RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
Ministère de L'enseignement Supérieur et de la Recherche Scientifique
UNIVERSITÉ IBN KHALDOUN TIARET
FACULTÉ DE MATHÉMATIQUES ET DE L'INFORMATIQUES
Département de Mathématiques

MÉMOIRE MASTER

Présenter en vue de l'obtention du diplôme de master

Spécialité :
«Mathématiques »

Option :
« Analyse fonctionnelle et équations différentielles »

Présenté Par :
Khiar Fatima Zohra & Sebbane Ikram

Sous L'intitulé :

Certaines méthodes pour résoudre un problème d'optimisation

Soutenu publiquement le 25 / 06 / 2024
À Tiaret devant le jury composé de :

M.	MAAZOUZ Kada	MCA	Université de Tiaret	Président
Mme.	SABIT Souhila	MCA	Université de Tiaret	Examinateur
M.	REZZOUG Nadir	MCB	Université de Tiaret	Encadreur

Année universitaire : 2023/2024

Remerciements

Nous tenons tout d'abord à remercier **ALLAH**, le tout puissant, qui nous a donné la force et le courage pour réaliser ce mémoire.

Nous remercions vivement **Monsieur N. Rezzoug** pour la confiance qu'il nous a accordé en acceptent de nous encadrer, pour sa disponibilité, son aide et ses suggestions. Nous prions ALLAH de accorder lui une bonne santé.

Nos plus sincères remerciements aux membres du jury Monsieur **K. Maazouz** et Madame **S. Sabit**, et qui nous ont fait l'honneur de examiner notre travail.

Nous adressons également nos remerciements à tous nos proches, qui nous ont toujours soutenus et encouragés tout au long de la réalisation de ce travail. Merci à tous.

★ ——————————————————————————— ★

* Dédicace *

Du fond de mon coeur, je dédie ce travail

À ma très chère mère

Vous m'avez donné la vie, la tendresse et le courage pour réussir tout ce que je peux t'offrir ne pourra exprimer l'amour et la reconnaissance que je vous porte. J'avoue vraiment que vous êtes pour moi la lumière qui me guide vers le chemin de la réussite. C'est àvous que je dois mon succès.
En témoignage, je vous offre ce modeste travail pour vous remercier de vos sacrifices consentis et pour l'affection dont m'avez toujours témoignée.

À ma très cher père "AISSA"

Vous avez toujours été à mes côtés pour me soutenir et m'encourager. L'épaule solide, l'œil attentif compréhensif et la personne la plus digne de mon estime et de mon respect. Aucune dédicace ne saurait exprimer mes sentiment, que dieu vous préserve et vous procure santé et longue vie.

À mes très chers frères :
Djamel, **Islam** et **Yassine**.
À toute la famille **KHIAR**.
À mon binôme **Ikram**,
et toute personne qui me connait.

K. Fatma Zohra

★ ★

* Dédicace *

À ma très chère mère

Je dédie ce succès à ma chère mère et à mon modèle de persévérance et d'efforts pour atteindre mon objectif Merci maman pour ton amour et tes sacrifices pour moi et tes précieux conseils et toute votre aide et votre présence dans ma vie.

À ma très cher père

Je dédie mes salutations à mon cher père qui m'a soutenu pour surmonter toutes les difficultés de la vie et qui m'a donné la motivation de poursuivre mes études. Et si ALLAH le veut, ce travail sera le fruit de ses efforts. Merci mon père pour votre éducation continue et votre soutien.

À mes grand-père et mes grand-mère maternelle.
À mon cher frère Sidhmed
et mes adorables sœurs Nadia, Djihan et Nossaiba.
À tous mes amies Fati, Ghezlan et Amel.

À toute la famille **SEBBANE**.
À mon binôme **FATIMA**.

S. Ikram

* Résumé *

Le but de ce travail est la présentation des méthodes plus utilisées pour résoudre un problème d'optimisation comme la méthode de direction conjugué, méthode de gradient conjugué, méthode de Newton et méthode de quasi-Newton.

Comme ces méthodes basées sur les recherches linéaires nous également présenter recherche linéaire inexacte d'Armijo, de Goldestein et de Wolfe.

* Abstract *

The aim of this work is the presentation of the most used methods to solve an optimization problem such as the conjugate direction method, conjugate gradient method, Newton method and quasi-Newton method.

Like these methods based on linear searches we also present inexact linear search of Armijo, Goldestein and Wolfe.

* ملخص*

الهدف من هذا العمل هو عرض الطرق الاكثر استخداما لحل مسألة التحسين مثل طريقة الاتجاه المترافق، وطريقة التدرج المرافق، وطريقة نيوتن، وطريقة شبه نيوتن. مثل هذه الأساليب المستندة الى عمليات البحث الخطية، نقدم أيضا بحثا خطيا غير دقيق عن Armijoو Goldstein وWolfe

Table des matières

NOTATIONS

Symbole	Signification
$\mathbb{R}$	Ensemble des nombres réels.
$\mathbb{R}^n$	$\overbrace{\mathbb{R} \times \mathbb{R} \times \ldots \times \mathbb{R}}^{n fois}$.
$\mathbb{R}^+$	L'ensemble des nombres réels positifs.
$\mathbb{C}$	L'ensemble des nombres complexes.
$J(x)$	Jacobienne.
$H(x)$	Hessienne.
$f(x)$	Fonction objective.
$x = (x_1, \ldots, x_n)$	Vecteur.
$\| \cdot \|$	Norme d'un vecteur.
$\langle \ , \ \rangle$	Produit scalaire.
$\nabla f(x)$	Gradient de f.
$M_{m,n}$	Matrice de m lignes et n colonnes.
A^t	Transposée d'une matrice A.
$L(\mathbb{R}^n, \mathbb{R})$	L'espace des fonctions continues de $\mathbb{R}^n$ dans $\mathbb{R}$.
$\widehat{x}$	Solution optimale.
H^{-1}	Matrice inverse.
$\frac{\partial f}{\partial x}$	Dérivée partielle de f par apport de x.
$\frac{\partial^2 f}{\partial x_i \partial x_j}$	Dérivée partielle seconde de f par apport de x.
(PQSC)	Problème quadratique sans contrainte.
(PNQSC)	Problème non quadratique sans contrainte.
(FR)	Fletcher Reeves.
(PRP)	Polak-Ribière-Polyak.
(DY)	Dai-Yuan.
(HZ)	Hestenes-Stiefel.

Introduction

L'optimisation est un processus visant à maximiser ou minimiser une fonction donnée, souvent appelée fonction coût ou fonction objectif. Cela implique la recherche des valeurs des variables qui rendent cette fonction la plus élevée (optimisation maximale) ou la plus basse (optimisation minimale) possible.

L'optimisation est largement utilisée dans divers domaines, tels que les sciences économiques pour la prise de décision rationnelle, l'ingénierie pour améliorer les performances des systèmes, la finance pour maximiser les profits, et bien d'autres encore. L'optimisation permet d'améliorer l'efficacité des processus, de réduire les coûts et d'améliorer les résultats dans de nombreux domaines de la vie quotidienne et industrielle. Il est un concept fondamental en mathématiques appliquées qui vise à trouver les meilleures solutions possibles pour atteindre un objectif spécifique dans divers domaines.
L'optimisation est l'étude des problèmes (P) qui écrire sur la forme suivant :

$$\min_{x \in X} f(x) \quad (P).$$

On dit que le problème (P) admet une solution s'il existe le choix $x_0 \in X$ tel que

$$\forall x \in X, \;\; f(x_0) \leq f(x),$$

alors $f(x_0)$ est un minimum de f sur X.

Pour résoudre un problème d'optimisation il y a plusieurs méthodes. Dans ce mémoire, on traite les méthodes plus utilisés. Ce travail est reparti en trois chapitre. Dans le premier, on présente des généralités sur la différentiabilité, dérivée directionnel, matrice hessienne, matrice définie positive, formules de Taylor, la convexité, les conditions d'optimalité et classification des problèmes d'optimisation.

Dans le deuxième chapitre, on introduit les recherches linéaires exactes et inexactes, en particulier les recherches d'Armijo, de Goldstein et de Wolfe.

Dans le dernier chapitre, on étudie la méthode de la direction conjuguée, et la méthode du gradient conjuguée dans le cas quadratique et dans le cas non quadratique puis on traite la méthode de Newton et la méthode de Quasi-Newton.

Chapitre 1

Préliminaires

1.1 Différentiabilité

Définition 1.1.1. *Soit U un ouvert de $\mathbb{R}^n$ et soit la fonction*

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

On dit que f est différentiable en $a \in U$ s'il existe une application linéaire L continue de $\mathbb{R}^n$ dans $\mathbb{R}$ telle que

$$\lim_{h \to 0} \frac{f(a+h) - f(a) - L(h)}{\|h\|} = 0$$

où $h = (h_1, \ldots, h_n)$.
Lorsque l'application L existe, elle s'appelle la différentielle de f en a, on la note $Df(a)$ donc

$$\lim_{h \to 0} \frac{f(a+h) - f(a) - Df(a)(h)}{\|h\|} = 0.$$

Remarque 1.1.1. *Si f est différentiable en tout point de U on dit que f est différentiable sur U.*

1.2 Fonction de classe C^1

Définition 1.2.1. *Une fonction f d'une variable réelle définie sur un intervalle I est dite de classe C^1 si elle est dérivable sur cet intervalle et si ses dérivées f' est continue sur cet intervalle.*

Propriétés 1.2.1.
a) Si f et g sont deux fonctions de classe C^1 sur un intervalle I alors les fonctions $f+g$ et $f\times g$ sont de classe C^1 sur I. Si de plus g ne s'annule pas sur I, alors $\frac{f}{g}$ est de classe C^1 sur I.
b) Si f est une fonction de classe C^1 sur un intervalle I et si g est une fonction de classe C^1 sur un intervalle J contenant l'intervalle $f(I)$, alors la fonction $g\circ f$ est de classeC^1 sur l'intervalle I.

Remarque 1.2.1.
La fonction f étant de classe C^1 sur l'intervalle I, elle est dérivable donc continue sur cet intervalle. L'image d'un intervalle par une fonction continue est un intervalle, on peut donc affirmer que $f(I)$ est un intervalle.

1.3 Dérivées partielles

Définition 1.3.1. *Soit $f : U \subset \mathbb{R}^n \longrightarrow \mathbb{R}$ une fonction et U un ouvert avec $a = (a_1, \cdots, a_n) \in U$. La dérivée partielle première de f par rapport à sa $i^{m\epsilon}$ variable en a notée $\frac{\partial f}{\partial x_i}(a)$ est définie par :*

$$\frac{\partial f}{\partial x_i}(a) = \lim_{h\to 0} \frac{f\left(a_1, a_2, \cdots, a_{i-1}, a_i + h, a_{i+1}, \cdots, a_n\right) - f\left(a_1, a_2, \cdots, a_n\right)}{h},$$

et en la note $\partial_i f(a)$ ou $D_i f(a)$.

1.4 Gradient

Définition 1.4.1. *On note par*

$$\nabla f(a) = \left(\frac{\partial f}{\partial x_1}, \cdots, \frac{\partial f}{\partial x_n}\right)(a).$$

Le gradient de f an point $a = (a_1, \ldots, a_n)$.
Le gradient est essentiel dans l'analyse des algorithmes d'optimisation.

Proposition 1.1. [10]*(Gradient de la composée)*
Supposons qu'on a deux ouverts $\Omega \subset \mathbb{R}^n$ et $U \subset \mathbb{R}$ et deux fonctions $f : \Omega \to \mathbb{R}$ et $g : U \to \mathbb{R}$ avec en plus $f(\Omega) \subset U$ (on peut alors définir $g \circ f : \Omega \to \mathbb{R}$).
Supposons que f et g sont de classe C^1. Alors g o f est aussi de classe C^1 avec

$$\nabla(g \circ f)(x) = g'(f(x))\nabla f(x), \quad \forall x \in \Omega.$$

1.5 Dérivée directionnelle

Définition 1.5.1.
Soient U un ouvert et $f : U \subset \mathbb{R}^n \longrightarrow \mathbb{R}$ une fonction, $a = (a_1, \cdots, a_n) \in U$ et $v = (v_1, \cdots, v_n)$ un vecteur de $\mathbb{R}^n$. La dérivée directionnelle de f suivant la direction de v au point a, on la note $D_v f(a)$ est définie par

$$D_v f(a) = \lim_{h \to 0} \frac{f(a_1 + hv_1, \cdots, a_n + hv_n) - f(a_1, \cdots, a_n)}{h}.$$

Proposition 1.2. *Soient U un ouvert et $f : U \subset \mathbb{R}^n \longrightarrow \mathbb{R}$ une fonction. Si f différentiable sur U. Alors*

$$D_v f(a) = \nabla f(a) v.$$

1.6 Matrice hessienne

Définition 1.6.1. *Soient U un ouvert de $\mathbb{R}^n$ et $f : U \subset \mathbb{R}^n \longrightarrow \mathbb{R}$ une fonction dont toutes les dérivées partielles d'ordre deux sont définies en a. Alors la matrice*

$$Hf(a) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2}(a) & \frac{\partial^2 f}{\partial x_2 \partial x_1}(a) & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_1}(a) \\ \frac{\partial^2 f}{\partial x_1 \partial x_2}(a) & \frac{\partial^2 f}{\partial x_2^2}(a) & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_2}(a) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_n}(a) & \frac{\partial^2 f}{\partial x_2 \partial x_n}(a) & \cdots & \frac{\partial^2 f}{\partial x_n^2}(a) \end{pmatrix}$$

est appelée matrice hessienne de f en a.

Théorème 1.6.1. *(Schwartz) Soit U un ouvert de $\mathbb{R}^2$ et $(x_0, y_0) \in U$ et $f : U \to \mathbb{R}$. On suppose que $\frac{\partial f}{\partial x}$, $\frac{\partial f}{\partial y}$ et $\frac{\partial^2 f}{\partial y \partial x}$ sont définies sur U et que $\frac{\partial^2 f}{\partial y \partial x}$ est continue en (x_0, y_0). Alors $\frac{\partial^2 f}{\partial x \partial y}(x_0, y_0)$ existe et également l'on a*

$$\frac{\partial^2 f}{\partial x \partial y}(x_0, y_0) = \frac{\partial^2 f}{\partial y \partial x}(x_0, y_0).$$

Exemple 1.1. *Soit $f(x_1, x_2, x_3) = e^{x_1} - x_1 x_2 x_3$.*
On a

$$\nabla f(x) = \begin{pmatrix} e^{x_1} - x_2 x_3 \\ -x_1 x_3 \\ -x_1 x_2 \end{pmatrix},$$

alors

$$H(x) = \begin{pmatrix} e^{x_1} & -x_3 & -x_2 \\ -x_3 & 0 & -x_1 \\ -x_2 & -x_1 & 0 \end{pmatrix}.$$

1.7 Formules de Taylor

1.7.1 Formule de Taylor avec reste intégrale

Théorème 1.7.1. *Soit U un ouvert de $\mathbb{R}^n$ et $a, h \in \mathbb{R}^n$ tel que $[a, a+h] \subset U$. Soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction deux fois différentiable sur U. Alors on a*

$$f(a+h) = f(a) + \sum_{i=1}^{n} h_i \frac{\partial f}{\partial x_i}(a) + \sum_{i,j=1}^{n} h_i h_j \int_0^1 (1-t) \frac{\partial^2 f}{\partial x_i \partial x_j}(a+th)dt.$$

1.7.2 Formule de Taylor-Lagrange

Théorème 1.7.2. *Soit U un ouvert de $\mathbb{R}^n$ et $a, h \in \mathbb{R}^n$ tels que $[a, a+h] \subset U$. Soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction deux fois différentiable sur U.*
La formule de Taylor -Lagrange d'ordre 2 est

$$f(a+h) = f(a) + \sum_{i=1}^{n} h_i \frac{\partial f}{\partial x_i}(a) + \frac{1}{2} \sum_{i,j=1}^{n} (h_i h_j) \frac{\partial^2 f}{\partial x_i \partial x_j}(a+\theta h) \ \ avec \ \ \theta \in]0,1[.$$

1.7.3 Formule de Taylor-Young

Théorème 1.7.3. *Soit U un ouvert de $\mathbb{R}^n$ et $a \in U$. Soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction de classe $C^2(U)$, la formule de Taylor-Young d'ordre 2 est*

$$f(a+h) = f(a) + \sum_{i=1}^{n} h_i \frac{\partial f}{\partial x_i}(a) + \frac{1}{2} \sum_{i,j=1}^{n} h_i h_j \frac{\partial^2 f}{\partial x_i \partial x_j}(a) + \varepsilon(h)\|h\|^2.$$

1.8 Convexité

1.8.1 Ensemble convexe

Définition 1.8.1. *Un ensemble $U \subset \mathbb{R}^n$ est dit convexe si*

$$\forall x, y \in U, \quad \forall t \in [0,1] \ on \ a \ tx + (1-t)y \in U.$$

Autrement dit U est convexe si pour tous points x, y de U, le signant $[x, y]$ est inclus dans U (c'est à dire, $\forall x, y \in U$ on a $[x, y] \subset U$).
Soient $x_1, x_2, \ldots, x_k \in \mathbb{R}^n$ et $\lambda_j \in \mathbb{R}$ tels que

$$\lambda_i \geq 0 \ et \ \sum_{j=1}^{k} \lambda_j = 1.$$

Alors toute expression de la forme $\sum_{j=1}^{k} \lambda_j x_j$ *s'appelle combinaison convexe des points* x_j *ou barycentre.*

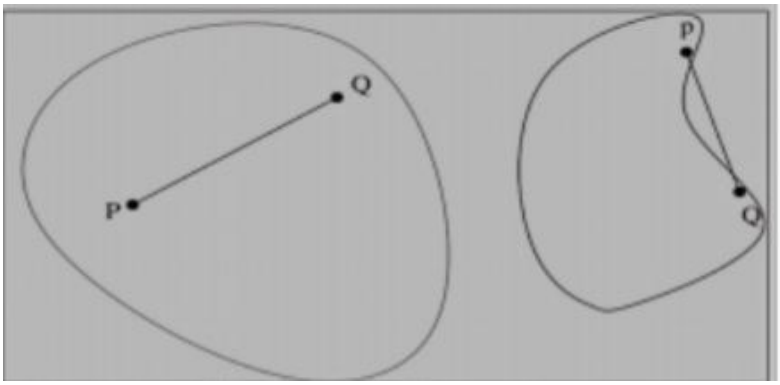

FIGURE 1.1 – Ensemble convexe et n'est pas convexe

Définition 1.8.2. *Soit* $S \subset \mathbb{R}^*$*, on appelle enveloppe convexe de* S *et on le note* $H(s)$*, l'ensemble de toutes les combinaisons convexe de* S*, en d'autre termes*

$$x \in H(S) \Leftrightarrow \exists x_1, x_2, ...x_k \in S \quad \exists \lambda_1, \lambda_2, ..., \lambda_k \in \mathbb{R}_+$$

tels que

$$\sum_{j=1}^{k} \lambda_j = 1 \quad et \quad x = \sum_{j=1}^{k} \lambda_j x_j.$$

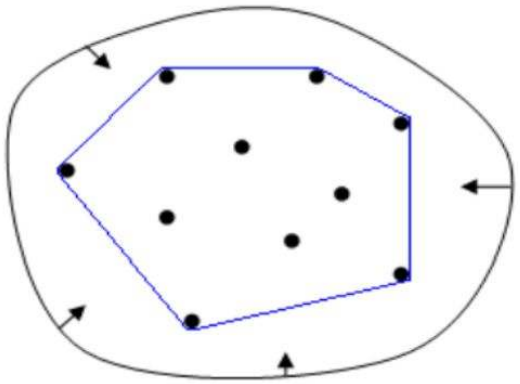

FIGURE 1.2 – Enveloppe convexe

1.8.2 Fonction convexe

Définition 1.8.3.
Soient S est un ensemble convexe non vide de $\mathbb{R}^n$ et $f : S \to \mathbb{R}$ une fonction.
a) f est dite convexe si et seulement si

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

pour tout $x_1, x_2 \in S$ et $\lambda \in [0,1]$.
b) f est dite strictement convexe si et seulement si

$$f(\lambda x_1 + (1-\lambda)x_2) < \lambda f(x_1) + (1-\lambda) f(x_2)$$

pour tout $x_1, x_2 \in S$ avec $x_1 \neq x_2$ et pour $\lambda \in]0,1[$.
c) f est dite fortement convexe ou uniformément convexe dans S si et seulement existe une constante $C > 0$ tel que

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2) - \frac{1}{2} C\lambda(1-\lambda) \|x_1 - x_2\|^2$$

pour tout $x_1, x_2 \in S$ et pour tout $\lambda \in [0,1]$.

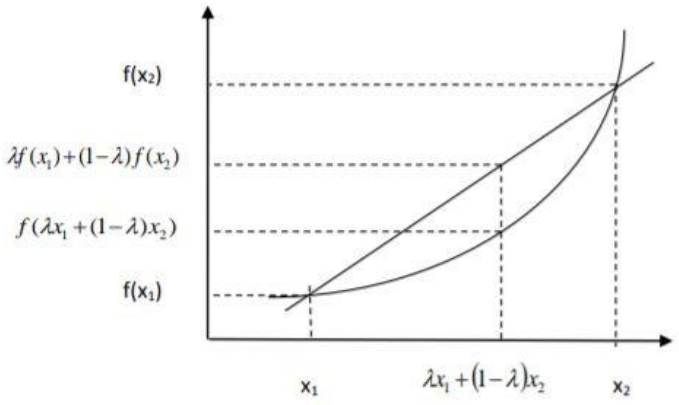

FIGURE 1.3 – Une fonction convexe

Proposition 1.3.
1) L'intersection d'une famille quelconque de convexes est convexe.
2) Le produit cartésien de deux ensembles convexes est convexe.
3) L'image directe d'un convexe par une application linéaire est convexe.

1.9 Matrice définie positive

Définition 1.9.1.
Une matrice $A \in M_n(R)$ est semi-définie positive si $\forall x \in \mathbb{R}^n,\ x^\top Ax \geq 0$.
Une matrice $A \in M_n(R)$ est définie positive si $\forall x \in \mathbb{R}^n \backslash \{0\},\ x^\top Ax > 0$.

Exemple 1.2. *On appelle matrice de Hilbert la matrice (symétrique d'ordre n) $H = (h_{i,j})$ telle que $h_{i,j} = \frac{1}{i+j-1}$, H est définie positive.*

1.10 Conditions d'optimalité

1.10.1 Conditions nécessaires d'optimalité

Théorème 1.10.1. [11] *Soit $f : \mathbb{R}^n \to \mathbb{R}$ différentiable au point $\widehat{x} \in \mathbb{R}^n$. Si $\widehat{x}$ est un minimum local de (P) alors $\nabla f(\widehat{x}) = 0$.*

Théorème 1.10.2. [11] *Soit $f : \mathbb{R}^n \to \mathbb{R}$ deux fois différentiable au point $\widehat{x} \in \mathbb{R}^n$. Si $\widehat{x}$ est un minimum local de (P) alors $\nabla(\widehat{x}) = 0$ et la matrice hessienne de f au point $\widehat{x}$, on note $H(\widehat{x})$, est semi définie positive.*

1.10.2 Conditions suffisantes d'optimalité

Théorème 1.10.3. [11] *Soit $f : \mathbb{R}^n \to \mathbb{R}$ deux fois différentiable au point $\widehat{x} \in \mathbb{R}^n$. Si $\nabla f(\widehat{x}) = 0$ et $H(\widehat{x})$ est définie positive alors $\widehat{x}$ est un minimum local strict de (P).*

1.11 Classification des problèmes d'optimisation

1.11.1 Forme générale d'un problème d'optimisation

Etant donné un ensemble $X \subset \mathbb{R}^n$ et une fonction $f : X \to \mathbb{R}$, la forme générale d'un problème d'optimisation est la suivante

$$\min\{f(x); x \in X\} \tag{1.1}$$

où $X \subset \mathbb{R}^n$ est appelé ensemble des solutions admissibles ou réalisables, $x \in \mathbb{R}^n$ appelle variable de décision, $f(x)$ s'appelle fonction objectif.

1.11.2 Problème d'optimisation sans contraintes

Si $X = \mathbb{R}^n$, le problème d'optimisation (1.1) s'appelle problème d'optimisation sans contraintes et aura donc la forme suivante

$$\min\left\{f(x); x \in \mathbb{R}^n\right\}. \tag{1.2}$$

Problème d'optimisation sans contraintes non linéaire

Le problème (1.2) est un problème d'optimisation sans contraintes non linéaire si f n'est pas affine.

Problème d'optimisation sans contraintes quadratique

Le problème (1.2) est un problème d'optimisation sans contraintes non linéaire quadratique si

$$f(x) = x^T Ax - bx$$

où A est une matrice symétrique d'ordre n, x^T est la transposé du vecteur x.

1.11.3 Problème d'optimisation avec contraintes

Soit $\mathcal{U}$ un ouvert non vide de $\mathbb{R}^n$ (le plus souvent $\mathcal{U} = \mathbb{R}^n$). Soit $f : \mathcal{U} \longrightarrow \mathbb{R}$ une application différentiable sur $\mathcal{U}$, et $\varphi_1, \varphi_2, \ldots, \varphi_p : \mathcal{U} \longrightarrow \mathbb{R}$ des applications de classe C^1 sur $\mathcal{U}$ (i.e φ_i est différentiable et $\nabla\varphi_i : x \mapsto \nabla\varphi_i(x)$ est continue ; c'est le cas en particulier lorsque φ_i est 2 fois différentiable). Soit le domaine $\mathcal{D}$

$$\mathcal{D} = \{\mathbf{x} \in \mathcal{U} \mid \varphi_i(\mathbf{x}) = 0, \forall i = 1, 2, \ldots, p\}.$$

Le problème

$$\min_{x \in \mathcal{D}} f(\mathbf{x})$$

s'appelle problème de minimisation sous contraintes.

Chapitre 2

Recherche linéaire

2.1 Direction de descente

Définition 2.1.1. [14] *Soit $f : \mathbb{R}^n \to \mathbb{R}$, un vecteur d non nul de $\mathbb{R}^n$ est dit une direction de descente, s'il existe un réel $\delta > 0$ tel que pour tout λ dans l'intervalle $]0, \delta[$ on a*

$$f(x + \lambda d) < f(x).$$

Théorème 2.1.1. [14] *Si f est différentiable en un point $x \in \mathbb{R}^n$ et $d \in \mathbb{R}^n$ telle que*

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

Alors d est une direction de descente en x.

Exemple de choix de direction de descente

Si on choisit $d_k = -\nabla f(x_k)$ et si $\nabla f(x_k) \neq 0$ on obtient la méthode du gradient. La méthode de Newton correspond à

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

Bien sur $-\nabla f(x_k)$ est une direction de descente

$$(\nabla f(x_k)^T d_k = -\nabla f(x_k)^T \nabla f(x_k) = -\|\nabla f(x_k)\|^2 < 0).$$

Pour la deuxième direction, si la matrice hessienne $H(x_k)$ est définie positive alors

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

est aussi une direction de descente.

Algorithme à direction de descente

Etape 0 :(initialisation)
On suppose qu'au début de l'itération k, on dispose d'un itéré $x_k \in \mathbb{R}^n$.
Etape l :
Test d'arrét : si $\|\nabla f(x_k)\| \simeq 0$, arrêt de l'algorithme ;
Etape 2 :
Choix d'une direction de descente $d_k \in \mathbb{R}^n$;
Etape 3 :
Recherche linéaire : déterminer un pas $\lambda_k > 0$ le long de d_k de manière à "faire décroitre f suffisamment" ;
Etape 4 :
Si la recherche linéaire réussie $x_{k+1} = x_k + \lambda_k d_k$
remplacer k par $k + 1$ et aller à l'étape 1.

2.2 Schémas générale des algorithmes d'optimisation sans contraintes

On suppose que d_k soit une direction de descente au point x_k. Ceci nous permet de considérer le point x_{k+1} successeur de x_k, de la manière suivante

$$x_{k+1} = x_k + \lambda_k d_k, \quad \lambda_k \in]0, +\delta[.$$

Vu la définition de direction de descente, on est assuré que

$$f(x_{k+1}) = f(x_k + \lambda_k d_k) < f(x_k).$$

Un bon choix de d_k et de λ_k permet ainsi de construire une multitude d'algorithmes d'optimisation.

2.3 Choix optimal du pas λ_k

Nous choisissons λ_k de façon optimale, c'est à dire qu'en partant du point x_k et la direction d_k pour f décroit le mieux possible au point

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

Ceci est possible si on impose à λ_k de vérifier la condition suivante

$$\forall \lambda \in]0, +\infty[, \quad f(x_k + \lambda_k d_k) \leq f(x_k + \lambda d_k). \tag{2.1}$$

Notons $\varphi_k(\lambda)$ la fonction d'une variable réelle suivante

$$\varphi_k(\lambda) = f(x_k + \lambda d_k), \quad \lambda \in]0, +\infty[. \tag{2.2}$$

(2.1) et (2.2) donnent

$$\varphi_k(\lambda_k) \leq \varphi_k(\lambda), \quad \lambda \in]0, +\infty[\tag{2.3}$$

ou encore λ_k est solution optimale c'est à dire

$$\varphi_k(\lambda_k) = min\{\varphi_k(\lambda) : \lambda \in]0, +\infty[\}. \tag{2.4}$$

Trouvez λ_k vérifiant (2.4) s'appelle recherche linéaire exacte. Ce travail à effectue à chaque itération k. Donc à chaque itération k, on doit résoudre problème d'optimisation dans $\mathbb{R}$.

Exemple de choix du pas λ_k

On choisit en générale λ_k de façon optimale, c'est à dire que λ_k doit vérifier

$$f(x_k + \lambda_k d_k) \leq f(x + \lambda d_k), \quad \lambda \in [0, +\infty[.$$

En d'autre termes on est ramené à étudier à chaque itération un problème de minimisation d'une variable réelle. C'est ce qu'on appelle recherche linéaire.

2.4 Recherche linéaire

L'algorithme de recherche linéaire vise à résoudre le problème d'optimisation sans contraintes (P), où l'objectif est de minimiser la fonction $f(x)$ avec x appartenant à $\mathbb{R}^n$. L'algorithme suit le schéma

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

Où λ_k est la solution optimale du problème d'optimisation suivant :

$$\min_{\lambda > 0} f\left(x_k + \lambda d_k\right).$$

Cela revient à trouver λ_k satisfaisant l'inégalité

$$f\left(x_k + \lambda_k d_k\right) \leq f\left(x_k + \lambda d_k\right)$$

pour tout $\lambda > 0$.
En d'autres termes, l'objectif est de minimiser la fonction

$$h_k(\lambda) = f(x_k + \lambda d_k).$$

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

Dans ce chapitre nous allons décrire les différentes manières de déterminer un pas $\lambda_k > 0$ le long d'une direction de descente d_k. C'est ce que l'on appelle faire de la recherche linéaire.
Il existe deux grandes classes de méthodes qui s'intéressent a' l'optimisation unidimensionnelle :
- Les recherches linéaires exactes.
- Les recherches linéaires inexactes.

2.4.1 Objectifs de la recherche linéaire

il s'agit de réaliser deux objectifs :

Premier objectif

Le premier objectif de la recherche linéaire est de réduire suffisamment la valeur de la fonction f. Cela se traduit généralement par la réalisation d'une inégalité de la forme :

$$f(x_k + \lambda d_k) \leq f(x_k) + "un\ terme\ negatif"$$

Le terme négatif, note ν_k, joue un rôle crucial dans la convergence de l'algorithme utilisant cette recherche linéaire.
Si $f(x_k)$ est minorée (c'est-à-dire qu'il existe une constante C telle que $f(x_k) \geq C$ pour tout k), alors ce terme négatif tend nécessairement vers zéro : $\nu_k \longrightarrow 0$. C'est souvent à partir de la convergence vers zéro de cette suite que l'on parvient à montrer que le gradient lui-même doit tendre vers zéro.
Le terme négatif devra prendre une forme bien particulière si on veut pouvoir en tirer de l'information. En particulier, il ne suffit pas d'imposer

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

Second objectif

Consiste d'empêcher le pas $\lambda_k > 0$ 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 du pas $\lambda_k > 0$ arbitrairement petit. Or ceci peut entraîner une "fausse convergence", c'est-à-dire la convergence des itérés vers un point non stationnaire. Si on prend :

$$0 < \lambda_k \leq \frac{\varepsilon}{2^k \|d_k\|}$$

la suite $\{x_k\}$ est de Cauchy, puisque pour $1 < l < k$ on a :

$$\|x_k - x_l\| = \left\| \sum_{i=1}^{i=k-1} \lambda_i d_i \right\| \leq \sum_{i=1}^{i=k-1} \frac{\varepsilon}{2^i} \to 0, \quad \text{lorsque } l \to \infty.$$

Donc $\{x_k\}$ converge, disons vers un point x^*. En prenant $l = 1$ et $k \to \infty$ dans l'estimation ci-dessus, on voit que $x^* \in \bar{B}(x_1, \varepsilon)$ et donc x^* ne saurait être solution s'il n'y a pas de solution dans $\bar{B}(x_1, \varepsilon)$. On a donc arbitrairement forcer la convergence de $\{x_k\}$ en prenant des pas très petits. Pour simplifier les notations, on définit : $h_k : \mathbb{R}_+ \to \mathbb{R}$ comme suit :

$$\lambda \mapsto \quad h_k(\lambda) = f(x_k + \lambda d_k).$$

2.4.2 Intervalle de sécurité

Dans la plupart des algorithmes d'optimisation modernes, on ne fait jamais de recherche linéaire exacte, car trouver λ_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 λ^* qui assure une décroissance suffisante de f.
Cela conduit à la notion d'intervalle de sécurité.

Définition 2.4.1. *On dit que $[\lambda_g, \lambda_d]$ est un intervalle de sécurité s'il permet de classer les valeurs de λ de la façon suivante :*
- Si $\lambda < \lambda_g$ alors λ est considéré trop petit,
- Si $\lambda_d \geq \lambda \geq \lambda_g$ alors λ est satisfaisant,
- Si $\lambda > \lambda_d$ alors 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 λ_g et λ_d.

Algorithme de base

Etape 0 : (initialisation)
$\lambda_g = \lambda_d = 0$, choisir $\lambda_1 > 0$, poser $k = 1$ et aller à l'étape 1.
Etape 1 :
Si λ_k convient, poser $\lambda^* = \lambda_k$ et on s'arrête.
Si λ_k est trop petit on prend $\lambda_{g,k+1} = \lambda_k, \lambda_g = \lambda_d$ et on va à l'étape 2.
Si λ_k est trop on prend $\lambda_{d,k+1} = \lambda_k, \lambda_d = \lambda_g$ et on va à l'étape 2.
Etape 2 :
Si $\lambda_{d,k+1} = 0$ déterminer $\lambda_{k+1} \in]\lambda_{g,k+1}, +\infty[$.
Si $\lambda_{d,k+1} \neq 0$ déterminer $\lambda_{k+1} \in]\lambda_{g,k+1}, \lambda_{d,k+1}[$.
Remplacer k par $k+1$ et aller à l'étape 1 .

2.5 Recherches linéaires exactes

Les règles de recherche linéaire exactes, telles que la règle de Cauchy et la règle de Curry, sont utilisées pour déterminer la meilleure ètape dans la minimisation d'une fonction objectif f.

La règle de Cauchy consiste à minimiser le critère le long de d_k en résolvant le problème de minimisation

$$\min h_k(\lambda), \quad \lambda \geq 0.$$

Le pas optimal résultant est appelé pas de Cauchy.

D'autre part, la règle de Curry cherche le plus petit point stationnaire de h_k qui fait décroitre la fonction, déterminant le pas de Curry comme solution du problème

$$\lambda_k = \inf \{\lambda \geq 0, h'_k(\lambda) < h_k(0)\}.$$

Cette approche est parfois qualifiée de recherche linéaire exacte.

En d'autres termes, la recherche linéaire exacte vise à trouver le pas optimal dans la direction d_k en utilisant des règles spécifiques telles que la règle de Cauchy ou la règle de Curry, ce qui conduit à la minimisation de la fonction objectif.

Remarque 2.5.1. *Ces deux règles ne sont utilisées que dans des cas particuliers, par exemple lorsque h_k est quadratique.*
Le mot exact prend sa signification dans le fait que si f est quadratique la solution de la recherche linéaire s'obtient de façon exacte et dans un nombre fini d'itérations.

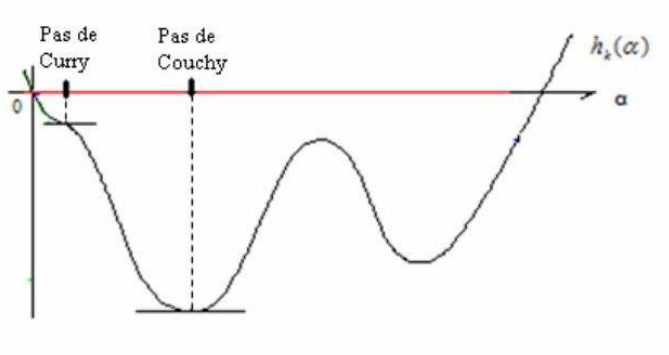

FIGURE 2.1

2.5.1 Les inconvénients des recherches linéaires exactes

Pour une fonction non linéaire arbitraire,
- Il peut ne pas exister de pas de Cauchy ou de Curry.
- La détermination de ces pas demande en général beaucoup d'observations et ne peut de toutes façons pas être faite avec précision.
- 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 (recherches linéaires inexactes), moins gourmandes en temps de calcul.

Au lieu de demander à λ_k de minimiser la fonction h_k, on préfère imposer des conditions moins restrictives, plus faciles à vérifier, qui permettent toute fois de contribuer à la convergence des algorithmes. En particulier, il n'y aura plus un unique pas (ou quelques pas) vérifiant ces conditions mais tout un intervalle de pas (ou plusieurs intervalles), ce qui rendra d'ailleurs leur recherche plus aisée. C'est ce que l'on fait avec les règles d'Armijo, de Goldstein et de Wolfe décrites dans la prochaine section.

2.6 Recherches linéaires inexactes

II existe plusieurs raisons pour lesquelles la méthode de recherche linéaire inexacte est souvent préférée dans les méthodes principales multidimensionnelles.

Sur une itération k de la dernière méthode nous avons l'itération courante $x_k \in \mathbb{R}^n$ et la direction de recherche $d_k \in \mathbb{R}^n$ qui est une direction de descente

pour la fonction objectif $f : \mathbb{R}^n \to \mathbb{R}$, on a :

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

Le but est de réduire "de façon importante" la valeur de l'objectif par un pas $x_k \longmapsto x_{k+1} = x_k + \lambda_k d_k$ de x_k dans la direction d_k. Pour cela de nombreux mathématiciens (Armijo, Goldstein, Wolfe, Albaali, Lemaréchal, Fletcher...) ont élaboré plusieurs règles (tests).

L'objectif de cette section consiste à présenter les principaux tests.
D'abord présentons le schéma d'une recherche linéaire inexacte. Elles reviennent à déterminer, par tâtonnement un intervalle $[\lambda_g, \lambda_d]$, où $\lambda^* \in [\lambda_g, \lambda_d]$, dans lequel :

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

Algorithme :(schéma général des recherches linéaires inexactes)

Etape 0 : (initialisation)
$\lambda_{g,1} = \lambda_{d,1} = 0$, choisir $\lambda_1 > 0$, poser $k = 1$ et aller à l'étape 1.
Etape 1 :
Si λ_k est satisfaisant (suivant un certain critère) : STOP $(\lambda^* = \lambda_k)$.

Si λ_k est trop petit (suivant un certain critère) : nouvel intervalle :
$[\lambda_{g,k+1} = \lambda_k\,,\ \lambda_{d,k+1} = \lambda_d]$ et aller à l'étape 2.

Si λ_k est trop grand (suivant un certain critère) : nouvel intervalle :
$[\ \lambda_{g,k+1} = \lambda_g, \lambda_{d,k+1} = \ \lambda_k]$et aller à l'étape 2.

Etape 2 :
Si $\lambda_{d,k+1} = 0$ déterminer $\lambda_{k+1} \in]\,\lambda_{g,k+1}, +\infty[$.

Si $\lambda_{d,k+1} \neq 0$ déterminer $\lambda_{k+1} \in]\,\lambda_{g,k+1}, \lambda_{d,k+1}[$ remplacer k par $k+1$ et aller à l'étape 1.
Il nous reste donc à décider selon quel(s) critère(s) λ est trop petit ou trop grand ou satisfaisant.

2.7 Recherche linéaire d'Armijo

2.7.1 Principe de la méthode

Il ne suffit pas de choisir à chaque itération k, λ_k tel que

$$f(x_k + \lambda_k d_k) < f(x_k). \tag{2.5}$$

Pour que la suite $\{x_k\}$ converge vers la solution optimale ou vers un point stationnaire. En effet si la fonction décroit très peu en passant du pointx_k vers le point

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

La suite $\{x_k\}$ diverge ou peut converger vers un point qui n'a rien à voir avec la solution optimale.
Pour éviter cet inconvénient, il faut imposer une condition caractérisant la notion de descente suffisante de la fonction. Il faut choisir λ_k de sorte que la fonction f décroit de suffisante en passant du point x_k au point passant du point

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

L'idée est que la diminution jugée suffisante soit proportionnelle si la taille du pas λ_k. Ainsi plus le pas sera grand, plus la diminution de la fonction f sera grande. Donc si $\gamma > 0$, on désire que

$$f(x_k) - f(x_k + \lambda_k d_k) \geq \lambda_k \gamma$$

ou encore

$$f(x_k + \lambda_k d_k) \leq f(x_k) - \lambda_k \gamma.$$

Le facteur γ ne peut pas être choisi de façon arbitraire. Surtout il doit varier d'une itération a l'autre.
Il est plus approprié de définir γ en fonction de la pente de la fonction au point x_k dans la direction d_k. En l'occurence, nous choisirons

$$\gamma = -\beta \nabla f(x_k)^T d_k$$
$$0 < \beta < 1.$$

Définition 2.7.1. ***Diminution suffisante(Condition d'Armijo)***

Soient $f : \mathbb{R}^n \to \mathbb{R}$ une fonction différentiable, un point $x_k \in \mathbb{R}^n$ et une direction (de descente) $d_k \in \mathbb{R}^n$ telle que

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

On dit que le pas $\lambda_k > 0$ vérifie la condition d'Armijo, si on a

$$f(x_k + \lambda_k d_k) \leq f(x_k) + \lambda_k \beta_1 \nabla f(x_k)^T d_k, \;\; 0 < \beta_1 < 1. \tag{2.6}$$

Test d'Armijo, Si

$$\varphi_k(\lambda) \leq \varphi_k(0) + \rho \phi_n'(0)\lambda \tag{2.7}$$

autrement dit,

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

alors λ convient. Si

$$\varphi_k(\lambda) > \varphi_k(0) + \rho h_k'(0)\lambda \tag{2.8}$$

autrement dit

$$f(x_k + \lambda d_k) > f(x_k) + \rho\lambda \nabla^T f(x_k) d_k$$

alors λ est trop grand.

2.7.2 Algorithme d'Armijo

Etape 0 : (initialisation)
$\lambda_{g,1} = \lambda_{d,1} = 0$ choisir $\lambda_1 > 0, \;\; \rho \in]0, 1[$ poser $k = 1$ et aller à l'étape 1.
Etape 1 :
Si $\varphi_k(\lambda_k) \leq \varphi_k(0) + \rho\varphi_k'(0)\lambda_k$: STOP $(\lambda^* = \lambda_k)$.
Si $\varphi_k(\lambda_k) > \varphi_k(0) + \rho\varphi_k'(0)\lambda_k$ alors $\lambda_{d,k+1} = \lambda_d, \;\; \lambda_{g,k+1} = \lambda_k$
et aller à l'étape 2.
Etape 2 :
Si $\lambda_{d,k+1} = 0$ déterminer $\alpha_{k+1} \in]\lambda_{g,k+1}, +\infty[$.
Si $\lambda_{d,k+1} \neq 0$ déterminer $\lambda_{k+1} \in]\lambda_{g,k+1}, \lambda_{d,k+1}[$ remplacer k par $k+1$ et aller à l'étape 1.

2.8 Recherche linéaire de Goldstein

2.8.1 Principe de la méthode

Supposons que l'itération k, on ait le point $x_k \in \mathbb{R}^n$, le vecteur de descente $d_k \in \mathbb{R}^n (\nabla f(x_k)^T d_k < 0)$.
On cherche le pas $\lambda_k > 0$ de Goldstein vérifiant les deux conditions suivantes

$$f(x_k + \lambda_k d_k) \leq f(x_k) + \rho\lambda_k \nabla f(x_k)^T d_k, \quad \rho \in]0, \frac{1}{2}[\quad (Goldstein\ 1)$$

$$et \quad f(x_k + \lambda_k d_k) \geq f(x_k) + (1-\rho)\lambda_k \nabla f(x_k)^T d_k, \quad \rho \in]0, \frac{1}{2}[. \quad (Goldstein\ 2).$$

Si on note

$$\varphi_k(\lambda) = f(x_k + \lambda d_k)$$

alors (Goldstein 1) et (Goldstein 2) deviennent

$$\varphi_k(\lambda_k) \leq \varphi_k(0) + \rho\lambda_k \varphi_k'(0)$$

et

$$\varphi_k(\lambda_k) \geq \varphi_k(0) + (1-\rho)\lambda_k \varphi_k'(0)$$

(Goldstein 1) assure une décroissance suffisante de f, les pas λ_k trop petits sont nocifs pour la convergence.

La suite $\{x_k\}$ pourrait converger prématurément vers un point x qui n'a rien à voir avec le problème d'optimisation.

2.8.2 Algorithme de Goldstein

Etape 1 : (Initialisation)
Choisir $\lambda_0 \in [0, 10^{100}]$ et $\rho \in]0, 1[$. Poser $a_0 = 0, b_0 = 10^{100}$. Poser $k = 0$ et aller à Etape2.

Etape 2 : (T est Goldstein 1)
Iteration k on a $[a_k, b_k]$ et λ_k calculez $\varphi_k(\lambda_k)$ Si $\varphi_k(\lambda_k) \leq \varphi_k(0) + p\lambda_k\varphi_k'(0)$ allez à Etape3. Sinon
Poser $b_{k+1} = \lambda_k, a_{k+1} = a_k$, et allez à Etape 4.

Etape 3 : (T est Goldstein 2)
Si $\varphi_k(a_k) \geq \varphi_k(0) + (1-p)a_k\varphi_k'(0)$ stop. $\lambda^* = \lambda_k$. Sinon poser
$a_{k+1} = a_k,\ b_{k+1} = b_k$ et allez à Etape 4.

Etape 4 :
Poser $\lambda_{k+1} = \frac{a_{k+1}+b_{k+1}}{2}$. Poser $k = k+1$ et allez à Etape 2.

2.9 Recherche linéaire de Wolfe

2.9.1 Algorithme de Wolfe

Etape 0 : (initialisation)
$\lambda_{g,1} = \lambda_{d,1} = 0$ choisir $\lambda_1 > 0, \rho \in [0,1], \sigma \in]\rho, 1|$ pose $k = 1$ et aller à Etape 1.
Etape 1 :
Si $\varphi_k(\lambda_k) \leq \varphi_k(0) + \rho\varphi'_k(0)\lambda_k$ et $\varphi'_k(\lambda) \geq \varphi'_k(0)$: $\quad STOP(\lambda^* = \lambda_k)$.
Si $\varphi_k(\lambda_k) > \varphi_k(0) + \rho\varphi'_k(0)\lambda_k$, alors $\lambda_{d,k+1} = \lambda_k, \lambda_{g,k+1} = \lambda_{g,k}$ aller à Etape 2.
Si $\varphi'_k(\lambda) < \sigma\varphi'_k(0)$ alors $\lambda_{d,k+1} = \lambda_{d,k}\lambda_{g,k+1} = \lambda_k$ et aller à Etape 2.
Etape 2 :
Si $\lambda_{d,k+1} = 0$ déterminer $\lambda_{k+1} \in]\lambda_{g,k+1}, +\infty[$. $Si\ \ \lambda_{d,k+1} \neq 0$
déterminer $\lambda_{k+1} \in\ ]\lambda_{g,k+1}, \lambda_{d,k+1}[$.

2.9.2 Recherche linéaire de Wolfe forte

Dans certains algorithmes d' optimisation, comme la méthode du gradient conjugué non linéaire, il peut être nécessaire d'utiliser une condition plus restrictive que la deuxième condition (2.8) est remplacée par

$$|\nabla^T f(x_k + \lambda_k d_k)| \leq \sigma|\nabla^T f(x_k)d_k| = -\nabla^T f(x_k)d_k,$$

on aura donc les conditions de Wolfe forte

$$f(x_k + \lambda_k d_k) \leq f(x_k) + \rho\lambda_k \nabla^T f(x_k)d_k \tag{2.9}$$

$$|\nabla f(x_k + \lambda_k d_k)^T d_k| \leq -\sigma\nabla^T f(x_k)d_k \tag{2.10}$$

autrement dit

$$\varphi_k(\lambda_k) \leq \varphi_k(0) + \rho\lambda_k\varphi'_k(0), \ \ 0 < \rho < \frac{1}{2} \ \ (Wolfe\ forte\ \ 1) \tag{2.11}$$

$$|\varphi'_k(\lambda_k)| \leq -\sigma\varphi'_k(0), \ \ 0 < \sigma < 1, \ \sigma > \rho \ \ (Wolfe\ forte\ \ 2) \tag{2.12}$$

2.9.3 Recherche linéaire de Wolfe faible

Etant donnée deux nombres réels ρ et o tel que $0 < \rho < \sigma < 1$, λ_k est acceptable par la recherche linéaire inexacte de Wolfe, si λ_k vérifie les conditions suivantes

$$f(x_k + \lambda_k d_k) \leq f(x_k) + \rho\lambda_k \nabla^T f(x_k)d_k, \ telque\ \ 0 < \rho < \frac{1}{2} \ \ (Wolfe\ \ 1) \tag{2.13}$$

$$\nabla^T f(x_k + \lambda_k d_k) d_k \geq \sigma \nabla^T f(x_k) d_k, telque\ 0 < \sigma < 1,\ \sigma > \rho\ \ (Wolfe\ \ 2) \quad (2.14)$$

autrement dit

$$\varphi_k(\lambda_k) \leq \varphi_k(0) + \rho \lambda_k \varphi'_k(0),\ 0 < \rho < \frac{1}{2} \quad (2.15)$$

$$\varphi'_k(\lambda_k) \geq \sigma \varphi'_k(0),\ 0 < \sigma < 1,\ \sigma > \rho \quad (2.16)$$

Chapitre 3

Certaines méthodes d'optimisation

3.1 Méthode à directions conjuguées

Définition 3.1.1. *Soit Q une matrice symétrique $n \times n$, définie positive. On dit que deux vecteurs x et y de $\mathbb{R}^n$ sont Qconjugués s'ils vérifient*

$$x^T Q y = 0$$

Théorème 3.1.1. [11] *Soit Q une matrice (n, n) symétrique et définie positive. Si les directions $y_0, y_1, ..., y_k$ avec $k \leq n-1$, sont non nulles et Q conjuguées, alors elles sont linéairement indépendantes.*

Remarque 3.1.1. *Il n'existe pas de caractère unique pour les vecteurs y_0, y_1, y_2 conjugués.*

Algorithme des directions conjuguées

Soit Q est une matrice (n, n) symétrique et définie positive et $b \in \mathbb{R}^n$. Considérons le problème de minimisation quadratique sans contraintes (PQSC) suivant :

$$\min_{x \in \mathbb{R}^n} \left\{ \frac{1}{2} x^T Q x - b^T x \right\} \quad (PQSC).$$

Algorithme

a)Initialisation

On se donne $x_0 \in \mathbb{R}^n$ quelconque et $[d_0, d_1, \ldots . d_{n-1}]Q$ conjuguées. Poser $k = 0$ et allez etape principale.

b) Etape principale

Pour $k \geq 0$. Calculer

$$g_k = \nabla f(x_k) = Q x_k - b.$$

Si $g_k = 0$. Stop. Sinon calculer

$$\alpha_k = -\frac{g_k^T d_k}{d_k^T Q d_k}$$

et

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

Poser $k = k + 1$ et allez à b) Etape principale.

Théorème 3.1.2. [11] *Partant d'un point initial $x_0 \in \mathbb{R}^n$ quelconque, l'algorithme des directions conjuguées précédent converge vers la solution optimale unique $\widehat{x}$ du problème (PQSC) dans n itérations, c'est à dire qu'on a*

$$x_n = \widehat{x} \text{ et } Qx_n = Q\widehat{x} = b. \tag{3.1}$$

Preuve 3.1. *Considérons le vecteur $\widehat{x} - x_0 \in \mathbb{R}^n$ D'autre part $\{d_0, \ldots\ldots d_{n-1}\}$ est un système libre et contient n vecteurs. Donc $\{d_0, \ldots\ldots d_{n-1}\}$ est une base de R^n Par conséquent il existe $\beta_{0,\ldots}, \beta_{n-1}$ scalaires réels tels que :*

$$\widehat{x} - x_0 = \beta_0 d_0 + \ldots\ldots + \beta_{n-1} d_{n-1}. \tag{3.2}$$

Multiplions les 2 membres de(3.2) par $d_k^T Q$, $0 \leq k \leq n-1$, on obtient

$$d_k^T Q(\widehat{x} - x_0) = \beta_k d_k^T Q d_k. \tag{3.3}$$

Donc

$$\beta_k = \frac{d_k^T Q(\widehat{x} - x_0)}{d_k^T Q d_k}. \tag{3.4}$$

Montrons maintenant que le numérateur de (3.4) est égal à $-d_k^T g_k$. En effet, d'après l'algorithme des directions conjuguées

$$\begin{aligned} x_k &= x_{k-1} + \alpha_{k-1} d_{k-1} \\ &= x_{k-2} + \alpha_{k-2} d_{k-2} + \alpha_{k-1} d_{k-1} \\ &= \ldots\ldots \\ &= x_0 + \alpha_0 d_0 + \ldots + \alpha_{k-1} d_{k-1} \end{aligned}$$

par conséquent

$$x_k - x_0 = \alpha_0 d_0 + \ldots + \alpha_{k-1} d_{k-1}, \quad k = 1, 2 \ldots\ldots n-1 \tag{3.5}$$

écrivons maintenant : $\widehat{x} - x_0$ sous la forme

$$\widehat{x} - x_0 = (\widehat{x} - x_k) + (x_k - x_0). \tag{3.6}$$

Multiplions (3.6) par $d_k^T Q$, on obtient

$$d_k^T Q(\widehat{x} - x_0) = d_k^T Q(\widehat{x} - x_k) \tag{3.7}$$

car

$$d_k^T Q(\widehat{x} - x_0) = \alpha_0 d_k^T Q d_0 + ... + \alpha_{k-1} d_k^T Q d_{k-1} = 0. \tag{3.8}$$

Calculons maintenant $d_k^T Q(\widehat{x} - x_k)$. On a

$$d_k^T Q\left(\widehat{x} - x_k\right) = d_k^T\left[Q\widehat{x} - Qx_k\right] = d_k^T\left[b - Qx_k\right] = -d_k^T\left(Qx_k - b\right) = -d_k^T g_k \tag{3.9}$$

(3.9) et (3.4)impliquent donc

$$\beta_k = -\frac{d_k g_k}{d_k Q d_k} = \alpha_k. \tag{3.10}$$

En conclusion on a

$$\widehat{x} - x_0 = \beta_0 d_0 + \ldots\ldots\ldots\ldots + \beta_{n-1} d_{n-1}$$

et

$$x_n - x_0 = \alpha_0 d_0 + \ldots\ldots\ldots\ldots + \alpha_{n-1} d_{n-1}.$$

Or

$$\alpha_k = \beta_k \quad k = 0, \ldots\ldots\ldots 1.$$

Donc

$$\widehat{x} - x_0 = x_n - x_0.$$

Ce qui implique

$$\widehat{x} = x_n.$$

□

Remarque 3.1.2. *Si on démarre du point x_1, alors la solution optimale est atteinte au point x_{n+1}, c'est à dire qu'on aura*

$$\widehat{x} = x_{n+1}.$$

Théorème 3.1.3. [11] *Soit Q une matrice (n, n)symétrique et définie positive et $b \in \mathbb{R}^n$ et $f(x) = \frac{1}{2} x^T Q x - b^T x$. Considérons la suite $\{x_k\}_{k=1,2,\ldots}$ de la manière suivante. On démarre par $x_1 \in \mathbb{R}^n$. A litération k, le successeur x_{k+1} de x_k est donné par la relation suivante :*

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

où $d_k \in \mathbb{R}^n$ est une direction de recherche et $\alpha_k \in \mathbb{R}^+$ est le pas de recherche obtenu par une recherche linéaire exacte, α_k vérifie

$$f(x_k + \alpha_k d_k) = \min_{\alpha>0} f(x_k + \alpha d_k). \tag{3.11}$$

Notons

$$g_{k+1} = \nabla f(x_{k+1}) = Qx_{k+1} - b. \tag{3.12}$$

Alors

$$g_{k+1} = g_k + \alpha_k Q d_k \tag{3.13}$$

et

$$g_{k+1} d_k = 0, \quad k = 0, 1, \ldots, n-1. \tag{3.14}$$

Preuve 3.2. *en effet. On a*

$$Q(x_{k+1} - x_k) = (Qx_{k+1} - b) - (Qx_k - b) = g_{k+1} - g_k. \tag{3.15}$$

D'un autre côté, on a

$$x_{k+1} = x_k + \alpha_k d_k. \tag{3.16}$$

Donc

$$(x_{k+1} - x_k) = \alpha_k d_k \tag{3.17}$$

(3.17) et (3.15) impliquent

$$g_{k+1} - g_k = \alpha_k Q d_k. \tag{3.18}$$

D'autre part et comme on l'a vu dans la preuve du théorème 2, considérons la fonction suivante

$$\varphi :]0, +\infty[\to \mathbb{R} : \alpha \to \varphi(\alpha) = f(x_k + \alpha d_k) \tag{3.19}$$

et le problème d'optimisation suivant

$$\varphi(\alpha_k) = \min\{\varphi(\alpha) : \alpha \in]0, +\infty[\}. \tag{3.20}$$

Q est définie positive.
Alors $f(x) = \frac{1}{2} x^T Q x - b^T x$ est strictement convexe sur $\mathbb{R}^n$.
Par conséquent $\varphi\alpha = f(x_k + \alpha d_k)$ est strictement convexe sur $]0, +\infty[$ qui est un ouvert convexe. Donc α_k solution optimale de(3.20) si et seulement si α_k vérifie $\varphi'(\alpha_k) = 0$.
Or

$$\varphi'(\alpha) = \nabla f(x_k + \alpha d_k)^T d_k. \tag{3.21}$$

Donc

$$\varphi'(\alpha_k) = \nabla f(x_k + \alpha_k d_k)^T d_k = \nabla f(x_{k+1})^T d_k = g_{k+1}^T d_k. \tag{3.22}$$

Par conséquent

$$\varphi'(\alpha_k) = 0 \Longleftrightarrow g_{k+1}^T d_k = 0. \tag{3.23}$$

□

Une des propriétés fondamentales de la méthode des direction conjuguées est théorème suivant :

Théorème 3.1.4. [11] *Dans la méthode des direction conjuguées, on a*

$$g_{k+1}^T d_i = 0, \quad k = 0, 1 \ldots\ldots .., n-1, \quad i = 0, \ldots\ldots k. \tag{3.24}$$

Preuve 3.3. *Remarquons d'abord que pour k fixé, la relation (3.24) contient les $k+1$ égalités suivantes*

$$\begin{cases} g_{k+1}^T d_0 = 0 \\ g_{k+1}^T d_1 = 0 \\ \ldots\ldots\ldots \\ g_{k+1}^T d_k = 0 \end{cases} \tag{3.25}$$

$$\begin{cases} k = 0 \text{ (1 relation)} : g_1 d_0 = 0 \\ k = 1 \text{ (2 relations)} : g_2 d_0 = 0, \quad g_2 d_1 = 0 \\ k = 2 (3 \text{ relations)} : g_3 d_0 = 0, \quad g_3 d_1 = 0, \quad g_3 d_2 = 0 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ k = n-1 (n \text{ relations)} : g_n d_0 = 0, \quad g_n d_1 = 0, \ldots, \quad g_n d_{n-1} = 0 \end{cases} \tag{3.26}$$

La preuve se fait par récurrence. Pour $k = 0$, la relation (3.24) est vraie. En effet, d'après le théorème 3 et en prenant $k = 0$ dans la relation(3.14), on obtient

$$g_1 d_0 = 0. \tag{3.27}$$

Supposons que la relation (3.24) soit vraie à l'ordre $k-1$, c'est à dire que nous avons

$$g_k^T d_i = 0 \quad i = 0, 1, \ldots\ldots k-1. \tag{3.28}$$

Montrons que la relation (3.24) reste vraie à l'ordre k, c'est à dire, montrons que ;

$$g_{k+1}^T d_i = 0, \quad i = 0, 1, \ldots . k. \tag{3.29}$$

En effet. On a en utilisant la relation (3.13)

$$g_{k+1}^T d_i = \left(g_k + \alpha_k Q d_k\right)^T d_i = g_k^T d_i + \alpha_k d_k^T Q d_i, \quad i = 0, \ldots, k. \tag{3.30}$$

a) $0 \leq i \leq k$ -1

La relation (3.28) implique que

$$g_k^T d_i = 0. \tag{3.31}$$

D'autre part, et d'après la Q conjugaison des vecteurs $d_0, d_1, \ldots, d_{n-1}$, on a

$$d_k^T Q d_i = 0 \tag{3.32}$$

(3.31) et (3.32) impliquent (3.30) pour $0 \leq i \leq k$ -1.

b) i=k

Reste a prouver la relation (3.24) pour $i = k$, c'est à dire montrons que

$$g_{k+1}^T d_k = 0. \tag{3.33}$$

Cette relation a été prouvée dans le théorème 3 (voir relation (3.14)). En conclusion la relation (3.24) a été démontrée pour tout $k \leq n-1$. □

3.2 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 suivant :

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

où f est régulière (continûment différentiable).

3.2.1 Algorithme du gradient conjugué

Supposons Q est une matrice symétrique définie positive de taille n et $b \in \mathbb{R}^n$. Considérons le problème de minimisation quadratique sans contraintes (PQSC) suivant :

$$\min_{x \in \mathbb{R}^n} \left\{ \frac{1}{2} x^T Q x - b^T x \right\}, \quad (PQSC).$$

Algorithme

Initialisation

On choisit $x_0 \in \mathbb{R}^n$ arbitraire et $d_0, d_1, \cdots d_{n-1}Q$ conjugués. On pose $k = 0$ et on passe à Étape principale.

Étape principale

Pour $k \geq 0$. Calculez

$$g_k = \nabla f(x_k) = Qx_k - b.$$

Si $g_k = 0$, stop. Sinon calcule

$$\alpha_k = -\frac{g_k^T d_k}{d_k^T Q d_k},$$

et

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

Poser $k = k + 1$ et allez à Etape principale.

Théorème 3.2.1. [11] *Il est énoncé que en partant d'un point initial x_0 dans $\mathbb{R}^n$ quelconque, l'algorithme des directions conjuguées converge vers la solution optimale unique $\widehat{x}$ du problème (PQSC) en n itérations, où*

$$x_n = \widehat{x} \ \text{ et } \ Qx_n = Q\widehat{x} = b.$$

Théorème 3.2.2. [11] *Supposons que Q est une matrice symétrique et définie positive de taille (n, n), et b est un vecteur dans $\mathbb{R}^n$, avec $f(x) = \frac{1}{2}x^T Qx - b^T x$. Considérons la suite $\{x_k\}_{k=1,2,\ldots}$ définie comme suit : à l'itération k, le successeur x_{k+1} de x_k est donné par la relation suivante :*

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

où $d_k \in \mathbb{R}^n$ est une direction de recherche et $\alpha_k \in \mathbb{R}^+$ est le pas de recherche obtenu par une recherche linéaire exacte, α_k vérifie

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

Notons

$$g_{k+1} = \nabla f(x_{k+1}) = Qx_{k+1} - b.$$

Alors

$$g_{k+1} = g_k + \alpha_k Q d_k,$$

et

$$g_{k+1} d_k = 0, \quad k = 0, 1, \ldots, n-1.$$

Théorème 3.2.3. [11] *Dans la méthode des direction conjuguées, on a*

$$g_{k+1}^T d_i = 0, \quad k = 0, 1 \ldots\ldots\ldots, n-1, \quad i = 0, \ldots\ldots k.$$

3.2.2 Méthode du gradient conjugué dans le cas quadratique

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

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

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 forte pente)

Algorithme de la méthode du gradient conjugué dans le cas quadratique

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

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

Algorithme

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 ($x^* = x_k$)."T est d'arrêt" si non aller à l'étape 2.

Etape 2 :
Définir $x_{k+1} = x_k + \lambda_k d_k$ avec

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

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

$$\beta_{k+1} = \frac{g_{k+1}^T A d_k}{d_k^T A d_k}.$$

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

Propriétés du gradient conjugué quadratique

Une autre formulation de la propriété fondamentale des gradients conjugués dans le cas quadratique est que les directions $\{d_k\}_{k=0}^{n-1}$ sont Q-conjuguées.
Ces directions obéissent à la relation suivante, comme illustré dans l'algorithme :

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

avec

$$\beta_k = \frac{g_{k+1}^T Q d_k}{d_k^T Q d_k}$$

l'algorithme du gradient conjugué, version quadratique converge vers la solution optimale en n itérations.

Théorème 3.2.4. [11]

$$g_k = \nabla f(x_k) = Qx_k - b$$

où $x_k, k = 0, \ldots, n-1$ sont obtenus par l'algorithme du gradient conjugue quadratique. Alors on a

$$g_{k+1}^T g_j = 0, \quad k = 0, 1, \ldots n-1, \quad j = 0, 1, \ldots\ldots k.$$

Preuve 3.4. *D'après les théorèmes présidents on a*

$$g_{k+1}^T d_j = 0, \quad k = 0, 1 \ldots\ldots, n-1, j = 0, \ldots\ldots k,$$

et

$$g_{k+1} = g_k + \alpha_k Q d_k, \quad k = 0, \ldots, n-1.$$

Fixons $k : 0 \leq k \leq n-1$ et considérons $0 \leq j \leq k$. D'après l'algorithme du gradient conjugué, on a

$$d_j = -g_j + \beta_{j-1} d_{j-1}, 0 \leq j \leq k$$

implique

$$0 = g_{k+1}^T (-g_j + \beta_{j-1} d_{j-1}) = -g_{k+1}^T g_j + \beta_{j-1} g_{k+1}^T d_{j-1}.$$

Or

$$g_{k+1}^T d_{j-1} = 0.$$

Par conséquent

$$g_{k+1}^T g_j = 0, \quad 0 \leq k \leq n-1, \quad 0 \leq j \leq k.$$

□

3.2.3 Méthode du gradient conjugué dans le cas non quadratique

Soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction non quadratique et (PNQSC) le problème de minimisation non quadratique sans contraintes suivant :

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

Pour élaborer l'algorithme du gradient conjugué dans le cas non quadratique, nous pouvons nous inspirer de l'algorithme du gradient conjugué quadratique établi précédemment. Cependant, contrairement au cas quadratique, nous n'avons pas de matrice Q, ce qui signifie que nous n'avons pas de directions conjuguées Q.

Malgré cela, l'algorithme du gradient conjugué dans le cas non quadratique génère une suite $\{x_k\}_{k\in\mathbb{N}}$ de la manière suivante :

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

où x_k est la k-ème itération, α_k est la taille du pas et d_k est la direction de recherche. L'algorithme démarre à partir d'un point x_0 arbitraire.

3.2.4 Différentes formes du gradient conjugué

Une autre formulation de la méthode du gradient conjugué est la suivante : Soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction non quadratique. Nous cherchons à résoudre le problème d'optimisation non quadratique sans contraintes (PNQSC) suivant :

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

La méthode du gradient conjugué génère une suite $\{x_k\}_{k\in N}$ de la manière suivante :

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

où $\alpha_k \in \mathbb{R}$ est déterminé par une recherche linéaire unidimensionnelle (comme la méthode d'Armijo, Goldstein ou Wolfe), et les directions d_k sont calculées récursivement selon les formules suivantes :

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

où $g_k = \nabla f(x_k)$ est le gradient de f au point x_k, et $\beta_k \in \mathbb{R}$ est un coefficient déterminé par la méthode spécifique utilisée.

En utilisant les notations suivantes :

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

nous pouvons obtenir différentes variantes de la méthode du gradient conjugué en choisissant différentes valeurs pour β_k. Les différentes valeurs attribuées à β_k définissent les différentes formes du gradient conjugué. Si on note

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

on obtient les variantes suivantes
- Gradient conjugué variante Fletcher Reeves(FR)

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

- Gradient conjugué variante Polak-Ribiére(PR)

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

- Gradient conjugué variante Hestenes-Stiefel(HS)

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

- Gradient conjugué variante Conjugate Descent Method(CD)

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

où $||.||$ est la norme Euclidienne. Récemment Dai et Yuan ont aussi introduit la forme suivante

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

Algorithme

Etape 1 :
Choisir $x_0 \in \mathbb{R}^n$. Quelconque et $\epsilon > 0$.
Etape 2
Poser $k = 0$. Calculez $g_0 = \nabla f(x_0)$. Posez $d_0 = -g_0$.
Etape 3 :
Calculez α_k en utilisant une recherche linéaire exacte ou inexacte d'Armijo ou de Goldstein ou de Wolfe ou de Wolfe Forte. Calculez $x_{k+1} = x_k + \alpha_k d_k$.
Etape 4 :
Si $\|\nabla f(x_{k+1})\| < \varepsilon_1$. Stop, $x^* = x_{k+1}$. Sinon allez à Etape 5.
Etape 5 :

Calculez $g_{k+1} = \nabla f(x_{k+1})$. Calculez β_k par l'une des manières suivantes

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

.

Calculez $d_{k+1} = -g_{k+1} + \beta_k d_k$.
Posez $k = k + 1$ et allez a Etape 3.

3.2.5 Les avantages de la méthode du gradient conjugué.

1- La consommation mémoire de l'algorithme est minime : on doit stocker les quatre vecteurs x_k, g_k, d_k, Ad_k (bien sur x_{k+1} prend la place de x_k au niveau de son calcul avec des remarques analogues pour $g_{k+1}, d_{k+1}, Ad_{k+1}$) et les scalaires λ_k, β_{k+1}.

2 - L'algorithme du gradient conjugué linéaire est surtout utile pour résoudre des grands systèmes creux, en effet il suffit de savoir appliquer la matrice A à un vecteur.

3 - La convergence peut être assez rapide : si A admet seulement $r(r < n)$ valeurs propres distincts la convergence a lieu en au plus r itération.

3.2.6 Convergence de méthode du gradient conjugué

les méthodes FR, DY et CD et plusieurs des méthodes ont en commun le terme $\|g_{k+1}\|^2$ comme numérateur. Une différence fondamentale qui les caractérise par rapport aux autres méthodes du gradient conjugué, où β_k est calculé différemment, est que les théorèmes de convergence globale nécessitent seulement la condition de Lipschitz pour la convergence générale et n'ont pas besoin de la condition de bornétude.

Synthèse des résultats de convergence des méthodes du gradient conjugué

Avant tout rappelons que les différentes variantes du gradient conjugué algorithme génèrent me suite $\{x_k\}_{k\in N}$ dans $\mathbb{R}^n$ définie par $x_0 \in \mathbb{R}^n$ quel. Conque et les itérations

$$x_{k+1} = x_k + \lambda_k d_k, \quad k = 0, 1, 2, \ldots (GC1)$$

avec λ_k est le résultat d'une recherche linéaire exacte ou inexacte du type Armijo ou Goldstein ou Wolfe ou Wolfe forte. Les directions d_k sont calculées itérativement par la formule

$$d_k = \begin{cases} -g_0 & si\ k = 0 \\ -g_k + \beta_{k-1} d_{k-1}, & k \geq 1 \end{cases} (GC2)$$

avec

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

et $\beta_k = \beta_k^{HS}$ ou $\beta_k = \beta_k^{FR}$ ou $\beta_k = \beta_k^{PRP}$ on $\beta_k = \beta_k^{CD}$ $(GC3)$
On $u\beta_k = \beta_k^{LS}$ ou $\beta_k = \beta_k^{DY}$ ou $\beta_k = \beta_k^{HZ}$ ou $\beta_k = \beta_k^{RMI}$.

Supposition 3.2.1. *Soit $f : \mathbb{R}^n \to \mathbb{R}$*
(i) L'ensemble $L = \{x \in \mathbb{R}^n; f(x) \leq f(x_1)\}$ est borné; où $x_1 \in \mathbb{R}^n$ est le point initial.
(ii) Sur un voisinage N de L. La fonction objectif f est continument différentiable et son gradient est lipschitzien (i.e) $\exists L > 0$ tel que

$$\|g(x) - g(\bar{x})\| \leq L\|x - \bar{x}\|, \forall x, \bar{x} \in N.$$

Théorème de Zoutendijk :[11] Considérons une méthode itérative quel conque générant une suite $\{x_k\}$ de la forme $x_{k+1} = x_k + \alpha_k d_k$,
d_k étant une direction de descente et α_k est une recherche linéaire inexacte de Wolfe. Supposons aussi que f vérifie la supposition. Alors on a

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

3.2.7 Résultats de convergence du gradient conjugué, version Fletcher- Reves

L'algorithme de la méthode de Fletcher Reeves :

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 Etape 1.
Etape 1 :
Si $g_k = 0$: STOP $(x^* = x_k)$."T est d'arrêt". Sinon aller à Etape 2.
Etape 2 :
Définir $x_{k+1} = x_k + \lambda_k d_k$ avec :

$$\lambda_k = min f(x_k + \lambda d_k).$$

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

où

$$\beta_{k+1}^{FR} = \frac{\|g_{k+1}\|^2}{\|g_k\|^2}.$$

Convergence de la méthode de Fletcher- Reves

Le premier résultat de convergence de la méthode du gradient conjugué non linéaire (version Flecher Reeves) avec la recherche linéaire inexacte de Wolfe forte suivante :

$$\left\{\begin{array}{c} f(x_k + \lambda_k d_k) \leq f(x_k) + \omega_1 \lambda_k d_k g_k g \\ \frac{T}{k} g_{k+1}| \leq -\omega_2 d_k^T g_k \end{array}\right\} \tag{3.34}$$

où $(\omega_2 < \frac{1}{2})$ a été démontré par Al-Baali[8]

-Touati Ahmed et Story [4] ont généralisé ce résultat pour

$$0 \leq \beta_k \leq \beta_k^{FR}. \tag{3.35}$$

- Gilbert et Nocedal[7] ont généralisé ce résultat pour

$$|\beta_k| \leq \beta_k^{FR} \tag{3.36}$$

- Récemment, Dai et Yuan[16] a montré que la méthode (FR)es globalement convergente avec la recherche linéaire inexacte de Wolfe relaxée.

On donne les principaux résultats de convergence de la méthode du gradient conjugué non linéaire avec la recherche linéaire inexacte Wolfe forte[Gilbertet et Nocedal], aussi avec la recherche linéaire inexacte de Wolfe [Dai et Yuan].

3.2.8 Résultats de convergence du gradient conjugué, version Polyak Ribière et Polyak

La variante dite de Polak-Ribière et Polyac (PRP) consiste à définir β_k. Cette méthode fút découverte par Polak, Ribière[5] et Polyak[2] L'algorithme de la méthode de (PRP) est le suivant :

Algorithme de la méthode de Polak-Ribière-Polyak

Etape 0 : (initialisation)
Soit x_0 le point de depart, $g_0 = \nabla f(x_0)$ poser $d_0 = -g_0$.
Poser $k = 0$ et aller à Etape 1.
Etape 1 :
Si $g_k = 0$: STOP $(x^* = x_k)$."Test d'arrêt". Sinon aller à Etape 2.
Etape 2 :
Définir $x_{k+1} = x_k + \lambda > 0_k d_k$ avec :

$$\lambda_k = min f(x_k + \lambda > 0 d_k)$$
$$d_{k+1} = -g_{k+1} + \beta_{k+1}^{PRP} d_k$$

$$\beta_{k+1}^{PRP} = \frac{g_{k+1}^T[g_{k+1} - g_k]}{\|g_k\|^2} = \frac{g_{k+1}y_k}{\|g_k\|^2}$$

.

Poser $k = k + 1$ et aller à Etape 1.

Convergence de la méthode de PRP

Pour étudier la convergence de cette méthode on a deux cas : Cas où f est fortement convexe Polak et Ribière ([5, 1969]) ont démontré la convergence de la méthode (PRP) à travers le proposition suivant :

Proposition 3.1. [13] *Si f est fortement convexe de classe C^1 avec un gradient lipschitzien, alors la méthode de Polak Ribiére et Polyac avec la recherche linéaire exacte génère une suite $\{x_k\}$ convengeant vers l'unique point x, réalisant le minimum de f.*

3.2.9 Résultats de convergence du gradient conjugué version Dai- Yuan

Récemment Dai et yuan ([15, 1998]) ont introduit une formule pour β_k sous la forme suivante :

$$\beta_k^{DY} = \frac{\|g_k\|^2}{d_{k-1}^T y_{k-1}} \tag{3.37}$$

oú

$$y_{k-1} = g_k - g_{k-1}.$$

Cette méthode possède plusieurs propriétés, par exemple elle possède la propriété de descente à chaque itération et elle converge si le pas est déterminé par la règle de (Wolfe faible, Armijo et Goldstein) lorsque f est strictement convexe.

Algorithme de la Méthode de Dai-Yuan

Etape0 : (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 $x_{k+1} = x_k + \lambda_k d_k$ avec λ_k vérifie les conditions

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

oú

$$\beta_{k+1}^{DY} = \frac{\|g_{k+1}\|^2}{d_k^T[g_{k+1} - g_k]} = \frac{\|g_{k+1}\|^2}{d_k^T y_k}.$$

Poser $k = k + 1$ et aller à Etape 1.

Convergence de la méthode de Dai-Yuan

Les résultats assurent la convergence de cette méthode pour une fonction strictement convexe avec une recherche linéaire inexacte d' Armijo et Goldstein.

3.3 Méthode de Newton

Soit f : $\mathbb{R}^n \to \mathbb{R}$. On supposera que $f \in C^2(\mathbb{R}^n)$.
Soit maintenant (P) le problème d'optimisation sans contraintes

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

La méthode de Newton[3] est attribuée au mathématicien, physicien et astronome anglais Isaac Newton $(1642 - 1727)$.
Le principe de la méthode de Newton consiste a minimiser successivement les approximations du second-ordre d'une fonction f, plus précisément si

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

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

$$\nabla f(x_k) + H(x_k)(x - x_k) = 0. \tag{3.38}$$

Supposons que H est inversible, alors le successeur x_{k+1} de x_k est donné par

$$x_{k+1} = x_k - H^{-1}(x_k)\nabla f(x_k). \tag{3.39}$$

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

3.3.1 Algorithme de Newton

Etape initiale
Soit ε un paramètre qui détermine le critère d'arrêt. Choisir un point initial. Poser $k = 1$ et aller a l'étape principale.
Etape principale
Si $\|\nabla f(x_k)\| < \varepsilon$, stop ; sinon, poser $x_{k+1} = x_k - H^{-1}(x_k)\nabla f(x_k)$, remplacer k par $k + 1$ et répéter l'étape principale.

3.3.2 Etude de la convergence

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

Théorème 3.3.1. [3] *Soit $f : \mathbb{R}^n \to \mathbb{R}$ trois fois continument différentiable. Considérons l'algorithme de Newton défini par l'application*

$$A(x) = x - H^{-1}(x)\nabla f(x)$$

Soit $\bar{x}$ tel que $\nabla f(\bar{x}) = 0$ et supposons que $H(\bar{x})^{-1}$ existe. Soit x_1 un point initial assez proche de $\bar{x}$ de sorte que cette proximité implique l'existence de $k_1, k_2 > 0$, avec $k_1 k_2 \|x_1 - \bar{x}\| < 1$ et vérifiant les deux propriétés suivantes
1. $\|H(x)^{-1}\| \leq k_1$.
2. $\|\nabla f(\bar{x}) - \nabla f(x) - H(x)(x - \bar{x})\| \leq k_2 \|x_1 - x_2\|^2$. Pour toux x satisfaisant

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

Alors, algorithme converge de façon super linéaire vers $\bar{x}$ avec une vitesse de convergence quadratique.

3.3.3 Avantages et inconvénients de la méthode de Newton

Avantages
Comme nous l'avons vu au théorème précédent, la méthode de Newton bénéfice d'une convergence quadratique, et c'est son principal avantage.
Inconvénients
1. Cette méthode fonctionne très bien pour les problème de petite dimension, lorsqu'on peut calculer facilement la matrice hessienne et son inverse. Cette procédure

nécessite des itérations plus nombreuses et couteuse dans les problèmes de grand taille.
2. Comme

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

on voit que le successeur x_{k+1} de x_k n'est pas toujours bien défini.
3. Même si $H^{-1}(x_k)$ est inversible, la direction $d_k = -H^{-1}(x_k)\nabla f(x_k)$ n'est pas toujours une directions de descente.(Si $h(x_k)$ est définie positive, alors d_k est une direction de descente).

Exemple 3.1.
$f(x) = 100(x_2 - x_1^2) + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3) + 101[(x_2 - 1)^2 + (x_4 - 1)^2]$
$+198(x_2 - 1)(x_4 - 1)$
cette fonction est la fonction de Wood, elle admet le point $(1, 1, 1, 1)$ *comme un minimum el le point* $(-1, 1, -1, 1)$ *comme un point selle.*
Nous allons appliquer la méthode de Newton pour minimiser cette fonction en prenant $x_0 = (-3, -1, -3, -1)$ *et* $\varepsilon = 10^{-4}$ *que la méthode de Newton à converge vers le point* $(-1, 1, -1, 1)$ *qui est le point selle de cette fonction.*
Le nombre d'itérations est $s = 15$
la solution est

$$x = (-9.67974IE - 019.471393E - 01, -9.695163E - 01, 9.512478E - 0)$$

la valeur de $f(x)$ *est* 7.876967.

tableau

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

3.4 Méthode de Quasi-Newton

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

3.4.1 Principe de la méthode

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

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

l'idée est de remplacer cette itération par

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

où λ_k est un paramètre fourni par une recherche linéaire le long de la direction

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

S_k est une approximation symétrique, définie positive, de $H^{-1}(x_k)$. Bien sur, plus S_k sera proche de $H^{-1}(x_k)$ plus l'algorithme convergera rapidement. L'objectif est

donc de trouver une bonne suite de matrices S_k faciles a construire, c'est à dire utilisant des informations seulement sur le gradient et qui converge vite vers des approximations de plus en plus précises, de l'inverse du hessien de f.
Pour réaliser cela, prenons $f \in C^2(R^n)$ et faisons un développement de $\nabla f(x)$ au voisinage de x_k

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

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

ou encore

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

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

$$S_k[\nabla f(x_{k+1}) - \nabla f(x_k)] \simeq (x_{k-1} - x_k) \tag{3.40}$$

mais comme x_{k+1} est calculé après S_k, il est peu probable que cette équation soit satisfaite, même approximativement. En revanche on peut toujours imposer que S_{k+1} satisfasse cette équation exactement, d'où

$$S_{k+1}[\nabla f(x_{k-1}) - \nabla f(x_k)] = (x_{k+1} - x_k) \tag{3.41}$$

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

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

C_k étant une matrice de correction qui intègre au mieux la nouvelle information fournie par x_{k+1} et $\nabla f(x_{k+1})$ de telle manière que S_{k+1} satisfasse la condition (3.41) Basées sur ce principe, les méthodes quasi -Newton différent l'une par rapport à l'autre, d'aprés la définition de la matrice C_k.
Commençons par la méthode de correction de rang un qui est considérée comme une introduction élémentaire aux deux autres méthodes (DFP et BFGS) décrites par la suite.

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

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

Montrons qu'il est facile de calculer a_k et u_k de telle sorte que la condition (3.41) soit satisfaite. Posons

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

la condition (3.41) s'écrit donc

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

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

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

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

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

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

3.4.2 Algorithme de Quasi-Newton

Etape initiale
Soit $\varepsilon > 0$ déterminant le critère d'arret. Choisir un point initial x_1 et une matrice symétrique définie positive S_1. Poser $y_1 = x_1,\ k = 1$ et aller aux étape principales.
Etapes principales
Etape 1 :
Si $\|\nabla f(y_k)\| < \varepsilon$. Stop ; sinon, poser $d_k = -S_k \nabla f(x_k)$ et soit λ_k la solution optimale du problème $\min f(y_k + \lambda d_k), \lambda \geq 0$.
Etape 2 :
Construire S_{k+1} comme suit

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

avec

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

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

Il est utile de citer le théorème suivant qui montre que lorsque H est constante, la condition (3.41) est non seulement vérifiée pour $i = k$ mais aussi pour tout $i < k$ et que dans ce cas la convergence est obtenue dans au plus n étapes.

Théorème 3.4.1. [11] *Si f est quadratique de matrice hessienne H définie positive et si $p_1, \ldots, p_n$ sont des vecteurs indépendants alors la méthode de correction de rang un converge au plus dans $(n+1)$ itérations et $(S_{n+1})^{-1} = H$.*

Exemple 3.2. *Considérons le problème suivant*

$$(P1)\begin{cases} min\frac{1}{2}x^tQx \\ x \in R^n \end{cases} \tag{3.42}$$

où

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

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

tableau

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

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

Conclusion

A travers ce travail nous avons présenté certains méthodes pour résoudre un problème d'optimisation.

Il est préférable de recherche des méthodes plus précises.

Bibliographie

[1] Bosora S, Khdir T. Méthode de la plus forte pente memoire master2, 2021, 2022.

[2] B.T. Polyak (1969), The Conjugate Gradient Method In Extremem Problems. Comput. Math. Math. Phys, 9, pp. 94 – 112.

[3] Chekroune S, Yahiaoui A, Zaich A. Résolution d'un problème d'optimisation les méthodes Quasi-Newton mémoire master2, 2020, 2021.

[4] D. Touati-Ahmed and C. Storey (1990), Efficient Hybrid Conjugate Gradient Techniques, JOTA, 64, pp. 379-397.

[5] E. Polak and G. Ribière (1969), Note Sur La Convergence De Directions Conjuguées, Rev. Française Informat. Recherche Operationelle, 3e année, 16, pp. 35-43.

[6] Fekih S, Kessar D. Utilisation des recherches linéaire résoudre un problème d'optimisation mémoire master2, 2022, 2023.

[7] J.C. Gilbert and J. Nocedal (1992), Global convergence properties of conjugate gradient methods for optimization, SIAM J. Optimization. Vol. 3, No.1, PP. 21 – 42.

[8] M. Al-Baali (1985), Descent property and global convergence of the Fletcher-Reeves method with inexacte line search. IMA J. Num. Anal.. Vol. 5, pp. 121-124.

[9] M. J. D. Powell (1971), On the convergence of the variable metric algorithm ; J. Inst. Math. Appl, 7, pp. 21 -36.

[10] M. Minoux (1983), Programmation Mathématique Théorie et Algorithmes , tome1, Dunod.

[11] Pr Rachid Benzine, OPTIMISATION CONVEXE OPTIMISATION SANS CONTRAINTES, 2007.

[12] R. Fletcher (1987), Practical methods of optimization, John. Wiley&Son. Chichster.

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

[14] TAHAR Bouali, Doctorat en Mathématiques Convergence des méthodes du gradient conjugué avec la recherche linéaire non monotone, 2014. 2015.

[15] Y. H. Dai and Y. Yuan (1999), A non linear conjugate gradient with a strong global conver- gence property, SIAM J.Optimization, Vol. 10(1) pp. 177-182.

[16] Y. H. Dai and Y. Yuan (1996), Convergence properties of the Fletche-Reeves method, IMA J Numer. Anal, Vol.16(2), PP.155-164.

[17] Y. H. Dai and Y. Yuan (1998), Some properties of a new conjugate gradient method, in : Advances in Nonlinear Programming, ed. Kluwer Academic, Boston, pp. 251-262.

[18] Y. H. Dai. On the Nonmonotone Line Search, Journal of Optimization theory and Applica- tions : Vol. 112, No. 2, pp. 315-330, February 2002.