \alpha_d$, et on va à l'étape 2.
- Si α_k est trop grand on prend $\alpha_{d,k+1} = \alpha_k, \alpha_{g,k+1} = \alpha_g$, et on va à l'étape 2.

▶ **Etape 2 :**

- Si $\alpha_{d,k+1} = 0$ déterminer $\alpha_{k+1} \in]\alpha_{g,k+1}, +\infty[$.
- Si $\alpha_{d,k+1} \neq 0$ déterminer $\alpha_{k+1} \in]\alpha_{g,k+1}, \alpha_{d,k+1}[$,

Remplacer k par $k+1$ et aller à l'étape 1.

Il faut maintenant préciser quelles sont les relations sur h_k qui va nous permettre de caractériser les valeurs de α convenables, ainsi que les techniques utilisées pour réduire l'intervalle.

2.2.3 Recherches linéaires inexactes

Schéma des recherches linéaires inexactes

Elles reviennent à déterminer, par tâtonnement un intervalle $[\alpha_g, \alpha_d]$ où $\hat{\alpha} \in [\alpha_g, \alpha_d]$, dans lequel :

$$h_k(\alpha_k) < h_k(0) \qquad (f(x_k + \alpha_k d_k) < f(x_k)).$$

Le schéma de l'algorithme est donc :

Algorithme (Schéma des recherches linéaires inexactes)

▶ **Etape 0 : (Initialisation) :** $\alpha_{g,1} = \alpha_{d,1} = 0$, choisir $\alpha_1 > 0$; poser $k = 1$ et aller à l'étape 1.

▶ **Etape 1 :**

- Si α_k est satisfaisant (suivant un certain critère) : STOP ($\hat{\alpha} = \alpha_k$).
- Si α_k est trop petit (suivant un certain critère) : nouvel intervalle : $[\alpha_{g,k+1} = \alpha_k, \alpha_{d,k+1} = \alpha_d]$ et aller à l'étape 2.
- Si α_k est trop grand (suivant un certain critère) : nouvel intervalle : $[\alpha_{g,k+1} = \alpha_g, \alpha_{d,k+1} = \alpha_k]$ et aller à l'étape 2.

▶ **Etape 2 :**

- Si $\alpha_{d,k+1} = 0$ déterminer $\alpha_{k+1} \in]\alpha_{d,k+1}, +\infty[$.
- Si $\alpha_{d,k+1} \neq 0$ déterminer $\alpha_{k+1} \in]\alpha_{g,k+1}, \alpha_{d,k+1}[$.

Remplacer k par $k+1$ et aller à l'étape 1.

Il nous reste donc à décider selon quel(s) critère(s) α est trop petit ou trop grand ou satisfaisant.

La règle d'Armijo

La règle d'**Armijo** (1966) impose une contrainte sur le choix de α_k suffisante pour minimiser localement h_k. Une condition naturelle est de demander que f décroisse autant qu'une $\rho \in]0,1[$ parfois appelée condition d'**Armijo** ou condition de décroissance linéaire :

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \rho \alpha_k \nabla^T f(x_k) d_k.$$

Autrement dit :

$$h_k(\alpha) \leq h_k(0) + \rho h'_k(0) \alpha.$$

Elle est de la forme $(f(x_k + \alpha_k d_k) < f(x_k))$, car ρ devra être choisi dans $]0,1[$. La fonction prenne une valeur plus petite que celle prise par la fonction affine :

$$\alpha \longrightarrow f(x_k) + \rho \alpha_k \nabla^T f(x_k) d_k. \qquad (*)$$

Test d'Armijo

♦ Si $h_k(\alpha) \leq h_k(0) + \rho h'_k(0) \alpha$, alors α convient.

♦ Si $h_k(\alpha) > h_k(0) + \rho h'_k(0) \alpha$, alors α est trop grand.

Algorithme (Règle d'Armijo)

▶ **Etape 0 : (Initialisation) :** $\alpha_{g,1} = \alpha_{d,1} = 0$, choisir $\alpha_1 > 0$, $\rho \in]0,1[$ poser $k = 1$ et aller à l'étape 1.

▶ **Etape 1 :**

- Si $h_k(\alpha_k) \leq h_k(0) + \rho h'_k(0) \alpha_k$: STOP ($\hat{\alpha} = \alpha_k$)

- Si $h_k(\alpha_k) > h_k(0) + \rho h'_k(0)\alpha_k$, alors $\alpha_{d,k+1} = \alpha_d$, $\alpha_{g,k+1} = \alpha_k$ et aller à l'étape 2.

► **Etape 2 :**

- Si $\alpha_{d,,k+1} = 0$ déterminer $\alpha_{k+1} \in]\alpha_{d,k+1}, +\infty[$.
- Si $\alpha_{d,k+1} \neq 0$ déterminer $\alpha_{k+1} \in]\alpha_{g,k+1}, \alpha_{d,k+1}[$,

Remplacer k par $k+1$ et aller à l'étape 1.

Remarque 2.2.2 ♦ *En pratique, la constante ρ est prise très petite, de manière a satisfaire $(*)$ le plus facilement possible. Typiquement, $\rho = 10^{-4}$. Notons que cette constante ne doit pas être adaptée aux données du problème et donc que l'on ne se trouve pas devant un choix de valeur délicat.*

♦ *Dans certains algorithmes, il est important de prendre $\rho < \frac{1}{2}$ pour que le pas α_k soit accepté lorsque x_k est proche d'une solution.*

♦ *L'inégalité $(\star)$ est toujours vérifiée si $\alpha_k > 0$ est suffisamment petit.*

♦ *On a vu qu'il était dangereux d'accepter des pas trop petits, cela pouvait conduire a une fausse convergence. Il faut donc un mécanisme supplémentaire qui empêche le pas d'être trop petit. On utilise souvent la technique de rebroussement due a* ***Armijo*** *en* 1966 *ou celle de* ***Goldstein****.*

La règle de Goldstein et Price

Dans la règle d'**Armijo** on assure la décroissance de la fonction objectif à chaque pas, mais ce n'est pas suffisant. Les conditions de **Goldstein** et **Price** *(*1969) suivantes sont, comme on va le prouver, suffisantes pour assurer la convergence sous certaines conditions et indépendamment de l'algorithme qui calcule le paramètre .Dans celle-ci, en ajoutant une deuxième inégalité à la règle d'**Armijo** on obtient la règle de **Goldstein** :

$$f(x_k) + \rho\alpha_k \nabla^T f(x_k) d_k \geq f(x_k + \alpha_k d_k) \geq f(x_k) + \sigma\alpha_k \nabla^T f(x_k) d_k;$$

$$\text{Autrement dit : } h_k(0) + \rho h'_k(0)\alpha \geq h_k(\alpha) \geq h_k(0) + \sigma h'_k(0)\alpha,$$

où ρ et σ sont deux constantes vérifiant $0 < \rho < \sigma < 1$, cette inégalité qui empêche le pas d'être trop petit.

Test de Goldstein et Price

♦ Si $h_k(\alpha) < h_k(0) + \sigma h'_k(0)\alpha$, alors α est trop petit.
♦ Si $h_k(\alpha) > h_k(0) + \rho h'_k(0)\alpha$, alors α est trop grand.
♦ Si $h_k(0) + \rho h'_k(0)\alpha \geq h_k(\alpha) \geq h_k(0) + \sigma h'_k(0)\alpha$, alors α convient.

Algorithme (Règle de Goldstein et Price)

▶ **Etape 0 : (Initialisation) :** $\alpha_{g,1} = \alpha_{d,1} = 0$, choisir $\alpha_1 > 0$, $\rho \in]0, 1[$ et $\sigma \in]\rho, 1[$, poser $k = 1$ et aller à l'étape 1.

▶ **Etape 1 :**

- Si $h_k(0) + \rho h'_k(0)\alpha \geq h_k(\alpha_k) \geq h_k(0) + \sigma h'_k(0)\alpha_k$: STOP ($\hat{\alpha} = \alpha_k$).
- Si $h_k(\alpha_k) > h_k(0) + \rho h'_k(0)\alpha_k$, alors $\alpha_{d,k+1} = \alpha_k, \alpha_{g,k+1} = \alpha_{g,k}$, et aller à l'étape 2.
- Si $h_k(\alpha_k) < h_k(0) + \sigma h'_k(0)\alpha_k$, alors $\alpha_{d,k+1} = \alpha_{d,k}, \alpha_{g,k+1} = \alpha_k$; et aller à l'étape 2.

▶ **Etape 2 :**

- Si $\alpha_{d,k+1} = 0$ déterminer $\alpha_{k+1} \in]\alpha_{d,k+1}, +\infty[$.
- Si $\alpha_{d,k+1} \neq 0$ déterminer $\alpha_{k+1} \in]\alpha_{g,k+1}, \alpha_{d,k+1}[$.

Théorème 2.2.1 *Soit $h_k : \mathbb{R}_+ \longrightarrow \mathbb{R}$ définie par $h_k(\alpha) = f(x_k + \alpha d_k)$ est continue et bornée inférieurement. Si d_k est une direction de descente en x_k et si $\rho \in]0, 1[$ et $\sigma \in]\rho, 1[$, alors l'ensemble des pas vérifiant la règle de **Goldstein** et **Price** est non vide.*

La règle de Wolfe

Les conditions de la règle de **Goldstein** et **Price** peuvent exclure un minimum ce qui est peut être un inconvénient (par exemple sur la fonction $f(x) = x^2$). La règle de **Wolfe** (1969) fait appel au calcul de $h'_k(\alpha_k)$, elle est donc en théorie plus coûteuse que la règle de **Goldstein**. Cependant dans de nombreuses applications, le calcul du gradient $\nabla f(x_k)$ représente un faible coût additionnel en comparaison du coût d'évaluation de $f(x_k)$, c'est pourquoi cette règle est très utilisée.

La règle de **Wolfe** a pour but de déterminer un pas α_k vérifiant les deux conditions suivantes :

1- La fonction f doit décroitre de manière significative :

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \rho\alpha_k \nabla^T f(x_k) d_k. \tag{W.1}$$

2- Le pas α_k doit être suffisamment grand :

$$\nabla^T f(x_k + \alpha_k d_k) d_k \geq \sigma \nabla^T f(x_k) d_k. \tag{W.2}$$

Autrement dit :

$$\begin{aligned} h_k(\alpha) &\leq h_k(0) + \rho h_k'(0)\,\alpha, \\ h_k'(\alpha) &\geq \sigma h_k'(0), \end{aligned}$$

avec $0 < \rho < \sigma < 1$.(typiquement $\rho = 0,1$ et $\sigma = 0,9$).

Test de Wolfe

- ♦ Si $h_k(\alpha) \leq h_k(0) + \rho h_k'(0)\,\alpha$ et $h_k'(\alpha) \geq \sigma h_k'(0)$ alors α convient.
- ♦ Si $h_k(\alpha) > h_k(0) + \rho h_k'(0)\,\alpha$, alors α est trop grand.
- ♦ Si $h_k'(\alpha) < \sigma h_k'(0)$, alors α est trop petit.

Algorithme (Règle de Wolfe)

► **Etape 0 : (Initialisation) :** $\alpha_{g,1} = \alpha_{d,1} = 0$, choisir $\alpha_1 > 0, \rho \in]0,1[$ et $\sigma \in]\rho, 1[$, poser $k = 1$ et aller à l'étape 1.

► **Etape 1 :**

- Si $h_k(\alpha_k) \leq h_k(0) + \rho h_k'(0)\,\alpha_k$ et $h_k'(\alpha_k) \geq \sigma h_k'(0)$: STOP ($\hat{\alpha} = \alpha_k$).
- Si $h_k(\alpha_k) > h_k(0) + \rho h_k'(0)\,\alpha_k$, alors $\alpha_{d,k+1} = \alpha_k, \alpha_{g,k+1} = \alpha_{g,k}$, et aller à l'étape 2.
- Si $h_k'(\alpha) < \sigma h_k'(0)$, alors $\alpha_{d,k+1} = \alpha_{d,k}, \alpha_{g,k+1} = \alpha_k$; et aller à l'étape 2.

► **Etape 2 :**

- Si $\alpha_{d,k+1} = 0$ déterminer $\alpha_{k+1} \in]\alpha_{d,k+1}, +\infty[$.
- Si $\alpha_{d,k+1} \neq 0$ déterminer $\alpha_{k+1} \in]\alpha_{g,k+1}, \alpha_{d,k+1}[$.

La règle de Wolfe forte

Pour certains algorithmes (par exemple le gradient conjugué non linéaire), il est parfois nécessaire d'avoir une condition plus restrictive que (W.2). Pour cela on obtient des contraintes plus fortes si on le remplace par :

$$\left|\nabla^T f(x_k + \alpha d_k) d_k\right| \leq -\sigma \nabla^T f(x)\, d_k. \tag{W.3}$$

Les (W.1) et (W.3) sont les conditions de **Wolfe** fortes (1971). Autrement dit :

$$h_k(\alpha) \leq h_k(0) + \rho h_k'(0)\,\alpha, \tag{Wf.1}$$

$$\left|h_k'(\alpha)\right| \leq -\sigma \nabla^T f(x)\, d_k, \tag{Wf.2}$$

avec $0 < \rho < \sigma < 1$.

Remarque 2.2.3 *On voit bien que les conditions de* ***Wolfe*** *forte impliquent les conditions de* ***Wolfe*** *faible. En effet,*

$$\begin{aligned} \left|\nabla^T f(x_k+\alpha d_k)d_k\right| &\leq -\sigma\nabla^T f(x)\, d_k \Rightarrow \\ \sigma\nabla^T f(x)\, d_k &\leq \nabla^T f(x_k+\alpha d_k)d_k \leq -\sigma\nabla^T f(x)\, d_k \Rightarrow \\ \sigma\nabla^T f(x)\, d_k &\leq \nabla^T f(x_k+\alpha d_k)d_k. \end{aligned}$$

2.3 Convergence des méthodes à direction de descente

2.3.1 Condition de Zoutendijk

On va étudier la contribution de la recherche linéaire inexacte à la convergence des algorithmes à directions de descente. Ce n'est qu'une contribution, parce que la recherche linéaire ne peut à elle seule assurer la convergence des itérés. On comprend bien que le choix de la direction de descente joue aussi un rôle. Cela se traduit par une condition, dite de **Zoutendijk**, dont on peut tirer quelques informations qualitatives intéressantes. On dit qu'une règle de recherche linéaire satisfait la condition de **Zoutendijk** s'il existe une constante $C > 0$ telle que pour tout indice $k \geq 1$ on ait :

$$f(x_{k+1}) \leq f(x_k) - C\left\|\nabla f(x_k)\right\|^2 \cos^2\theta_k, \qquad (Z.1)$$

où θ_k est l'angle que fait d_k avec $-\nabla f(x_k)$, défini par :

$$\cos\theta_k = \frac{-\nabla^T f(x_k)d_k}{\left\|\nabla^T f(x_k)\right\|\left\|d_k\right\|}.$$

Proposition. 2.3.1 *Si la suite $\{x_k\}$ générée par un algorithme d'optimisation vérifie la condition de* ***Zoutendijk*** *(Z.1) et si la suite $\{f(x_k)\}$ est minorée, alors :*

$$\sum_{k\geq 1}\left\|\nabla f(x_k)\right\|^2\cos^2\theta_k < \infty. \qquad (Z.2)$$

Preuve. En sommant les inégalités (Z.1), on a :

$$\sum_{i=1}^{k}\left\|\nabla f(x_i)\right\|^2\cos^2\theta_i \leq \frac{1}{C}\left(f(x_1) - f(x_{k+1})\right).$$

Comme $\{f(x_k)\}$ est minorée, donc il existe un C' telle que :

$$-f(x_{k+1}) \leq -C'.$$

Alors :

$$\left(\sum_{i=1}^{k} \|\nabla f(x_i)\|^2 \cos^2\theta_i \leq \frac{1}{C}\left(f(x_1) - C'\right) < \infty\right) \Rightarrow \left(\sum_{k\geq 1} \|\nabla f(x_k)\|^2 \cos^2\theta_k < \infty\right).$$

■

Les deux propositions suivantes précisent les circonstances dans les quelles la condition de **Zoutendijk** (*Z.1*) est vérifie avec les règles d'**Armijo** et de **Wolfe**.

Proposition. 2.3.2 *Soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction contiûment différentiable dans un voisinage de Γ donnée par :*

$$\Gamma = \{x \in \mathbb{R}^n : f(x) \leq f(x_1)\}.$$

*On considère un algorithme à directions de descente d_k, qui génère une suite $\{x_k\}$ en utilisant la recherche linéaire d'**Armijo** avec $\alpha_1 > 0$. Alors il existe une constante $C > 0$ telle que, pour tout $k \geq 1$, l'une des conditions :*

$$f(x_{k+1}) \leq f(x_k) - C\nabla^T f(x_k)d_k, \qquad (Z'.1)$$

où :

$$f(x_{k+1}) \leq f(x_k) - C\|\nabla f(x_k)\|^2 \cos^2\theta_k, \qquad (Z'.2)$$

est vérifiée.

Proposition. 2.3.3 *Soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction continument différentiable dans un voisinage de Γ. On considère un algorithme à directions de descente d_k, qui génère une suite $\{x_k\}$ en utilisant la recherche linéaire de **Wolfe**. Alors il existe une constante $C > 0$ telle que, pour tout $k \geq 1$, la condition de **Zoutendijk** (Z.1) est vérifiée.*

CHAPITRE 3

Gradient conjugué à trois termes pour la résolution des problèmes d'optimisation sans contraintes

Ce chapitre introduit une classe importante d'algorithmes de résolution des problèmes d'optimisation sans contraintes.

Le concept central est celui de direction de descente. On le retrouvera dans des contextes variés, également pour résoudre des problèmes avec contraintes. Tous les algorithmes d'optimisation n'entrent pas dans ce cadre. Une autre classe importante de méthodes se fonde sur la notion de région de confiance.

On commencera notre chapitre par une synthèse sur les différentes méthodes d'optimisation sans contraintes, et on termine par ce chapitre à un algorithme de gradient conjugué à trois terme pour résoudre les problèmes d'optimisation sans contrainte de grand taille.

3.1 Méthodes numériques utilisées pour résoudre les problèmes d'optimisation sans contraintes

Les méthodes numériques utilisées pour résoudre les problèmes d'optimisation sans contraintes sont itératives. On part d'un point initial x_0 et à la k-*ième* itération, si on a x_k, le successeur x_{k+1} est obtenu en utilisant les informations utilisées à l'itération k. Il est souhaitable que la suite x_k , ainsi construite converge vers la solution optimale du problème. On classe les méthodes d'optimisation sans contraintes en deux classes

essentielles : les méthodes à recherches linéaires et les méthodes à régions de confiance.

3.1.1 Méthode du gradient (Méthode de la plus forte pente)

La méthode (ou algorithme) du **Gradient** fait partie d'une classe plus grande des méthodes numériques appelées **méthodes de descente**. Expliquons rapidement l'idée directrice de ces méthodes.

On veut minimiser une fonction J. Pour cela on se donne un point de départ arbitraire x_o. Pour construire l'itéré suivant x_1 il faut penser qu'on veut se rapprocher du minimum de J ; on veut donc que $J(x_1) < J(x_o)$. On cherche alors x_1 sous la forme $x_1 = x_o + \rho_1 d_1$ où d_1 est un vecteur non nul de $\mathbb{R}^n$ et $\rho 1$ un réel strictement positif. En pratique donc, on cherche d_1 et ρ_1 pour que $J(x_o+\rho_1 d_1) < J(x_o)$. On ne peut pas toujours trouver d_1. Quand d_1 existe on dit que c'est une **direction de descente** et $\rho 1$ est le **pas de descente**. La direction et le pas de descente peuvent être fixes ou changer à chaque itération. Le schéma général d'une méthode de descente est le suivant :

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

où ρ_k et d_k sont choisis de telle sorte que $J(x_k + \rho_k d_k) \leq J(x_k)$.

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

$$J(x_k + \rho_k d_k) = J(x_k) + \rho_k(\nabla J(x_k), d_k) + o(\rho_k d_k).$$

Comme on veut $J(x_k + \rho_k d_k) < J(x_k)$, on peut choisir en première approximation $d_k = -\nabla J(x_k)$. La méthode ainsi obtenue s'appelle l'algorithme du **Gradient**. Le pas ρ_k est choisi constant ou variable.

Algorithme du Gradient

1. Initialisation

$k = 0$: choix de x_0 et de $\rho_0 > 0$.

2. Itération k

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

3. Critère d'arrêt

- Si $\|x_{k+1} - x_k\| < \varepsilon$, STOP.
- Sinon, on pose $k = k + 1$ et on retourne à 2.

La méthode du gradient est l'une des plus simples et plus célèbres méthodes d'optimisation sans contraintes. Pour beaucoup de problèmes, la méthode du gradient travaille de façon performante dans les premières itérations, mais lorsqu'on s'approche d'un point stationnaire la méthode devient très lente. Ce phénomène est appelé phénomène du *zig-zaguing*.

3.1.2 Méthode du gradient conjugué

Les méthodes du gradient conjugué sont utilisées pour résoudre les problèmes d'optimisation non linéaires sans contraintes spécialement les problèmes de grandes tailles. On l'utilise aussi pour résoudre les grands systèmes linéaires.

Elles reposent sur le concept des directions conjuguées parce que les gradients successifs sont orthogonaux entre eux et aux directions précédentes.

L'idée initiale était de trouver une suite de directions de descente permettant de résoudre le problème :

$$(\mathcal{P}) : \begin{cases} \min\limits_{x \in \mathbb{R}^n} f(x) \ , \end{cases}$$

où f est régulière (continûment différentiable).

Le principe général d'une méthode à directions conjuguées

Donnons la définition de "*conjugaison*" :

Définition 3.1.1 *Soit A une matrice symétrique $n \times n$, définie positive. On dit que deux vecteurs x et y de $\mathbb{R}^n$ sont $A-$conjugués (ou conjugués par rapport à A) s'ils vérifient :*

$$x^T A y = 0.$$

Description de la méthode

Soit $\{d_0, d_1, ..., d_n\}$ une famille de vecteurs A-conjugués. On appelle alors méthode de directions conjuguées toute méthode itérative appliquée à une fonction quadratique strictement convexe de n variables :

$$q(x) = \frac{1}{2} x^T A x + b^T x + c.$$

Avec $x \in \mathbb{R}^n$ et $A \in \mathcal{M}_{n\times n}$ est symétrique et définie positive, $b \in \mathbb{R}^n$ et $c \in \mathbb{R}$, conduisant à l'*optimum* en n étapes au plus. Cette méthode est de la forme suivante :

$$\begin{cases} x_0 \text{ donné.} \\ x_{k+1} = x_k + \alpha_k d_k, \end{cases},$$

où α_k *est optimal* et $d_1, d_2, .., d_n$ possédant la propriété d'être *mutuellement conjuguées* par rapport à la fonction quadratique.

Si l'on note $g_k = \nabla q\,(x_k)$, la méthode se construit comme suit :

Calcul de α_k

Comme α_k minimise q dans la direction d_k, on a :

$$\forall k \ ; q'\,(\alpha_k) = d_k^T \nabla q(x_{k+1}) = 0,$$

$$d_k^T \nabla q(x_{k+1}) = d_k^T (Ax_{k+1} + b) = 0.$$

Soit

$$d_k^T A(x_k + \alpha_k d_k) + d_k^T b = 0,$$

d'où l'on tire

$$\alpha_k = \frac{-d_k^T\,(Ax_k + b)}{d_k^T A d_k}.$$

Comment construire les directions A-conjuguées ?

Des directions A-conjuguées $d_0, ..., d_k$ peuvent être générées à partir d'un ensemble de vecteurs linéairement indépendants $\xi_0, ..., \xi_k$ en utilisant la procédure dite de **Gram-Schmidt**, de telle sorte que pour tout i entre 0 et k, le sous-espace généré par $d_0, ..., d_i$ soit égale au sous-espace généré par $\xi_0, ..., \xi_i$.

Alors d_{i+1} est construite comme suit :

$$d_{i+1} = \xi_{i+1} + \sum_{m=0}^{i} \varphi_{(i+1)m} d_m.$$

Nous pouvons noter que si d_{i+1} est construite d'une telle manière, elle est effectivement linéairement indépendante avec $d_0, ..., d_i$.

En effet, le sous-espace généré par les directions $d_0, ..., d_i$ est le même que le sous-espace généré par les directions $\xi_0, ..., \xi_i$, et ξ_{i+1} est linéairement indépendant de $\xi_0, ..., \xi_i$.

ξ_{i+1} ne fait donc pas partie du sous-espace généré par les combinaisons linéaires de la forme $\sum_{m=0}^{i} \varphi_{(i+1)m} d_m$, de sorte que d_{i+1} n'en fait pas partie non plus et est donc linéairement

indépendante des $d_0, ..., d_i$.

Les coefficients $\varphi_{(i+1)m}$, eux sont choisis de manière à assurer la A-conjugaison des $d_0, ..., d_{i+1}$.

Méthode de gradient conjugué dans le cas quadratique

La méthode du gradient conjugué quadratique est obtenue en appliquant la procédure de **Gram-Schmidt** aux gradients $\nabla q(x_0), ..., \nabla q(x_{n-1})$, c'est-à-dire en posant $\xi_0 = -\nabla q(x_0), ..., \xi_{n-1} = -\nabla q(x_{n-1})$.

En outre, nous avons :

$$\nabla q(x) = Ax + b, \text{ et } \nabla^2 q(x) = A.$$

Notons que la méthode se termine si $\nabla q(x_k) = 0$.

La particularité intéressante de la méthode du gradient conjugué est que le membre de droite de l'équation donnant la valeur de d_{k+1} dans la procédure de **Gram-Schmidt** peut être grandement simplifié.

Notons que la méthode du gradient conjugué est inspirée de celle du gradient (plus profonde pente).

Algorithme de la méthode du gradient conjugué pour les fonctions quadratiques

On suppose ici que la fonction à minimiser est quadratique sous la forme :

$$q(x) = \frac{1}{2}x^T Ax + b^T x + c.$$

Si l'on note $g_k = \nabla f(x_k)$, l'algorithme prend la forme suivante.

Cet algorithme consiste à générer une suite d'itérés $\{x_k\}$ sous la forme :

$$x_{k+1} = x_k + \alpha_k d_k.$$

L'idée de la méthode est :

1- Construire itérativement des directions $d_0, ..., d_k$ mutuellement conjuguées :

A chaque étape k la direction d_k est obtenue comme combinaison linéaire du gradient en x_k et de la direction précédente d_{k-1} c'est-à-dire :

$$d_{k+1} = -\nabla q(x_{k+1}) + \beta_{k+1} d_k,$$

les coefficients β_{k+1}étant choisis de telle manière que d_k soit conjuguée avec toutes les directions précédentes. autrement dit :

$$d_{k+1}^T A d_k = 0,$$

On en déduit :

$$\begin{aligned} d_{k+1}^T A d_k &= 0 \Rightarrow \left(-\nabla q(x_{k+1}) + \beta_{k+1} d_k\right)^T A d_k = 0 \\ &\Rightarrow -\nabla^T q(x_{k+1}) A d_k + \beta_{k+1} d_k^T A d_k = 0 \\ &\Rightarrow \beta_{k+1} = \frac{\nabla^T q(x_{k+1}) A d_k}{d_k^T A d_k} = \frac{g_{k+1}^T A d_k}{d_k^T A d_k}. \end{aligned}$$

2- Déterminer le pas α_k :

En particulier, une façon de choisir α_k consiste à résoudre le problème d'optimisation unidimensionnelle suivant :

$$\alpha_k = \min_{\alpha>0} f(x_k + \alpha d_k),$$

on en déduit :

$$\alpha_k = -\frac{1}{A d_k} g_k \frac{d_k^T}{d_k^T} = \frac{-d_k^T g_k}{d_k^T A d_k}.$$

Le pas α_k obtenu ainsi s'appelle le pas optimal.

Algorithme du gradient conjugué "quadratique"

Etape 0 : (initialisation)

Soit x_0 le point de départ, $g_0 = \nabla q(x_0) = Ax_0 + b$, poser $d_0 = -g_0$,

poser $k = 0$ et aller à l'étape 1.

Etape 1 :

si $g_k = 0$: STOP ($x^* = x_k$)."Test d'arrêt".

si non aller à l'étape 2.

Etape 2 :

Prendre $x_{k+1} = x_k + \alpha_k d_k$ avec :

$$\begin{aligned} \alpha_k &= \frac{-d_k^T g_k}{d_k^T A d_k}, \\ d_{k+1} &= -g_{k+1} + \beta_{k+1} d_k, \\ \beta_{k+1} &= \frac{g_{k+1}^T A d_k}{d_k^T A d_k}. \end{aligned}$$

Poser $k = k + 1$ et aller à l'étape 1.

Différentes formules de β_{k+1} dans le cas quadratique

Les différentes valeurs attribuées à β_k définissent les différentes formes du gradient conjugué.

Si on note $y_{k-1} = g_k - g_{k-1}, s_k = x_{k+1} - x_k$ on obtient les variantes suivantes :

1- Gradient conjugué variante **Hestenes - Stiefel (HS)** :

$$\beta_k^{HS} = \frac{g_{k+1}^T y_k}{d_k^T y_k}.$$

2- Gradient conjugué variante **Fletcher Reeves (FR)** :

$$\beta_k^{FR} = \frac{\|g_k\|^2}{\|g_{k-1}\|^2}.$$

3- Gradient conjugué variante **Daniel (D)** :

$$\beta_k^{D} = \frac{g_{k+1}^T \nabla^2 f(x_k)\, d_k}{d_k^T \nabla^2 f(x_k)\, d_k}.$$

4- Gradient conjugué variante **Polak-Ribière-Polyak (PRP)** :

$$\beta_k^{PRP} = \frac{g_k^T y_k}{\|g_{k-1}\|^2}.$$

5- Gradient conjugué variante descente – **Fletcher (CD)** :

$$\beta_k^{CD} = -\frac{\|g_k\|^2}{d_{k-1}^T g_{k-1}}.$$

6- Gradient conjugué variante **Liu - Storey (LS)** :

$$\beta_k^{LS} = -\frac{g_{k+1}^T y_k}{d_k^T g_k}.$$

7- Gradient conjugué variante de **Dai-Yuan (DY)** :

$$\beta_k^{DY} = \frac{\|g_k\|^2}{d_{k-1}^T y_{k-1}}.$$

8- Gradient conjugué variante de **Dai–Liao (DL)** :

$$\beta_k^{DL} = \frac{g_{k+1}^T (y_k - t s_k)}{y_k^T s_k}.$$

9- Gradient conjugué variante **Hager-Zhang (HZ)** :

$$\beta_k^{HZ} = \left(y_k - 2d_k \frac{\|y_k\|^2}{d_k^T y_k} \right)^T \frac{g_{k+1}}{d_k^T y_k}.$$

10- Gradient conjugué variante de **Z. Wei** :

$$\beta_k^{*} = \frac{g_k^T \left(g_k - \frac{\|g_k\|}{\|g_{k-1}\|} g_{k-1} \right)}{\|g_{k-1}\|^2}.$$

11- Gradient conjugué variante de **Hao Fan, Zhibin Zhu et Anwa Zhou** :

$$\beta_k^{MN} = \frac{\|g_k\|^2 - \frac{\|g_k\|}{\|g_{k-1}\|} g_k^T g_{k-1}}{\mu \left| g_k^T d_{k-1} \right| + \|g_{k-1}\|^2}.$$

12- Gradient conjugué variante **Rivaie-Mustafa-Ismail-Leong (RMIL)** :

$$\beta_k^{RMIL} = \frac{g_k^T (g_k - g_{k-1})}{\|d_{k-1}\|^2}.$$

Remarque 3.1.1 *Dans le cas quadratique on a vu que :*

$$\beta_{k+1}^{HS} = \beta_{k+1}^{PRP} = \beta_{k+1}^{FR} = \beta_{k+1}^{CD} = \beta_{k+1}^{DY}.$$

Dans le cas non quadratique, ces quantités ont en général des valeurs différentes.

Méthode du gradient conjugué dans le cas non quadratique

On s'intéresse dans cette section à la minimisation d'une fonction $f : \mathbb{R}^n \to \mathbb{R}$, non nécessairement quadratique :

$$\min_{x \in \mathbb{R}^n} f(x).$$

Les méthodes du gradient conjugué génèrent des suites $\{x_k\}_{k=0,1,2...}$de la forme suivante :

$$x_{k+1} = x_k + \alpha_k d_k.$$

Le pas $\alpha_k \in \mathbb{R}$ étant déterminé par une recherche linéaire. la direction d_k est définie par la formule de récurrence suivante ($\beta_k \in \mathbb{R}$) :

$$d_k = \begin{cases} -g_k & \text{si } k = 1, \\ -g_k + \beta_k d_{k-1} & \text{si } k \geq 2. \end{cases}$$

Ces méthodes sont des extensions de la méthode du gradient conjugué linéaire du cas quadratique, si β_k prend l'une des valeurs :

$$\begin{aligned}\beta_k^{PRP} &= \frac{g_k^T y_k}{\|g_{k-1}\|^2},\\ \beta_k^{FR} &= \frac{\|g_k\|^2}{\|g_{k-1}\|^2},\\ \beta_k^{CD} &= \frac{\|g_k\|^2}{-d_{k-1}^T g_{k-1}}.\end{aligned}$$

où $y_{k-1} = g_k - g_{k-1}$.

Algorithme de la méthode du gradient conjugué pour les fonctions quelconques

Etape 0 : (initialisation)

Soit x_0 le point de départ, $g_0 = \nabla f(x_0)$, poser $d_0 = -g_0$.

Poser $k = 0$ et aller à l'étape 1.

Etape 1 :

Si $g_k = 0$: STOP ($x^* = x_k$)."Test d'arrêt".

Si non aller à l'étape 2.

Etape 2 :

Définir $x_{k+1} = x_k + \alpha_k d_k$ avec :

α_k : calculer par la recherche linéaire.

$$d_{k+1} = -g_{k+1} + \beta_{k+1} d_k,$$

où :

β_{k+1} : défini selon la méthode.

Poser $k = k + 1$ et aller à l'étape 1.

Remarque 3.1.2 *Dans le cas quadratique avec recherche linéaire exacte, on a vu que :*

$$\beta_k^{PRP} = \beta_k^{FR} = \beta_k^{CD} = \beta_k^{DY}.$$

3.1.3 Méthode de Newton

La méthode de **Newton** n'est pas une méthode d'optimisation à proprement parler. C'est en réalité une méthode utilisée pour résoudre des équations non linéaires de la forme

$F(x) = 0$ où F est une fonction de $\mathbb{R}^n$ dans $\mathbb{R}^n$. Nous allons d'abord la décrire puis montrer comment on peut l'appliquer à la recherche de minimum.

Description de la méthode

Considérons le problème d'optimisation sans contraintes $(\mathcal{P})$:

$$(\mathcal{P}) : \begin{cases} \min\limits_{x\in\mathbb{R}^n} f(x) \ , \end{cases}$$

Où $f : \mathbb{R}^n \longrightarrow \mathbb{R}$.

Le principe de la méthode de Newton consiste à minimiser successivement les approximations du second ordre de f, plus précisément si

$$f(x) = f(x) + \nabla f(x_k)^t (x - x_k) + (x - x_k)^t H(x_k)(x - x_k) + o\|x - x_k\|^2 .$$

Posons :

$$q(x) = f(x) + \nabla f(x_k)^t (x - x_k) + (x - x_k)^t H(x_k)(x - x_k) .$$

Soit x_{k+1} l'optimum de q, alors il vérifie $\nabla q(x_{k+1}) = 0$, soit en remplaçant :

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

ou encore :

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

donc :

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

Algorithme de Newton

Etape initiale : Soit $\varepsilon > 0$, critère d'arrêt. Choisir x_1 point initial, poser $k = 1$ et aller a l'étape principale.

Etape principale : Si $\|\nabla f(x_k)\| \leq \varepsilon$ stop, sinon poser :

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

Remplacer k par $k + 1$ est aller a l'étape principale.

3.1.4 Méthode de quasi Newton ou quasi-Newtoniennes

Une méthode de **quasi Newton** est une méthode de type :

$$\begin{cases} x_{k+1} = x_k + \lambda_k d_k, \\ d_k = -B_k g_k, \end{cases} , \tag{3.1}$$

ou bien :

$$\begin{cases} x_{k+1} = x_k + \lambda_k d_k, \\ d_k = -S_k^{-1} g_k, \end{cases} , \tag{3.2}$$

où B_k (respectivement S_k) est une matrice destinée à approcher l'inverse du Héssien de f (respectivement le Héssien) de f en x_k. Le problème posé est : quelle stratégie à adopter pour faire cette approximation ?

On peut par exemple poser $B_1 = I$, mais comment ensuite mettre à jour l'approximation B_k au cours des itérations ?

L'idée est la suivante :

Prenons $f \in C^2(\mathbb{R}^n)$, et faisons un développement de $\nabla f(x)$ au voisinage de x_k :

$$\begin{aligned} \nabla f(x) &= \nabla f(x_k) + H(x_k)(x - x_k) + 0(\|x - x_k\|) \\ &\simeq \nabla f(x_k) + H(x_k)(x - x_k), \end{aligned}$$

ce qui implique :

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

Les approximations sons exacts si f est quadratique. En particulier avec $x = x_{k+1}$ et si B_k était une bonne approximation de $[H(x_k)]^{-1}$ alors :

$$B_k [g(x_{k+1}) - g(x_k)] \simeq x_{k+1} - x_k.$$

On peut imposer que B_{k+1} satisfait cette équation exactement d'où

$$B_{k+1} [g(x_{k+1}) - g(x_k)] = x_{k+1} - x_k. \tag{3.3}$$

Formules de mise à jour de l'approximation du Héssien

Le principe de la mise à jour consiste à une itération donnée de l'algorithme :

$$\begin{cases} x_{k+1} = x_k + \lambda_k d_k \\ d_k = -B_k g_k \end{cases} ,$$

à appliquer une formule de type :

$$B_{k+1} = B_k + \Delta_k, \tag{3.4}$$

avec Δ_k symétrique, assurant la relation de **quasi Newton**. ainsi que B_{k+1} définie positive, sous l'hypothèse que B_k est définie positive.

La formule (3.4) permet d'utiliser les nouvelles informations obtenues lors de l'étape k de l'algorithme, c'est à dire essentiellement le gradient $g_{k+1} = \nabla f(x_{k+1})$ au point x_{k+1}, obtenu par une recherche linéaire (exacte ou approchée) dans la direction d_k. Il existe différentes formules de type (3.1). Suivant que Δ_k est de rang un ou deux, on parlera de correction de rang un ou de rang deux.

Méthode de correction de rang un

Etend donné que $[H(x_k)]^{-1}$est symétrique, la formule de mise à jour de l'approximation du Hessien B_k est la suivante :

$$B_{k+1} = B_k + a_k u_k u_k^T, \quad u_k \in \mathbb{R}^n,$$

donc la condition de **quasi Newton** s'écrit comme suit :

$$s_k = \left(B_k + a_k u_k u_k^T\right) y_k,$$

ou encore :

$$s_k - B_k y_k = a_k u_k u_k^T y_k.$$

D'où l'on déduit que u_k est proportionnel à $s_k - B_k y_k$, avec un facteur qui peut être pris en compte dans a_k. Un choix évident pour vérifier cette dernière équation est de prendre $u_k = s_k - B_k y_k$ et a_k tel que $a_k\left(u_k^T y_k\right) = 1$, on obtient :

$$B_{k+1} = B_k + \frac{(s_k - B_k y_k)(s_k - B_k y_k)^T}{(s_k - B_k y_k)^T y_k}. \tag{3.5}$$

Méthode de Davidon Fletcher Powell (DFP)

Cette méthode a été proposée par **Davidon** en 1959 et développé plus tard en 1963 par **Fletcher**. La formule de mise à jour de **DFP** est une formule de correction de rang deux. De façon plus précise construisons B_{k+1} en fonction de B_k de la forme :

$$B_{k+1} = B_k + A_k + \Delta_k, \tag{3.6}$$

avec Δ_k et A_k deux matrices de rang un tel que :

$$A_k = a_k u_k u_k^T, \ \Delta_k = b_k v_k v_k^T,$$

a_k, b_k sont des constantes, u_k, v_k sont deux vecteurs de $\mathbb{R}^n$.

B_{k+1} doit satisfaire la condition **quasi Newton** c'est à dire :

$$x_{k+1} - x_k = B_{k+1}\left[g_{k+1} - g_k\right].$$

Si on pose par suite :

$$s_k = x_{k+1} - x_k,\ y_k = g_{k+1} - g_k,$$

Donc :

$$\begin{aligned} s_k &= B_{k+1} y_k \\ &= \left(B_k + a_k u_k u_k^T + b_k v_k v_k^T\right) y_k, \end{aligned} \tag{3.7}$$

par suite :

$$a_k u_k u_k^T y_k + b_k v_k v_k^T y_k = s_k - B_k y_k.$$

Un choix évident pour satisfaire cette équation est de prendre :

$$u_k = s_k,\ v_k = B_k y_k,\ a_k u_k^T y_k = 1,\ b_k v_k^T y_k = -1,$$

d'où :

$$B_{k+1} = B_k + \frac{s_k s_k^T}{s_k^T y_k} - \frac{B_k y_k^T y_k B_k}{y_k^T B_k y_k}. \tag{3.8}$$

Algorithme de Davidon Fletcher Powell (DFP)

Etape initiale :

Soit $\varepsilon > 0$, déterminer le critère d'arrêt. Choisir un point initial x_1 et une matrice symétrique définie positive B_1 quelconque (par exemple $B_1 = I$) poser $k = 1$, et aller aux étapes principales.

Etapes principales :

Etape 1 : Si $\|\nabla f(x_k)\| < \varepsilon$ stop ; sinon, poser $d_k = -B_k g_k$ et déterminer le pas optimal λ_k solution optimale du problème $\min f(x_k + \lambda d_k)$, $\lambda \geq 0$. et poser $x_{k+1} = x_k + \lambda_k d_k$.

Etape 2 : Construire B_{k+1} comme suit :

$$B_{k+1} = B_k + \frac{s_k s_k^T}{s_k^T y_k} - \frac{B_k y_k y_k^T B_k}{y_k^T B_k y_k},$$

avec :

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

Remplacer k par $k+1$ et aller a l'étape 1.

Méthode de Broyden, Fletcher, Goldfarb et Shanno(BFGS)

La formule de mise à jour de **Broyden**, **Fletcher**, **Goldfarb** et **Shanno** est une formule de correction de rang deux, qui s'obtient à partir de la formule **DFP** en intervertissant les rôles de s_k et y_k. La formule obtenue permet de mettre à jour une approximation B_k de Hessien lui même et non de son inverse comme dans le cas de la méthode **DFP**. On exigera que posée dans les mêmes propriétés, à savoir B_{k+1} reste définie positive si B_k l'est et bien sur l'équation d'approximation de quasi **Newton** doit être vérifiée, c'est à dire :

$$B_{k+1}s_k = y_k.$$

On obtient donc :

$$B_{k+1} = B_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k}. \tag{3.9}$$

Algorithme de Broyden, Fletcher, Goldfarb et Shanno(BFGS)

Etape initiale : Soit $\varepsilon > 0$, déterminer le critère d'arrêt. Choisir x_1 point initial et B_1 définie positive quelconque (par exemple $B_1 = I$). Poser $k = 1$ et aller aux étapes principales.

Etapes principales :

Etape 1 : Si $\|\nabla f(x_k)\| < \varepsilon$ stop ; sinon, poser $d_k = -B_k g_k$ et déterminer le pas optimal λ_k solution optimale du problème $\min f(x_k + \lambda d_k)$, $\lambda \geq 0$ et poser $x_{k+1} = x_k + \lambda_k d_k$.

Etape 2 : Construire B_{k+1} comme suit :

$$B_{k+1} = B_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k},$$

avec :

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

Remplacer k par $k+1$ et aller a l'étape 1.

Les méthodes de classe Broyden

Une méthode de classe **Broyden** est une méthode de **quasi-Newton** où l'approximation de l'inverse du Hessien prendre la formule suivant :

$$\begin{aligned} B_{k+1} &= B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k} + \phi \left(s_k^T B_k s_k\right) v_k v_k^T, \qquad (3.10) \\ \text{tel que } \phi &\in [0,1]\,,\ v_k^T = \left[\frac{y_k}{y_k^T s_k} - \frac{B_k s_k}{s_k^T B_k s_k}\right]. \end{aligned}$$

♦ Si $\phi = 0$ on obtient la méthode **BFGS**, car la formule (3.10) devient :

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T S_k},$$

c'est exactement la formule de l'approximation de la méthode **BFGS**.

♦ Si $\phi = 1$ on obtient la méthode **DFP**.

3.2 Méthode du gradient conjugué à trois termes

Pour améliorer l'efficacité de la méthode du gradient conjugué classique, un type de méthodes de gradient conjugué de trois-terme a été largement étudié récemment. La première méthode générale à trois termes du gradient conjugué a été proposée par **Beale** dans [8]. Dans [19], **Nazareth** a présenté un autre type de méthode de gradient conjugué de trois terme, Dans [27], un algorithme de gradient conjugué de descente **PRP** modifié a été développé, où le direction de la recherche est obtenue par une formule à trois termes. Dans [28], la méthode de gradient conjugué **HS** a été modifiée par une méthode du gradient conjugué de descente à trois termes. Très récemment, **Andrei** [4-5] a étudié trois types de méthodes à trois termes.

3.2.1 Principe général d'une méthode du Gradient conjugué à trois termes

Condition de conjugaison de Dai-Liao

Donnons la définition :

Définition 3.2.1 *Soit* $f : \mathbb{R}^n \to \mathbb{R}$, d *un vecteur non nul de* $\mathbb{R}^n$ *est dit une direction de*

conjugaison de ***Dai-Liao****, si*

$$d_{k+1}^T y_k = -t_k g_{k+1}^T s_k, \qquad (*)$$

avec $t_k > 0$*, aussi bien qu'être suffisamment descente.*

Les méthodes du Gradient Conjugué à trois termes utilisent l'idée suivante pour résoudre le problème $(\mathcal{P})$.

Elles construisent la suite des itérés $\{x_k\}_{k\geq 1}$, approchant une solution x_k de $(\mathcal{P})$ par la récurrence :

$$x_{k+1} = x_k + \alpha_k d_k, \text{ pour } k \geq 1, \qquad (3.11)$$

où α_k est appelé le pas et d_k la direction de recherche de f en x_k.

Pour définir une méthode du Gradient Conjugué à trois termes il faut donc spécifier deux choses :

$\diamondsuit$ Dire comment la direction d_k est calculée ; la manière de procéder donne le nom à l'algorithme.

$\diamondsuit$ Dire comment on détermine le pas α_k; c'est ce que l'on appelle : la recherche linéaire.

Quelques méthodes du Gradient Conjugué à trois termes

Décrivons cette classe d'algorithmes de manière précise.

La première méthode générale à trois termes du gradient conjugué a été proposé par **E.M.L. Beale** dans [8], où en incorporant une direction d_t de redémarrage $(t \leq k-1)$, la direction de recherche a été déterminée par :

$$d_{k+1} = -g_{k+1} + \beta_k d_k + \gamma_k d_t. \qquad (3.12)$$

Dans l'algorithme de **Beale** [8], le paramètre β_k peut être donnée par β_k^{HS}, β_k^{FR} ou β_k^{DY}, etc..., et le paramètre γ_k est donnée par :

$$\gamma_k = \begin{cases} 0 & k = t+1, \\ \frac{g_{k+1}^T y_t}{d_t^T y_t} & k > t+1. \end{cases} . \qquad (3.13)$$

Dans [19], **Nazareth** a présenté un autre type de méthode de gradient conjugué à trois termes, où la direction de recherche est calculé par :

$$d_{k+1} = -y_k + \frac{y_k^T y_k}{y_k^T d_k} d_k + \frac{y_{k-1}^T y_k}{y_{k-1}^T d_{k-1}} d_{k-1}, \qquad (3.14)$$

avec $d_{-1} = d_0 = 0$. Il est prouvé que si f est une fonction d'objectif quadratique convexe, alors pour toute pas α_k, les directions de recherche générés par (3.14) sont conjugués dans

la matrice de coefficient du terme quadratique en f. Dans [26], un algorithme de gradient conjugué **PRP** modifié décent a été développé, où la direction de recherche est obtenu par la formule à trois termes suivants :

$$d_{k+1} = -g_{k+1} + \frac{g_{k+1}^T y_k}{g_k^T g_k} d_k - \frac{g_{k+1}^T d_k}{g_k^T g_k} y_k.$$

Dans [27], la méthode de gradient conjugué **HS** a été modifiée par une méthode de gradient conjugué à trois termes de descente. Ça lit :

$$d_{k+1} = -g_{k+1} + \frac{g_{k+1}^T y_k}{s_k^T g_k} s_k - \frac{g_{k+1}^T s_k}{s_k^T y_k} y_k,$$

où $s_k = x_{k+1} - x_k$.

Différent des méthodes de gradient conjugué classiques, les méthodes de trois termes ci-dessus ont une propriété remarquable que la direction construite est suffisamment descente, à savoir pour chaque $k \geq 0$, il satisfait que $g_k^T d_k \leq -c \|g_k\|^2$, où c est une constante donnée. Très récemment, **Andrei** dans [4-6] a enquêté sur les trois types de méthodes de gradient de descente à trois terme suivants :

$$d_{k+1} = -\frac{y_k^T s_k}{\|g_k\|^2} g_{k+1} + \frac{y_k^T g_{k+1}}{\|g_k\|^2} s_k - \frac{s_k^T g_{k+1}}{\|g_k\|^2} y_k, \tag{3.15}$$

$$d_{k+1} = -g_{k+1} - \left(\left(1 + \frac{\|y_k\|^2}{y_k^T s_k} \right) \frac{s_k^T g_{k+1}}{y_k^T s_k} - \frac{y_k^T g_{k+1}}{y_k^T s_k} \right) s_k - \frac{s_k^T g_{k+1}}{y_k^T s_k} y_k, \tag{3.16}$$

et

$$d_{k+1} = -g_{k+1} - \left(\left(1 + 2\frac{\|y_k\|^2}{y_k^T s_k} \right) \frac{s_k^T g_{k+1}}{y_k^T s_k} - \frac{y_k^T g_{k+1}}{y_k^T s_k} \right) s_k - \frac{s_k^T g_{k+1}}{y_k^T s_k} y_k. \tag{3.17}$$

respectivement.

3.3 Nouvelle méthode du gradient conjugué à trois termes

3.3.1 Le nouveau Algorithme. Définition et propriétés générales

Description de la méthode

Dans cette section, nous allons exposer l'idée de proposer une nouvelle méthode de gradient conjugué spectrale et de développer un nouvel algorithme. Il est bien connu que

si la matrice Hessienne de f est définie positive, alors la direction le plus efficace à x_k est la direction de **Newton** :

$$d_{k+1} = -\nabla^2 f(x_{k+1})^{-1} g_{k+1}. \tag{3.18}$$

Ainsi, pour la direction de **Newton** d_{k+1}, il est vrai que :

$$s_k^T \nabla^2 f(x_{k+1}) d_{k+1} = -s_k^T g_{k+1}. \tag{3.19}$$

En vertu de **la condition de sécant** $\boxed{\nabla^2 f(x_{k+1}) s_k = y_k}$, (3.19) peut être approximée par :

$$y_k^T d_{k+1} = -s_k^T g_{k+1}. \tag{3.20}$$

Par conséquent, si une direction d_{k+1} est nécessaire pour satisfaire (3.20), alors il peut être considéré comme une direction de **Newton** approximative. Ensuite, nous allons construire une direction d_{k+1} de telle sorte que les deux (3.20) et $(*)$ sont satisfaits.

Noter que la différence entre (3.16) et (3.17) se trouve dans le second terme. Nous essayons de remplacer ce terme par :

$$-\left(\left(1+\tau\frac{\|y_k\|^2}{y_k^T s_k}\right)\frac{s_k^T g_{k+1}}{y_k^T s_k} - \frac{y_k^T g_{k+1}}{y_k^T s_k}\right) s_k,$$

où τ est une constante scalaire de telle sorte que la direction obtenue satisfait (3.20) et $(*)$. Nous avons d'abord besoin que (3.20) est vraie. Ensuite, il est facile d'obtenir que $\tau = -1$. En outre, nous pouvons prouver le résultat suivant.

Proposition. 3.3.1 *Soit d_{k+1} donnée par :*

$$d_{k+1} = -g_{k+1} - \left(\left(1-\frac{\|y_k\|^2}{y_k^T s_k}\right)\frac{s_k^T g_{k+1}}{y_k^T s_k} - \frac{y_k^T g_{k+1}}{y_k^T s_k}\right) s_k - \frac{s_k^T g_{k+1}}{y_k^T s_k} y_k. \tag{3.21}$$

Alors, $(*)$ est vérifiée.

Preuve. Par calcul direct, on peut vérifier que d_{k+1} satisfait $(*)$, où $t_k = 1 > 0$. ■

La proposition précédente montre que d_{k+1} est une direction conjugué satisfaisant la condition de conjugaison de Dai-Liao, aussi bien qu'être une direction approximative de **Newton**.

Réécrire d_{k+1} en (3.21) comme :

$$d_{k+1} = -Q_k g_{k+1}, \tag{3.22}$$

où : Q_k est une matrice, donnée par :

$$Q_k = I - \frac{s_k y_k^T + y_k s_k^T}{y_k^T s_k} + \left(1 - \frac{\|y_k\|^2}{y_k^T s_k}\right) \frac{s_k s_k^T}{y_k^T s_k}. \tag{3.23}$$

Rappelant la méthode d'actualisation (de mise à jour) de **BFGS** classique suivante :

$$H_{k+1} = H_k - \frac{s_k y_k^T H_k + y_k s_k^T}{y_k^T s_k} + \left(1 + \frac{y_k^T H_k y_k}{y_k^T s_k}\right) \frac{s_k s_k^T}{y_k^T s_k}. \tag{3.24}$$

On voit qu'il existe une relation étroite entre (3.23) et (3.24). Actuellement pour $H_k = I$ et modifier le signe devant $y_k^T s_k$ dans le troisième terme de (3.24), nous savons que (3.24) se révèle être (3.23). De ce point de vue, d_{k+1} dans (3.21) est une direction spectrale similaire à celle obtenue par la méthode de **quasi-Newton**.

Cependant, d_{k+1} donné par (3.21) n'est pas toujours une direction de descente. Actuellement, $y_k^T s_k < \|y_k\|^2$, alors d_{k+1} peut être montée. Pour cette raison, nous modifions (3.21) comme suivant :

$$d_{k+1} = -g_{k+1} - \delta_k s_k - \eta_k y_k, \tag{3.25}$$

où :

$$\delta_k = \left(1 - \min\left\{1, \frac{\|y_k\|^2}{y_k^T s_k}\right\}\right) \frac{s_k^T g_{k+1}}{y_k^T s_k} - \frac{y_k^T g_{k+1}}{y_k^T s_k}, \tag{3.26}$$

et

$$\eta_k = \frac{y_k^T g_{k+1}}{y_k^T s_k}. \tag{3.27}$$

Remarque 3.3.1 *Dans la section qui suit, nous allons prouver que d_{k+1} déterminée par (3.25), (3.26) et (3.27) est une direction de descente.*

Remarque 3.3.2 *Il est noter que si $y_k^T s_k < \|y_k\|^2$ est vérifiée, alors (3.25) se réduit à la méthode dans [26]. Cela signifie que notre méthode est une extension de celui de [26].*

Avec la construction de la direction, nous trouvons un pas par la stratégie de recherche linéaire standard de **Wolfe** [24-25] :

$$\begin{cases} f(x_{k+1}) \leq f(x_k) + \rho \alpha_k g_k^T d_k \\ g(x_{k+1})^T \geq \sigma g_k^T d_k \end{cases}, \tag{3.28}$$

où $0 < \rho < \sigma < 1$.(ρ et σ sont deux constantes).

Algorithme de la méthode du gradient conjugué à trois termes (MTHREECG)

▶ **Etape 1 (Initialisation) :** Choisir $x_0 \in \text{dom}(f)$ et calculer $f_0 = f(x_0)$, $g_0 = \nabla f(x_0)$ et $d_0 = -g_0$.Pour $k = 0$. Poser $k = 1$ et aller à l'étape 2.

▶ **Etape 2 (Critère d'arrêt) :** Si $\|g_k\|_\infty < \epsilon$, alors STOP, sinon aller à l'étape 3.

▶ **Etape 3 (Détermination du pas) :** En utilisant la recherche linéaire de wolfe (3.28) pour déterminer le pas α_k.

▶ **Etape 4 :** Calculer $z = x_k + \alpha_k d_k$, $g_z := \nabla f(z)$ et $y_k = g_k - g_z$.

▶ **Etape 5 :** Calculer $\bar{a}_k = \alpha_k g_k^T d_k$ et $\bar{b}_k = -\alpha_k y_k^T d_k$.

▶ **Etape 6 (Accélération du pas) :** Si $\bar{b}_k > 0$, alors calculer $\left(\varepsilon_k = -\frac{\bar{a}_k}{\bar{b}_k}\right)$, et actualiser l'itération comme $x_{k+1} = x_k + \varepsilon_k \alpha_k d_k$.. Sinon, poser $x_{k+1} = z$. Calculer $f_{k+1}, g_{k+1}, s_k = x_{k+1} - x_k$ et $y_k = g_{k+1} - g_k$.

▶ **Etape 7 (Actualisation des coefficients) :** Calculer δ_k et η_k par (3.26) et (3.27), respectivement.

▶ **Etape 8 (Actualisation du direction) :** Calculer une nouvelle direction par (3.25).

▶ **Etape 9 (Critère de redémarrage de Powell) :** Si $\left|g_{k+1}^T g_k\right| > 0.2 \left\|g_{k+1}\right\|^2$, alors poser $d_{k+1} = -g_{k+1}$.

▶ **Etape 10 :** Poser $k := k + 1$. Aller à l'étape 2.

Remarque 3.3.3 *Comme pour les résultats existants (voir, par exemple, [1,7,12-16] et [18]), la mise en place de la convergence de l'algorithme nécessite que le pas obtenu satisfait les conditions de la recherche linéaire de* ***Wolfe*** *forte suivant :*

$$\begin{cases} f(x_{k+1}) \leq f(x_k) + \rho \alpha_k g_k^T d_k \\ \left|g(x_{k+1})^T d_k\right| \leq -\sigma g_k^T d_k \end{cases} . \tag{3.29}$$

Dans ce cas, l'étape 3 doit être mis en œuvre par (3.29).

Remarque 3.3.4 *Dans [3],* ***N. Andrei*** *a proposé un programme d'accélération du pas comme suit. Pour $\eta > 0$, on a :*

$$f(x_k + \eta \alpha_k d_k) = f(x_k) + \eta \alpha_k g_k^T d_k + \frac{1}{2} \eta^2 \alpha_k^2 d_k^T \nabla^2 f(x_k) d_k + o\left(\left\|\eta \alpha_k d_k^T\right\|^2\right),$$

et

$$f(x_k + \alpha_k d_k) = f(x_k) + \alpha_k g_k^T d_k + \frac{1}{2} \alpha_k^2 d_k^T \nabla^2 f(x_k) d_k + o\left(\left\|\alpha_k d_k^T\right\|^2\right).$$

Définir :

$$\Psi_k(\eta) = f(x_k + \eta \alpha_k d_k) - f(x_k + \alpha_k d_k).$$

Alors, de $\Psi_k'(\eta) = 0$, *on obtient :*

$$\eta_k = -\frac{a_k}{b_k + 2\alpha_k \epsilon_k}, \tag{3.30}$$

telle que $\Psi_k(\eta_k) \leq 0$, *où* $a_k = \alpha_k g_k^T d_k < 0, b_k = \alpha_k^2 d_k^T \nabla^2 f(x_k) d_k$, *et* $\epsilon_k = o\left(\alpha_k \|d_k\|^2\right)$. *Ceci,* $\eta_k \alpha_k$ *est une amélioration de* α_k.

étant donné que la méthode proposée dans le présent document ne dépend pas sur le calcul de $\nabla^2 f(x_k)$, nous remplaçons le b_k ci-dessus à l'étape 5 de l'algorithme par $\bar{b}_k = -\alpha_k y_k^T d_k$. Par la suite, à l'étape 6 de l'algorithme, nous améliorons le pas en réglant $\left(\varepsilon_k := -\frac{a_k}{\bar{b}_k}\right)$ et $x_{k+1} := x_k + \varepsilon_k \alpha_k d_k$ si $\bar{b}_k > 0$. Il est clair que ε_k est une forme modifiée de η_k, où ϵ_k est enlevé, et a_k et b_k sont remplacés par $\bar{a}_k$ et $\bar{b}_k$, respectivement. Il peut être prouvé que l'inégalité suivante est (voir Proposition 3.1 dans [3]) :

$$f(x_{k+1}) = f(x_k + \varepsilon_k \alpha_k d_k) \leq f(x_k + \alpha_k d_k). \tag{3.31}$$

A savoir, $\varepsilon_k \alpha_k$ est une accélération du pas α_k.

Remarque 3.3.5 *Il a été prouvé dans [9] que la méthode du gradient conjugué standard sans redémarrage est au plus linéairement convergente. Dans [22] et [5], il a été suggéré que la direction est redémarré par* $d_{k+1} = -g_{k+1}$, *dans le cas où* $|g_k^T g_{k+1}| > 0.2 \|g_{k+1}\|^2$, *que l'on appelle la stratégie de redémarrage de* ***Powell****. Dans l'étape 9 de l'algorithme, la stratégie de redémarrage de Powell est utilisé pour améliorer l'efficacité de l'algorithme.*

3.3.2 Convergence globale du nouveau Algorithme

Dans cette section, nous avons étudier la convergence globale de l'algorithme de la méthode du gradient conjugué à trois termes. Nous indiquons d'abord les hypothèses suivantes douces, qui sont nécessaires pour prouver les principaux résultats dans le présent document.

Hypothèse 3.3.1 *L'ensemble* Ω *définie par :*

$$\Omega = \{x \in \mathbb{R}^n / f(x) \leq f(x_0)\},$$

est borné.

Hypothèse 3.3.2 *Dans un voisinage* V *de* Ω, f *est continûment différentiable et son gradient* g *est Lipschitzienne, c'est à dire, il existe une constante* $L > 0$ *tel que :*

$$\|g(x) - g(y)\| \leq L \|x - y\|, \forall x, y \in V. \tag{3.32}$$

Comme $f(x_k)$ décroît quand $k \to +\infty$, il résulte de l'hypothèse précédente que la suite $\{x_k\}$ générée par l'algorithme de la méthode du gradient conjugué à trois termes est clairement contenue dans une région bornée. Par conséquence, il existe une suite convergente de $\{x_k\}$. Sans perte de généralité, on suppose que $\{x_k\}$ est convergente. En outre, à partir les deux hypothèses précédentes, il est conclu qu'il existe une constante $\gamma_1 > 0$ tel que pour tout $x \in \Omega, \|g(x)\| \leq \gamma_1$. Ainsi, la suite $\{g_k\}$ est également bornée.

Lemme 3.3.1 *Soit $f : \mathbb{R}^n \to \mathbb{R}^n$ une fonction continûment différentiable. Supposons que la recherche linéaire satisfait aux conditions de* ***Wolfe*** (3.28). *Alors, d_{k+1} définie par (3.25), (3.26) et (3.27) est une direction de descente.*

Preuve. Des conditions de **Wolfe** (3.28), nous avons :

$$y_k^T s_k > 0.$$

Si :

$$y_k^T s_k < \|y_k\|^2,$$

alors :

$$g_{k+1}^T d_{k+1} = -\|g_{k+1}\|^2 < 0.$$

Sinon

$$g_{k+1}^T d_{k+1} = -\|g_{k+1}\|^2 - \left(1 - \min\left\{1, \frac{\|y_k\|^2}{y_k^T s_k}\right\}\right) \frac{\left(g_{k+1}^T s_k\right)^2}{y_k^T s_k} \leq 0.$$

■

Lemme 3.3.2 *Supposons que la recherche linéaire satisfait aux conditions de* ***Wolfe*** *(3.28). Alors d_{k+1} définie par (3.25), (3.26) et (3.27) satisfait à la condition de conjugaison $(*)$.*

Preuve. Par calcul direct, nous obtenons :

$$y_k^T d_{k+1} = -\left(1 - \min\left\{1, \frac{\|y_k\|^2}{y_k^T s_k}\right\} + \frac{\|y_k\|^2}{y_k^T s_k}\right) + s_k^T g_{k+1},$$

où

$$t_k = 1 - \min\left\{1, \frac{\|y_k\|^2}{y_k^T s_k}\right\} + \frac{\|y_k\|^2}{y_k^T s_k} > 0.$$

En particulier, si $\|y_k\|^2 < y_k^T s_k$, alors $t_k = 1$. Par conséquence, la direction d_{k+1} définie par (3.25)-(3.29) satisfait à la condition de conjugaison $(*)$. ■

Lemme 3.3.3 *Supposons que d_k est une direction de descente et que ∇f satisfait à la condition de Lipschitz :*

$$\|\nabla f(x) - \nabla f(x_k)\| \leq L \|x - x_k\|,$$

*pour tous x sur le segment reliant x_k et x_{k+1}. Sous la recherche linéaire de **Wolfe** (3.28), il est vrai que :*

$$\alpha_k \geq \frac{(1-\sigma)\left|g_k^T d_k\right|}{L\|d_k\|^2}.$$

Preuve. Soustraire $g_k^T d_k$ des deux côtés de la première inégalité dans (3.28), on obtient que :

$$(\sigma - 1) g_k^T d_k \leq (g_{k+1} - g_k)^T d_k = y_k^T d_k \leq \|y_k\| \|d_k\| \leq \alpha_k L \|d_k\|^2. \tag{3.33}$$

Comme d_k est descente et $0 < \sigma < 1$, le résultat désiré découle directement de (3.33). ■

Lemme 3.3.4 *Soient $\{d_k\}$ et $\{\alpha_k\}$ sont deux suites générées par l'algorithme de la méthode du gradient conjugué à trois termes. Supposons que les **hypothèses 3.3.1** et **3.3.2** sont vérifiées. Alors :*

$$\sum_{k=0}^{\infty} \frac{\left(g_k^T d_k\right)^2}{\|d_k\|^2} < +\infty. \tag{3.34}$$

Preuve. À partir de (3.28) et le **lemme 3.3.3**, il en résulte que :

$$f_k - f_{k+1} \geq -\rho \alpha_k g_k^T d_k \geq \rho \frac{(1-\sigma)\left|g_k^T d_k\right|}{L\|d_k\|^2}.$$

Alors, par **l'hypothèse 3.3.1**, (3.34) est prouvée. ■

Lemme 3.3.5 *Soient $\{d_k\}$ et $\{\alpha_k\}$ sont deux suites générées par l'algorithme de la méthode du gradient conjugué à trois termes, où α_k est obtenu par la recherche linéaire de **Wolfe** forte. Supposons que les hypothèses **3.3.1** et **3.3.2** sont vérifiées. Si :*

$$\sum_{k=0}^{\infty} \frac{1}{\|d_k\|^2} = +\infty, \tag{3.35}$$

Alors :

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

Preuve. Dénoté :

$$\rho_k = \frac{d_k^T g_k}{\|g_k\| \|d_k\|}.$$

Compte tenu du **lemme 3.3.1**, il est clair que :

$$d_k^T g_k \leq - \|g_k\|^2 ,$$

Ainsi :

$$\rho_k \leq -\frac{\|g_k\|}{\|d_k\|}.$$

Il donne :

$$\rho_k^2 \geq \frac{\|g_k\|^2}{\|d_k\|^2}.$$

Supposons que le résultat souhaité est pas vrai, c'est-à-dire :

$$\lim_{k\to\infty} \inf \|g_k\| \neq 0,$$

alors, il existe $\gamma > 0$ tel que :

$$\|g_k\| \geq \gamma > 0, \forall k \geq 1.$$

Par conséquent

$$\frac{\gamma^2}{\|d_k\|^2} < \frac{\|g_k\|^2}{\|d_k\|^2} < \rho_k^2 = \frac{\left(d_k^T g_k\right)^2}{\|d_k\|^2 \|g_k\|^2} < \frac{\left(d_k^T g_k\right)^2}{\gamma^2 \|d_k\|^2}.$$

Du **lemme 3.3.4**, nous savons que :

$$\sum_{k=0}^{\infty} \frac{\left(g_k^T d_k\right)^2}{\|d_k\|^2} < +\infty,$$

ce qui implique que :

$$\sum_{k=0}^{\infty} \frac{1}{\|d_k\|^2} < +\infty \text{ est convergent.}$$

Cela contredit la condition (3.35). La preuve du résultat souhaité est terminée. ■

Théorème 3.3.1 *Soient $\{d_k\}$ et $\{\alpha_k\}$ sont deux suites générées par l'algorithme de la méthode du gradient conjugué à trois termes, où α_k est obtenu par la recherche linéaire de* ***Wolfe*** *forte. Supposons que les* ***hypothèses 3.3.1*** *et* ***3.3.2*** *sont vérifiées. Si f est une fonction uniformément convexe sur S, c'est à dire qu'il existe une constante $\mu > 0$ tel que :*

$$\left(\nabla f(x) - \nabla f(y)\right)^T (x - y) \leq \mu \|x - y\|^2 , \forall x, y \in V.$$

Alors :

$$\lim_{k\to\infty} \|g_k\| = 0. \tag{3.36}$$

Preuve. Par la continuité de Lipschitz, nous savons que :

$$\|y_k\| \leq L \|s_k\|.$$

D'autre part, par la convexité uniforme, il donne :

$$y_k^T s_k \geq \mu \|s_k\|^2.$$

Ainsi :

$$\begin{aligned} |\delta_k| &\leq \frac{\left|s_k^T g_{k+1}\right|}{\left|y_k^T s_k\right|} + \frac{\|y_k\|^2 \left|s_k^T g_{k+1}\right|}{\left|y_k^T s_k\right|^2} + \frac{\left|y_k^T g_{k+1}\right|}{\left|y_k^T s_k\right|} \\ &\leq \frac{\gamma}{\mu \|s_k\|} + \frac{L^2 \gamma}{\mu^2 \|s_k\|} + \frac{L\gamma}{\mu \|s_k\|} \\ &= \frac{\gamma}{\mu}\left(1 + L + \frac{L^2}{\mu}\right)\frac{1}{\|s_k\|}. \end{aligned} \tag{3.37}$$

Depuis,

$$|\eta_k| = \frac{\left|s_k^T g_{k+1}\right|}{\left|y_k^T s_k\right|} \leq \frac{\left\|s_k^T\right\| \|g_{k+1}\|}{\mu \|s_k\|^2} \leq \frac{\gamma}{\mu \|s_k\|},$$

il est conclu que :

$$d_{k+1} \leq g_{k+1} + |\delta_k| \|s_k\| + |\eta_k| \|y_k\| \leq \gamma + \frac{\gamma}{\mu}\left(2 + L + \frac{L^2}{\mu}\right).$$

Par conséquent, d_{k+1} est borné. Compte tenu du **lemme 3.3.5**, (3.36) est vraie. La preuve a été complétée. ■

3.4 Tests numériques

Dans cette partie nous allons effectuer des tests numériques pour montrer l'efficacité et la performance du nouveau algorithme (**MTHREECG**).

3.4.1 Principe :

Nous testons l'algorithme (**MTHREECG**) en mettant en œuvre pour résoudre les **75** problèmes de test de référence de [2]. Pour chaque problème de test, nous changeons la dimension $n = 1000, 2000, ..., 10000$, respectivement. Dans la recherche linéaire de **Wolfe**, les paramètres $\rho = 0.0001$ et $\sigma = 0,8$. La tolérance Prend le critère d'arrêt de l'algorithme $\varepsilon = 10^{-6}$.

3.4.2 Etapes :

Nous comparons **MTHREECG** avec **THREECG** dans [7], la **CG_DESCENT** dans [14], et la méthode du gradient conjugué à trois termes de **Beale** dans [10].

3.4.3 Codes de programmation :

Tous les codes des procédures informatiques sont écrites en Fortran 77, et sont mises en œuvre sur PC avec Processeur 2,9 GHz, 4 Go de mémoire RAM et système d'exploitation Windows XP. La figure du code Fortran Version 1.4 (14 Novembre 2005) de la (CG_DESCENT) est téléchargé à partir du site Web :
http ://clas.ufl.edu/users/hager/papers/CG/Archive/CG_DESCENT-F-1.4.tar.gz.

Le code de la (ThreeCG) et la recherche linéaire de **Wolfe** est téléchargé à partir du site Web de **N.Andrei** :
http ://camo.ici.ro/neculai/THREECG/threecg.for.

3.4.4 Profils de performance numérique :

Les profils de performance numérique de la méthode du gradient conjugué **MTHREECG**, **THREECG** et **CG_DESCENT** basés sur le temps CPU.

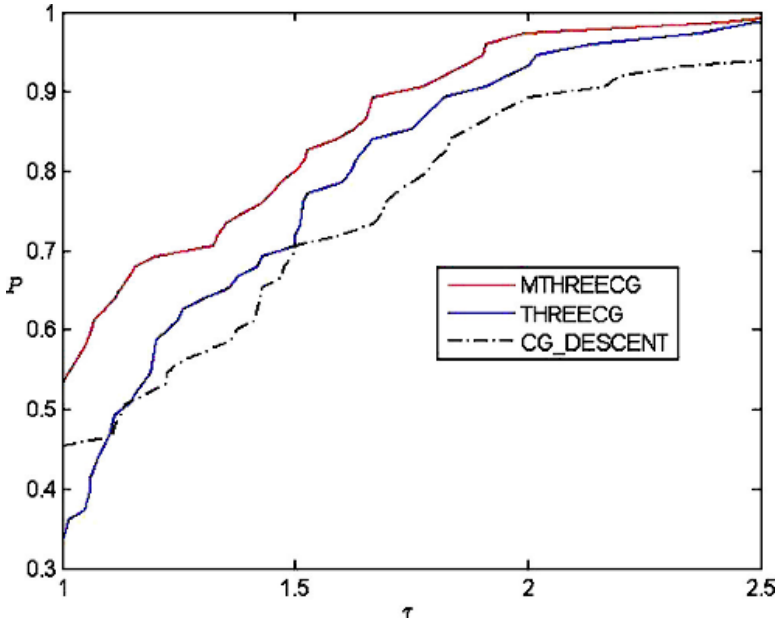

FIG. 3.1 – Performance basée sur le temps CPU.

Les profils de performance numérique de la méthode du gradient conjugué **MTHREECG**, **THREECG** et **CG_DESCENT** basés sur le nombre d'itérations.

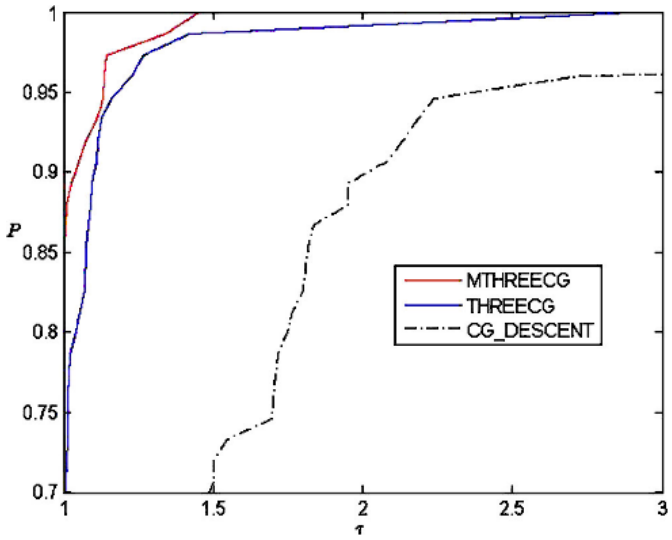

FIG. 3.2 – Performance basée sur le nombre d'itérations.

Les profils de performance numérique de la méthode du gradient conjugué **MTHREECG**, **THREECG, CG_DESCENT** et **BEALE** basés sur le temps CPU.

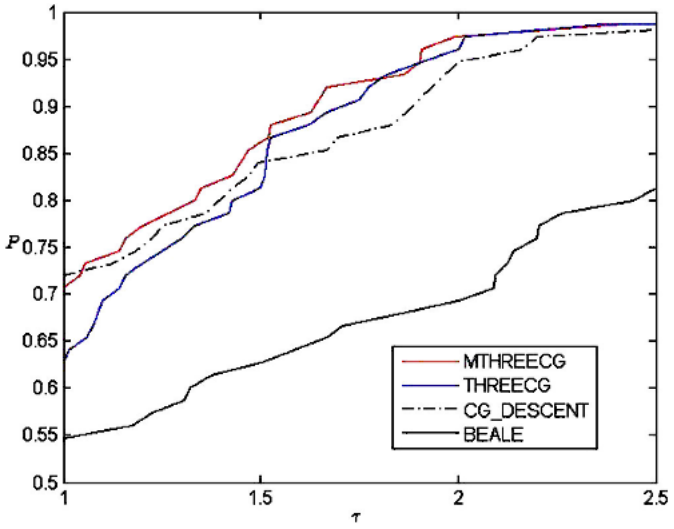

FIG. 3.3 – Performance basée sur le temps CPU

Conclusion

Il est clair d'après les figures. 3.1, 3.2 et 3.3, que le nouveau algorithme est numériquement plus performante que les autres méthodes du gradient conjugué **THREECG**, **CG_DESCENT** et **BEALE**

Conclusion générale

Durant les 60 dernières années, beaucoup de méthodes du gradient conjugué ont été élaborées et par conséquent une multitude de résultats de convergence ont été démontrées. Dans ce mémoire nous avons présenté une synthèse générale sur les méthodes de gradient conjugué à trois termes.

Ensuite, nous avons étudiée une nouvelle famille des méthodes de gradient conjugué à trois terme, la propriété de descente est assurée à chaque itération et certains résultats généraux de convergence sont montrés pour cette famille avec la recherche linéaire de **Wolfe** fortes.

Les résultats numériques montrent que cette méthode est plus performante que d'autres méthodes de gradient conjugué assez connues.

Bibliographie

[1]- A.Y. Al-Bayati, Hawraz N. Al-khayat, *A global convergent spectral conjugate gradient method*,Aust. J. Basic Appl. Sci. 7 (2013) 303-309.

[2]- N. Andrei, *An unconstrained optimization test functions collection*, Adv. Model. Optim. 10 (2008) 147-161.

[3]- N. Andrei, *Acceleration of conjugate gradient algorithm for unconstrained optimization*, Appl. Math. Comput. 213 (2009) 361-369.

[4]- N. Andrei, *A modified Polak-Ribière-Polyak conjugate gradient algorithm for unconstrained optimization*, Optimization 60 (2011) 1457-1471.

[5]- N. Andrei, *A simple three-term conjugate gradient algorithm for unconstrained optimization*, J. Comput. Appl. Math. 241 (2013) 19-29.

[6]- N. Andrei, *On three-term conjugate gradient algorithms for unconstrained optimization*, Appl. Math. Comput. 219 (2013) 6316-6327.

[7]- N. Andrei, *Another conjugate gradient algorithm with guaranteed descent and conjugacy conditions*, J. Optim. Theory Appl. 159(1) (2013) 159-182, http ://dx.doi.org/10.1007/s10957-013-0285-9.

[8]- E.M.L Beale, *A derivation of conjugate gradients*, in : F.A Lootsma (Ed.), *Numerical Methods for Nonlinear optimization*, Academic Press, London, 1972, pp. 39-43.

[9]- H.P. Crowder, P. Wolfe, *Linear convergence of the conjugate gradient method*, IBM J. Res. Dev. 16 (1969) 431-433.

[10]- Y.H. Dai, Y. Yuan, *Convergence properties of the Beale-Powell restart algorithm*, Sci. China Ser. A 41 (1998) 1142-1150.

[11]- Y.H. Dai, Y. Yuan, *A nonlinear conjugate gradient method with a strong global convergence property,* SIAM J. Optim. 10 (1999) 177-182.

[12]- Y.H. Dai, Y. Yuan, *A three-parameter family of nonlinear conjugate gradient methods, Math. Comput. 70 (2000) 1155-1167.*

[13]- R. Fletcher, C.M. Reeves, *Function minimization by conjugate gradients*, Comput. J. 7 (1964) 149-154.

[14]- W.W. Hager, H. Zhang, Algorithme 851 : CG_DESCENT, *a conjugate gradient method with guaranteed descent*, ACM Trans. Math. Softw. 32 (2006) 113-137.

[15]- M.R. Hestenes, E.L. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. Natl. Bur. Stand. 49(6) (1952) 409-436.

[16]- H.D. Huang, Y.J. Li, Z.X. Wei, *Global convergence of a modified PRP conjugate gradient method*, J. Math. Res. Expo. 30 (2010) 141-148.

[17]- S. Huang, Z. Wan, S.H. Deng, *A modified projected conjugate gradient method for unconstrained optimization problems*, ANZIAM J. 54(03) (2013) 143-152.

[18]- C. Li, L. Fang, X.L. Cao, *Global convergence of a kind of conjugate gradient method*, TELKOMNIKA 11 (2013) 544-549.

[19]- L. Nazareth, *A conjugate direction algorithm without line search*, J. Optim. Theory Appl. 23 (1997) 373-387

[20]- E. Polak and G. Ribière, *Note sur la convergence de directions conjugées*, Rev. Francaise Informat Recherche Opertionelle, 3e Année 16 (1969), pp. 35–43.

[21]- B. T. Polyak, *The conjugate gradient method in extreme problems*, USSR Comp. Math. Math. Phys., 9 (1969), pp. 94–112.

[22]- M.J.D Powell, *Restart procedures of the conjugate gradient method*, Math. Program. 2 (1977) 241-254.

[23]- D. Songhai, W. Zhong, *A three-term conjugate gradient algorithm for large-scale unconstrained optimization problems*, Applied Numerical Mathematics, 92(2015)70–81.

[24]- Z. Wan, Z.L. Yang, Y.L. Wang, *New spectral PRP conjugate gradient method for unconstrained optimization*, Appl. Math. Left. 24(1) (2011) 16-22.

[25]- P. Wolfe, *Convergence conditions for ascent methods*, SIAM Rev. 11 (1968) 226-235.

[26]- J. Zhang, Y.Xiao, Z. Wei, *Nonlinear conjugate gradient methods with sufficient descent condition for large-scale unconstrained optimization*, Math. Probl. Eng. 2009 (2009),

http ://dx.doi.org/10.1155/2009/234290, Article ID 243290.

[27]- L. Zhang, W. Zhou, D.H. Li, *A descent modified Polak-Ribière-Polyak conjugate gradient method and its global convergence,* IAM J. Numer. Anal. 26 (2006) 629-640.

[28]- L. Zhang, W. Zhou, D.H. Li, *Some descent three-term conjugate gradient methods and their global convergence*, Optim. Methods Softw. 22 (2007) 697-711.