RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITÉ DR. YAHIA FARÉS DE MEDEA
FACULTÉ DES SCIENCES
DÉPARTEMENT DE MATHÉMATIQUES ET INFORMATIQUE



MEMOIRE

présenté en vue de l'obtention du

DIPLOME DE MASTER EN MATHEMATIQUES

Spécialité : Analyse et Modélisation Mathématique

par :

BENYAHIA Lynda

Titre : Sur quelques algorithmes pour équations avec contrainte convexe

---

Soutenu publiquement le 13 juin 2017 devant le jury composé de

Président du jury :	LEMDANI Rachid	*MAA*	*U.Y.F.M*
Examinateur :	LEGHMIZI Mohamed Lamine	*MCA*	*U.Y.F.M*
Encadreur :	ALLECHE Boualem	*Pr*	*U.Y.F.M*

ANNEE UNIVERSITAIRE : 2016-2017

Remerciements

Je tiens à adresser mes remerciements les plus chaleureux et ma profonde gratitude à mon encadreur Monsieur. B.Alleche, professeur à l'université de Médéa pour m'avoir proposé le sujet de ce mémoire. C'est grâce à sa grande disponibilité, ses conseils, ses orientations, et ses encouragements que j'ai pu mener à bien ce travail.

Mes remerciements vont également à Mr.R.Lemdani,MAA à l'université de Médéa, pour avoir bien voulu me faire l'honneur d'accepter de présider le jury.

De même je remercie Mr.M.L.Leghmizi, pour l'honneur qu'ils m'ont fait de bien vouloir accepter de faire partie du jury.

Je remercie également toutes les personnes qui m'ont facilité la tâche et tout ceux qui ont contribué de prés ou de loin à l'élaboration de ce mémoire.

Enfin, Merci à toute ma famille, en particulier, ma mère, mon père, ma soeur et mes frères.

Benyahia Lynda

Résumé

Soit $F : \mathbb{R}^n \to \mathbb{R}^n$ est une fonction continue et monotone. On cherche à résoudre les équations avec contraintes convexes suivantes, notées CCE(F, C) :

$$F(x^*) = 0 \quad \textit{telle que } x^* \in \mathrm{C}.$$

La nouvelle méthode de projection de gradient conjugué hybride est proposée pour les équations avec contraintes convexes.
À chaque itération, la nouvelle méthode est entièrement exempte de toute évaluation dérivée.
En outre, la direction de recherche générée par la méthode proposée satisfait la propriété de descendance suffisante, qui est indépendante de la recherche linéaire.
Sous condition que la fonction est continue monotone, Nous établissons la convergence globale de la nouvelle méthode.
Par rapport aux méthodes traditionnelles pour résoudre ce problème, notre nouvelle méthode n'exige pas la continuité (Lipschitz) du fonction.
Certains tests numériques à grande échelle sont effectués, ce qui montre que la méthode proposée est efficace et prometteuse.

Mots clés : Méthode de gradient conjugué . Equations avec contrainte convexe . Convergence globale

Abstract

Consider the following convex constrained equations, denoted by CCE(F, C) :

$$F(x^*) = 0 \quad \textit{subject to } x^* \in \mathrm{C}.$$

where $F : \mathbb{R}^n \to \mathbb{R}^n$ be a continuous and monotone mapping.
A new hybrid conjugate gradient projection method is proposed for the convex constrained equations.
At each iteration, the new method is fully free from any derivative evaluations.
Furthermore, the search direction generated by the proposed method satisfies the sufficient descent property, which is independent of the line search.
Under the condition that the underlying mapping is continuous and monotone, we establish the global convergence of the new method.
Compared with traditional methods for solving such problem, our new method does not require the Lipschitz continuity of the underlying mapping.
Some large-scale numerical tests are performed and reported, which show that the proposed method is efficient and promising.

Key words : Conjugate gradient method . Convex constrained equations . Global convergence.

Table des matières

Introduction **v**

1 Notions de base (Optimisation sans contraintes) **1**

1.1 Quelques définitions 1

1.1.1 Analyse convexe 1

1.1.2 Calcul matriciel 2

1.1.3 Différentiabilité 5

1.2 Les problèmes de minimisation sans contraintes 7

1.2.1 Conditions d'optimalité des problèmes d'optimisation sans contraintes 8

1.2.2 Les algorithmes des problèmes de minimisation sans contraintes 11

1.2.3 Convergence globale des algorithmes 12

1.3 Méthodes du gradient conjugué 14

1.3.1 Description de la méthode 14

1.3.2 Algorithme de La méthode du gradient conjugué pour les fonctions quelconques 15

1.3.3 la convergence globale de les méthodes du gradient conjugué 16

2 Optimisation avec contrainte convexe **19**

2.1 Préliminaires 19

2.1.1 Projection sur un convexe fermé 19

2.1.2 Méthode de gradient projeté 22

2.1.3 Agorithme de gradient projeté et leur convergence 23

2.2 Quelles que méthodes traditionnel pour résoudre CCE(F, C) 24

2.2.1 Méthode de gradient conjugué descente (CGD) 24

2.2.2 La convergence globale de (CGD) 26
2.2.3 Méthode de gradient conjugué auto-adaptatif à trois termes (TTGD) 27
2.2.4 La convergence globale de (TTGD) 29
2.3 Nouveau méthode de projection par gradient conjugué hybride 30
2.3.1 Algorithme de projection par gradient conjugué hybride 30
2.3.2 La convergence globale 35

3 Application dans "*MATLAB*" 37
3.1 *PROBLEME* "**1**" 38
3.1.1 Programmes de (NHCGP) et (CGD) et (TTGD) 38
3.1.2 Résultats numériques de (NHCGP) et (CGD) et (TTGD) 42
3.2 *PROBLEME* "**2**" 43
3.2.1 Programmes de (NHCGP) et (CGD) et (TTGD) 43
3.2.2 Résultats numériques de (NHCGP) et (CGD) et (TTGD) 47

Bibliographie 48

Introduction

Dans la vie courante, nous sommes fréquemment confrontés à des problèmes "d'optimisation" plus ou moins complexes. Cela peut commencer au moment où l'on tente de ranger son bureau, de placer son mobilier, et aller jusqu'à un processus industriel, par exemple pour la planification des différentes tâches.
Ces problèmes peuvent être exprimés sous la forme générale d'un "problème d'optimisation".
L'optimisation peut être définie comme la science qui détermine la meilleure solution à certains problèmes mathématiquement définis, qui sont souvent des modèles de physique réel.
C'est une technique qui permet de "quantifier" les compromis entre des critères parfois non commensurables.
L'optimisation recouvre l'étude des critères d'optimalité pour les différents problèmes, la détermination des méthodes algorithmiques de solution, l'étude de la structure de telles méthodes et l'expérimentation de l'ordinateur avec ces méthodes avec vrais problèmes de la vie.
D'un point de vue mathématique, l'optimisation consiste à rechercher le minimum ou le maximum d'une fonction avec ou sans contraintes.

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

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

. -Comment résoudre ce problème ou bien quelles sont les méthodes utilisé pour résoudre (p)?.

2) Soit $f : \mathbb{R}^n \to \mathbb{R}$. On consider le problème de minimisation avec contraintes convexes suivant :

$$(P) \ : \ \min\{f(x) : x \in \mathrm{C} \subseteq \mathbb{R}^n\}$$

La solution de (p) est une solution de $\mathrm{CCE}(F, \mathrm{C})$. Le $\mathrm{CCE}(F, \mathrm{C})$ a été utilisé dans de nombreux domaines scientifiques Les systèmes d'équilibre chimique , les problèmes d'équilibre économique,

la détection par compactage , etc.,donc quelles sont les anciennes méthodes utilisées pour résoudre les problèmes du type (P)?.
Dans ces dernières années sont proposés une nouvelle méthode pour résoudre $\mathrm{CCE}(F, \mathrm{C})$ est la méthode de projection de gradient conjugué hybride.
Quel est l'algorithme de cette méthode ?,et est ce qu'elle vérifie la convergence globale ?.

3) Nous testons la nouvelle méthode ,et comparons avec les anciennes méthodes.
Quelle est la meilleure méthode ?.

Chapitre 1

Notions de base (Optimisation sans contraintes)

1.1 Quelques définitions

1.1.1 Analyse convexe

Ensembles convexes

Définition 1.1.1 *Soient* E *un espace vectoriel de dimension finie sur* $\mathbb{R}$ *et* $x, y \in \mathrm{E}$. *On appelle segment de* E, *un ensemble noté et défini comme suit :*

$$[x; y] := (1 - t)x + ty : t \in [0; 1].$$

On dit qu'une partie C *de* E *est convexe si pour tout* $x, y \in \mathrm{C}$ *, le segment* $[x; y]$ *est contenu dans* C.

Proposition 1.1.2

1. *Si* $\mathrm{C}_1, ..., \mathrm{C}_n$ *sont des espaces convexes. Alors l'intersection* $\bigcap \mathrm{C}_i$ *est convexe.*
2. *Pour* $i = 1, ..., k$ *soient* $\mathrm{C}_i \in \mathbb{R}^{ni}$ *des ensembles convexes. Alors* $\mathrm{C} = \mathrm{C}_1 * ... * \mathrm{C}_k$ *est un ensemble convexe de* $\mathbb{R}^{n_1 * ... * n_k}$.
3. *Soit* $\mathrm{C} \subset \mathbb{R}^n$ *un ensemble convexe et* $f : \mathbb{R}^n \to \mathbb{R}^m$ *une application affine. Alors* $f(\mathrm{C}) = \{f(x), x \in \mathrm{C}\}$ *est convexe.*

Fonctions convexes

$$f : \mathrm{C} \to \mathbb{R}^n, \mathrm{C} \subset \mathbb{R}^n.$$

est convexe sur C si son épigraphe est un ensemble convexe.

Définition 1.1.3 *Soit* $\mathrm{C} \subset \mathbb{R}^n$ *un ensemble convexe non vide.Une fonction* $f : \mathrm{C} \subset \mathbb{R}^n \to \mathbb{R}$ *est une fonction convexe si et seulement si pour tout* $x, y \in \mathrm{C}$ *, on a :*

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

-On dit que $f : \mathrm{C} \subset \mathbb{R}^n \to \mathbb{R}$ *est strictement convexe si pour tout* $x, y \in \mathrm{C}$ *avec* $x \neq y$*,on a :*

$$f(tx + (1-t)y) < tf(x) + (1-t)f(y), \ \forall t \in]0,1[.$$

-On dit que $f : \mathrm{C} \subset \mathbb{R}^n \to \mathbb{R}$ *est fortement convexe ou uniformément convexe de module* $\delta > 0$ *(*$\delta-$*convexe), si pour tout* $x, y \in \mathrm{C}$ *,on a :*

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y) - \frac{\delta}{2}t(1-t)\|y-x\|^2, \ \forall t \in]0,1[.$$

-On dit que $f : \mathrm{C} \subset \mathbb{R}^n \to \mathbb{R}$ *est quasi-convexe,si pour tout* $x, y \in \mathrm{C}$ *,on a :*

$$f(tx + (1-t)y) \leq \max[f(x), f(y)], \ \forall t \in]0,1[.$$

-On dit que $f : \mathrm{C} \subset \mathbb{R}^n \to \mathbb{R}$ *est strictement quasi-convexe,si pour tout* $x, y \in \mathrm{C}$ *avec* $f(x) \neq f(y)$ *,on a :*

$$f(tx + (1-t)y) < \max[f(x), f(y)], \ \forall t \in]0,1[.$$

1.1.2 Calcul matriciel

Définition 1.1.4 *- Soit* A *une matrice carrée d'ordre n. S'il existe une matrice carrée* B *d'ordre n telle que* $\mathrm{AB} = \mathrm{BA} = \mathrm{I}$*, alors* B *est appelée la matrice inverse de* A *et est notée* A^{-1}*.*

- Une matrice carrée A *est dite symétrique si* $\mathrm{A} = \mathrm{A}^t$*.*

- Le rang d'une matrice A *est donné par le nombre de lignes ou de colonnes linéairement indépendantes de* A*.*

- Soit A *une matrice carrée, le polynôme caractéristique de* A *est donné par :* $p(\lambda) = |A - \lambda I|$ *,*

$p(\lambda)$ est un polynôme de degré n donc possèdant au plus n racines réelles distinctes.
- Une matrice A *est singuliére si et seulement si* $\det(\mathrm{A}) = 0$*, est dite* A *non singulière si* $\det(\mathrm{A}) \neq 0$.
- Les valeurs propres de la matrice A *sont les racines de $p(.)$; i,e., les valeurs λ_i telles que $p(\lambda_i) = 0$.*
- Une matrice A *est dite orthogonale si* $\mathrm{A}^{-1} = \mathrm{A}^t$.
- Soit $\mathcal{M}_n$ l'anneau des matrices carrées $n \times n$ dans $\mathbb{R}$. Une matrice $\mathrm{A} \in \mathcal{M}_n$ est dite semi-définie positive si :

$$\forall x \in \mathbb{R}^n : x^T \mathrm{A} x \geq 0.$$

elle est dite définie positive si :

$$\forall x \in \mathbb{R}^n_* : x^T \mathrm{A} x > 0.$$

Indépendance linéaire

Soit $\upsilon^{(1)}, \upsilon^{(2)}, ..., \upsilon^{(k)}$ un ensemble de vecteurs dans $\mathbb{R}^n$. On dit que cet ensemble est linéairement indépendants si :

$$\alpha_1 \upsilon^{(1)} + \alpha_2 \upsilon^{(2)} + ... + \alpha_k \upsilon^{(k)} = 0 \Rightarrow \alpha_1 = \alpha_2 = ... = \alpha_k = 0.$$

Produit scalaire

Le produit scalaire de deux vecteurs x et y de $\mathbb{R}^n$ est :

$$x^t y = \sum x_i y_i.$$

Normes vectorielles

Une norme d'un espace vectoriel E est une application $\|.\|$ de E dans $\mathbb{R}^+$ qui vérifie les propriétés suivantes :

$$\forall x \in \mathrm{E}, \|x\| \geq 0.$$

$$\|x\| = 0 \Leftrightarrow x = 0.$$

$$\forall \lambda \in \mathbb{R}, x \in \mathrm{E}, \|\lambda x\| = |\lambda| \|x\|.$$

$$\forall x, y \in \mathrm{E}, \|x + y\| \leq \|x\| + \|y\|.$$

Dans $\mathbb{R}^n$, les trois normes les plus courantes sont la norme infinie, la norme 1 et la norme euclidienne.

- norme infinie :

$$\|x\|_\infty = \max_{1\leq i\leq n} |x_i|.$$

- norme 1 :

$$\|x\|_1 = \sum |x_i|.$$

- norme 2 ou norme euclidienne :

$$\|x\|_2 = \sqrt{x^t x} = (\sum x_i^2)^{\frac{1}{2}}.$$

ou $x = (x_1, ..., x_n)^t$.
La norme euclidienne est donc définie par le produit scalaire $x^t y$.

Normes matricielles

Soit $\|.\|$ une norme quelconque sur $\mathbb{R}^n$, on appelle norme matricielle subordonnée de la matrice carrée A d'ordre n

$$\|\mathrm{A}\| = \max_{x\in\mathbb{R}^n, \|x\|\neq 0} \frac{\|\mathrm{A}x\|}{\|x\|} = \max_{x\in\mathbb{R}^n, \|x\|=1} \|\mathrm{A}x\|.$$

Orthogonalité dans $\mathbb{R}^n$

$$x \perp y = x^t y = 0.$$

Théorème 1.1.5 *Toute matrice symétrique a des valeurs propres réelles et des vecteurs propres réels. Elle est diagonalisable. De plus, les vecteurs propres forment une base orthonormée.*

Théorème 1.1.6 *Si* A *est une matrice définie positive d'ordre n, alors :*

1. A *est non singulière ; i.e,* A^{-1} *existe.*
2. $\mathrm{A}_{ii} > 0$ *pour* $i = 1, ..., n$.

Théorème 1.1.7 *Si* A *est une matrice symétrique et* D *est une matrice diagonale contenant les valeurs propres de* A *; alors il existe une matrice orthogonale* Q *telle que* $\mathrm{D} = \mathrm{Q}^{-1}\mathrm{AQ} = \mathrm{Q}^{\mathrm{t}}\mathrm{AQ}$

1.1.3 Différentiabilité

On se place dans $\mathbb{R}^n$, $n < \infty$, considéré comme un espace vectoriel normé muni de la norme euclidienne notée $\|.\|$. Soit Ω un ouvert de $\mathbb{R}^n$.

Gradient

•On note par

$$(\bigtriangledown f(x))^T = (\frac{\partial f}{\partial x_1}, ..., \frac{\partial f}{\partial x_n})(x)$$

le gradient de f au point $x = (x_1, .., x_n)$.
•On dit que x^* est un point stationnaire de f si $\nabla f(x^*) = 0$.

Définition 1.1.8 *Soit $f : \mathbb{R}^n \to \mathbb{R}$. $y = f(x)$ définit une surface dans $\mathbb{R}^{n+1}$.*
$f(x) = r$, avec r constant, définissent des courbes de niveau ou des contours de cette surface.
Un contour de niveau r est l'ensemble des points $\mathrm{R}(r) = \{x/f(x) = r\}$.

Propriété 1.1.1 *Soit $f : \mathbb{R}^n \to \mathbb{R}$ et ∇f son gradient,* R *le contour, alors le gradient ∇f est orthogonal au contour.*

Démonstration :
Soit le contour R de niveau $f(x_0)$ passant par $x_0 : \mathrm{R} = \{x/f(x) = f(x_0)\}$.
Soit une courbure dans $\mathbb{R}$ paramétrée par $\upsilon(t) : \mathbb{R}^n \to \mathbb{R}$ avec $\upsilon(t_0) = x_0$ et $\upsilon(t) = x \in \mathrm{R}$.
La tangente à cette courbe en x_0 est de direction :

$$\frac{d\upsilon}{dt/_{t=t_0}}.$$

Nous avons :

$$\begin{aligned} f(\upsilon(t)) = f(\upsilon(t_0)) &= f(x_0) \\ &\Rightarrow \frac{d}{dt} f(\upsilon(t))/_{t=t_0} = 0 \\ &\Leftrightarrow \frac{\partial f}{\partial x_1}(\upsilon(t_0)) \frac{d\upsilon_1}{dt}/_{t=t_0} + ... + \frac{\partial f}{\partial x_n}(\upsilon(t_0)) \frac{d\upsilon_n}{dt}/_{t=t_0} = 0 \\ &\Leftrightarrow \nabla^T f(x_0) \frac{d\upsilon}{dt}/_{t=t_0} = 0. \end{aligned}$$

Ce qui veut dire que $\nabla f(x_0)$ est orthogonal à R en $x_0, \forall x_0$.

Hessien

•On appelle Hessien de f la matrice symétrique de $\mathcal{M}_n(\mathbb{R})$

$$\mathrm{H}(x) = \nabla(\nabla^T f)(x) = \nabla^2 f(x) = (\frac{\partial^2 f}{\partial x_i \partial x_j})(x),\ i = 1, ..., n,\ j = 1, ..., n.$$

Fonction convexe différentiable

Soit $\mathrm{C} \subseteq \mathbb{R}^n$ un sous ensemble non vide.$f : \mathbb{R}^n \to \mathbb{R}$ et $\overline{x} \in \mathrm{C}$,$f$ est dite différentiable au point $\overline{x}$; s'il existe un vecteur $\overline{x}\ \nabla f(\overline{x}) \in \mathbb{R}^n$ et une fonction $\alpha : \mathbb{R}^n \to \mathbb{R}$ telle que :

$$f(x) = f(\overline{x}) + \nabla f^T(x - \overline{x}) + \|x - \overline{x}\|\alpha(\overline{x}, x - \overline{x}),$$

où $\alpha(\overline{x}, x - \overline{x}) \to_{x \to \overline{x}} 0$.

Fonction convexe deux foix différentiable

Soit $\mathrm{C} \subseteq \mathbb{R}^n$ un sous ensemble non vide.$f : \mathbb{R}^n \to \mathbb{R}$ et $\overline{x} \in \mathrm{C}$,$f$ est dite deux foix différentiable au point $\overline{x}$; s'il existe un vecteur $\overline{x}\ \nabla f(\overline{x}) \in \mathbb{R}^n$ et une matrice symétrique $\mathrm{H}(\overline{x})$ d'ordre (n, n) appellée matrice hessienne, et une fonction $\alpha : \mathbb{R}^n \to \mathbb{R}$ telle que :

$$\forall x \in \mathrm{C} : f(x) = f(\overline{x}) + \nabla f^T(x - \overline{x}) + \frac{1}{2}(x - \overline{x})^T \mathrm{H}(\overline{x})(x - \overline{x}) + \|x - \overline{x}\|^2 \alpha(\overline{x}, x - \overline{x}),$$

où $\alpha(\overline{x}, x - \overline{x}) \to_{x \to \overline{x}} 0$.

Dérivée directionnelle

Définition 1.1.9 *On appelle dérivée directionnelle de f dans la direction d au point x, notée $\delta f(x, d)$, la limite (éventuellement ∞) du rapport :*

$$\frac{f(x + hd) - f(x)}{h}\ lorsque\ h\ tend\ vers\ 0.$$

Autrement dit :

$$\delta f(x, d) = \lim_{h \to 0} \frac{f(x + hd) - f(x)}{h} = \nabla^T f(x) d.$$

• *Si $\|d\| = 1$: la dérivée directionnelle est le taux d'accroissement de f dans la direction d au point x.*

Remarque 1.1.1 • *Le taux d'accroissement est maximal dans la direction du gradient.*
• *Le gradient indique la direction de la plus grande pente.*

Direction de descente

Soit $f : \mathbb{R}^n \to \mathbb{R}$ $\overline{x} \in \mathbb{R}^n$, d un vecteur non nul de $\mathbb{R}^n$ est dit une direction de descente au point $\overline{x}$ si et seulement s'il existe un réel $\delta > 0$ tel que pour tout α dans l'intervalle $]0, \delta[$, on a :

$$f(\overline{x} + \alpha d) < f(\overline{x}).$$

Donnons maintenant une condition suffisante pour que d soit une direction de descente.

Théorème 1.1.10 *Soit $f : \mathbb{R}^n \to \mathbb{R}$, différentiable au point $\overline{x} \in \mathbb{R}^n$ et $d \in \mathbb{R}^n$ telle que :*

$$\nabla^T f(\overline{x}) d < 0,$$

Alors d est une direction de descente en $\overline{x}$.

Remarque 1.1.2 *L'ensemble des directions de descente de f en x,*

$$\{d \in \mathbb{R}^n : \nabla^T f(x).d < 0\},$$

Forme un demi-espace ouvert de $\mathbb{R}^n$.

1.2 Les problèmes de minimisation sans contraintes

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

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

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

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

La construction d'algorithmes consacrés à la recherche efficace de minimum d'une fonction sans contraintes est un problème complexe, car il ne suffit pas d'élaborer un algorithme il faut montrer

de plus qu'il emporte sur ceux connus.
On compare des algorithmes en se basant sur des plusieurs critères, par exemple, la précision du résultat, le nombre d'évalutions fonctionnelles et celui d'évaluations du gradient, la vitesse de la convergence, le temps de calcul, l'occupation de la mémoire nécessaire, ...
Même en se fixant des critères de comparaison, on ne peut pas classer les algorithmes ni en indiquant lequel est le meilleur ou le pire. Le fait est qu'on obtient les estimations de l'un des critères précédents pour des classes des problèmes, et un algorithme mauvais pour une vaste classe peut s'averer efficace pour une autre, plus restreinte. L'utilisateur doit posseder tout un arsenal d'algorithmes pour être en mesure de faire face à chaque problème posé.

Résultats d'existence et d'unicité

L'existence et l'unicité de la solution du problème (P) résultent en général de propriétes de convexité de la fonction f et de l'ensemble C sur lequel elle est définie. En effet :

Théorème 1.2.1 *Soit* $f : \mathrm{C} \subset \mathbb{R}^n \to \mathbb{R}$ *une fonction strictement convexe et* C *un ensemble convexe. Le minimum de* f *sur* C *; existe et est unique.*

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

Définition 1.2.2 *Considérons le problème de minimisation sans contraintes* (P) .
a) $x^* \in \mathbb{R}^n$ *s'appelle minimum global du problème* (P) *si :*

$$f(x^*) \leq f(x), \forall x \in \mathbb{R}^n. \tag{1.1}$$

b) x^* *est un minimum local de* (P) *s'il existe un voisinage* $\mathrm{V}_\varepsilon(x^*)$ *de* x^* *tel que :*

$$f(x^*) \leq f(x), \forall x \in \mathrm{V}_\varepsilon(x^*). \tag{1.2}$$

c) x^* *est un minimum strict de* (P) *s'il existe un voisinage* $\mathrm{V}_\varepsilon(x^*)$ *de* x^* *tel que :*

$$f(x^*) < f(x), \forall x \in \mathrm{V}_\varepsilon(x^*) \text{ et } x \neq x^*. \tag{1.3}$$

Conditions nécessaires d'optimalité

Condition nécessaire d'optimalité d'ordre 1 :

Théorème 1.2.3 *Soit $f : \mathbb{R}^n \to \mathbb{R}$ telle que f soit différentiable au point $\overline{x} \in \mathbb{R}^n$. Soit $d \in \mathbb{R}^n$ telle que $\nabla f(\overline{x})^t d < 0$. Alors il existe $\delta > 0$ tel que $f(\overline{x} + \alpha d) < f(\overline{x})$ pour tout $\alpha \in]0, \delta[$.*
La direction d s'appelle dans ce cas direction de descente.

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

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

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

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

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

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

et par conséquent on obtient :

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

Corollaire 1.2.1 *Soit $f : \mathbb{R}^n \to \mathbb{R}$ différentiable en $\overline{x}$, si $\overline{x}$ est un minimum local alors $\nabla f(\overline{x}) = 0$.*

Preuve.
Par contre, on suppose que $\nabla f(\overline{x}) \neq 0$. Si on pose $d = -\nabla f(\overline{x})$, on obtient :

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

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

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

mais ceci est contradictoire avec le fait que $\overline{x}$ est unminimum local, d'où $\nabla f(\overline{x}) = 0$.

Condition nécessaire d'optimalité d'ordre 2 :

Théorème 1.2.4 *Soit $f : \mathbb{R}^n \to \mathbb{R}$ deux fois différentiable en $\overline{x}$. Supposons que $\overline{x}$ soit minmum local. Alors $\nabla f(\overline{x}) = 0$ et $\mathrm{H}(\overline{x})$ est semi définie positive.*

Conditions suffisantes d'optimalité

Cas non convexe

Théorème 1.2.5 *Soit $f : \mathbb{R}^n \to \mathbb{R}$ deux fois différentiable en $\overline{x}$. Si $\nabla f(\overline{x}) = 0$ et $\mathrm{H}(\overline{x})$ est définie positive, alors $\overline{x}$ est un minmum local strict.*

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

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

$$o\lambda(\overline{x}; x-\overline{x}) \to 0 \text{ pour } x \to \overline{x}.$$

Supposons que $\overline{x}$ n'est pas un minimum local strict. Donc il existe une suite x_k convergente vers $\overline{x}$ telle que :

$$f(x_k) \leq f(\overline{x}), x_k \neq \overline{x}, \forall k.$$

Posons $d_k = (x_k - \overline{x})/\|x_k - \overline{x}\|$. Donc $\|d_k\| = 1$ et on obtient à partir de (1) :

$$\frac{f(x_k) - f(\overline{x})}{\|x_k - \overline{x}\|^2} = \frac{1}{2}d_k^t\mathrm{H}(\overline{x})d_k + \lambda(\overline{x}; x_k - \overline{x}) \leq 0, \forall k \ , \ (*)$$

et comme $\|d_k\| = 1$,$\forall k$ alors $\exists\{d_k\}_{k\in N_1\subset\mathbb{N}}$ telle que $d_k \to d$ pour $k \to \infty$ et $k \in \mathrm{N}_1$.
Alors $(*)$ donne : $d^t\mathrm{H}(\overline{x})d \leq 0$, ce qui conterdit le fait que $\mathrm{H}(\overline{x})$ est définie positive car $\|d\| = 1$ (donc $d \neq 0$). Donc $\overline{x}$ est un minimum local strict.

Cas convexe

Théorème 1.2.6 *Soit $f : \mathbb{R}^n \to \mathbb{R}$ telle que f est convexe et différentiable. Alors x^* est un minimum global de f si et seulement si $\nabla f(x^*) = 0$.*

Théorème 1.2.7 *Soit $f : \mathbb{R}^n \to \mathbb{R}$ telle que f est pseudo convexe. Soit $x^* \in \mathbb{R}^n$ tel que $\nabla f(x^*) = 0$. , alors x^* est un minimum global de f.*

Remarque 1.2.1 *Les théorèmes* (1.2.6) *et* (1.2.7) *demeurent vrais si on remplace $\mathbb{R}^n$ par un ouvert* S *de $\mathbb{R}^n$.*

Remarque 1.2.2 *Dans le cas où f est convexe, alors tout minimum local est aussi global. De plus si f est strictement convexe, alors tout minimum local devient non seulement global mais aussi unique .*

1.2.2 Les algorithmes des problèmes de minimisation sans contraintes

Aspect général des algorithmes

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

$$x_{k+1} = x_k + \alpha_k d_k, \tag{1.4}$$

où d_k détermine la direction de déplacement à partir du point x_k et α_k est un facteur numérique dont le grandeur donne la longueur du pas dans la direction d_k.
Le type d'algorithme permettant de résoudre le problème (P) sera déterminé dés qu'on définit les procédes de construction du vecteur d_k et de calcul de α_k à chaque itération.
La façon avec laquelle on construit les vecteurs d_k et les scalaires α_k détermine directement les propriètés du procesuss et spécialement en ce qui concerne la convergence de la suite $\{x_k\}$, la vitesse de la convergence,....
Pour s'approcher de la solution optimale du problème (P) (dans le cas général, c'est un point en lequel ont lieu peut être avec une certaine précision les conditions nécessaires d'optimalité de f),on se déplace naturellement à partir du point x_k dans la direction de la décroissance de la fonction f.

Fonctions multivoques et algorithmes

Une application multivoque est une application A qui à $x \in \mathbb{R}^n$ fait correspondre un sous ensemble $\mathrm{A}(x)$ de $\mathbb{R}^n$.
Etant donné un point x_k. En appliquant les instructions d'un certain algorithme, on obtient un nouveau point x_{k+1}. Cette procédure peut être décrite par une application multivoque A applée application algorithmique. Donc étant donné un point $x_1 \in \mathbb{R}^n$, l'application algorithmique génère une suite $x_1, x_2, ...$, où $x_{k+1} \in \mathbb{R}^n$ pour tout k.
La notion d'application algorithmique fermé est directement liée à la convergence des algorithmes.

Définition 1.2.8 *Soient* X *et* Y *deux sous ensembles fermés non vides de* $\mathbb{R}^n$ *et* $\mathbb{R}^m$ *respectivement et* $\mathrm{A} : \mathrm{X} \Rightarrow \mathrm{Y}$ *une application multivoque.* A *est dite fermée au point* $x \in \mathbb{R}^n$ *si*

$$\left\{ \begin{array}{l} x_k \in \mathrm{X}, x_k \to x \\ y_k \in \mathrm{Y}, y_k \to y \end{array} \right\} \textit{implique que } y \in \mathrm{A}(x). \tag{1.5}$$

Donnons un théorème de convergence qui utilise les fonctions multivoques et qui nous sera utile par la suite. Avant celà, notons qu'à cause de la non convexité, de la taille du problème et d'autres

difficultés, on peut arrêter le procesuss itératif si on trouve un point appartient à un ensemble spécifique qu'on appelle ensemble des solutions Ω.

Voici ci-dessous quelques exemples typiques de cet ensemble.
$\Omega = \{x^* : \nabla f(x^*) = 0\}$.
$\Omega = \{x^* : x^*$ *est une solution optimale locale du problème* $(P)\}$.
$\Omega = \{x^* : f(x^*) < \nu^* + \varepsilon\}$ *où* $\varepsilon > 0$ *est une tolérance définie à l'avance et* ν^* *est la valeur minimale de la fonction objective.*

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

Théorème 1.2.9 *Soient* X *un ensemble non vide férmé dans* $\mathbb{R}^n$ *et* $\Omega \subseteq \mathrm{X}$ *un ensemble des solutions non vide. Soit* $\alpha : \mathbb{R}^n \to \mathbb{R}$ *une fonction continue, on considère que l'application algorithmique* $\mathrm{C} : \mathrm{X} \to \mathrm{X}$ *vérifie la propriété suivante :* " *étant donné* $x \in \mathrm{X}$ *alors* $\alpha(y) \leq \alpha(x)$ *pour* $y \in \mathrm{C}(x)$". *Soit* $\mathrm{B} : \mathrm{X} \to \mathrm{X}$ *une application algorithmique férmée sur le complémentaire de* Ω *qui vérifie :* $\alpha(y) < \alpha(x)$ *pour* $y \in \mathrm{B}(x)$ *et* $x \neq \Omega$.*Maintenant considérons l'application algorithmique composé* $\mathrm{A} = \mathrm{CB}$. *Soit* $x_1 \in \mathrm{X}$, *la suite* x_k *est générée comme suit :*
Si $x_k \in \Omega$, *stop ; sinon poser* $x_{k+1} \in \mathrm{A}(x_k)$, *remplacer* k *par* $(k+1)$ *et répéter. Supposons que* $\Lambda = x \in \mathrm{X}, \alpha(x) \leq \alpha(x_1)$ *est compact. Alors, ou bien l'algorithme s'arrête à un nombre fini d'itérations par un point* $\in \Omega$, *ou tous les points d'accumulation de* x_k *appartenants à* Ω.

1.2.3 Convergence globale des algorithmes

Définition 1.2.10 *Nous dirons qu'un algorithme décrit par une application multivoque* A, *est globalement convergent (ou encore : possède la propriété de convergence globale) si, quel que soit le point de départ* x_1 *choisi, la suite* x_k *engendrée par* $x_{k+1} \in \mathrm{A}(x_k)$ *(ou une sous suite) converge vers un point satisfaisant les conditions nécessaires d'optimalité.*

Modes et vitesse de convergence

-La convergence globale d'un algorithme ayant été établie, nous nous intéressons 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 e près (ε fixé à l'avance) de

l'optimum 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 considérablement.

-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 x^*.

-Ceci conduit à attribuer à chaque algorithme un indice d'efficacité appelé sa vitesse de convergence.

Nous introduisons ici les principales définitions de base qui seront abondamment utilisées par la suite.

-Plaçons nous dans $\mathbb{R}^n$, où $\|.\|$ désigne la norme euclidienne et considérons une suite x_k convergeant vers x^*.

$$\cdot\, Si \limsup \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} = \lambda < 1 \tag{1.6}$$

On dit que la convergence est linéaire et λ est le taux de convergence associé.

$$\cdot\, Si \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} \to 0 quand k \to \infty. \tag{1.7}$$

on dit que la convergence est superlinéaire.

Plus précisément si $\exists p > 1$ tel que :

$$\limsup_{k\to\infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^p} < +\infty. \tag{1.8}$$

on dit que la convergence est superlinéaire d'ordre p.

En particulier si :

$$\limsup_{k\to\infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^2} < +\infty. \tag{1.9}$$

on dit que la convergence est quadratique (superlinéaire d'ordre 2).

1.3 Méthodes 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 .
L'idée initiale était de trouver une suite de directions de descente permettant de résoudre le problème (p).

1.3.1 Description de la méthode

On s'intéresse dans suite à la minimisation d'une fonction $f : \mathbb{R}^n \to \mathbb{R}$

$$\min\{f(x); x \in \mathbb{R}^n\} \tag{1.10}$$

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

$$x_{k+1} = x_k + \alpha_k d_k \tag{1.11}$$

Le pas $\alpha_k \in \mathbb{R}$ étant déterminé par une recherche linéaire tel que :

$$\alpha_k = \min f(x_k + \alpha d_k) \quad , \quad \alpha > 0 \tag{1.12}$$

la direction d_k est définie par la formule de récurrence suivante ($\beta_k \in \mathbb{R}$)

$$d_k = \begin{cases} -g_k & \text{si } k = 1 \\ -g_k + \beta_k d_{k-1} & \text{si } k \geq 2. \end{cases} \tag{1.13}$$

Différentes formules de β_{k+1}

$\diamondsuit$ **Formule de Hestenes-Stiefel** Cette méthode été découverte en 1952 par Hestenes et Stiefels

$$\beta_{k+1}^{HS} = \frac{g_{k+1}^T y_k}{d_k^T y_k}, \tag{1.14}$$

où $y_k = g_{k+1} - g_k$.

$\diamondsuit$**Formule de Fletcher-Reeves** Cette méthode été découverte en 1964 par Fletcher et Reeves ,

$$\beta_{k+1}^{FR} = \frac{\|g_{k+1}\|^2}{\|g_k\|^2}. \tag{1.15}$$

♢**Formule de Polak-Ribière-Polyak** Cette méthode été découverte en 1969 par Polak, Ri-bière et Ployak ,

$$\beta_{k+1}^{PRP} = \frac{g_{k+1}^T y_k}{\|g_k\|^2}. \tag{1.16}$$

♢**Formule de la descente conjuguée** Cette méthode été découverte en 1987 par Fletcher ,

$$\beta_{k+1}^{CD} = \frac{\|g_{k+1}\|^2}{-d_k^T g_k}. \tag{1.17}$$

♢**Formule de Dai-Yuan** Cette méthode été découverte en 1999 par Dai et Yuan ,

$$\beta_{k+1}^{DY} = \frac{\|g_{k+1}\|^2}{d_k^T y_k}. \tag{1.18}$$

Remarque 1.3.1 *Dans le cas quadratique avec recherche linéaire exacte 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.

1.3.2 Algorithme de La méthode du gradient conjugué pour les fonctions quelconques

Algorithme 1.3.1 *(les différentes méthodes du gradient conjugué non linéaire)*

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

$$\alpha_k \ : calculer\ par\ la\ recherche\ linéaire$$

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

où

$$\beta_{k+1} \ : d\acute{e}fini \ \ selon \ \ la \ \ m\acute{e}thode$$

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

1.3.3 la convergence globale de les méthodes du gradient conjugué

Définition 1.3.1 *Nous dirons qu'une méthode de gradient conjugué converge globalement si l'une des deux conditions suivantes est vérifiée :*
-a) $g_{k_0} = 0$, *pour un certain* $k_0 \in \mathbb{N}$.
-b) $\liminf_{k\to\infty} \|g_k\| = 0$.

Rappelons dans ce qui suit les deux conditions suivantes que toutes les méthodes du gradient conjugué doivent vérifier (ou au moins l'une d'entre elles) pour prouver les résultats de convergence.

Conditions C1 et C2 (de Lipschitz et de bornetude)

Condition C1 Soit $f : \mathbb{R}^n \to \mathbb{R}$. On dit que f vérifie la condition C1 si f est contiuement différentiable dans un voisinage V(Γ) *de* $\Gamma = \{x \in \mathbb{R}^n : f(x) \leq f(x_1) : x_1 \in \mathbb{R}_n$ point initial $\}$
et si $\nabla f(x)$ vérifie la condition de Lipschitz dans V(Γ) , c'est à dire , il existe une constante $L > 0$ telle que :

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| : \ \forall x, y \in \mathrm{V}(\Gamma).$$

Condition C2 L'ensemble $\Gamma = \{x \in \mathbb{R}^n : f(x) \leq f(x_1)\}$ est borné, i.e., il existe une constante $B < \infty$ telle que :

$$\|x\| \leq B : \ \forall x \in \Gamma.$$

Théorème de Zoutendijk

L'outil de base utilisé par les différentes variantes du gradient conjugué avec une recherche linéaire inexacte est le théorème suivant du à Zoutendijk.

Théorème 1.3.2 *Considérons une méthode itérative quelconque 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 inéxacte de Wolfe. Supposons aussi que f vérifie la Condition C1*. Alors on a :*

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

Une autre version du théorème de Zoutendijk se trouve dans [th suivant]. Ce sera l'objet du théorème suivant :

Théorème 1.3.3 *Considérons une méthode itérative quelconque 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 inéxacte de Wolfe forte. Supposons aussi que f vérifie la Condition C1*. Alors ou bien :*

$$\liminf_{k\to\infty} \|g_k\| = 0,$$

ou

$$\sum_{k=0}^{\infty} \frac{\|g_k\|^4}{\|d_k\|^2} < \infty.$$

Utilisation du théorème de Zoutendijk pour démontrer la convergence globale

On a : $\sum_{k=0}^{\infty} \frac{(g_k^T d_k)^2}{\|d_k\|^2} < \infty$ est équivalente à $\sum_{k=1}^{\infty} \cos^2(\theta_k)\|g_k\|^2 < \infty$, où θ_k est l'angle que fait d_k avec $-g_k$.
Il est évident de voir que si :

$$\cos(\theta_k) \geq \delta > 0 \quad , \quad \forall k \in \mathbb{N},$$

alors on a :

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

Dans les différentes méthodes du gradient conjugué on n'arrive pas à démontrer le résultat précédent i.e. $\lim_{k\to\infty} \|g_k\| = 0$, mais seulement :

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

Cette condition implique qu'il existe une sous suite de $\|g_k\|$ qui tend vers zéro.

Conditions suffisantes associés au théorème de Zoutendijk pour démontrer la relation $\liminf_{k\to\infty} \|g_k\| = 0$.

Pour démontrer $\liminf_{k\to\infty} \|g_k\| = 0$ on associe au théorème de Zoutendijk les 2 hypothèses suivantes :

Hypothèse1

La descente suffisante est assurée, i.e., il existe une constante c indépendante de k telle que :

$$g_k^T d_k \leq -c\|g_k\|^2.$$

Hypothèse2

Il existe une constante β telle que :

$$\|d_k\|^2 \leq \beta k.$$

On a donc

$$\left\{ \begin{array}{c} \textbf{relation de Zoutendijk} \\ \textbf{Hypothése1} \\ \textbf{Hypothése2} \end{array} \right\} \Rightarrow \{\liminf_{k\to\infty} \|g_k\| = 0\}.$$

Remarque 1.3.2 *:*
-La condition ***Hypothèse 1*** *est plus forte que la descente.En effet :*

$$g_k^T d_k \leq -\|g_k\|^2 \Rightarrow g_k^T d_k < 0.$$

-La condition ***Hypothèse 1*** *n'est pas nécessaire pour avoir :*

$$\liminf_{k\to\infty} \|g_k\| = 0. \tag{1.19}$$

Dai-Yuan ont montré la convergence globale seulement avec la condition de descente :

$$g_k^T d_k < 0.$$

-D'autre part la condition ***Hypothèse1****,(i.e.*$g_k^T d_k \leq -c\|g_k\|^2$*) n'est pas suffisante toute seule, pour avoir la convergence globale.*

Chapitre 2

Optimisation avec contrainte convexe

Soit C un ensemble convexe fermé non vide de $\mathbb{R}^n$, et $F : \mathbb{R}^n \to \mathbb{R}^n$ est une fonction continue et monotone.
Autrement dit, $\forall x, y \in \mathbb{R}^n$, nous avons :

$$(F(x) - F(y))^T (x - y) \geq 0. \tag{2.1}$$

Nous considérons les équations avec contraintes convexes suivantes, notées CCE(F, C) :

$$F(x^*) = 0 \quad , x^* \in \mathrm{C}. \tag{2.2}$$

Dans ce chapitre on va essayer de présenter quelques des méthodes pour la minimisation des fonctions avec contraintes convexes. ,Nous allons introduire quelques préliminaires et méthodes traditionelles pour résoudre (CCE). On terminera ce chapitre par la nouvelle méthode et prouver sa convergence globale.

2.1 Préliminaires

Rappellons d'abord les résultats fondamentals suivant :

2.1.1 Projection sur un convexe fermé

Théorème 2.1.1 *Soient* $\mathrm{C} \in \mathbb{R}^n$ *un ensemble convexe fermé non-vide ,* $x \in \mathbb{R}^n$*. Alors il existe un unique élément* $\mathrm{P}_x \in \mathrm{C}$ *[on peut le noter pour plus de précision par* $\mathrm{P}_\mathrm{C}(x)$*], tel que :*

$$\|x - \mathrm{P}_\mathrm{C}(x)\| = \inf_{y \in \mathrm{C}} \|x - y\| = \min_{y \in \mathrm{C}} \|x - y\|.$$

Ou bien :

$$\mathrm{P_C}(x) = \arg\min\{\|x-y\|, y \in \mathrm{C}\}. \tag{2.3}$$

L'élément $\mathrm{P_C}(x) \in \mathrm{C}$ *s'appelle la projection de* x *sur* C *en* $\mathbb{R}^n$. *En plus on a :*

a) *L'élément* $\mathrm{P_C}(x) \in \mathrm{C}$ *satisfait aussi :*

$$\langle x - \mathrm{P_C}(x), y - \mathrm{P_C}(x)\rangle \leq 0 \ , \ \forall y \in \mathrm{C}. \tag{2.4}$$

b) *Si* $z \in \mathrm{C}$ *est tel que :*

$$\langle x - z, y - z\rangle \leq 0 \ , \ \forall y \in \mathrm{C}. \tag{2.5}$$

alors :$z = \mathrm{P_C}(x)$ (*donc* $z = \mathrm{P_C}(x)$ *est l'unique élément de* C *satisfaisant* (2.5)).

c) *On a :*

$$\|\mathrm{P_C}(x_1) - \mathrm{P_C}(x_2)\| \leq \|x_1 - x_2\| \ , \ \forall x_1, x_2 \in \mathbb{R}^n. \tag{2.6}$$

(*la fonction* $\mathrm{P_C}$ *n'augmente pas les distances*). *Ceci implique que* $\mathrm{P_C}$ *est une fonction lipschitzienne, donc continue.*

d) *On a :*

$$x = \mathrm{P_C}(x) \ , \quad \textit{si et seulement si} \quad x \in \mathrm{C}. \tag{2.7}$$

e) *Si* C *est le sous-espace affine de* $\mathbb{R}^n$ *donné par* $\mathrm{C} = a + \mathrm{C}_0$ *avec* $a \in \mathbb{R}^n$ *et* C_0 *un sous-espace vectoriel de* $\mathbb{R}^n$ *alors l'inégalité (**a**) devient l'égalité :*

$$\langle x - \mathrm{P_C}(x), y\rangle = 0 \ , \quad \forall x \in \mathrm{C}_0. \tag{2.8}$$

(*c'est à dire* $x - \mathrm{P_C}(x) \perp \mathrm{C}_0$).

Idée de la preuve :

Tout revient à minimiser sur C, la fonction $g : \mathbb{R}^n \to \mathbb{R}$ définie par :

$$g(y) = \|y - x\|^2.$$

Montrons d'abord que g est une fonction elliptique. Pour cela, On peut écrire :

$$g(y) = \langle x - y, x - y\rangle = \langle y, y\rangle - 2\langle y, x\rangle + \langle x, x\rangle.$$

ce qui donne :

$$\nabla g(y) = 2y - 2x,$$

$$\nabla^2 g(y) = 2I_n \text{ (}\textit{donc constante indpendente de } y\text{)}.$$

Comme $2I_n$ est une matrice symmétrique et définie positive (matrice SDP), alors g est elliptique. Comme C est fermé convexe alors il existe un unique élément $\mathrm{P}_x = \mathrm{P}_{\mathrm{C}}(x) \in \mathrm{C}$, tel que $g(\mathrm{P}_{\mathrm{C}}(x)) \leq g(y), \forall y \in \mathrm{C}$ c'est à dire :

$$\|x - \mathrm{P}_{\mathrm{C}}(x)\| \leq \|x - y\| \ , \ \forall y \in \mathrm{C}.$$

ce qui donne le résultat principal du théorème. Nous avons aussi :

$$\langle \nabla g(\mathrm{P}_{\mathrm{C}}(x)), y - \mathrm{P}_{\mathrm{C}}(x) \rangle \geq 0 \ , \ \forall y \in \mathrm{C},$$

ce qui nous donne facilement :

$$\langle x - \mathrm{P}_{\mathrm{C}}(x), y - \mathrm{P}_{\mathrm{C}}(x) \rangle \leq 0 \ , \ \forall y \in \mathrm{C}.$$

Ce qui prove **a)**.

-La partie **b)** est une conséquence de l'ellipticité et de l'unicité.

D'autre part, soient $x_1, x_2 \in \mathbb{R}^n$. On a alors :

$$\langle x_1 - \mathrm{P}x_1, y - \mathrm{P}x_1 \rangle \leq 0 \ , \ \forall y \in \mathrm{C},$$

$$\langle x_2 - \mathrm{P}x_2, y - \mathrm{P}x_2 \rangle \leq 0 \ , \ \forall y \in \mathrm{C}.$$

En prennant $y = \mathrm{P}x_2$ dans la première inégalité et $y = \mathrm{P}x_1$ dans la deuxième et en faisantla somme, on trouve :

$$\langle \mathrm{P}x_2 - \mathrm{P}x_1, x_1 - \mathrm{P}x_1 + \mathrm{P}x_2 \rangle \leq 0,$$

ce qui donne :

$$\|\mathrm{P}x_2 - \mathrm{P}x_1\|^2 \leq \langle \mathrm{P}x_2 - \mathrm{P}x_1, x_2 - x_1 \rangle \leq \|\mathrm{P}x_2 - \mathrm{P}x_1\| \|x_2 - x_1\|.$$

(on a utilisé l'inégalité de Cauchy-Schwarz).

- Ceci nous donne **c)** par simplification.

-La partie **d)** est évidente.

-Pour la partie **e)**, soient C_0 un sous-espace vectoriel de $\mathbb{R}_n$, $a \in \mathbb{R}_n$, $\mathrm{C} = a + \mathrm{C}_0$ et $h \in \mathrm{C}_0$

arbitraire et fixé. On va prendre en (1) : $y = \mathrm{P}x \pm h$ qui est un élément de C car $\mathrm{P}x \in \mathrm{C}$ et $h \in \mathrm{C}_0$. Ceci donne :

$$\pm\langle x - \mathrm{P}x, h\rangle \leq 0,$$

d'où on déduit :

$$\langle x - \mathrm{P}x, h\rangle = 0.$$

ce qui finit la preuve.

2.1.2 Méthode de gradient projeté

Revenons au problème de minimisation suivant :

$$\{\min f(x), x \in \mathbb{R}^n, \quad \textit{sous la contrainte} : \ x \in \mathrm{C}\} \tag{2.9}$$

On supposera dans la suite que l'ensemble C des contraintes est convexe et fermeé.
Supposons que $x^* \in \mathrm{C}$ tel que :

$$f(x^*) = \min_{x \in \mathrm{C}} f(x) \tag{2.10}$$

Alors :

$$\langle \nabla f(x^*), x - x^*\rangle \geq 0 \ , \ \forall x \in \mathrm{C}. \tag{2.11}$$

qui est équivalente pour tout $\alpha > 0$ à l'inégalité :

$$\alpha\langle -\nabla f(x^*), x - x^*\rangle \leq 0 \ , \ \forall x \in \mathrm{C}.$$

Ceci donne :

$$\langle [x^* - \alpha\nabla f(x^*)] - x^*, x - x^*\rangle \leq 0 \ , \ \forall x \in \mathrm{C}.$$

En utilisant le Théorème de projection cette dernière inégalité est équivalente à l'égalitè :

$$x^* = \mathrm{P}_{\mathrm{C}}(x^* - \alpha\nabla f(x^*)).$$

Alors la solution du problème de minimisation est un point fixe de l'application $g : \mathrm{C} \mapsto \mathrm{R}^n$ donnée par :

$$g(x) = \mathrm{P}_{\mathrm{C}}(x - \alpha\nabla f(x)).$$

Une idée naturelle pour approcher numériquement ce point fixe est d'utiliser une méthode iterative, qui consiste à construire une suite $x_k \in \mathrm{C}$ par reccurence :

$$x_{k+1} = \mathrm{P}_{\mathrm{C}}(x_k - \alpha_k\nabla f(x_k)) \ , \ k \in N,$$

Avec $\alpha > 0$ donné. Il est aussi possible de faire varier α à chaque pas.
Donc l'algorithme général sera dans ce cas :

$$\begin{cases} x_{k+1} = \mathrm{P_C}(x_k - \alpha_k \nabla f(x_k)) \ , \ k \in N \\ x_0 \qquad\qquad\qquad\qquad\qquad\qquad \text{donné} \in \mathrm{C}. \end{cases} \tag{2.12}$$

avec $\alpha_k > 0$ données. C'est la méthode de gradient avec projection à pas variable.
On l'appelle méthode de gradient avec projection à pas fixe si α_k est indépendent de k.

2.1.3 Agorithme de gradient projeté et leur convergence

Algorithme 2.1.1 *(Méthode de gradient projeté)*

Etape0 : (initialisation)

— *Soit* x_0 *le point de départ,* $\mathrm{P_C}$ *un opérateur de projection sur* C

— *Poser* $k = 0$ *et aller à l'étape 1.*

Etape 1 :

— *Si* $\nabla f(x_k) = 0$ *: STOP* $(x^* = x_k)$*." Test d'arrêt"*

— *Si non aller à l'étape 2.*

Etape 2 :

— *Définir* $x_{k+1} = \mathrm{P_C}(z_k)$ *avec :*

$$z_k = x_k - \alpha_k \nabla f(x_k)$$

α : *est le pas calculé par la méthode de gradient choisie*

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

Théorème 2.1.2 *Soient* $\mathrm{C} \in \mathbb{R}^n$ *un ensemble fermé convexe et non vide ,et* $f : \mathbb{R}^n \to \mathbb{R}$ *une fonction telle que* ∇f *est lipschitzienne.*
(Donc il existe $M > 0$ *et* $c > 0$ *tels que* $\forall x, y \in \mathbb{R}^n$ *on a :*

$$\langle \nabla f(x) - \nabla f(y), x - y \rangle \geq c\|x - y\|^2,$$

et

$$\|\nabla f(x) - \nabla f(y)\| \leq M\|x - y\|).$$

Supposons qu'il existe des nombres réels a_1 et a_2 satisfaisant :

$$0 < a_1 \leq a_2 < \frac{2c}{M^2},$$

tels que :

$$a_1 \leq \alpha_k \leq a_2 \quad \forall k \in N.$$

Alors la méthode (2.12) *converge.*
(*c'est à dire, il existe* $C \geq 0$ *et* $\beta \in [0, 1[$ *tels que* $\|x_k - x^*\| \leq C\beta_k, \quad \forall k \in N$)

Preuve.
Par définition nous avons :

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

et aussi

$$x^* = \mathrm{P_C}(x^* - \alpha_k \nabla f(x^*)).$$

En faisant la différence et en utilisant la Partie **c)** du (Théorème de projection), on déduit :

$$\|x_{k+1} - x^*\| \leq \|x_k - x^* - \alpha_k[\nabla f(x_k) - \nabla f(x^*)]\|.$$

c'est à dire :

$$\|x_{k+1} - x^*\|^2 \leq \|x_k - x^*\|^2 + \alpha_k^2 \|\nabla f(x_k) - \nabla f(x^*)\|^2 - 2\alpha_k \langle x_k - x^*, \nabla f(x_k) - \nabla f(x^*)\rangle.$$

En utilisant les hypothèses sur f on déduit :

$$\|x_{k+1} - x^*\|^2 \leq (1 - 2c\alpha_k + M^2\alpha_k^2)\|x_k - x^*\|^2.$$

Remarque 2.1.1 *Cette méthode est applicable uniquement si on peut calculer facilement la projection* $\mathrm{P_C}$.

2.2 Quelles que méthodes traditionnel pour résoudre CCE(F, C)

2.2.1 Méthode de gradient conjugué descente (CGD)

Dans cette section, nous allons étendre la méthode du gradient conjugué descente avec β_k définie par :

$$\beta_k^{HZ} = \frac{1}{\langle d_{k-1}, y_{k-1}\rangle} \langle y_{k-1} - 2d_{k-1} \frac{\|y_{k-1}\|^2}{\langle d_{k-1}, y_{k-1}\rangle}, g_k\rangle, \tag{2.13}$$

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

Pour résoudre CCE(F, C) .Nous énonçons les étapes suivantes :

Algorithme 2.2.1 *(Méthode de* CGD*)*

Etape0 : (initialisation)

— *Soient $x_0 \in$ C le point de départ, les paramètre $0 < \rho < 1, \beta_k > 0, 0 < \sigma < 1$.*

— *Poser $k = 0$ et aller à l'étape 1.*

Etape 1 :

— *Si $F(x_k) = 0$: STOP, si non aller à l'étape 2.*

Etape 2 :

— *Définir d_k par :*

$$d_k = \begin{cases} -F(x_k), & si\ k = 0, \\ -F(x_k) + \eta d_{k-1}, & si\ k \geq 1. \end{cases} \tag{2.14}$$

Avec :

$$\eta = \max\{0, \beta_k^{HZ}\},$$

$$\beta_k^{HZ} = \frac{1}{\langle d_{k-1}, y_{k-1}\rangle}\langle y_{k-1} - 2d_{k-1}\frac{\|y_{k-1}\|^2}{\langle d_{k-1}, y_{k-1}\rangle}, F(x_k)\rangle \tag{2.15}$$

Tels que :

$$y_k = F(x_{k+1}) - F(x_k) + \lambda_k \alpha_k \|F(x_k)\| d_k$$

$$\lambda_k = 1 + \|F(x_k)\|^{-1} \max\left\{0, -\frac{\langle F(x_{k+1}) - F(x_k), \alpha_k d_k\rangle}{\|\alpha_k d_k\|^2}\right\}$$

$$\alpha_k = \beta_k \rho^{m_k}$$

Avec m_k étant Le plus petit entier non négatif m tel que :

$$-\langle F(x_k + \alpha_k d_k), d_k\rangle \geq \sigma \alpha_k \|d_k\|^2 \tag{2.16}$$

— *Si $d_k = 0$: STOP, si non aller à l'étape 3.*

Etape 3 :

— *Trouvez le point d'essai $z_k = x_k + \alpha_k d_k$*

— *Si $F(z_k) = 0$: STOP, Si non aller à l'étape 4.*

Etape 4 :

— *Définir* x_{k+1} *par*

$$x_{k+1} = P_c[x_k - \xi_k F(z_k)], \tag{2.17}$$

Avec :

$$\xi_k = \frac{\langle F(z_k), x_k - z_k\rangle}{\|F(z_k)\|^2}, \tag{2.18}$$

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

Remarque 2.2.1 *Soit* $\{d_k\}$ *la direction générée par (2.14), supposons que* $\langle y_{k-1}, d_{k-1}\rangle \neq 0$, *pour tout* $k > 1$, *il obtient :*

$$\langle F_k, d_k\rangle \leq -\frac{7}{8}\|F_k\|^2. \tag{2.19}$$

2.2.2 La convergence globale de (CGD)

Dans ce suit supposons que F vérifie la condition de Lipschitz dans C.

Lemme 2.2.1 *La recherche de linéaire est bien définie. C'est-à-dire, pour tout* $k > 0$,*il existe un nombre positif* α_k *qui satisfait* (2.16).

Lemme 2.2.2 *Supposons que* $F(.)$ *soit monotone et continue sur* $\mathbb{R}_n$,*soit* $\{x_k\}$ *une générées par l'Algorithme* (2.2.1), *alors il existe* $M > 0$, *tel que :*

$$\|F(x_k)\| \leq M. \tag{2.20}$$

Preuve.

Soit $\overline{x} \in$ C *solution de* (2.2),*On obtient :*

$$\begin{aligned}
\|x_{k+1} - \overline{x}\|^2 &= \|\mathrm{P_C}[x_k - \xi_k F(z_k)] - \overline{x}\|^2 \\
&\leq \|x_k - \xi_k F(z_k) - \overline{x}\|^2 \\
&= \|x_k - \overline{x}\|^2 - 2\xi_k\langle F(z_k), x_k - \overline{x}\rangle + \xi_k^2\|F(z_k)\|^2 \\
&\leq \|x_k - \overline{x}\|^2 - 2\xi_k\langle F(z_k), x_k - z_k\rangle + \xi_k^2\|F(z_k)\|^2 \\
&= \|x_k - \overline{x}\|^2 - \frac{\langle F(z_k), x_k - z_k\rangle^2}{\|F(z_k)\|^2} \\
&\leq \|x_k - \overline{x}\|^2.
\end{aligned} \tag{2.21}$$

Alors $\|x_k - \overline{x}\| \leq \|x_0 - \overline{x}\|$, $\forall k$ *donc :*

$$\|F(x_k)\| = \|F(x_k) - F(\overline{x})\| \leq L\|x_0 - \overline{x}\|.$$

On prend : $M = L\|x_0 - \overline{x}\|$.

Lemme 2.2.3 *Supposons que* $F(.)$ *soit monotone et continue sur* $\mathbb{R}^n$ *, et soit* $\{x_k\}$ *et* $\{z_k\}$ *les suites générées par l'Algorithme* (2.2.1)*, alors* $\{x_k\}$ *et* $\{z_k\}$ *sont bornées,*

$$\lim_{k\to\infty} \|x_k - z_k\| = 0.$$

En particulie :

$$\lim_{k\to\infty} \alpha_k \|d_k\| = 0. \tag{2.22}$$

Théorème 2.2.1 *Supposons que les conditions dans Lemme* (2.2.3) *restent. Alors la suite* $\{x_k\}$ *générée par Algorithme* (2.2.1) *vérifié :*

$$\liminf_{k\to\infty} \|F_k\| = 0. \tag{2.23}$$

Preuve. (*voir [8]*)

2.2.3 Méthode de gradient conjugué auto-adaptatif à trois termes (TTGD)

Dans cette section, nous allons étendre la méthode du gradient conjugué auto-adaptatif à trois termes avec β_k définie par :

$$\beta_k^{HS} = \frac{g_k^T y_{k-1}}{d_{k-1}^T y_{k-1}}. \tag{2.24}$$

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

Pour résoudre CCE(F, C) .Nous énonçons les étapes suivantes.

Algorithme 2.2.2 *(Méthode de* TTGD*)*

Etape0 : (initialisation)

— *Soient* $x_0 \in$ C *le point de départ, les paramètre* $0 < \rho < 1, 0 < \gamma < 2, 0 < \sigma < \delta, \tau > 0, \eta > 0$.

— *Poser* $k = 0$ *et aller à l'étape 1.*

Etape 1 :

— *Si* $\|F(x_k)\| = 0$ *: STOP, si non aller à l'étape 2.*

Etape 2 :

— *Si* $k = 0$ *ou* $s_{k-1}^T y_{k-1} < \sigma\|F(x_{k-1})\|^\tau s_{k-1}^T s_{k-1}$ *ou* $s_{k-1}^T y_{k-1} > \delta\|F(x_{k-1})\|^\tau s_{k-1}^T s_{k-1}$

Définir d_k *par :*

$$d_k = -F(x_k). \tag{2.25}$$

— *Sinon*

$$d_k = -F(x_k) + \beta_k^{HS} d_{k-1} - \theta_k y_{k-1}, \tag{2.26}$$

Avec

$$\beta_k^{HS} = \frac{F(x_k)^T y_{k-1}}{d_{k-1}^T y_{k-1}},\ \theta_k = \frac{F(x_k)^T d_{k-1}}{d_{k-1}^T y_{k-1}} \tag{2.27}$$

$$s_{k-1} = z_{k-1} - x_{k-1}, y_{k-1} = F(z_{k-1}) - F(x_{k-1}). \tag{2.28}$$

— *Si $d_k = 0$: STOP, Si non aller à l'étape 3.*

Etape 3 :

— *Trouvez le point d'essai $z_k = x_k + \alpha_k d_k$ avec $\alpha_k = \rho^{m_k}$ avec m_k étant Le plus petit entier non négatif m tel que :*

$$-\langle F(x_k + \alpha_k d_k), d_k\rangle \geq \eta\alpha_k \|d_k\|^2 \tag{2.29}$$

— *Si $F(z_k) = 0$: STOP, Si non aller à l'étape 4.*

Etape 4 :

— *Définir x_{k+1} par :*

$$x_{k+1} = P_c[x_k - \gamma\xi_k F(z_k)], \tag{2.30}$$

avec :

$$\xi_k = \frac{\langle F(z_k), x_k - z_k\rangle}{\|F(z_k)\|^2} \tag{2.31}$$

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

Remarque 2.2.2 (2.25) *et* (2.26) *implique que : $F(x_k)^T d_k = -\|F(x_k)\|^2$.Par l'inégalité de Cauchy-Schwarz, nous avons :*

$$\|F(x_k)\| \leq \|d_k\|. \tag{2.32}$$

D'autre part, de (2.26)*, nous avons :*

$$\begin{aligned}\|d_k\| &\leq \|F(x_k)\| + |\beta_k^{HS}|\|d_{k-1}\| + |\theta_k|\|y_{k-1}\| \\ &= \|F(x_k)\| + |\frac{F(x_k)^T y_{k-1}}{d_{k-1}^T y_{k-1}}|\|d(x_k)\| + |\frac{F(x_k)^T d_{k-1}}{d_{k-1}^T y_{k-1}}|\|y(x_k)\| \\ &\leq 3\|F(x_k)\|.\end{aligned} \tag{2.33}$$

Combinant (2.32) *et* (2.33)*, nous avons :*

$$\|F(x_k)\| \leq \|d_k\| \leq 3\|F(x_k)\|. \tag{2.34}$$

2.2.4 La convergence globale de (TTGD)

Lemme 2.2.4 *Supposons que $F(.)$ est monotone et continue sur $\mathbb{R}^n$, et soit $\{x_k\}$ et $\{z_k\}$ les suites générées par l'Algorithme* (2.2.2)*,Ensuite nous avons :*

$$\alpha_k \geq \rho \min\{1, \frac{\|F(x_k)\|^2}{(L+\eta)\|d_k\|^2}\}. \tag{2.35}$$

Preuve.

Si $\alpha_k \neq \rho$,nous avons $\frac{\alpha_k}{\rho}$ ne satisfait pas (2.29) c'est :

$$-F(x_k + \frac{\alpha_k}{\rho}d_k)^T < \eta\frac{\alpha_k}{\rho}\|d_k\|^2.$$

En combinant ceci avec (2.30), nous avons

$$\begin{aligned}
\|F(x_k)\|^2 &= -F(x_k)^T d_k \\
&= (F(x_k + \frac{\alpha_k}{\rho}d_k) - F(x_k))^T d_k - F(x_k + \frac{\alpha_k}{\rho}d_k)^T d_k \\
&\leq L\frac{\alpha_k}{\rho}\|d_k\|^2 + \eta\frac{\alpha_k}{\rho}\|d_k\|^2 \\
&= \frac{L+\eta}{\rho}\alpha_k\|d_k\|^2,
\end{aligned}$$

Où l'inégalité tient puisque $F(.)$ est une fonction continue (Lipschitz).implique :

$$\alpha_k \geq \rho\frac{\|F(x_k)\|^2}{(L+\eta)\|d_k\|^2}.$$

Cela complète la preuve.

Lemme 2.2.5 *Supposons que $F(.)$ est monotone et continue sur $\mathbb{R}^n$, et soit $\{x_k\}$ et $\{z_k\}$ les suites générées par l'Algorithme* (2.2.2)*.Ensuite nous avons :*

$$\|x_{k+1} - x^*\|^2 \leq \|x_k - x^*\|^2 - \gamma(2-\gamma)\frac{\eta^2\|x_k - w_k\|^4}{\|F(w_k)\|^2},$$

Où $x^* \in$ C, et la suite $\{x_k\}$ et $\{z_k\}$ sont bornées.c-à-d :

$$\lim_{k\to\infty}\|x_k - z_k\| = 0.$$

Théorème 2.2.2 *Supposons que $F(.)$ est monotone et continue. Alors la suite $\{x_k\}$ générée par Algorithme* (2.2.2)*vérifié :*

$$\liminf_{k\to\infty}\|F_k\| = 0. \tag{2.36}$$

Preuve.(voir [15])

2.3 Nouveau méthode de projection par gradient conjugué hybride

Dans cette section, nous proposons une nouvelle méthode pour (2.2), Cette méthode est considéré comme un combinaison de la méthode du gradient conjugué et Le célèbre hyperplan méthode de projection.

2.3.1 Algorithme de projection par gradient conjugué hybride

Dans cette section, nous allons étendre la méthode du gradient conjugué avec β_k définie par :

$$\beta_k^N = \frac{\|g_k\|^2 - \max\{0, \frac{\|g_k\|}{\|g_{k-1}\|} g_k^T g_{k-1}\}}{\max\{\mu \|d_{k-1}\| \|g_k\|, d_{k-1}^T (g_k - g_{k-1})\}}, \tag{2.37}$$

Pour résoudre CCE(F, C) .Nous énonçons les étapes de la nouvelle méthode comme suit :

Algorithme 2.3.1 *(Méthode de projection par gradient conjugué hybride)*

Etape0 : (initialisation)

— *Soient* $x_0 \in \mathrm{C}$ *le point de départ, les paramètre* $0 < \rho < 1, \mu > 1, 0 < \gamma < 2, \sigma > 0$.

— *poser* $\beta_0 = 1$ *est* $0 < \beta_{\min} < \beta_{\max}$.

— *Poser* $k = 0$ *et aller à l'étape 1.*

Etape 1 :

— *Si* $F(x_k) = 0$ *: STOP.*

— *Si non aller à l'étape 2.*

Etape 2 :

— *Définir* d_k *par*

$$d_k = \begin{cases} -F(x_k), & si\ k = 0, \\ -F(x_k) + \beta_k^{NN} d_{k-1}, & si\ k \geq 1. \end{cases} \tag{2.38}$$

avec :

$$\beta_k^{NN} = \frac{\|F(x_k)\|^2 - \max\{0, \frac{\|F(x_k)\|}{\|F(x_{k-1})\|} F(x_k)^T F(x_{k-1})\}}{\max\{\mu \|d_{k-1}\| \|F(x_k)\|, d_{k-1}^T (F(x_k) - F(x_{k-1}))\}}, \tag{2.39}$$

— *Si* $d_k = 0$ *: STOP .*

— *Si non aller à l'étape 3.*

Etape 3 :

— *Trouvez le point d'essai* $z_k = x_k + \alpha_k d_k$,*avec* $\alpha_k = \beta_k \rho^{m_k}$ *avec* m_k *étant Le plus petit entier non négatif* m *tel que :*

$$-\langle F(x_k + \alpha_k d_k), d_k\rangle \geq \sigma \alpha_k \|d_k\|^2 \tag{2.40}$$

— *Si* $F(z_k) = 0$ *: STOP.*

— *Si non aller à l'étape 4.*

Etape 4 :

— *Définir* x_{k+1} *par*

$$x_{k+1} = P_c[x_k - \gamma \xi_k F(z_k)]. \tag{2.41}$$

avec :

$$\xi_k = \frac{\langle F(z_k), x_k - z_k\rangle}{\|F(z_k)\|^2}. \tag{2.42}$$

— *Choisissez une longueur d'étape initiale* β_{k+1} *telle que* $\beta_{k+1} \in [\beta_{min}, \beta_{max}]$.

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

Remarque 2.3.1 *Évidemment,la direction de recherche* d_k *définie par* (2.38) *et le paramètre* β_k^{NN} *Définies par* (2.39) *sont similaires à celles de* (2.37) *et* (2.38)*, respectivement. Par conséquent, l'algorithme* (2.3.1) *peut être considéré comme une méthode hybride de les méthodes* HS *et* DY*.*

Remarque 2.3.2 *le paramètre* β_k^{NN} *Défini par* (2.39) *est non négatif. En fait, Cauchy-Schwartz L'inégalité, nous avons :*

$$\begin{aligned}
\beta_k^{NN} &= \frac{\|F(x_k)\|^2 - \max\{0, \frac{\|F(x_k)\|}{\|F(x_{k-1})\|} F(x_k)^T F(x_{k-1})\}}{\max\{\mu \|d_{k-1}\| \|F(x_k)\|, d_{k-1}^T (F(x_k) - F(x_{k-1}))\}} \\
&\geq \frac{\|F(x_k)\|^2 - \max\{0, \frac{\|F(x_k)\|}{\|F(x_{k-1})\|} \|F(x_k)\| \|F(x_{k-1})\|\}}{\max\{\mu \|d_{k-1}\| \|F(x_k)\|, d_{k-1}^T (F(x_k) - F(x_{k-1}))\}}, \\
&= 0.
\end{aligned}$$

Lemme 2.3.1 *La direction de recherche* d_k *définie par* (2.38) *vérifie :*

$$F(x_k)^T d_k \leq -(1 - \frac{1}{\mu})\|F(x_k)\|^2, \qquad \forall k \geq 0. \tag{2.43}$$

Preuve.

Pour $k = 0$.On a $d_0 = -F(x_0)$, et $F(x_0)^T d_0 = -\|F(x_0)\|^2$, qui vérifie (2.43) avec $\mu > 1$.

On suppose que (2.43)est satisfaite pour un entier $k - 1$.

Maintenant nous commençons à prouver que (2.43) est Satisfait pour k.

i) Si $F(x_k)^T F(x_{k-1}) \leq 0$. et d_{k-1} satisfaisant (2.43),on a $\|d_{k-1}\| > 0$.

Ainsi, à partir de (2.39), on obtient :

$$\beta_k^{NN} = \frac{\|F(x_k)\|^2}{\max\{\mu\|d_{k-1}\|\|F(x_k)\|, d_{k-1}^T(F(x_k) - F(x_{k-1}))\}} \leq \frac{\|F(xk)\|^2}{\mu\|d_{k-1}\|\|F(x_k)\|}.$$

Puis, de (2.38), on peut déduire que :

$$\begin{aligned} F(x_k)^T d_k &\leq -\|F(x_k)\|^2 + \frac{\|F(x_k)\|^2}{\mu\|d_{k-1}\|\|F(x_k)\|} F(x_k)^T d_{k-1} \\ &\leq -\|F(x_k)\|^2 + \frac{\|F(x_k)\|^2}{\mu\|d_{k-1}\|\|F(x_k)\|}\|F(x_k)\|\|d_{k-1}\| \\ &= -(1 - \frac{1}{\mu})\|F(x_k)\|^2. \end{aligned}$$

ii) Si $F(x_k)^T F(x_{k-1}) > 0$ et $d_{k-1}^T(F(x_k) - F(x_{k-1})) \geq \mu\|d_{k-1}\|\|F(x_k)\|$, puis :

$$\begin{aligned} F(x_k)^T d_k &= -\|F(x_k)\|^2 + \frac{\|F(x_k)\|^2 - \frac{\|F(x_k)\|}{\|F(x_{k-1})\|} F(x_k)^T F(x_{k-1})}{d_{k-1}^T(F(x_k) - F(x_{k-1}))} F(x_k)^T d_{k-1} \\ &\leq -\|F(x_k)\|^2 + \frac{\|F(x_k)\|^2 - \frac{\|F(x_k)\|}{\|F(x_{k-1})\|} F(x_k)^T F(x_{k-1})}{d_{k-1}^T(F(x_k) - F(x_{k-1}))}\|F(x_k)\|\|d_{k-1}\| \\ &\leq -\|F(x_k)\|^2 + \frac{\|F(x_k)\|^2}{\mu\|F(x_k)\|\|d_{k-1}\|}\|F(x_k)\|\|d_{k-1}\| \\ &= -(1 - \frac{1}{\mu})\|F(x_k)\|^2. \end{aligned}$$

iii) Si $F(x_k)^T F(x_{k-1}) > 0$ et $d_{k-1}^T(F(x_k) - F(x_{k-1})) < \mu\|d_{k-1}\|\|F(x_k)\|$, puis :

$$\begin{aligned} F(x_k)^T d_k &\leq -\|F(x_k)\|^2 + \frac{\|F(x_k)\|^2 - \frac{\|F(x_k)\|}{\|F(x_{k-1})\|} F(x_k)^T F(x_{k-1})}{\mu\|F(x_k)\|\|d_{k-1}\|}\|F(x_k)\|\|d_{k-1}\| \\ &\leq -\|F(x_k)\|^2 + \frac{\|F(x_k)\|^2}{\mu\|F(x_k)\|\|d_{k-1}\|}\|F(x_k)\|\|d_{k-1}\| \\ &= -(1 - \frac{1}{\mu})\|F(x_k)\|^2. \end{aligned}$$

Par conséquent, pour tout $k \geq 0$, l'équation (2.43) tient toujours.

Lemme 2.3.2 *Pour tout $k \geq 0$, nous avons :*

$$c_1\|F(x_k)\| \leq \|d_k\| \leq c_2\|F(x_k)\|. \tag{2.44}$$

Où c_1, c_2 sont des constantes positives.

Preuve.

Tout d'abord, à partir de (2.43) et l'inégalité de Cauchy-Schwartz, on obtient :

$$\|d_k\| \geq (1 - \frac{1}{\mu})\|F(x_k)\|.$$

D'autre part, de la définition de d_k, nous avons :

$$\begin{aligned}\|d_k\| \leq \|F(x_k)\| + \beta_k^{NN}\|d_{k-1}\| &\leq \|F(x_k)\| + \frac{\|F(x_k)\|^2}{\mu\|F(x_k)\|\|d_{k-1}\|}\|d_{k-1}\| \\ &= -(1 - \frac{1}{\mu})\|F(x_k)\|.\end{aligned}$$

Combiner les deux inégalités ci-dessus avec $c_1 = 1 - 1/\mu$, $c_2 = 1 + 1/\mu$ implique que (2.43) pour tout $k \geq 0$.

Remarque 2.3.3 *L'équation* (2.44) *indique que nous obtenons une solution de* CCE(F, C)*, si* $d_k = 0$.
Par conséquent, Algorithm (2.3.1) *est terminé si* $d_k = 0$.
De plus, à partir de (2.40)*, on obtient* $d_k = 0$ *si* $F(z_k) = 0$.
Ainsi, l'algorithme (2.3.1) *est également terminé si* $F(z_k) = 0$.

Lemme 2.3.3 *La recherche de linéaire dans* (2.40) *est bien définie. C'est-à-dire, pour tout* $k \geq 0$*, il existe un entier non négatif* m_k *satisfaisant* (2.40).

Preuve.

(Par contradiction)
Supposons qu'il existe $k_0 \geq 0$ tel que (2.40) non satisfait pour tout entier non négatif m, c'est-à-dire :

$$-\langle F(x_{k_0} + \beta_{k_0}\rho^m d_{k_0}), d_{k_0}\rangle < \sigma\beta_{k_0}\rho^m\|d_{k_0}\|^2, \quad \forall m \geq 1.$$

En laissant $m \to \infty$ et en utilisant la continuité de $F(.)$ donc :

$$-\langle F(x_{k_0}), d_{k_0}\rangle \leq 0,$$

En combinant avec la propriété de descente suffisante (2.43), on a $F(x_{k_0}) = 0$.
Evidemment, à partir des étapes 1 et 3 de l'algorithme (2.3.1), on a $F(x_{k_0}) \neq 0$ si la recherche linéaire de (2.40) est Exécuté, ce qui contredit $F(x_{k_0}) = 0$.

Dans ce qui suit, on suppose que $\|F(x_k)\| \neq 0$ et $\|d_k\| \neq 0$ pour tout k, c'est-à-dire, Algorithme (2.3.1) génère une suite infinie $\{x_k\}$. Maintenant, nous prouvons certaines propriétés de la Séquence $\{x_k\}$.

Lemme 2.3.4 *Supposons que $F(.)$ est monotone et continue sur $\mathbb{R}^n$, et soit $\{x_k\}$ et $\{z_k\}$ les suites générées par l'Algorithme* (2.3.1)*, alors $\{x_k\}$ et $\{z_k\}$ sont toutes deux bornées,ie :*

$$\lim_{k\to\infty} \|x_k - z_k\| = 0. \tag{2.45}$$

Preuve.

Aprés (2.40),nous avons :

$$\langle f(z_k), x_k - z_k\rangle \geq \sigma\|x_k - z_k\|^2 > 0. \tag{2.46}$$

Pour tout $x^* \in \mathrm{C}^*$, de (2.37), il obtient :

$$\begin{aligned} \|x_{k+1} - x^*\|^2 &= \|P_C[x_k - \gamma\xi_k F(z_k)] - x^*\| \\ &\leq \|x_k - \gamma\xi_k F(z_k) - x^*\|^2 \\ &= \|x_k - x^*\|^2 - 2\gamma\xi_k\langle F(z_k), x_k - x^*\rangle + \gamma^2\xi_k^2\|F(z_k)\|^2. \end{aligned} \tag{2.47}$$

Par la monotonie de la fonction $F()$, nous avons :

$$\begin{aligned} \langle f(z_k), x_k - x^*\rangle &= \langle f(z_k), x_k - z_k\rangle - \langle f(z_k), z_k - x^*\rangle \\ &\geq \langle f(z_k), x_k - z_k\rangle - \langle f(x^*), z_k - x^*\rangle \\ &= \langle f(z_k), x_k - z_k\rangle. \end{aligned} \tag{2.48}$$

En remplaçant (2.46) et (2.48) par (2.47), nous avons :

$$\begin{aligned} \|x_{k+1} - x^*\|^2 &\leq \|x_k - x^*\|^2 - 2\gamma\xi_k\langle F(z_k), x_k - z_k\rangle + \gamma^2\xi_k^2\|F(z_k)\|^2 \\ &= \|x_k - x^*\|^2 - \gamma(2-\gamma)\frac{\langle F(z_k), x_k - z_k\rangle^2}{\|F(z_k)\|^2} \\ &\leq \|x_k - x^*\|^2 - \gamma(2-\gamma)\frac{\sigma^2\|x_k - z_k\|^2}{\|F(z_k)\|^2}, \end{aligned} \tag{2.49}$$

Qui avec $\gamma \in (0, 2)$ indique que pour tout k,

$$\|x_{k+1} - x^*\| \leq \|x_k - x^*\|. \tag{2.50}$$

Qui montre que la suite $\{x_k\}$ est bornée.

Par (2.44), il détient que $\{d_k\}$ est borné et donc est $\{z_k\}$.

Alors, par la continuité de $F(.)$, il existe une constante $M > 0$ telle que $\|F(z_k)\| \leq M \ \forall k$.
Il résulte donc de (2.49) que :

$$\gamma(2-\gamma)\frac{\sigma^2}{M^2}\sum_{k=0}^{\infty}\|x_k - z_k\|^2 \leq \sum_{k=0}^{\infty}(\|x_k - x^*\|^2 - \|x_{k+1} - x^*\|^2) < \infty.$$

Ce qui implique que l'assertion (2.45) est valide.

2.3.2 La convergence globale

Théorème 2.3.1 *Supposons que les conditions dans Lemme* (2.3.4) *restent. Alors la suite* $\{x_k\}$ *générée par Algorithme* (2.3.1) *converge globalement vers une solution de* CCE(F, C).

Preuve.
Nous examinons les deux cas suivants :
$-(i)$ Si $\liminf_{k\to\infty\|F(x_k)\|} = 0$, avec la continuité de $F(.)$ implique que la suite $\{x_k\}$ présente un point d'accumulation $\overline{x}$ tel que $F(\overline{x}) = 0$.
Dans (2.50),on déduit que $\{x_k - \overline{x}\}$ converge, et puisque $\overline{x}$ est un point d'accumulation de $\{x_k\}$, donc $\{x_k\}$ converge vers $\overline{x}$.
$-(ii)$ Si $\liminf_{k\to\infty\|F(x_k)\|} > 0$, alors il existe $\epsilon > 0$, tel que :

$$\|F(x_k)\| \geq \epsilon, \qquad \forall k \geq 0.$$

Alors, par (2.44), on a $\|d_k\| \geq c_1\epsilon \ , \ \forall k \geq 0$. Combinant avec (2.45), nous obtenons :

$$\lim_{k\to\infty}\alpha_k = 0.$$

Par conséquent, à partir de la recherche linéaire de (2.40), pour k suffisamment grand, nous avoir :

$$-\langle F(x_k + \beta_k\rho^{m_{k-1}}d_k), d_k\rangle < \sigma\beta_k\rho^{m_{k-1}}\|d_k\|^2. \tag{2.51}$$

Puisque $\{x_k\}, \{d_k\}$ sont bornés, nous pouvons choisir une suite et laisser $k \to \infty$ dans (2.51), on peut obtenir :

$$-\langle F(\overline{x}), \overline{d}\rangle \leq 0, \tag{2.52}$$

Où $\overline{x}, \overline{d}$ sont des points limites de sous-suite correspondantes. D'autre part, par (2.43), on obtient :

$$-\langle F(x_k), d_k\rangle \geq (1 - \frac{1}{\mu})\|F(x_k)\|^2 \quad \forall k \geq 0.$$

Soit $k \to \infty$ dans l'inégalité ci-dessus, on obtient :

$$-\langle F(\overline{x}), \overline{d}\rangle \geq (1 - \frac{1}{\mu})\|F(\overline{x})\|^2, \tag{2.53}$$

Ainsi, par (2.52) et (2.53), on obtient $\|F(\overline{x})\| = 0$, ce qui contredit le fait que :

$$\liminf_{k\to\infty} \|F(x_k)\| > 0.$$

Par conséquent $\liminf_{k\to\infty} \|F(x_k)\| > 0$ ne tient pas.

Chapitre 3

Application dans "$MATLAB$"

Dans cette chapitre, nous testons Nouveau Algorithme de projection par gradient conjugué hybride **(NHCGP)**, et les comparons avec la méthode de projection de gradient spectral **(TTGD)** et la méthode de gradient conjugué **(CGD)**.

-Nous testons l'efficacité des trois méthodes sur deux problèmes simples.
-Avec les fonctions de test ecrit comme suit :

$$F(x) = (f_1(x), f_2(x), ..., f_n(x))^T.$$

On va calculer :

— N :indique les dimensions.

— Iter : indique le nombre d'itérations.

— CPU : indique le temps par secondes.

— $\|F_x\|$: indique la norme de la fonction finale lorsque le programme est arrêté.

3.1 *PROBLEME* "1"

Définir $f_i(x)$ par :

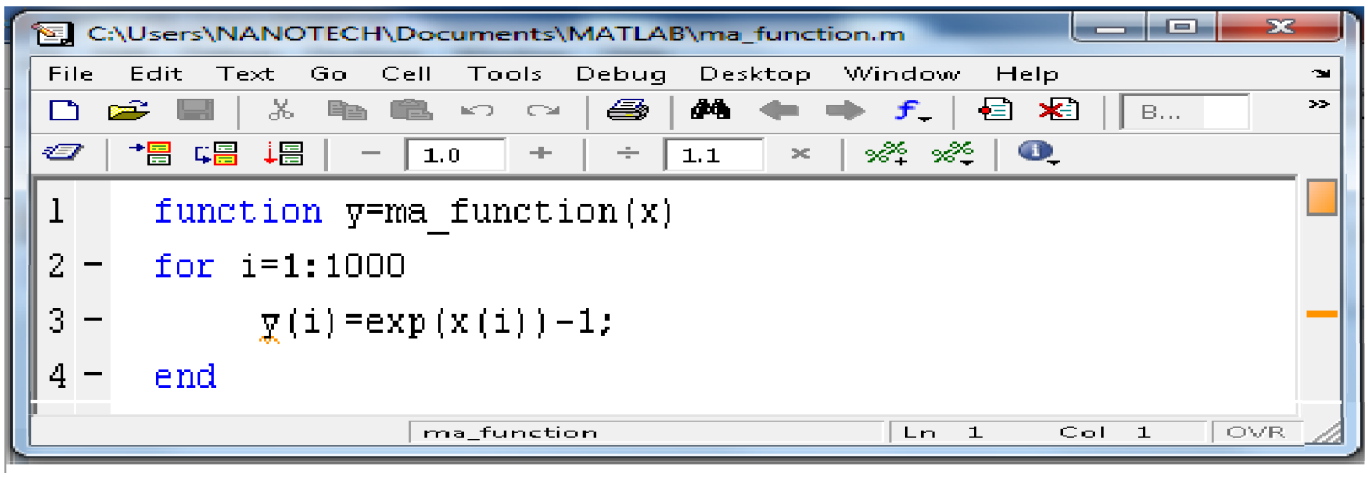
$$f_i(x) = \exp(x_i) - 1, \ \ pour \ \ i = \{1, 2, ..., n\}, \ \ et \ \mathrm{C} = \mathbb{R}^n_+. \tag{3.1}$$

De toute évidence, cet Problème a une solution unique $x^* = (0, 0, ..., 0)^T$.
Ici, on pose $x_0 = (1, 1, ..., 1)^T$.

3.1.1 Programmes de (NHCGP) et (CGD) et (TTGD)

-Les paramètres utilisés Dans l'algorithme de (NHCGP) sont définis comme :
$\rho = 0.5$, $\sigma = 10^{-4}$, $\gamma = 1.65$, $\mu = 1.4$.
La taille initiale de l'étape 2 est : $\beta_k = 1$ pour tout k.
-Les paramètres utilisés Dans l'algorithme de (CGD) sont définis comme : $\rho = 0.39$, $\sigma = 10^{-4}$.
La taille initiale de l'étape 2 est : $\beta_k = 1$ pour tout k.
-Les paramètres utilisés Dans l'algorithme de (TTGD) sont définis comme :
$\rho = 0.5$, $\sigma = 10^{-6}$, $\gamma = 1.5$, $\delta = 1.9$, $\tau = 2$, $\eta = 0.2$.
On arrête l'itération si le nombre d'itérations dépasse 1000 ou l'inégalité $\|F(x_k)\| \leq 10^{-5}$ est satisfait.

Programme de fonction



Programme de (NHCGP)

C:\Users\NANOTECH\Documents\MATLAB\NMPGC.m

```
%les entrées%
rho=0.5;sigma=10.^(-4);esp=1e-5;b=1;gamma=1.65;mu=1.4;n=input('donner n=');
%xo le poit départ et y dans C
x0=1;y0=0;
 for k=2:n
     x0=[x0 1];y0=[y0 0];
 end
nitr=0;x=x0;F=ma_function(x);d=-(F);
%les sorties%
while norm(F)>esp
     if norm(d)~=0
         %calcul alpha(k)%
         for m=0:10
             ak=b*rho.^(m);
             zk=ak.*d+x;  Fk=ma_function(zk);
             if sigma*ak*norm(d)*norm(d)<=-Fk*transpose(d)
                 a=ak; z=zk; Fz=Fk;
                 break
             end
         end
         if norm(Fz)==0
             x=z; F=Fz;
         else
             %calcul xi(k)%
             xi=(Fz*transpose(x-z))/(norm(Fz)*norm(Fz));
             %calcul x_next
             pc=x-gamma.*xi .*Fz;
             if pc<y0
              y=y0;
             else
              y=pc;
             end
             x_next=y; F_next=ma_function(x_next);
             %calcul beta(k)%
             if  (norm(F_next)/norm(F))*(F)*transpose(F_next)< 0
                 G=0;
             else
                 G=(norm(F_next)/norm(F))*(F)*transpose(F_next);
             end
             if  mu*norm(d)*norm(F_next)<(F_next-F)*transpose(d)
                 H=(F_next-F)*transpose(d);
             else
                 H=mu*norm(d)*norm(F_next);
             end
             beta=(norm(F)*norm(F)-G)/H;
             d_next =(beta.*d) - (F_next);
         end
     end
   nitr=nitr+1; x=x_next; F=F_next; d=d_next;
end
x_racine=x' , FX=norm(F) , iter=nitr
```

Programme de (CGD)

C:\Users\NANOTECH\Documents\MATLAB\CGD.m*

```
%les entrées%
rho=0.39;sigma=10.^(-4);esp=1e-5;b=1;n=input('donner n=');
%xo le poit départ et y dans C
x0=1;y0=0;
 for k=2:n
     x0=[x0 1];y0=[y0 0];
 end
x=x0; nitr=0; F=ma_function(x); d=-(F);
%les sorties%
while norm(F)>esp
      if norm(d)>esp
         %calcul alpha(k)%
         for m=1:100000
             ak=b*rho.^(m);
             zk=ak.*d+x; Fk=ma_function(zk);
             if sigma*ak*norm(d)*norm(d)<=-Fk*transpose(d)
                a=ak; z=zk; Fz=Fk;
                break
             end
         end
         if norm(Fz)==0
            x=z; F=Fz;
         else
            %calcul xi(k)%
            xi=(Fz*transpose(x-z))/(norm(Fz)*norm(Fz));
            %calcul x_next
            pc=x-(xi .*Fz);
            if pc<y0
               y=y0;
            else
               y=pc;
            end
            x_next=y; F_next=ma_function(x_next);
          %calcul beta(k)%
          if((F_next-F)*transpose(d))/norm(a*d).^(2)<0
            lamda=1;
          else
            lamda=1+((((F_next-F)*transpose(d))./norm(a*d).^(2))./norm(F));
          end
          Y=F_next-F+(lamda*a*norm(F)*d);
          M=Y-(2.*d.*(norm(Y).^(2))./(d*transpose(Y)));
          betta=(M*transpose(F_next))/(d*transpose(Y));
          if betta<0
             beta=0;
          else
             beta=betta;
          end
          d_next =(beta.*d) - (F_next);
         end
      end
      nitr=nitr+1; x=x_next; F=F_next; d=d_next;
end
x racine=x', FX=norm(F), iter=nitr
```

Programme de (TTGD)

C:\Users\NANOTECH\Documents\MATLAB\TTGD.m*

```
%les entrées%
rho=0.5;sigma=10.^(-6);esp=1e-5;gamma=1.5;delta=1.9;tau=2;eta=0.2;
n=input('donner n=');
%xo le poit départ et y dans C
x0=1;y0=0;
 for k=2:n
     x0=[x0 1]; y0=[y0 0];
 end
x=x0;F=ma_function(x);d=-(F);nitr=0;
%les sorties%
while norm(F)>esp
      if norm(d)>esp
         %calcul alpha(k)%
         for m=1:100000
             ak=rho.^(m);
             zk=ak.*d+x; Fk=ma_function(zk);
             if eta*ak*norm(d)*norm(d)<=-Fk*transpose(d)
                a=ak; z=zk; Fz=Fk;
                break
             end
          end
          if norm(Fz)==0
             x=z;F=Fz;
          else
             %calcul xi(k)%
             xi=(Fz*transpose(x-z))/(norm(Fz)*norm(Fz));
             %calcul x_next
             pc =x-(gamma.*xi.*Fz);
             if pc<y0
              y=y0;
             else
              y=pc;
             end
             x_next=y; F_next=ma_function(x_next);
             %calcul beta(k)%
             Y=Fz-F;
             beta=(F*transpose(Fz-F))./(d*transpose(Fz-F));
             %calcul teta(k)%
             teta=(F*transpose(d))./(d*transpose(Y));
             s=z-x;
             if s*transpose(Y)< sigma* (norm(F)).^(tau).*s*transpose(s)
                d_next=-F;
             else
               if delta* (norm(F)).^(tau).*s*transpose(s)<s*transpose(Y)
                d_next=-F;
               end
              d_next =(beta.*d) - (F_next)-(teta.*Y);
             end
             nitr=nitr+1; x=x_next; F=F_next; d=d_next;
          end
      end
end
x_racine=x', FX=norm(F), iter=nitr
```

3.1.2 Résultats numériques de (NHCGP) et (CGD) et (TTGD)

Tableau 1(Résultats numériques avec différentes dimensions)

dimensions	*Méthode*	*Iter*	*CPU*	$\|F_x\|$
10	NHCGP	1	0.0001	$0.00000e+000$
	CGD	8	0.0016	$8.4788e-006$
	TTGD	9	0.0001	$6.3442e-006$
100	NHCGP	1	0.0005	$0.00000e+000$
	CGD	9	0.0031	$5.8987e-006$
	TTGD	10	0.0028	$5.0155e-006$
1000	NHCGP	1	0.0005	$0.00000e+000$
	CGD	10	0.0178	$4.1037e-006$
	TTGD	11	0.0088	$3.9651e-006$
10000	NHCGP	1	0.0064	$0.00000e+000$
	CGD	11	0.0923	$2.8550e-006$
	TTGD	12	0.0349	$3.1347e-006$
100000	NHCGP	1	0.7000	$0.00000e+000$
	CGD	11	1.3336	$9.0282e-006$
	TTGD	12	0.7758	$9.9128e-006$

3.2 *PROBLEME* "2"

Définir $f_i(x)$ par :

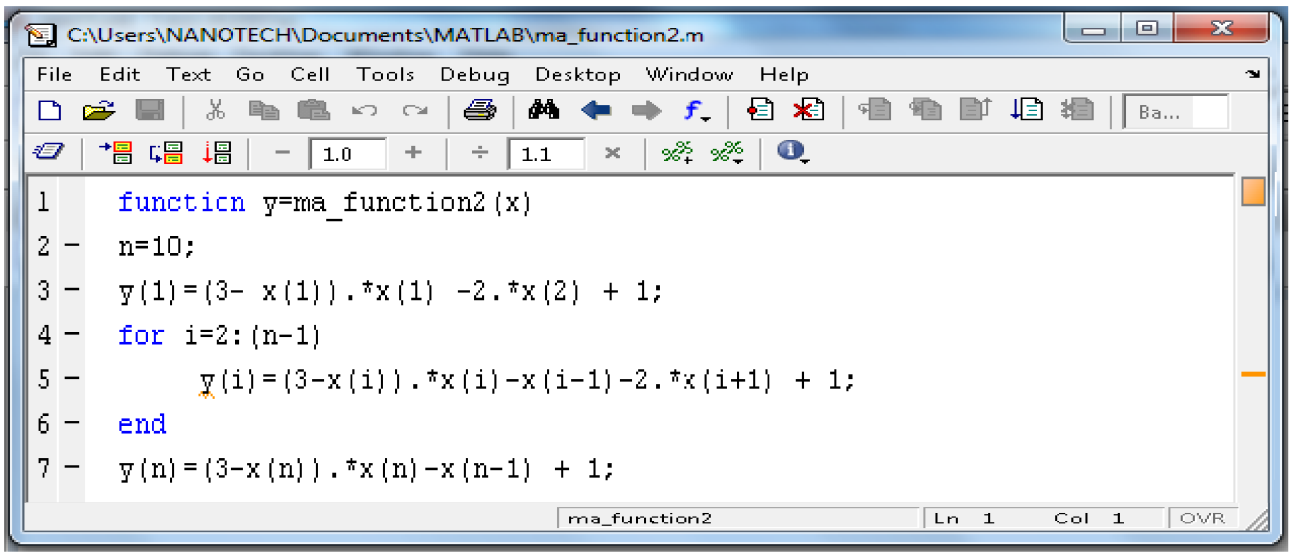
$$\begin{aligned} f_1(x) &= (3-x_1)x_1 - 2x_2 + 1, \\ f_i(x) &= (3-x_i)x_i - x_{i-1} - 2x_{i+1} + 1, \ \textit{pour } i = \{2,3,...,n-1\}, \\ f_n(x) &= (3-x_n)x_n - x_{n-1} + 1, \\ &\textit{et } \mathrm{C} = \mathbb{R}^n. \end{aligned} \tag{3.2}$$

Ici, on pose $x_0 = (-1,-1,...,-1)^T$

3.2.1 Programmes de (NHCGP) et (CGD) et (TTGD)

-Les paramètres utilisés Dans l'algorithme de (NHCGP) sont définis comme :
$\rho = 0.5$, $\sigma = 10^{-4}$, $\gamma = 1.65$, $\mu = 1.4$.
La taille initiale de l'étape 2 est : $\beta_k = 1$ pour tout k.
-Les paramètres utilisés Dans l'algorithme de (CGD) sont définis comme : $\rho = 0.5$, $\sigma = 10^{-4}$.
La taille initiale de l'étape 2 est : $\beta_k = 1$ pour tout k.
-Les paramètres utilisés Dans l'algorithme de (TTGD) sont définis comme :
$\rho = 0.5$, $\sigma = 10^{-6}$, $\gamma = 1.8$, $\delta = 0.9$, $\tau = 2$, $\eta = 0.2$.
On arrête l'itération si le nombre d'itérations dépasse 1000 ou l'inégalité $\|F(x_k)\| \leq 10^{-5}$ est satisfait.

Programme de fonction



Programme de (NHCGP)

C:\Users\NANOTECH\Documents\MATLAB\NMPGC2.m

```
%les entrées%
rho=0.5;sigma=10.^(-4);esp=1e-5;b=1;gamma=1.65;mu=1.4;n=input('donner n=');
nitr=0;x0=-1;
 for k=2:n
     x0=[x0 -1];
 end
x=x0;F=ma_function2(x);d=-(F);
%les sorties%
while norm(F)>esp
         if norm(d)>esp
            %calcul alpha(k)%
            for m=1:100000
                ak=b*rho.^(m);
                zk=ak.*d+x; Fk=ma_function2(zk);
                if sigma*ak*norm(d)*norm(d)<=-Fk*transpose(d)
                   a=ak; z=zk; Fz=Fk;
                   break
                end
            end
            if norm(Fz)==0
              x=z;F=Fz;
            else
            %calcul xi(k)%
            xi=(Fz*transpose(x-z))/(norm(Fz)*norm(Fz));
            %calcul x_next
            x_next=x-(gamma.*xi .*Fz); F_next=ma_function2(x_next);
            %calcul beta(k)%
            if  (norm(F_next)/norm(F))*(F)*transpose(F_next)< 0
                  G=0;
            else
                  G=(norm(F_next)/norm(F))*(F)*transpose(F_next);
            end
            if  mu*norm(d)*norm(F_next)<(F_next-F)*transpose(d)
                  H=(F_next-F)*transpose(d);
            else
                  H=mu*norm(d)*norm(F_next);
            end
              beta=(norm(F)*norm(F)-G)/H;
              d_next =(beta.*d) - (F_next);
            end
         end
         nitr=nitr+1; x=x_next; F=F_next; d=d_next;
end
x racine=x', FX=norm(F), iter=nitr
```

Programme de (CGD)

C:\Users\NANOTECH\Documents\MATLAB\CGD2.m

```
%les entrées%
rho=0.5;sigma=10.^(-4);esp=1e-5;b=1;n=input('donner n=');
%xo le poit départ
x0=-1;
 for k=2:n
     x0=[x0 -1];
 end
x=x0; F=ma_function2(x); d=-(F);nitr=0;
%les sorties%
while norm(F)>esp
     if ncrm(d)>esp
        %calcul alpha(k)%
        fcr m=1:100000
            ak=b*rho.^(m);
            zk=ak.*d+x; Fk=ma_function2(zk);
            if sigma*ak*norm(d)*norm(d)<=-Fk*transpose(d)
               a=ak; z=zk; Fz=Fk;
               break
            end
        end
        if norm(Fz)==0
          x=z; F=Fz;
        else
          %calcul xi(k)%
          xi=(Fz*transpose(x-z))/(norm(Fz)*norm(Fz));
          %calcul x_next
          x_next=x-xi.*Fz; F_next=ma_function2(x_next);
          %calcul beta(k)%
          if((F_next-F)*transpose(d))/norm(a*d).^(2)<0
            lamda=1;
          else
             lamda=1+[(((F_next-F)*transpose(d))/ncrm(a*d).^(2))/norm(F)];
          end
          Y=F_next-F+(lamda.*a.*norm(F).*d);
          M=Y-(2.*d.*(norm(Y).^(2)))/(d*transpose(Y));
          betta=(M*transpose(F_next))/(d*transpose(Y));
          if betta<0
             beta=0;
          else
             beta=betta;
          end
          d_next =(beta.*d) - (F_next);
        end
     end
     nitr=nitr+1; x=x_next; F=F_next; d=d_next;
end
x_racine=x', FX=norm(F), iter=nitr
```

Programme de (TTGD)

C:\Users\NANOTECH\Documents\MATLAB\TTGD2.m

```
%les entrées%
rho=0.5;sigma=10.^(-6);esp=1e-5;gamma=1.8;delta=0.9;tau=2;eta=0.2;
n=input('donner n=');
%xo le poit départ et y dans C
x0=-1;
 for k=2:n
     x0=[x0 -1];
 end
x=x0;nitr=0;F=ma_function2(x);d=-(F);
%les sorties%
while norm(F)>esp
    if norm(d)>esp
       %calcul alpha(k)%
       for m=1:100000
           ak=rho.^(m);
           zk=ak.*d+x; Fk=ma_function2(zk);
           if eta*ak*norm(d)*norm(d)<=-Fk*transpose(d)
              a=ak;z=zk;Fz=Fk;
              break
           end
       end
       if norm(Fz)==0
          x=z;F=Fz;
       else
       %calcul xi(k)%
        xi=(Fz*transpose(x-z))/(norm(Fz)*norm(Fz));
        %calcul x_next
        x_next=x-gamma.*xi.*Fz; F_next=ma_function2(x_next);
        %calcul beta(k)%
        Y=Fz-F;
        beta=(F*transpose(Y))./(d*transpose(Y));
        %calcul teta(k)%
        teta=(F*transpose(d))./(d*transpose(Y));
        s=z-x;
        if s*transpose(Y)< sigma* (norm(F)).^(tau).*s*transpose(s)
           d_next=-F;
        else
           if delta* (norm(F)).^(tau).*s*transpose(s)<s*transpose(Y)
              d_next=-F;
           end
           d_next =(beta.*d) - (F_next)-(teta.*Y);
        end
        nitr=nitr+1; x=x_next; F=F_next; d=d_next;
       end
    end
end
x_racine=x', FX=norm(F), iter=nitr
```

3.2.2 Résultats numériques de (NHCGP) et (CGD) et (TTGD)

Tableau 2(Résultats numériques avec différentes dimensions)

dimensions	*Méthode*	*Iter*	*CPU*	$\|F_x\|$
10	NHCGP	22	0.0012	$5.8678e-006$
	CGD	25	0.0064	$7.9884e-006$
	TTGD	22	0.0057	$5.8680e-006$
100	NHCGP	22	0.0095	$7.0217e-006$
	CGD	27	0.0109	$8.0707e-006$
	TTGD	26	0.0128	$8.6256e-006$
1000	NHCGP	22	0.0189	$7.0217e-006$
	CGD	27	0.0306	$8.0707e-006$
	TTGD	26	0.0333	$8.6256e-006$
10000	NHCGP	22	0.1118	$7.0217e-006$
	CGD	27	0.2154	$8.0707e-006$
	TTGD	26	0.2260	$8.6256e-006$
100000	NHCGP	22	1.1781	$7.0217e-006$
	CGD	27	2.3248	$8.0707e-006$
	TTGD	26	2.1221	$8.6256e-006$

Conclusion

Les résultats numériques donnés dans les tableaux 1 et 2 montrent que :

1. les trois méthodes peuvent résoudre tous les problèmes testés avec succès.
2. pour le Problème 1 , Algorithme NHCGP s'avère évidemment mieux que CGD à partir de l'Iter et CPU time, et il est comparable à TTGD.

Dans l'ensemble, bien que notre paramètre β_K^{NN} est un peu plus compliqué, l'Algorithme NHCGP résultant est assez efficace pour les équations avec contraintes, en particulier pour les problèmes à grande échelle.

Bibliographie

[1] Cheng,W.Y. : A PRP type method for systems ofmonotone equations.Math. Comput.Model. $50(1-2), 15-20(2009)$.

[2] Dai, Y.H., Yuan, Y.X. : A nonlinear conjugate gradient method with a strong global convergence property. SIAM J. Optim. $10, 177-182(2000)$.

[3] Dirkse, S.P., Ferris, M.C. :MCPLIB : a collection of nonlinear mixed complementarity problems. Optim. Methods Softw. $5, 319-345(1995)$.

[4] Fletcher, R., Reeves, C. : Function minimization by conjugate gradients. J. Comput. $7, 149-154(1964)$.

[5] Gilber, J.C., Nocedal, J. :Global convergence properties of conjugate gradientmethods for optimization. SIAM J. Optim. $2, 21-42(1992)$.

[6] Han, D.R. : A proximal decomposition algorithm for variational inequality problems. J. Comput. Appl. Math. $161(1), 231-244(2003)$.

[7] Hestenes, M.R., Stiefel, E.L. : Method of conjugate gradient for solving linear systems. J. Res. Natl. Bur. Stand. $49, 409-432(1952)$.

[8] Hu, Y.P., Wei, Z.X. : Wei-Yao-Liu conjugate gradient projection algorithm for nonlinear monotone equations with convex constraints. Int. J. Comput.Math. (2014).
doi :10.1080/00207160.2014.977879.

[9] Jian, J.B., Han, L., Jiang, X.Z. : A hybrid conjugate gradient method with descent property for unconstrained optimization. Appl. Math. Model. $39(3-4), 1281-1290(2015)$.

[10] Liu, S.Y., Huang, Y.Y, Jiao, H.W. : Sufficient descent conjugate gradient methods for solving convex constrained nonlinear monotone equations. Abstr. Appl. Anal. $2014, 12(2014)$.

[11] Meintjes, K., Morgan, A.P : A methodology for solving chemical equilibrium systems. Appl. Math. Comput. $22, 333 - 361(1987)$.

[12] Polak, B., Ribière, G. : Note surla convergence des méthodes de directions conjuguées. Rev. Francaise Imformat Rech. Opertionelle $16, 35 - 43(1969)$.

[13] Sun, M., Liu, J. : Three derivative-free projection methods for large-scale nonlinear equations with convex constraints. J. Appl. Math. Comput. $47(1 - 2), 265 - 276(2015)$.

[14] Wang, C.W.,Wang, Y.J., Xu, C.L. : A projection method for a system of nonlinear monotone equations with convex constraints. Math. Methods Oper. Res. $66, 33 - 46(2007)$.

[15] Wang, X.Y., Li, S.J., Kou, X.P. : A self-adaptive three-term conjugate gradient method for monotone nonlinear equations with convex constraints. Calcolo (2015).
doi :$10.1007/s10092 - 015 - 0140 - 5$.

[16] Wei, Z.X., Yao, S.W., Liu, L.Y. : The convergence properties of some new conjugate gradient methods. Appl. Math. Comput. $183, 1341 - 1350(2006)$.

[17] Xiao, Y.H., Zhu, H. : A conjugate gradient method to solve convex constrained monotone equations with applications in compressive sensing. J. Math. Anal. Appl. $405, 310 - 319(2013)$.

[18] Yao, S.W., Wei, Z.X., Huang, H. : A note about WYL's conjugate gradient method and its application. Appl. Math. Comput. $191, 381 - 388(2007)$.

[19] Y. H. Dai, Analyses of conjugate gradient methods, Ph.D. thesis, Institute of Computational Mathematics and Scientific/Engineering Computing, Chinese Academy of Sciences,(1997).

[20] Y. H. Dai, New properties of a nonlinear conjugate gradient method, Numer. Math., $89(2001), pp.83 - 98$.

[21] Yu, Z.S., Lin, J., Sun, J., Xiao, Y.H., Liu, L.Y., Li, Z.H. : Spectral gradient projection method for monotone nonlinear equations with convex constraints. Appl. Numer. Math. $59, 2416 - 2423(2009)$.

[22] Zhang, L., Zhou,W.J. : Spectral gradient projection method for solving nonlinear monotone equations. J. Comput. Appl. Math. $196(2), 478 - 484(2006)$.