République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Faculté des Sciences Exactes

Département de Recherche Opérationnelle

Mémoire de fin de cycle

En vue de l'obtention du diplôme de Master en Mathématiques Appliquées

Option : Modélisation Mathématique et Technique de Décision

Thème

Minimisation d'une forme quadratique sous une boule euclidienne : cas convexe et non convexe

Présenté par

Ferhat El Hadi
Gouasmia Imane

Soutenu le 02 juillet 2017 devant le jury composé de :

Mlle AOUDIA Zohra	M.A.A	Présidente
Mr TOUATI Sofiane	M.A.B	Examinateur
Mr BIBI Mohand Ouamer	Professeur	Rapporteur

Promotion 2016-2017

Remerciements

Un grand merci revient à Dieu le tout puissant qui lui seul nous a donné la volonté de réaliser ce modeste travail.

*Un grand merci à notre promoteur, Pr **M.O.BIBI**, pour ses conseils et son aide et qui a mis à notre disposition tout le nécessaire pour réaliser ce travail.*

*Nous remercions M^{lle} **Z.Aoudia** et Mr **S.Touati** pour avoir accepté de juger ce travail.*

Nous remercions tous les membres de l'unité de recherche $LaMOS$ pour leur aide, leur orientation et leur participation pour atteindre le but recherché.

Nous tenons à remercier nos parents pour leur encouragement et leur soutien et les enseignants pour l'aide et la patience dont ils ont fait preuve tout au long de ce travail, ainsi que leurs orientations afin d'accéder à l'objectif tracé.

Sans oublier les bons collègues, nous les remercions pour toutes les belles journées qu'on a passées ensemble.

Et tous les autres sans exception.

Ainsi, à tous ceux qui ont contribué de près ou de loin à la réalisation de ce modeste travail, nous leur disons :

Merci pour tout.

Dédicaces

Je dédie ce modeste travail :

A la mémoire de mon père que j'aurai aimé voir.

A ma chère mère, à mes sœurs, au fils de ma sœur Aymen et à son père Rabeh.

A ma première enseignante de mathématiques Z.Gareh, et à sa famille.

A toute la famille Ferhat et la famille Rammach.

A tous mes amis notamment: B.Wassim, B.Aymen (pousticha),.....

A toute ma promotion et spécialement: K.Amine (Lemzabi), B.Abdeldjalil, D.Widad,.....

A mes collègues et leurs familles.

A tous mes professeurs.

Une dédicace spécial à ma très chère binôme "Amouna" qui m'a vraiment poussé, encouragé et surtout supporté durant l'élaboration du projet et à toute sa famille....

Mr El Hadi,

Dédicaces

A mon très cher *père,* qui a mis en moi tant d'espoir et de confiance.

A ma très adorée *mère,* qui a attendu ce jour avec impatience.

A mon très cher frère *Abd Raouf* pour son soutien moral.

A mes très chères soeurs *Nassima, Karima, Radia* et à leurs *maris*, à ma chère nièce *Arwa* et mes neveux *Youcef, Islam et Raid.*

A toute ma famille.

A *El-hadi*, mon ami partenaire "binôme" qui a tant donné pour que nous achevions ce travail dans les meilleures conditions et à toute sa famille.

A tous mes amis et amies pour leur amitié, leurs encouragement, leur soutien.

A tous ceux qui m'ont aidée et qui me sont chers. Tous mes remerciements.

Liste des symboles

Symboles utilisés dans le document

x^T : transposée du vecteur x.

D^T : transposée de la matrice D.

$H^{-1}(x)$: inverse de la matrice $H(x)$.

$\geq$: supérieur ou égal.

$\leq$: inférieur ou égal.

$\neq$ différent.

$\in$: appartient.

$\forall$: pour tout (quantificateur universel).

$\exists$: il existe (quantificateur universel).

$\exists!$: il existe un et un seul.

$\subset$: inclus (est contenu).

$\Rightarrow$: implique.

$\Leftrightarrow$: si et seulement si (ssi).

$\sum_{i=1}^{n} a_i$: somme par rapport à l'indice i , équivaut à $a_1 + a_2 + ... + a_n$.

$\frac{\partial \varphi(x)}{\partial x}$: dérivée de φ par apport à x.

$\nabla\varphi(x)$: gradient de la fonction $\varphi(x)$.

($\nabla^2\varphi(x)$ ou $H(x)$) : Hessien de la fonction $\varphi(x)$.

(PQQ) : problème de minimisation quadratique sous contraintes quadratiques.

Dédicaces

Sommaire des symboles

TABLE DES MATIÈRES

Introduction Générale

La programmation mathématique est une branche des mathématiques appliquées ayant pour objet l'étude théorique des problèmes d'optimisation, ainsi que la conception et la mise en œuvre des algorithmes de résolution.

La présence du terme "programmation" dans le nom donné à cette discipline peut s'expliquer historiquement par le fait que les premières recherches et les premières applications se sont développées dans le contexte de l'économie et de la recherche opérationnelle.

Dans de nombreuses applications pratiques, les variables d'une fonction donnée sont soumises à certaines conditions ou contraintes. Ces contraintes peuvent être formulées sous forme d'égalités ou d'inégalités.
Par exemple, si un producteur fabrique deux biens, il peut vouloir minimiser le coût total tout en étant obligé de fabriquer une quantité totale spécifiée. De même, une compagnie peut désirer maximiser ses ventes résultant de deux publicités alors qu'elle doit observer la contrainte du budget de publicité. Enfin, le consommateur désirant maximiser la fonction d'utilité provenant de la consommation de certains biens est restreint par son budget.

La résolution numérique d'un problème quadratique nécessite l'utilisation de méthodes spécifiques. La plupart de ces méthodes sont itératives et génèrent une suite de vecteurs $x^{(i)}$. Les itérés successifs sont construits de façon à faire converger cette suite vers la solution du problème x^*. On peut alors utiliser plusieurs méthodes pour la résolution de ces problèmes quadratiques. Dans ce travail, nous nous intéressons à trois méthodes de résolution d'un problème quadratique sous contrainte quadratique de type inégalité (PQQ) :

$$\begin{cases} min\varphi(x) = \frac{1}{2}x^tDx + c^tx, \\ g(x) = \frac{1}{2}\|x\|^2 = \frac{1}{2}x^tI_nx \leq b, \\ x \in R^n. \end{cases}$$

Après un bref rappel de quelques notions sur l'algèbre linéaire et sur les formes quadratiques dans le premier chapitre, nous avons présenté dans le second les méthodes de résolution des (PQQ) convexes, basées sur les conditions nécessaires et suffisantes d'optimalité pour ce type de problèmes, ensuite, nous avons rappelé quelques notions de l'optimisation non convexe, puis nous avons utilisé l'algorithme de la programmation (DC) pour résoudre les problèmes (PQQ) non convexes.

Dans le dernier chapitre, nous avons fait une étude numérique selon le temps d'exécution et le nombre d'itérations des méthodes, et ce, après avoir implémenté ces dernières sous le langage MATLAB, exécutées sur des exemples d'application.

En fin, ce travail s'achève par une conclusion générale.

CHAPITRE 1

RAPPELS MATHEMATIQUES

1.1 Introduction

Dans ce chapitre, nous rappelons quelques définitions et propriétés des fonctions quadratiques et des matrices semi-définies positives.

Les propriétés non démontrées dans ce chapitre sont pour la plupart classiques et relevant d'un cours d'algèbre linéaire ou bilinéaire de base .

1.2 Algèbre matricielle

1.2.1 Espace vectoriel :

Un vecteur (point) x de R^n est une collection ordonnée $x = (x_1, x_2, ..., x_n)^T$ des n réels $x_j, j \in \{1, 2, ..., n\}$, appelés composantes de x. Le nombre n est appelé dimension du vecteur.
L'espace R^n est l'ensemble de toutes les collections de ce type. L'espace R^n est muni des deux opérations linéaires de base :

– ***Addition :*** **La somme de deux vecteurs $x, y \in R^n$ est un vecteur de R^n, défini par :**

$$x + y = (x_1 + y_1, x_2 + y_2, ..., x_n + y_n)^T, \tag{1.1}$$

– ***Multiplication par des réels :*** **Soit $x = (x_1, x_2, ..., x_n)^T$ un vecteur de R^n. La multiplication du vecteur x par le réel α est aussi un vecteur de R^n, défini comme suit :**

$$\alpha x = (\alpha x_1, \alpha x_2, ..., \alpha x_n)^T. \tag{1.2}$$

La structure que nous obtenons – l'ensemble de tous les vecteurs n-dimensionnels avec les deux opérations qu'on vient de définir– s'appelle l'espace vectoriel réel R^n n-dimensionnel.

Sous-espace linéaire [1] :

Un sous-ensemble L non vide de R^n est appelé sous-espace linéaire de R^n si les deux opérations (1.1) et (1.2) sont stables dans L, autrement dit :

$$\forall x, y \in L, \forall \alpha, \beta \in R \Rightarrow \alpha x + \beta y \in L.$$

Combinaison linéaire [1] :

Étant donnés k vecteurs $a_1, a_2, ..., a_k$ et un ensemble de k scalaires $\{\alpha_1, \alpha_2, ..., \alpha_k\}$, le vecteur b est appelé combinaison linéaire des vecteurs $a_i, 1 \leq i \leq k$, s'il s'écrit sous la forme :

$$b = \alpha_1 a_1 + \alpha_2 a_2 + ... + \alpha_k a_k. \tag{1.3}$$

L'expression (1.3) peut être écrite sous forme de produit d'une matrice et d'un vecteur comme suit :

$$b = (a_1, a_2, \ldots, a_k) \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \cdots \\ \alpha_k \end{pmatrix} = A\alpha,$$

où a_i est la i^{eme} colonne de la matrice A, et α_i est le i^{eme} élément du vecteur colonne α.
Une combinaison linéaire où tous les coefficients sont nuls est appelée combinaison linéaire triviale, dans le cas contraire elle est appelée combinaison linéaire non triviale.

1.2.2 Indépendance et dépendance linéaire :

Les vecteurs $(a_1, a_2, \cdots, a_k)$ sont dits linéairement indépendants si

$$\alpha_1 a_1 + \alpha_2 a_2 + \cdots + \alpha_k a_k = 0 \Rightarrow \alpha_1 = \alpha_2 = \cdots = \alpha_k = 0.$$

Les vecteurs a_i, $i = \overline{i,k}$, sont dits linéairement dépendants s'il existe une combinaison linéaire nulle de ces vecteurs, où au moins un des scalaires α_i, $i = \overline{i,k}$, est non nul.

1.2.3 Matrice :

Une matrice à m lignes et n colonnes et à coefficients dans R est un tableau rectangulaire de nombres réels pouvant se présenter de la manière suivante :

$$A = (a_{ij}, i \in I, j \in J) = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix},$$

où I=$\{1,2,\cdots m\}$ et J=$\{1,2,\cdots n\}$ représentent respectivement l'ensemble des indices de lignes et de colonnes de A.
Les termes (coefficients) représentés dans le tableau constituent les éléments de la matrice A et ils sont caractérisés par leurs valeurs et leurs positions. Par exemple, les indices de ligne i et de colonne j indiquent la position de l'élément a_{ij} dans A. Une matrice composée de m lignes et de n colonnes est dite d'ordre $(m \times n)$. Une matrice d'ordre $(m \times 1)$, est appelée vecteur-colonne, tandis qu'une matrice d'ordre $(1 \times n)$, est appelée vecteur-ligne. Pour des calculs pratiques, la matrice A se note aussi :

$$A = (a_1, a_2, ..., a_j, ..., a_n) = \begin{pmatrix} A_1^T \\ A_2^T \\ \vdots \\ A_m^T \end{pmatrix},$$

où le vecteur colonne a_j est représenté comme suit :

$$a_j = A(I, j) = \begin{pmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{mj} \end{pmatrix}.$$

1.2.4 Transposée d'une matrice :

Soit A une matrice d'ordre $m \times n$. La transposée de A est définie comme une nouvelle matrice, notée A^T d'ordre $(n \times m)$ et qui a pour colonnes les lignes de A et pour lignes les colonnes de A, telle que :

$$A^T = (a_{ij},\ i = \overline{1,n},\ j = \overline{1,m}).$$

Remarque 1.1
Une matrice carrée A est dite symétrique si on a $A = A^T$.

1.2.5 L'inverse d'une matrice :

Définition.1.1 [2] :

Une matrice carrée A est inversible ou régulière si son déterminant est non nul , sinon elle est dite singulière ou non inversible.

1.2.6 Noyau et image d'une matrice :

Définition1.2 [1] :

Soit A une matrice d'ordre $m \times n$. Le noyau de A est l'ensemble

$$N(A) = \{x \in R^n : Ax = 0\}.$$

L'image ou l'espace image de A est l'ensemble

$$R(A) = \{y \in R^m : \exists x \in R^n, Ax = y\}.$$

Proposition 1 :

- **$N(A)$ est un sous-espace vectoriel de R^n.**
- **$R(A)$ est le sous-espace vectoriel de R^m, engendré par les colonnes de A.**

En effet, soient $\{a_1, a_2, ..., a_n\}$ l'ensemble des colonnes de A, où $a_j \in R^m$. Soit $y \in R(A)$. Par définition, il existe $x \in R^n$ tel que $Ax = y$. Alors

$$y = Ax = \sum_{j=1}^{n} x_j a_j.$$

Donc, tout élément y de $R(A)$ est une combinaison linéaire des colonnes de A. Réciproquement, toute combinaison linéaire de $\{a_1, a_2, ..., a_n\}$ est un élément de $R(A)$.

1.2.7 Rang d'une matrice :

Définition 1.3 :

On appelle rang d'une matrice, la dimension de l'image de la matrice A :

$$rang(A) = dimR(A) \qquad (1.4)$$

Théorème 1 : ***(théorème du noyau-image)***

Soit A une matrice d'ordre $m \times n$, alors

$$dimR(A) + dimN(A) = n.$$

1.2.8 Décomposition des matrices en blocs :

Il est souvent utile de décomposer une matrice en blocs. Par exemple, la matrice A d'ordre 3

$$A = \left(\begin{array}{cc|c} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ \hline a_{31} & a_{32} & a_{33} \end{array}\right)$$

a été décomposée en 4 blocs :

$$A_{11} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, A_{12} = \begin{pmatrix} a_{13} \\ a_{23} \end{pmatrix}, A_{21} = (a_{31}\ a_{32}), A_{22} = (a_{33}).$$

On peut maintenant écrire la matrice A sous la forme :

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix},$$

où les éléments de la matrice A sont eux-mêmes des matrices.
Pour multiplier des matrices décomposées en blocs, il faut s'assurer d'abord que les blocs sont d'un ordre convenable.
Si le produit de deux matrices AR est défini, c'est à dire, si A a autant de colonnes que R a de lignes, les colonnes de A et les lignes de R doivent être décomposées de la même façon si l'on veut multiplier A et R après les avoir décomposées en blocs, si c'est bien le cas, on applique à ces nouveaux éléments les règles habituelles de la multiplication matricielle avec des éléments scalaires ; ainsi si $R = \begin{pmatrix} R_{11} \\ R_{21} \end{pmatrix}$, alors $AR = \begin{pmatrix} A_{11}R_{11} + A_{12}R_{21} \\ A_{21}R_{11} + A_{22}R_{21} \end{pmatrix}$. Cette opération est appelée *multiplication des matrices par blocs.*

1.2.9 Discussion générale sur l'existence et le nombre de solutions d'un système linéaire

Soient m et n deux nombres entiers. Un système de m équations linéaires à n inconnues $x_1, x_2, ..., x_n$ s'écrit comme suit :

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1, \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2, \\ \quad\vdots \quad\quad \vdots \quad \vdots \quad\quad \vdots \quad\quad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m, \end{cases} \tag{1.5}$$

où les coefficients a_{ij} sont des réels. Les nombres $b_1, b_2, ..., b_m$ sont appelés les membres libres du système (1.5) ou les seconds membres.
Alors le système (1.5) peut s'écrire sous la forme matricielle suivante :

$$Ax = b. \tag{1.6}$$

Tout vecteur x vérifiant les équations (1.5) s'appelle solution du système. Le système (1.5) est dit compatible s'il possède une ou plusieurs solutions. Dans le cas contraire, il est dit incompatible ou impossible. D'une manière générale, le système (1.6) possède une solution si le vecteur $b \in R(A)$, i.e, au sous-espace engendré par les vecteurs colonnes de la matrice A. Lorsque le vecteur b est nul, le système (1.6) est dit homogène. Tout système homogène possède la solution triviale $x = 0$.

Définition 1.4 :

Le système linéaire (1.6) est dit de rang complet en lignes si $rang(A) = m$**,** $m \leq n$**, et de rang complet en colonnes si** $rang(A) = n, m \geq n$**.**

Lemme 1 :

Soit $m \leq n$ **et** $rang(A) = m$**. Alors le système** $Ax = b$ **admet toujours des solutions, quel que soit le second membre** b **:**

a) une solution et une seule si $m = n$**,**

b) une infinité de solutions si $m < n$.

1.2.10 Espace complémentaire orthogonal :

Le produit scalaire (produit euclidien) de deux n-vecteurs x **et** y **est le scalaire défini par :**

$$x^T y = \sum_{i=1}^{n} x_i y_i.$$

Le produit scalaire possède les propriétés suivantes :

- **Commutativité :** $x^T y = y^T x$ **.**
- **Distributivité par rapport à l'addition des vecteurs :** $x^T(y + z) = x^T y + x^T z$**.**
- **Positivité :** $x^T x \geq 0$ **et** $x^T x = 0 \Leftrightarrow x = 0$.
- **La norme euclidienne d'un vecteur** x **est notée :** $\|x\|_2 = \sqrt{x^t x} = \sqrt{\sum_{i=1}^{n} x_i^2}$ **.**

Si $x^T y = 0$**, les vecteurs** x **et** y **sont dits orthogonaux et on note** $x \perp y$.

Deux parties A **et** B **de** R^n **sont dites orthogonales si :** $\forall a \in A, \quad \forall b \in B, \quad a^T b = 0$.

Soit A **une partie de** R^n**. On appelle orthogonal de** A**, le sous-ensemble de** R^n **défini par :**

$$A^{\perp} = \{x \in R^n / \forall a \in A, x^T a = 0\}.$$

Proposition 1 :

$A^{\perp}$ **est un sous-espace vectoriel de** R^n **:**

- $A \subset B \Rightarrow B^{\perp} \subset A^{\perp}$ **.**
- $A \subset (A^{\perp})^{\perp}$.
- **Si** A **est un sous-espace vectoriel de** R^n**, alors** $(A^{\perp})^{\perp} = A$**.**

Pour chaque sous-espace L **de** R^n**, il existe un ensemble complémentaire** $\overline{L}$ **dont les éléments sont définis comme suit :** $y \in \overline{L}$ **si, pour chaque** $x \in L, x^T y = 0$**, i.e,** y **est orthogonal à chaque vecteur dans** L**.**

Le sous-ensemble $\overline{L}$ **est un sous-espace, puisque chaque combinaison de vecteurs de** $\overline{L}$ **est orthogonale à chaque vecteur de** L**.**

Le sous-espace $\overline{L}$ **est appelé complément orthogonal de** L**, et les deux sous-espaces n'ont que le vecteur nul en intersection. Si la dimension de** L **est** p**, alors la dimension de** $\overline{L}$ **est** $n - p$.

Proposition 2 :

Soit A une matrice d'ordre $m \times n$, alors on a :

- $N(A)^{\perp} = R(A^T)$,
- $R(A)^{\perp} = N(A^T)$.

En effet, pour tout x du noyau $N(A)$ de A, on a : $Ax = 0$. **Donc pour tout** $i = 1, 2, ..., m, (Ax)_i = 0, (Ax)_i$ **est obtenu en multipliant le vecteur ligne** A_i^T **par le vecteur colonne** x **:**

$$(Ax)_i = (a_{i1}, a_{i2}, \ldots, a_{in}) \begin{pmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{pmatrix} = 0, \text{ d'où } A_i^T \perp x.$$

On déduit que $N(A) \perp R(A^T)$. **De plus** $dimR(A^T) = dimR(A) = n - dimN(A)$, **d'où le résultat. Par passage à la transposée, on déduit aussi que :** $R(A)^{\perp} = N(A^T)$.

1.2.11 Forme bilinéaire :

Définition.1.5 [6] :

On appelle forme bilinéaire sur $E \subseteq R^n$ **une application** $b : E \times E \to R$, **telle que :**

1. **Pour tout** $x \in$ E**,l'application partielle**

$$b_x : E \xrightarrow[y \mapsto\ b(x,y)]{} K$$

est une forme linéaire sur E.

2. **Pour tout** $y \in$ E**,l'application partielle**

$$b_y : E \xrightarrow[x \mapsto\ b(x,y)]{} K$$

est une forme linéaire sur E.

– **On dit que** b **est symétrique si** $b(x, y) = b(y, x)$ **pour tout** x **et tout** y **dans** E.

– **A toute forme bilinéaire sur E est associée une forme quadratique.**

$$q : E \xrightarrow[x \mapsto\ q(x)=b(x,x)]{} K$$

– **Une forme quadratique est associée à une forme unique bilinéaire symétrique que l'on appelle sa forme polaire qui s'écrit comme suit :**

$$b(x, y) = \frac{1}{2}\Big(q(x + y) - q(x) - q(y)\Big).$$

1.2.12 Forme quadratique :

Définition : 1.3 [2]

Une forme quadratique dans R^n est une fonction réelle de n variables $x_1, x_2, ..., x_n$, ayant la forme suivante :

$$\varphi(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_i x_j. \tag{1.7}$$

Représentation matricielle d'une forme quadratique :

Soient $x^T = (x_1, x_2, \ldots, x_n)$,$A = (a_{ij}, 1 \leq i \leq n, 1 \leq j \leq n)$.
Par exemple, dans le cas de trois variables, on aura :

$$\varphi(x) = \sum_{i=1}^{3}\sum_{j=1}^{3} a_{ij}x_i x_j, \tag{1.8}$$

où

$$\varphi(x) = a_{11}x_1^2 + a_{12}x_1x_2 + a_{13}x_1x_3 + a_{21}x_2x_1 + a_{22}x_2^2 + a_{23}x_2x_3 + a_{31}x_3x_1 + a_{32}x_3x_2 + a_{33}x_3^2$$

$$= x_1(a_{11}x_1 + a_{12}x_2 + a_{13}x_3) + x_2(a_{21}x_1 + a_{22}x_2 + a_{23}x_3) + x_3(a_{31}x_1 + a_{32}x_2 + a_{33}x_3)$$

$$= (x_1, x_2, x_3)\begin{pmatrix} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 \end{pmatrix}$$

$$= (x_1, x_2, x_3)\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

$$= x^T Ax. \tag{1.9}$$

La forme $(1, 8)$ peut encore s'écrire comme suit :

$$\varphi(x) = a_{11}x_1^2 + a_{22}x_2^2 + a_{33}x_3^2 + (a_{12} + a_{21})x_1x_2 + (a_{13} + a_{31})x_1x_3 + (a_{23} + a_{32})x_2x_3$$

En posant $d_{ij} = d_{ji} = \frac{a_{ij}+a_{ji}}{2}, 1 \leq i, j \leq n$, on remarque que pour $i \neq j$,le coefficient devant le produit

$x_i x_j$, est égal à $(a_{ij} + a_{ji})$. On aura alors :

$$\varphi(x) = x^T Ax = x^T Dx. \qquad \textbf{(1.10)}$$

où $D = (d_{ij}, 1 \leq i, j \leq n)$ est une matrice symétrique.

1.2.13 Le gradient d'une forme quadratique :

Soit $\varphi(x) = x^T Dx$ une forme quadratique, où $x \in R^n$, $D^T = D = (d_1, d_2, .., d_n)$ et d_i est un vecteur-colonne :

$$d_i = \begin{pmatrix} d_{1i} \\ d_{2i} \\ \vdots \\ d_{ni} \end{pmatrix}.$$

On a alors :

$$Dx = \begin{pmatrix} d_{11}x_1 + d_{12}x_2 + \ldots + d_{1n}x_n \\ d_{21}x_1 + d_{22}x_2 + \ldots + d_{2n}x_n \\ \vdots \quad \vdots \quad \ddots \quad \vdots \\ d_{n1}x_1 + d_{n2}x_2 + \ldots + d_{nn}x_n \end{pmatrix} = \begin{pmatrix} d_1^T x \\ d_2^T x \\ \vdots \\ d_n^T x \end{pmatrix}.$$

D'où

$$\varphi(x) = x_1 d_1^T x + x_2 d_2^T x + ... + x_n d_n^T x.$$

Par dérivation de la fonction φ par rapport à chaque composante x_j et cela pour $j = 1, 2, ..., n$, on obtient les expressions suivantes :

$$\begin{aligned}
\tfrac{\partial \varphi(x)}{\partial x_1} &= d_1^T x + x_1 d_{11} + x_2 d_{21} + ... + x_n d_{n1} = 2d_1^T x, \\
\tfrac{\partial \varphi(x)}{\partial x_2} &= d_2^T x + x_1 d_{21} + x_2 d_{22} + ... + x_n d_{2n} = 2d_2^T x, \\
&\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\
\tfrac{\partial \varphi(x)}{\partial x_n} &= d_n^T x + x_1 d_{n1} + x_2 d_{n2} + ... + x_n d_{nn} = 2d_n^T x.
\end{aligned}$$

Finalement, on obtient :

$$\nabla \varphi(x) = 2 \begin{pmatrix} d_1^T x \\ d_2^T x \\ \vdots \\ d_n^T x \end{pmatrix} = 2Dx.$$

1.2.14 Le hessien d'une forme quadratique :

Soit une fonction réelle de classe C^2 , $\varphi : R^n \to R$ .Le hessien de la fonction $\varphi(x)$ est donné par :

$$\nabla^2\varphi(x) = (\nabla\frac{\partial\varphi}{\partial x_1}, \nabla\frac{\partial\varphi}{\partial x_2}, \ldots, \nabla\frac{\partial\varphi}{\partial x_j}, \ldots, \nabla\frac{\partial\varphi}{\partial x_n}) = \begin{pmatrix} \frac{\partial^2\varphi}{\partial^2 x_1} & \frac{\partial^2\varphi}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2\varphi}{\partial x_1 \partial x_n} \\ \frac{\partial^2\varphi}{\partial x_2 \partial x_1} & \frac{\partial^2\varphi}{\partial^2 x_2} & \cdots & \frac{\partial^2\varphi}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2\varphi}{\partial x_n \partial x_1} & \frac{\partial^2\varphi}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2\varphi}{\partial^2 x_n} \end{pmatrix}.$$

Donc le hessien de la forme quadratique (1.10) est :$\nabla^2\varphi(x) = 2D$ **.**

1.2.15 La dérivée directionnelle :

Soit $\varphi : R^n \to R$, une fonction de classe C^1. La dérivée directionnelle de φ dans la direction d au point x est :

$$D\varphi(x, d) = \nabla\varphi(x)d.$$

Exemple :

Par exemple, la matrice associée à la forme quadratique $\varphi(x) = 9x_1^2 + 7x_2^2 - 8x_1x_2 + 6x_2x_3$ est telle que :

$$D = \begin{pmatrix} 9 & -4 & 0 \\ -4 & 7 & 3 \\ 0 & 3 & 0 \end{pmatrix}.$$

$$\nabla\varphi(x) = \begin{pmatrix} 18x_1 - 8x_2 \\ 14x_2 - 8x_1 + 6x_3 \\ 6x_2 \end{pmatrix} = 2Dx$$**, est le gradient de φ au point x .**

et $$\nabla\varphi^2(x) = \begin{pmatrix} 18 & -8 & 0 \\ -8 & 14 & 6 \\ 0 & 6 & 0 \end{pmatrix} = 2D$$**, est le hessien de φ au point x.**

1.2.16 La convexité :

1.2.16.1 Les ensembles convexes [2] :

Un ensemble $K \subset \mathbb{R}^n$ est dit convexe si pour tout couple $(x, y) \in K$ et $\lambda \in [0, 1]$, on a :

$$\lambda x + (1 - \lambda)y \in K.$$

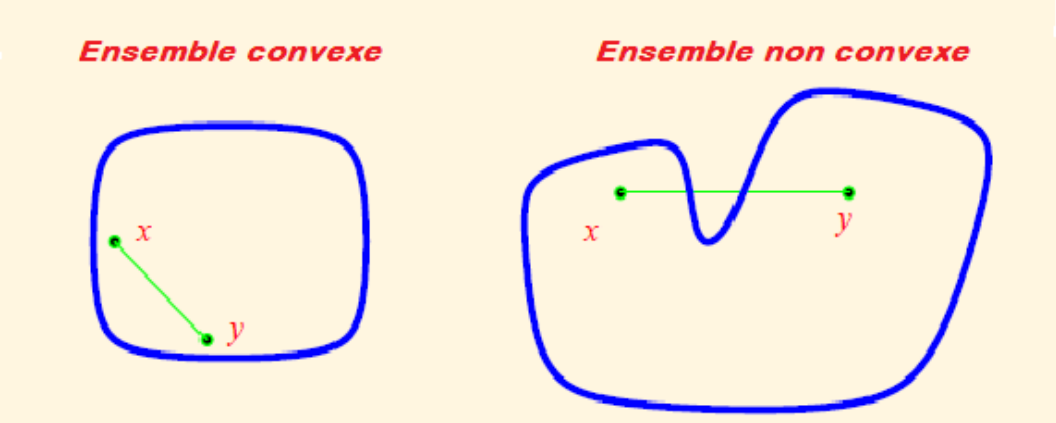

Figure 1.1 – Ensemble convexe et non convexe

Géométriquement, cette notion s'interprète comme suit :
" Pour tout segment reliant deux points quelconques x et y de K, le segment $[x, y]$ doit être aussi inclus dans K ."

1.2.16.2 Les fonctions convexes [2] :

Définition.1.6 :

On dit qu'une fonction $\varphi : K \to R$, définie sur un ensemble convexe K, est convexe si elle vérifie l'inégalité suivante :

$$\forall (x, y) \in K^2, \forall \lambda \in [0, 1], \quad \varphi(\lambda x + (1 - \lambda)y) \leq \lambda\varphi(x) + (1 - \lambda)\varphi(y).$$

La fonction φ est dite strictement convexe si l'inégalité ci-dessus est stricte pour $x \neq y$ et $\lambda \in]0, 1[$.

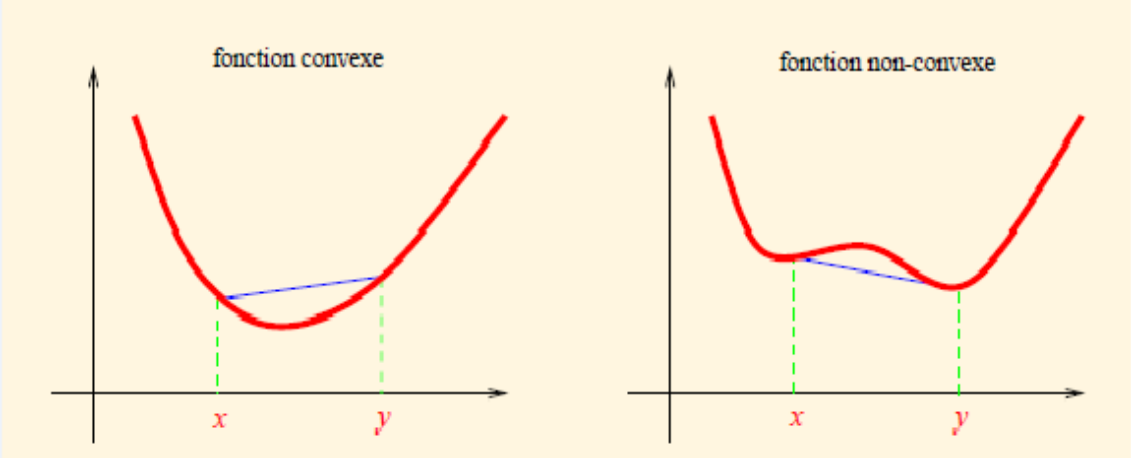

Figure 1.2 – Exemple de fonction convexe et non convexe

1.2.16.3 Enveloppe convexe [7] :

Étant donné un sous-ensemble A de R^n, l'espace R^n est un convexe contenant A. De plus, l'intersection de convexes contenant A étant convexe, on peut poser la définition suivante.

Définition.1.7 :

L'enveloppe convexe de $A \subset R^n$ est le plus petit convexe (au sens de l'inclusion) qui contient A. Elle est notée $Conv(A)$.

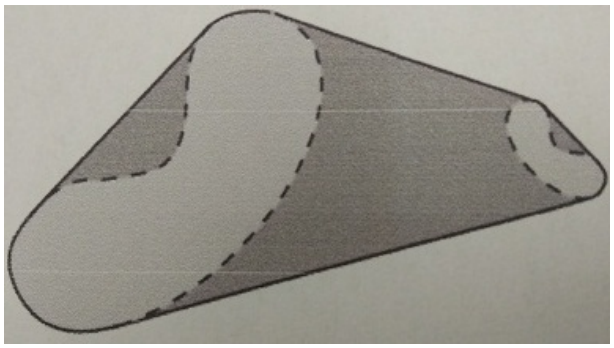

Figure 1.3 – Enveloppe convexe (en clair et en sombre) d'un sous-ensemble (en clair)

Propriétés :

– **Soit φ une fonction réelle de classe $\mathcal{C}^1$, définie sur un ensemble convexe $K \subset \mathbb{R}^n$. Alors φ est convexe, si et seulement si :**

$$\varphi(y) - \varphi(x) \geq \nabla^T \varphi(x)(y - x), \forall x, y \in K.$$

– **Soit φ une fonction réelle de classe $\mathcal{C}^2$, définie sur un ensemble convexe $K \subset \mathbb{R}^n$. Alors φ est convexe, si et seulement si :**

$$(y-x)^T H(x)(y-x) \geq 0, \forall x, y \ \in K,$$

où $H(x) = \nabla^2\varphi(x)$.

Définition.1.7 :

Soit une forme quadratique $\varphi(x) = x^T Dx$. Alors

– **φ est dite définie positive si** $\varphi(x) > 0, \quad \forall x \neq 0$ **;**
– **φ est dite définie négative si** $\varphi(x) < 0, \quad \forall x \neq 0$ **;**
– **φ est dite semi-définie positive ou non négative si** $\varphi(x) \geq 0, \quad \forall x \in R^n$ **;**
– **φ est dite semi-définie négative si** $\varphi(x) \leq 0, \quad \forall x \in R^n$ **;**
– **La forme quadratique $\varphi(x)$ est dite non définie si $\varphi(x)$ est positive pour certaines valeurs de x et négative pour d'autres.**

1.2.17 Critère de Sylvester :

Pour caractériser une forme quadratique définie positive ou semi-définie positive, on se sert du critère de Sylvester. Pour l'établir considérons une matrice symétrique.

$$D = \begin{pmatrix} d_{11} & d_{12} & \dots & d_{1n} \\ d_{21} & d_{22} & \dots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \dots & d_{nn} \end{pmatrix}.$$

Définition.1.8 [2] :

Le mineur de la matrice D formé des lignes $(i_1, i_2, ..., i_p)$ et des colonnes $(j_1, j_2, ..., j_p)$ sera noté comme suit :

$$D\begin{pmatrix} i_1, i_2, \dots, i_p \\ j_1, j_2, \dots, j_p \end{pmatrix} = \begin{vmatrix} d_{i_1 j_1} & d_{i_1 j_2} & \dots & d_{i_1 j_p} \\ d_{i_2 j_1} & d_{i_2 j_2} & \dots & d_{i_2 j_p} \\ \vdots & \vdots & \ddots & \vdots \\ d_{i_p j_1} & d_{i_p j_2} & \dots & d_{i_p j_p} \end{vmatrix}.$$

Ce mineur est dit principal si $i_1 = j_1, i_2 = j_2, \dots, i_p = j_p$, c'est à dire s'il est formé des lignes et des colonnes portant les même numéros. Les mineurs suivants :

$$D_1 = d_{11}, \quad D_2 = \begin{vmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{vmatrix}, \dots, \quad D_n = \begin{vmatrix} d_{11} & d_{12} & \dots & d_{1n} \\ d_{21} & d_{22} & \dots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \dots & d_{nn} \end{vmatrix},$$

sont appelés mineurs principaux successifs.

Nous avons alors le critère de Sylvester qui se formule comme suit :

Théorème 2 [2] : ***(Critère de Sylvester)***

1. **Pour que la matrice D soit définie positive, il est nécessaire et suffisant que tous ses mineurs principaux successifs soient strictement positifs :**

$$D_1 > 0, D_2 > 0, \ldots, D_n > 0;$$

2. **Pour que la matrice D soit semi-définie positive, il est nécessaire et suffisant que tous ses mineurs principaux soient non négatifs :**

$$D\begin{pmatrix} i_1, i_2, \ldots, i_p \\ j_1, j_2, \ldots, j_p \end{pmatrix} \geq 0, 1 \leq i_1 < i_2 < \ldots < i_p \leq n, p = 1, 2 \ldots n.$$

3. **Une matrice D est définie négative** $\Leftrightarrow (-1)^p D_p > 0, p = 1, 2, \ldots, n.$
4. **Une matrice D est semi-définie négative ($i.e$, définie non positive) est équivalent à dire que :**

$$(-1)^p D\begin{pmatrix} i_1, i_2, \ldots, i_p \\ j_1, j_2, \ldots, j_p \end{pmatrix} \geq 0, 1 \leq i_1 < i_2 < \ldots < i_p \leq n, p = 1, 2 \ldots n.$$

Remarque : **La condition**

$$D_1 \geq 0, D_2 \geq 0, \ldots, D_n \geq 0,$$

n'est pas suffisante pour que la matrice D soit définie non négative.

Contre-exemple :

$$D = \begin{pmatrix} 0 & 0 \\ 0 & -4 \end{pmatrix}$$

On a $D_1 \geq 0, D_2 \geq 0$ mais la matrice D est semi définie négative car $D\begin{pmatrix} i_1 \\ j_1 \end{pmatrix} = D\begin{pmatrix} 2 \\ 2 \end{pmatrix} = -4 < 0.$

• Le critère de Sylvester n'est valable que pour les matrices symétriques.

Propriétés :

Soit une forme quadratique $\varphi(x) = x^T Dx$, où D est une matrice symétrique.

• φ est convexe si et seulement si la matrice D est définie non négative ($D \geq 0$).

• Un élément diagonal d'une matrice symétrique D semi-définie positive ne peut s'annuler que si les autres éléments de la même ligne et colonne s'annulent aussi.

• Soit D définie non négative. Si x est un point quelconque mais fixé de $\mathbb{R}^n$ tel que $x^T Dx = 0$, on a alors $Dx = 0$.

CHAPITRE 2

LA PROGRAMMATION QUADRATIQUE

2.1 Introduction

Dans ce chapitre, nous rappelons les concepts mathématiques que nous jugeons nécessaires pour la présentation de ce projet. Par la suite, nous présentons les conditions nécessaires et suffisantes pour l'optimalité des problèmes quadratiques convexes et non convexes, avec certaines méthodes de résolution.

2.2 Minimisation non linéaire sans contraintes

Dans cette section, nous décrivons les conditions nécessaires et suffisantes d'optimalité pour la minimisation d'une fonction non linéaire différentiable sans contraintes, ensuite les appliquer pour le cas où la fonction est convexe.

Compacité du domaine [3] :

Théorème 1 : ***(Existence d'extremum sur un domaine compact.)***

Si K est un compact ($i.e$, est fermé et borné) de R^n, et $\varphi : K \to R$ est continue, alors φ admet un minimum ainsi qu'un maximum global sur K.

2.2.1 Conditions nécessaires de minimalité locale

Définition [2] 2.1 : Soit φ une fonction réelle, définie sur un ensemble ouvert K de R^n . La fonction φ admet ***un minimum local*** en $x^* \in K$ si :

$$\exists B(x^*, \varepsilon) = \{x / \|x - x^*\| < \varepsilon\} \subset K | \varphi(x) \geq \varphi(x^*), \forall x \in B(x^*, \varepsilon).$$

Définition [2]2.2 : Soit φ une fonction réelle, définie sur un ensemble ouvert K de R^n . La fonction φ admet ***un minimum local*** en $x^* \in K$ au sens strict si :

$$\exists B(x^*, \varepsilon) \subset K \ \mid \ \varphi(x) > \varphi(x^*), \forall x \in B(x^*, \varepsilon), x \neq x^*.$$

Définition 2.3 : Soit une fonction réelle φ , définie sur un ensemble ouvert K de R^n . La fonction φ admet ***un minimum global*** en $x^* \in K$ si :

$$\varphi(x) \geq \varphi(x^*), \forall x \in K.$$

Définition 2.4 : Soit φ une fonction définie et différentiable sur R^n. Un point x^* est appelé point stationnaire de φ si $\nabla\varphi(x^*) = 0$.

Théorème 2 [2] : Si x^* est un minimum local (ou global) de φ sur R^n et si φ est différentiable en x^*, alors :

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

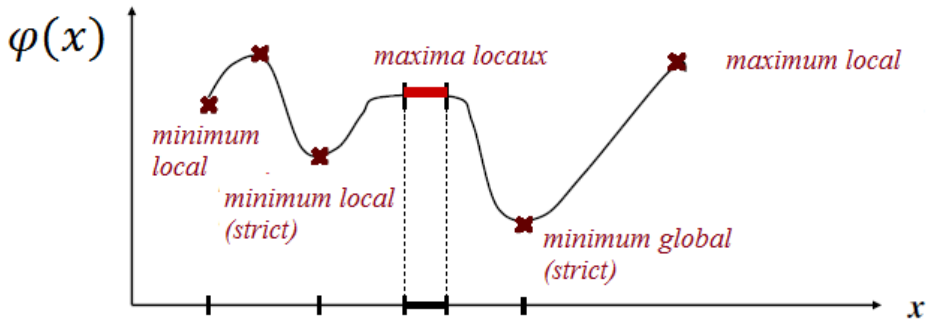

Figure 2.1 – Représentation de la différence entre un point minimum local et un point minimum global

Théorème 3 [5] : Si x^* est un minimum local (global) de φ sur R^n et si φ est deux fois différentiable en x^*, alors on a
(i) $\nabla\varphi(x^*) = 0$ (stationnarité),
(ii) $H(x^*)$ est semi défini positif, où $H(x^*) = \nabla^2\varphi(x^*)$ est le hessien de φ au point x^*.

Preuve : Soit x^* un minimum local de φ. Soit le développement de Taylor de φ au voisinage de x^* :

$$\varphi(x) - \varphi(x^*) = \nabla^T\varphi(x^*)(x - x^*) + \tfrac{1}{2}(x - x^*)^T H(x^*)(x - x^*) + \|(x - x^*)\|^2\varepsilon(x - x^*),$$

avec $\varepsilon(x - x^*) \to 0$ quand $x \to x^*$.
Si $\nabla\varphi(x^*) \neq 0$ alors en choisissant $x = x^* - \theta\nabla\varphi(x^*)$ on aurait $\varphi(x) < \varphi(x^*)$ pour $\theta > 0$ suffisamment petit, ce qui contredit le fait que x^* soit un minimum local de φ. Donc la condition (i) est bien nécessaire.
Si la matrice $H(x^*)$ n'est pas semi-définie positive,*i.e*, il existe un vecteur $d \in R^n, d \neq 0$,tel que $d^T H(x^*)d < 0$. En choisissant $x = x^* + \theta d$ pour $\theta > 0$ suffisamment petit on aura $\varphi(x) < \varphi(x^*)$, vu que $\nabla\varphi(x^*) = 0$, ce qui contredit le fait que x^* soit un minimum. Donc la condition (ii) est aussi nécessaire.

2.2.2 Conditions suffisantes de minimalité locale

Théorème 4 [2] : Si x^* est un point où φ est deux fois différentiable, et si :
(i) $\nabla\varphi(x^*) = 0$ (stationnarité),
(ii) $H(x^*)$ est définie positive,
alors x^* est un minimum local strict de φ sur R^n.

Preuve : Considérons le point x^* vérifiant les conditions (i) et (ii) du théorème .Soit le développement de Taylor de φ en x^* :

$$\varphi(x) = \varphi(x^*) + \frac{1}{2}(x - x^*)^T H(x^*)(x - x^*) + \|(x - x^*)\|^2\varepsilon(x - x^*),$$

avec $\varepsilon(x - x^*) \to 0$ quand $x \to x^*$. Au voisinage de x^*, tout vecteur x peut s'écrire sous cette forme : $x = x^* + \theta d$, où d est une direction telle que $\|d\| = 1$. On a alors :

$$\varphi(x) = \varphi(x^* + \theta d) = \varphi(x^*) + \frac{\theta^2}{2}d^T H(x^*)d + \theta^2\varepsilon(d), \tag{2.1}$$

avec $\varepsilon(d) \to 0$ quand $\theta \to 0$ .En vertu de (ii) on a $d^T H(x^*)d > 0$, alors pour un $\theta > 0$ suffisamment petit, on aura $\varphi(x^* + \theta d) > \varphi(x^*)$.
Ce qui prouve que x^* est bien un minimum local strict de φ.

2.2.3 Minimisation convexe sans contraintes

Théorème 5 [3] :

Si φ est une fonction convexe continûment différentiable, une condition nécessaire et suffisante pour que x^* soit un minimum global de φ sur R^n et que $\nabla\varphi(x^*) = 0$. Autrement dit, dans le cas des fonctions convexes, la stationnarité à elle seule constitue une condition nécessaire et suffisante de minimalité globale.

Définition 2.5 [3] : *L'épigraphe* d'une fonction φ, noté épi(φ) est l'ensemble suivant des vecteurs à $(n+1)$ composantes (x, μ) :

$$épi(\varphi) = \{(x,\mu)/\varphi(x) \leq \mu, x \in R^n, \mu \in R\} \subset R^{n+1}.$$

Définition 2.6 : Soit $\varphi : K \subset R^n \to R$, une fonction convexe définie sur un ensemble K convexe. Alors $\forall a \in R$ ***l'ensemble de niveau*** :

$$N_a = \{x \in K : \varphi(x) \leq a\}$$

est convexe.

Applications coercives [3] :

Définition 2.7 : Soit une application $\varphi : D \subset R^n \to R$, continue et convexe. Elle est dite coercive si D est fermé, non borné et si :

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

(souvent $D = R^n$).

Théorème 5 [3] : (Une application coercive.)

Une application coercive φ admet un minimum global(et aucun maximum global). Si $-\varphi$ est coercive, φ admet un maximum global (et aucun minimum global).

2.3 Minimisation non linéaire avec contraintes

Dans cette section, nous citons les conditions nécessaires et suffisantes d'optimalité pour la minimisation d'une fonction non linéaire différentiable sous contraintes, nous présentons certaines propriétés à propos du cas convexe, puis nous traitons le problème de minimisation d'une fonction quadratique convexe avec une contrainte quadratique convexe.

2.3.1 Position du problème :

On considère un problème d'optimisation non linéaire avec contraintes non linéaires, qui se formule de la manière suivante :

$$\begin{cases} min \ \ \varphi(x), \\ g(x) \leq 0, \\ x \in R^n. \end{cases} \tag{2.1}$$

notons par K l'ensemble $K = \{x \in R^n \ : \ g(x) \leq 0\}$, et supposons que les fonctions φ et g sont de classe C^2 (deux fois continûment différentiables sur R^n).

Définition 2.8

1. Un vecteur $x \in R^n$ est appelé solution réalisable ou plan du problème (2.1) s'il vérifie la contrainte du problème, c'est-à-dire, que $x \in K$.
2. Une solution réalisable x^* est appelée solution optimale du problème (2.1) si

$$\varphi(x^*) \leq \varphi(x),\ \forall x \in K,$$

et on note $\min\limits_{x \in K} \varphi(x) = \varphi(x^*)$.

2.3.2 Condition nécessaires et suffisantes de minimalité de Karush-Kuhn-Tucker (KKT)

Définition 2.9 [4] : La fonction $L(x, \lambda) = \varphi(x) + \lambda g(x)$ est appelée ***fonction de Lagrange*** associée au problème de minimisation de φ sur K, où le scalaire $\lambda \geq 0$, est appelé ***multiplicateur de Lagrange.***

Le théorème suivant est fondamental et donne une condition nécessaire d'optimalité locale pour un problème d'optimisation avec contraintes du type (2.1) :

Théorème 6 [4] . ***(condition nécessaire, ordre 1) :*** On suppose que φ et g sont continûment différentiables sur K. Si φ restreinte à K présente un extremum en $x^* \in K$ alors le lagrangien admet un point critique $(x^*; \lambda^*)$.

Le théorème suivant donne La version ≪ faible ≫ pour les conditions suffisantes d'optimalité locale pour un problème d'optimisation avec contraintes du type (2.1) :

Théorème 7 [4] ***(Condition suffisante d'ordre 2) :*** On suppose que φ et g sont deux fois continûment différentiables (non nécessairement convexes) sur R^n.

Une condition suffisante pour que x^* soit un point minimum *local* de (2.1) est que :

(a) il existe un voisinage $V(x^*)$ de x^* dans lequel les fonctions $\varphi(x)$ et $g(x)$ soient convexes.

(b) il existe $\lambda^* \geq 0$ tel que (x^*, λ^*) soit un point-col restreint au voisinage $V(x^*)$, c'est à dire vérifiant :

$$L(x^*, \lambda) \leq L(x^*, \lambda^*) \leq L(x, \lambda^*) \quad \forall x \in K \cap V(x^*) \forall \lambda \geq 0.$$

2.3.3 Programmation convexe

Un problème de programmation mathématique convexe(resp.strictement convexe), consiste à minimiser une fonction convexe(resp.strictement convexe) sur un domaine convexe.

Propriétés des problèmes convexes [2,8] :

Propriété. 2.1 : L'ensemble M des points où φ atteint son minimum est convexe.

Propriété. 2.2 : Tout problème strictement convexe admet au plus une solution.

Propriété. 2.3 : Tout minimum local est un minimum global.

Propriété. 2.4 : Si φ est strictement convexe, alors son minimum global est atteint en un seul point x^*.

Supposons que les fonctions φ et g sont convexes, le théorème suivant décrit une condition nécessaire et suffisante pour que $x^* \in K$ soit un point de minimum pour le problème (2.1).

Théorème 8 [4] : Théorème de Karush-Kuhn-Tuker (1951) :

Le point x^* est un minimum de φ sur K si et seulement si $\exists \lambda^* \geq 0$ tel que :

$$\begin{cases} (i) \quad \nabla\varphi(x^*) + \lambda^* \nabla g(x^*) = 0, \\ (ii) \quad \lambda^* g(x^*) = 0, \\ (iii) \quad g(x^*) \leq 0. \end{cases}$$

2.4 Minimisation d'une fonction quadratique convexe sous une contrainte quadratique convexe

2.4.1 Position du problème

Nous nous intéressons au cas où le problème (2.1) prend cette forme :

$$\min_{x \in K} \ \varphi(x) = \frac{1}{2} x^T D x + c^T x, \tag{2.2}$$

où $K = \{x \in R^n \backslash \ \ g(x) = \frac{1}{2}\|x\|^2 \leq b\}$ désigne l'ensemble des solutions réalisables. On remarque bien que les fonctions φ et $g(x) = \frac{1}{2}\|x\|^2 - b = \frac{1}{2} x^t I_n x - b$ sont de classe C^2 (deux fois continûment différentiables sur R^n), où I_n est la matrice identité d'ordre n.

Pour que la fonction objectif φ soit convexe, la matrice D est supposée semi-définie positive.

La matrice I_n est définie positive, ceci nous assure la convexité du domaine des solutions réalisables. On est ainsi en présence d'un problème d'optimisation différentiable convexe.

2.4.2 Condition d'optimalité globale

La fonction de Lagrange associée au problème (2.2) est :

$$L(x, \lambda) = \frac{1}{2} x^t (D + \lambda I_n) x + c^t x - b\lambda.$$

Le point x^* est une solution du problème (2.2) si et seulement s'il existe un nombre réel unique $\lambda^* \geq 0$ tel que :

(i) $(D + \lambda^* I)x^* = -c$.

(ii) $\|x^*\|^2 \leq b$.

(iii) $\lambda^* = 0$ si $x^* \in int(K)$ (x^* est à l'intérieur de l'ensemble K) où $\lambda^* > 0$ si $\|x^*\|^2 = b$ (x^* est un point à la frontière de l'ensemble K).

2.4.3 Méthodes de résolution

Cette section porte sur une classe particulière de techniques d'optimisation qui servent à résoudre les (PQQC). Les méthodes envisagées sont itératives : elles construisent, au fur et à mesure des itérations, une suite $\{x^k\}$ qui convergera vers un minimum global x^*.

Méthode du gradient projeté (Rosen 1960)

Une idée assez naturelle, pour adapter les méthodes d'optimisation sans contraintes aux problèmes avec contraintes, est de projeter à chaque étape le déplacement sur la frontière du domaine, afin de s'assurer que le nouveau point obtenu appartient à l'ensemble des solutions réalisables.
En modifiant ainsi le moins possible la méthode d'origine, on peut espérer conserver son efficacité. De nombreux algorithmes basés sur ce schéma général ont été proposés.
Pour la méthode du gradient, par exemple, on est conduit à projeter le gradient sur la frontière du domaine. Ceci donne un cheminement le long de la frontière dans la direction de la plus forte pente « relative », c'est-à-dire autorisée par les contraintes ; un des premiers algorithmes construits suivant ce principe est le gradient projeté de Rosen (1960).

La méthode du gradient projeté est essentiellement intéressante dans le cas des contraintes linéaires. Elle peut être généralisée dans le cas où les contraintes sont non linéaires mais, comme nous allons le voir, des difficultés apparaissent qui rendent son emploi beaucoup plus délicat.[4]

Principe de la méthode :

une itération consiste à chercher depuis un point initial x_0 une direction de descente dans l'hyperplan tangent et à trouver dans cette direction la meilleure solution, cette direction est celle qui maximise la dérivée directionnelle de φ.

Théorème 9 [4]

La solution optimale du problème :

$$\begin{cases} \min\limits_{d\in R^n} \; d^t\nabla\varphi(x_0) = d^t(Dx_0 + c), \\ x_0^t d = 0, \\ \|d\| = 1, \end{cases}$$

est $d = \frac{\overline{y}}{\|\overline{y}\|}$, où $\overline{y}$ est la projection de $-\nabla\varphi(x_0)$ sur $S^0 = \{y | x_0^t y = 0\}$ donnée par :

$$\overline{y} = -P^0\nabla\varphi(x_0) = -(I_n - \frac{1}{\|x_0\|^2}xx^t)(Dx_0 + c).$$

La matrice P^0 est la matrice de projection sur S^0.

Algorithme de la méthode [4] :

(a) Initialisation : $x_k = x_0$.
(b) Tant que x_k n'est pas acceptable :
Déterminer si $\|x_k\|^2 = b$ ou non.
Si $\|x_k\|^2 = b$:
Calculer $P^0 = I_n - \frac{1}{\|x_k\|^2} x_k x_k^t$.
Sinon :
$P^0 = I_n$.
(c) Calculer la direction de descente $y_k = -P^0(Dx_k - c)$.
(d) Si $y_k \neq 0$ alors :
Calculer le pas $\theta_{max} = max\{\theta \mid \|x_k + \theta y_k\|^2 \leq b\}$.
Calculer x_{k+1} tel que $\varphi(x_{k+1}) = \min\limits_{0<\theta<\theta_{max}} \varphi(x_k + \theta y_k)$.
$x_k \leftarrow x_{k+1}$.
Sinon :
Poser $u = -\frac{1}{2\|x_k\|^2} x_k^t (Dx_k + c)$.
(e) Si $u \geq 0$ alors x_k vérifie les conditions de KKT, x_k est acceptable.
Sinon :
La contrainte n'est plus considérée comme saturée. Aller à (b).

Méthodes de pénalité [4,10]

Les méthodes de pénalité sont des méthodes duales, i.e, le concept de base de ces méthodes est de transformer la résolution du problème sous contraintes en une suite de résolutions de problèmes sans contraintes en associant à l'objectif une pénalité dès qu'une contrainte est violée.

Principe des méthodes de pénalité [10] :

La fonction objectif $\varphi(x)$ du problème (2.2) est alors remplacée par la fonction suivante à minimiser :

$$P(x, \theta) = \varphi(x) + \theta h(x),$$

où $h(x)$ est la fonction pénalité, continue, dépendant des contraintes $g(x)$, θ est un coefficient de pénalité, toujours positif. La fonction de pénalité est choisie de telle façon que la possibilité de réalisation soit garantie dans tous les processus de recherche de l'optimum. Cette caractéristique est très importante pour éviter un arrêt prématuré de l'algorithme d'optimisation.

Suivant les types de contraintes et le type de fonction $h(x)$, on distingue les méthodes de pénalités intérieures et les méthodes de pénalités extérieures.

Méthodes de pénalités extérieures :

La fonction $h(x)$ est utilisée afin de défavoriser les positions non admissibles. La fonction de pénalité doit être continue et à dérivées continues :

$$h(x) = \big[max\big(0, g(x)\big)\big]^2.$$

Les méthodes de pénalités sont en général utilisées de manière itérative : une suite de valeurs croissantes θ^k de θ est générée et à chaque itération k du processus, le problème d'optimisation sans contraintes suivant est résolu :

$$P(x, \theta, k) = \varphi(x) + \theta^k \big[max\big(0, g(x)\big)\big]^2. \tag{2.3}$$

Lorsque k tend vers l'infini le problème (2.3) devient équivalent à notre problème contraint .

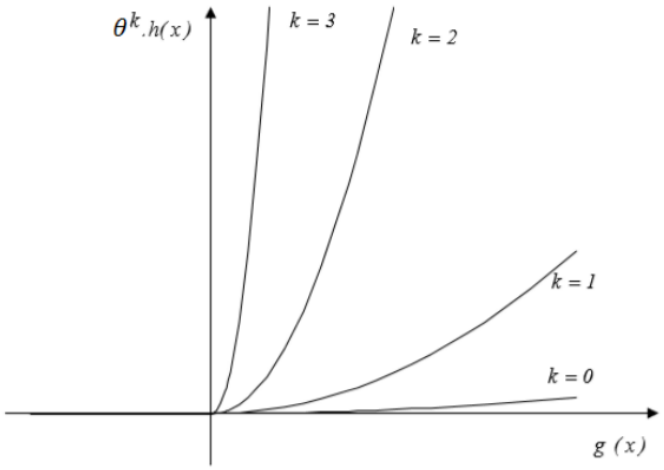

Figure 2.2 – Fonctions de pénalité extérieure pour $\theta = 10$.

L'avantage de cette méthode est que le point de départ n'est pas nécessairement admissible tout en garantissant que le point final sera dans le domaine admissible ou presque. La fonction de pénalité extérieure est continue dans tout le domaine d'étude, admissible comme non admissible, mais elle présente l'inconvénient de conduire à un optimum réalisable seulement quand k tend vers l'infini et d'approcher ce point par une série de solutions non admissibles.

Algorithme de la méthode[10] :

1. Initialisation : choix de $x^0 \in K$, $\theta^1 > 0$;
2. Pour $k = 1, 2, ...$
2.1. Trouver un point stationnaire approché x^k de $P(x^k, \theta^k)$ (en partant par exemple de x^{k-1}) vérifiant :

$$\|\nabla P(x^k, \theta^k) \leq \epsilon\|$$

2.2 Si x^k est satisfaisant(vérifie approximativement les conditions de KKT), on arrête ;
Sinon
2.3 Choisir $\theta^{k+1} > \theta^k$ et aller en 2.

Théorème 10 : (Convergence de la pénalisation extérieure)[10]

Soit $h(\mathbb{R}^n \to \mathbb{R})$ une fonction de pénalisation extérieure vérifiant :

- $h(x) \geq 0$ $(\forall x)$.
- $h(x) = 0 \iff x \in K = \{x/g(x) \leq 0\}$.
- h continue.

On suppose d'autre part, que φ est une fonction continue, et que K est fermé, et que l'une des deux conditions suivantes est vérifiée :

i) La fonction $\varphi(x) \to +\infty$ quand $\|x\| \to +\infty$.

ii) L'ensemble K est borné et $h(x) \to +\infty$ quand $\|x\| \to +\infty$.

Alors, lorsque le coefficient de pénalité θ tend vert $+\infty$:

- La suite $\bar{x}(\theta)$, ($\bar{x}(\theta)$ un minimum de $P(x, \theta)$), admet au moins un point d'accumulation, et tout point d'accumulation de cette suite est une solution optimale globale du *problème* (2.2) :

- $h\big(\bar{x}(\theta)\big) \to 0$

Méthodes de pénalités intérieures

Dans le cas de la pénalité intérieure, on cherche à définir la fonction $h(x)$ de telle sorte que, plus la contrainte devient active, c'est-à-dire plus x se rapproche de la frontière du domaine admissible, plus la fonction de pénalisation $h(x)$ croit et tend vers l'infini et par conséquent, moins on a de chance de trouver le minimum proche de la frontière du domaine admissible. Cette caractéristique montre que cette technique ne convient pas pour résoudre les problèmes possédant des contraintes d'égalités.

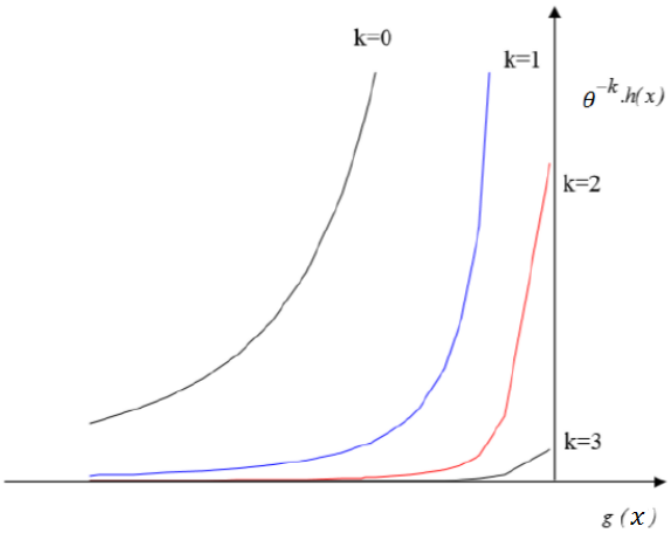

Figure 2.3 – Fonction de pénalité intérieure pour $\theta = 10$.

La fonction de pénalisation intérieure présente l'avantage de conduire toujours à une séquence de solutions réalisables. Néanmoins, elle a l'inconvénient d'être discontinue sur l'interface entre les domaines admissible et interdit. En plus, le point de départ doit obligatoirement être dans la région admissible, ce qui conduit à la

nécessité d'un algorithme supplémentaire pour le trouver.

La fonction de pénalisation intérieure P est définie par

$$P(x,t) = \varphi(x) + \theta h(x),$$

où $h : R^n \to R$ est définie comme suit :

$$h(x) = -\frac{1}{g(x)} \quad ou \quad h(x) = -\frac{1}{\big(g(x)\big)^2} \quad ou \quad h(x) = -\log\big(-g(x)\big).$$

Cette fonction vérifie :

• $h(x) \leq 0 \ \forall x \in int(K)$.

• $h(x) \to +\infty$ lorsque x tend vers la $fontière$ de K.

• Et comme la fonction g est continue, $h(x)$ est continue sur l'intérieur de K.

Une telle fonction est appelée une fonction de pénalisation intérieure (on dit encore, fonction "*barrière*").

Le paramètre réel θ positif, est appelé le *coefficient de pénalité.*

Algorithme de la méthode :

1. Initialisation : choix de $x^0 \in K$, $\theta^1 > 0$;
2. Pour $k = 1, 2, ...$
2.1.Trouver un point stationnaire approché x^k de $P(x^k, \theta^k)$(en partant par exemple de x^{k-1}) ;
2.2. Si x^k est satisfaisant(vérifie approximativement les conditions de KKT), on arrête ;
Sinon
2.3. Choisir $\theta^{k+1} \in]0, \frac{\theta^k}{2}[$ (par exemple), et aller en 2.

Théorème 11 : (Convergence de la pénalisation intérieure)

Supposons que tout $x \in K$ est limite d'une suite de points appartenant $\grave{a}$ l'intérieur vérifiant :

• $h(x) \geq 0 \ \forall x \in int(K)$.

• $h(x) \to +\infty$ quand x tend vers la $frontière$ de K.

• h continue sur $int(K)$.

On suppose, d'autre part, que φ est une fonction continue, et que l'une des deux conditions suivantes soit vérifiée :

i) $\varphi(x) \to +\infty$ quand $\|x\| \to +\infty$.

ii) K est borné .

Alors, lorsque le coefficient de pénalité θ tend vers 0 :

• la suite $\bar{x}(\theta)$, ($\bar{x}(\theta)$ un minimum de $P(x, \theta)$), admet au moins un point d'accumulation et tout point d'accumulation de la suite $\bar{x}(\theta)$ est un optimum global du *problème* (2.2).

• La quantité $\theta h(\bar{x}(\theta))$ tend vers 0.

2.5 Minimisation d'une fonction quadratique non convexe sous une contrainte quadratique convexe

Position du problème :

On considère un problème de minimisation quadratique avec contrainte quadratique d'inégalité (PQQ), se formulant de la manière suivante :

$$\min_{x\in K} \ \varphi(x) = \frac{1}{2}x^T Dx + c^T x, \tag{2.4}$$

où $K = \{x \in R^n \ : \ g(x) = \|x\|^2 \leq b\}$ désigne l'ensemble des solutions réalisables, et D est une matrice symétrique qui n'est pas semi-définie positive ($D \not\succeq 0$).Notons que la fonction $g(x) = \|x\|^2 - b = x^t I_n x - b$ est convexe, où I_n est la matrice-identité d'ordre n.

Détecter le minimum global du problème (2.4) est, en général, une tache difficile,mais la structure spéciale de $\varphi(x)$ et K rend la tâche relativement simple.

2.5.1 La programmation DC

Soit $X = R^n$, Y est le dual de X qui peut être défini par X lui même. On note par $\Gamma_0(X)$ l'ensemble des fonctions continues convexes sur X.

la fonction conjuguée g^* de $g \in \Gamma_0(X)$ est définie par([9]) :

$$g^*(y) = sup\{xy - g(x) : x \in X\}.$$

2.5.2 Fonctions DC

Une fonction $\varphi : K \to [-\infty, +\infty]$ définie sur un ensemble convexe K de R^n, est dite DC sur K si elle peut s'écrire comme la différence de deux fonctions convexes sur K, i.e,

$$\varphi(x) = g(x) - h(x) \quad g, h \in \Gamma_0(K).$$

Le programme DC est donné comme suit ([11]) :

$$(P) \quad \alpha = inf_{x\in K}\{\varphi(x) = g(x) - h(x)\} \quad g, h \in \Gamma_0(K).$$

Avec la convention $+\infty - (+\infty) = +\infty$, il est clair que la finitude de α implique que $Dg \subset Dh$ et $Dh^* \subset Dg^*$ le problème dual est donné par :

$$(D) \quad \alpha = inf_{y\in Y}\{h^*(y) - g^*(y)\}.$$

Proposition [11] : toute fonction deux fois continûment différentiable est DC sur un ensemble convexe compact.

Pour le problème (2.4), nous proposons la composition DC suivante [9] :

$$\varphi(x) = \frac{1}{2}x^t Dx + c^t x + \Psi_K(x) = g_1(x) - h_1(x),$$

avec $g_1(x) = \frac{1}{2}\theta\|x\|^2 + c^t x + \Psi_K(x)$ et $h_1(x) = \frac{1}{2}x^t(\theta I - D)x$, où θ est un réel positif tel que la matrice $(\theta I - D)$ soit semi-définie positive, $K = \{x \in R^n : \|x\|^2 \leq b\}$ et Ψ_K est une fonction indicatrice, i.e,

$$\Psi_K(x) = \begin{cases} 0 & \text{Si } x \in K, \\ +\infty & \text{Sinon.} \end{cases}$$

Il est aussi clair que $g_1, h_1 \in \Gamma_0(K)$ et le problème (2.4) prend la forme ([9]) :

$$(P_{dc}) \quad min\{g_1(x) - h_1(x) : x \in R^n\}.$$

2.5.3 Algorithme de DC (DCA)

Il s'agit d'une nouvelle méthode du gradient, basée sur l'optimalité et la dualité en optimisation DC. Cette approche est complètement différente des méthodes classiques d'optimisation convexe.

Dans les DCA, la construction algorithmique cherche à exploiter la structure DC du problème. La suite des directions de descente est obtenue en calculant une suite de gradients non directement à partir de la fonction φ, mais des composantes convexes des problèmes primal et dual ([11]).

Algorithme général DCA [11]

Etape 0. x^0 donnée.
Etape 1. Pour chaque k, x^k étant connu, déterminer $y^k \in \partial h(x^k)$.
Etape 2. Trouver $x^{k+1} \in \partial g(y^k)$.
Etape 3. Si le test d'arrêt est vérifié STOP ; Sinon $k \leftarrow k+1$ et aller en Etape 1.

Le DCA simplifié appliqué sur le problème (2.4) peut être formulé de cette manière :

Soit $x^0 \in K$ donné et $k \geq 1$.

Faire :

$$y^k = (\theta I - D)x^k$$

et

$$x^{k+1} = argmin\{\frac{1}{2}\theta\|x\|^2 + x^t(c - y^k) + \Psi_K(x) : x \in R^n\}.$$

Il est bien clair que x^{k+1} est la projection de $\frac{1}{\theta}(y^k - c)$ sur K, i.e,

$$x^{k+1} = P_K(x^k - \frac{1}{\theta}(Dx^k + c)). \qquad (1)$$

Algorithme DCA [9]

Soit $\varepsilon > 0$ et $x \in K$ donnés. Initialiser $k = 0$.
(a) Si $\|(\theta I - D)x^k - c\| \leq \theta\sqrt{b}$ prendre $x^{k+1} = \frac{1}{\theta}\left((\theta I - D)x^k - c\right)$.
(b) Sinon, prendre $x^{k+1} = \frac{\sqrt{b}[(\theta I - D)x^k - c]}{\|(\theta I - D)x^k - c\|}$.
Si $\|x^{k+1} - x^k\| \leq \varepsilon$, arrêter. Sinon $k \leftarrow 1$ et aller en (a).

Remarques

(i) Si $c = 0$, nous choisissons x^0 tel que $y^0 = (\theta I - D)x^0 \neq 0$, puisque $y^0 = 0$ implique $x^k = 0, \ \ \forall k \geq 1$.

(ii) En pratique, la convergence de DCA dépend de la valeur de θ. Le meilleur θ^* est défini par :

$$\theta^* = inf\{\theta > 0 : (\theta I - D) \succeq 0\}.$$

(iii) Le multiplicateur de Lagrange λ^* peut être calculé par la formule suivante :

$$\lambda^* = -(1/b)(x^t Dx + c^t x).$$

Théorème 12 [9] :

(i) $\varphi(x^{k+1}) \leq \varphi(x^k) - \frac{1}{2}(\theta + \lambda)\|x^{k+1} - x^k\|^2$, où λ est la plus petite valeur propre de $(\theta I - D)$.

(ii) $\varphi(x^k) \searrow \alpha_1 \geq \alpha$ et $\lim\limits_{k \to +\infty} \|x^{k+1} - x^k\| = 0$.

(iii) Chaque point $x^* = \lim\limits_{k \to +\infty} x^k$ est un point de **KKT**, i.e, il existe $\lambda^* \geq 0$ tel que $(D - \lambda^* I)x^* = -c$, $\lambda^*(\|x^*\|^2 - b) = 0$ et $\|x^*\|^2 \leq b$.

Preuve [9] :

Pour tout $x \in K$ on a :

$$\varphi(x) = \varphi(x^k) + (x - x^k)^t(Dx^k + c) + \frac{1}{2}(x - x^k)^t D(x - x^k).$$

Donc

$$\varphi(x^{k+1}) = \varphi(x^k) + (x^{k+1} - x^k)^t(Dx^k + c) + \frac{1}{2}(x^{k+1} - x^k)^t D(x^{k+1} - x^k). \qquad (2)$$

En utilisant la définition (1) de x^{k+1} on aura :

$$(x^k - x^{k+1})^t(x^k - \frac{1}{\theta}(Dx^k + c) - x^{k+1}) \leq 0,$$

i.e,

$$(x^{k+1} - x^k)^t(Dx^k + c) \leq -\theta||x^{k+1} - x^k||^2. \qquad (3)$$

En combinant (2) et (3) on aura :

$$\varphi(x^{k+1}) \leq \varphi(x^k) - \frac{1}{2}\theta||x^{k+1} - x^k||^2 + \frac{1}{2}(x^{k+1} - x^k)^t D(x^{k+1} - x^k).$$

Finalement, on obtient :

$$\varphi(x^{k+1}) \leq \varphi(x^k) - \theta||x^{k+1} - x^k||^2 + \frac{1}{2}(x^{k+1} - x^k)^t(\theta I - D)(x^{k+1} - x^k).$$

Soit λ la plus petite valeur propre de la matrice semi-définie positive $(\theta I - D)$. Alors on aura :

$$\varphi(x^{k+1}) \leq \varphi(x^k) - \frac{\theta + \lambda}{2}||x^{k+1} - x^k||^2.$$

La suite $\{\varphi(x^k)\}$ est décroissante et bornée inférieurement d'où elle converge vers $\alpha_1 \geq \alpha$. De là, la suite $\{||x^{k+1} - x^k||\}$ converge vers 0 puisque $\theta > 0$ et $\lambda \geq 0$. Ceci, nous conduit à prouver (iii).

CHAPITRE 3

IMPLÉMENTATION DES MÉTHODES SOUS MATLAB

3.1 Introduction

Dans ce chapitre, nous avons implémenté les algorithmes de pénalité extérieure, DCA sous le langage de programmation MATLAB. Cette programmation nous a permis de faire une étude numérique pour prouver leur efficacité.

Les résultats obtenus par les méthodes (temps d'exécution, nombre d'itérations) sont illustrés dans les tableaux ci-après.

3.2 Choix du langage

Le choix s'est porté sur l'emploi du langage du logiciel MATLAB 7.9.0 (R2009b), car il répond aux critères suivants :

– La maniabilité du langage : constitué d'un ensemble de possibilités faisant en sorte que le programmeur travaille avec aisance, assuré d'une part par la syntaxe du langage et d'autre part par un aspect visuel clair représentatif à la fois du détail et du global.

– Le bagage du langage : il contient une interface graphique puissante ainsi qu'une grande variété de méthodes scientifiques implémentées (prédéfinies).[8,14]

3.2.1 Généralités sur le langage .

Le MATLAB est un logiciel parfaitement dédié à la résolution de problèmes d'analyse numérique ou de traitement du signal. Il permet d'effectuer des calculs matriciels ou de visualiser les résultats sous forme graphique. La formulation des problèmes s'apparente à la formulation mathématique des problèmes à résoudre. L'utilisation de ce logiciel consiste à lancer des lignes de commandes, qui peuvent le plus souvent ressembler à la programmation en C.[8,14]

Le nom MATLAB vient de MATrix LABoratory, les éléments de données de base manipulés par MATLAB étant des matrices (mais pouvant évidemment se réduire à des vecteurs et des scalaires) qui ne nécessitent ni dimensionnement ni déclaration de type. Contrairement aux langages de programmation classiques, les fonctions du MATLAB permettent de manipuler directement et interactivement ces données matricielles, le rendant ainsi particulièrement efficace en calcul numérique, analyse et visualisation de données en particulier. Il existe deux modes de fonctionnement sur MATLAB :

Le mode interactif : Les instructions sont exécutées au fur et à mesure qu'elles sont données par l'usager.

Le mode exécutif : Dans ce cas, l'utilisateur utilise un fichier "$M-file$" contenant toutes les instructions à exécuter.

3.2.2 Programmation avec MATLAB .

Il y a deux façons d'écrire des fonctions MATLAB :

(i) Soit directement dans la fenêtre de commandes,

(ii) Soit en utilisant l'éditeur de développement de MATLAB, en sauvegardant les programmes dans des fichiers texte avec l'extension "$.m$". **Fichiers** $^*.m$

Les programmes sauvegardés dans les fichiers MATLAB(*.m) sont alors directement utilisables comme des fonctions MATLAB à partir de la fenêtre de commande. Pour cela, le fichier doit se trouver dans le répertoire MATLAB, qui est en pratique le dossier Work.

Création d'une fonction

La création d'une fonction dans MATLAB se fait par la syntaxe suivante : function$[s1, s2, ...] = nom - fonction(e1, e2, ...)$. Les variables $s1, s2, ...$, sont les arguments de sortie (S) de la fonction et les variables $e1, e2, ...$, sont les arguments d'entrée (E).

Attention : le fichier $M(M-file)$ doit avoir le même nom que la fonction qu'il contient.

Quelques définitions des commandes élémentaires en MATLAB [8,14]

commandes	***définitions***
$function$	définition de fonction
$while$	instruction de répétition avec test
if	test conditionnel
$else$	complète if
$norm(H)$	norme 2 de H (peut s'appliquer à un vecteur V)
$inv(H)$	inverse de H comme (H^{-1})
$'$	transposition de matrice
$tic\ \ toc$	temps d'exécution
$D = rand(n)$	générer un matrice aléatoire D d'ordre $(n \times n)$
$eval(g)$	La valeur de g
$fminsearch(P, x)$	$min\ P(x), \quad x \in R^n$

3.3 Implémentation des méthodes

3.3.1 Code de l'algorithme de pénalité extérieure sous Matlab

```
function [x,iter,phiva]=pen_ext(phi,g,x,theta,a,b)
tic
s = 0;c=[];h=[];
if(eval(g)>0)
   c=[c;'(',g,')^2'];
else
    c=[];
end
h=c(1,:);
iter = 0;
s = (max(s,eval(g)))^2;
while (s > 0)
         P=[phi,'+',num2str(theta),'*(',h,')'];
         P=inline(P);
         [x,phiva]=fminsearch(P,x);
         s=0;
         iter = iter + 1;
         s = (max(s,eval(g)))^2;
         theta=theta*a;
end
[y,l]=fminsearch(phi,x);
if (norm(y)^2<=b)&&(l<phiva)
    x=y;
    phiva=l;
    iter = iter + 1;
end
toc
end
```

3.3.2 Application numérique

Considérons le problème (PQQ) suivant :

$$\begin{cases} min\ \varphi(x) = x_1^2 + \frac{3}{2}x_2^2 - x_1x_2 + 3x_1 - 2x_2 - 4, \\ g(x) = ||x||^2 = x_1^2 + x_2^2 \leq 1, \\ x \in R^2. \end{cases} \tag{3.1}$$

les fonctions φ et g sont strictement convexes, d'où le problème (3.1) est bien un problème de programmation quadratique convexe.

Nous avons résolu le problème (3.1) par la méthode de pénalité extérieure en initialisant :

- Le point initial $x^0 = (2, -2)^t$.
- Le coefficient de pénalité $\theta^0 = 1$.
- Le coefficient d'augmentation $a = 2$ (fixer au cours des itérations, il doit être différent de 1).

La solution du problème est illustré dans la figure suivante :

```
introduire la fonction objectif phi
x(1)^2+(3/2)*x(2)^2-x(1)*x(2)+3*x(1)-2*x(2)-4

phi =

x(1)^2+(3/2)*x(2)^2-x(1)*x(2)+3*x(1)-2*x(2)-4

introduire la contrainte g
x(1)^2+x(2)^2-1

g =

x(1)^2+x(2)^2-1

introduire la valeur de x point de départ
[2;-2]
introduire le coefficient de pénalité > 0
1
introduire le coefficient d'augmentaion de la coef_pen >1
2
b=1
Elapsed time is 1.825097 seconds.
la solution obtenue par la methode de pénalité:

x =

   -0.9627
    0.2705

le nombre d'iterations est: 26
la valeur minimale de la foction objectif est: -6.1322
>>
```

Figure 3.1 – Résultat de l'exécution

3.3.3 Étude de la méthode

Nous avons résolu le problème (3.1) par la méthode de pénalité extérieure en fixant le point initial $x^0 = (2, -2)^t$, le coefficient de pénalité $\theta^0 = 1$ et en variant le coefficient d'augmentation $a > 1$; les résultats obtenus sont illustrés dans le tableau suivant :

a	Nombre d'itérations	Temps d'exécution en (s)
2	26	1,953236
10	10	1,175033
20	8	1,102616
10^2	6	0,978910
10^3	4	0,881514
10^6	3	0,835820

Commentaire :

La convergence de cet algorithme dépend fortement au choix du coefficient d'augmentation a, i.e, l'augmentation du paramètre a implique la rapidité de la résolution, et ce, en moins d'itérations.

Le même raisonnement est fait si nous fixons a et nous varions θ^0, car $\theta^{k+1} = \theta^k \times a$.

Le même problème (3.1) est résolu avec la même méthode, mais en fixant $a = 10$, $\theta^0 = 1$ et en varient le point de départ x^0 (il doit être un point extérieur), les résultats obtenus sont illustrés dans le tableau suivant :

$\|\|x^* - x^0\|\|$	Nombre d'itérations	Temps d'exécution en (s)
$3,7326$	10	$1,264124$
$29,1604$	11	$1,286863$
$283,7151$	12	$1,349517$
$2,82 \times 10^3$	14	$1,439444$

Commentaire :

La convergence de cet algorithme dépend aussi du choix du point initial x^0, i.e, pour x^0 proche de la solution approchée, l'algorithme est rapide et il fait moins d'itérations.

3.3.4 Code de l'algorithme de la composition DC sous Matlab

```
function [y,iter,phiva,lambda]=DCA(D,c,b,x,eps)
tic
n=length(D); iter=1;
theta= max(abs(eig(D)));
K=theta*eye(n)-D;
if norm(K*x-c)<=theta*sqrt(b)
    y=(1/theta)*(K*x-c);
   else
    y=(sqrt(b)*(1/theta)*(K*x-c))/(norm(K*x-c));
end
if norm(x)>eps
    er=(norm(y-x)^2)/(norm(x)^2);
else
    er=norm(y-x)^2;
end
while (er>eps)
    x=y;
   if norm(K*x-c)<=theta*sqrt(b)
       y=(1/theta)*(K*x-c);
   else
     y=(sqrt(b)*((1/theta)*(K*x-c)))/(norm(K*x-c));
   end
   iter=iter+1;
   if norm(x)>eps
    er=(norm(y-x)^2)/(norm(x)^2);
   else
    er=norm(y-x)^2;
   end
end
 phiva=(1/2)*y'*D*y+c'*y;
 lambda=-(1/b)*(y'*D*y+c'*y);
 toc
end
```

3.3.5 Application numérique

Considérons le problème (PQQ) suivant :

$$\begin{cases} min\ \varphi(x) = -x_1^2 - \frac{5}{2}x_2^2 + 4x_1x_2 - 3x_1 - 2x_2, \\ g(x) = ||x||^2 = x_1^2 + x_2^2 \leq 4, \\ x \in R^2. \end{cases} \tag{3.2}$$

la fonction φ n'est pas convexe, d'où le problème (3.2) est bien un problème de programmation quadratique non convexe.

Nous avons résolu le problème (3.2) par la méthode de décomposition DC en initialisant le point initial $x^0 = (\sqrt{2}, \sqrt{2})^t$.

La solution du problème est illustré dans la figure suivante :

```
D=[-2 4;4 -5]
b=4
c=[-3;-4]
eps=0.5
Les valeurs propres de D sont :

eig_D =

   -7.7720
    0.7720

La matrice D est définie non positive
Elapsed time is 0.004253 seconds.
la solution approchée est

y =

    0.4585
    0.7898

Le multiplicateur de lagrange est : 1.2942
Le nombre d'iterations est: 2
La valeur minimale de la foction objectif est: -4.8558
>>
```

Figure 3.2 – Résultat de l'exécution

Remarques :

• Pour $\theta = max|\lambda_i|$, la matrice $(\theta I - D)$ est toujours semi-définie positive, où les λ_i sont les valeurs propres associées à la matrice D.

• Le point x^0 tel que $x_i^0 = \sqrt{\frac{b}{n}}$, $i = 1, ...n$, est toujours réalisable $\forall\ b \in R,\ b > 0$.

Conclusion Générale

Dans ce travail, nous avons d'abord rappelé dans le premier et deuxième chapitre les notions fondamentales d'algèbre linéaire et d'optimisation quadratique convexe et non convexe avec contraintes.

Nous avons aussi proposé quelques méthodes pour la résolution des (PQQ), à savoir :

- La méthode de gradient projeté et les méthodes de pénalité pour résoudre les (PQQ) convexes.
- La décomposition DC pour résoudre les (PQQ) non convexes.

Dans le troisième chapitre, nous avons implémenté les méthodes sous le langage MATLAB. Cette implémentation nous a permis de faire des applications numériques des méthodes avec le temps d'exécution et le nombre d'itérations.

Perspectives

Résoudre un (PQQ) par la méthode de support.
Etudier un (PQQ) général non convexe.

Bibliographie

[1] B. Brahmi. Méthodes primale et duale pour la résolution des problèmes de programmation quadratique convexe. Mémoire de Magister, Université de Béjaia, 2006.

[2] N.Ikhaneche. Méthode de support pour la minimisation d'une fonctionnelle quadratique convexe. Mémoire de Magister. Département de Recherche Opérationnelle, Université de Béjaia, 2004. .

[3] D.Zdenek. Optimal quadratic programming algorithms. Édition Springer, New York, 2009.

[4] M. Minoux. Programmation Mathématique : théorie et algorithmes. Édition Tic Doc, 2nd Ed, Paris, 2008.

[5] R. Fletcher. Practical methods of optimization. Volume 2. Constrained Optimization. A Wiley-Interserscience Publication, -John wiley and sons edition, New York 1987.

[6] K.Hassaini. Cours d'algèbre 4, 2^{eme} année Licence, Université de Béjaïa, 2014.

[7] B.Aizel, A.Ouazib et M. O. Bibi. Minimisation d'une forme quadratique concave sous contraintes linéaires d'inégalité. Mémoire de Master, Université de Béjaia, 2014.

[8] E.Ferhat, N.Dhamnia et K.Hassaini. Algorithmes de minimisation d'une fonction quadratique convexe sous contraintes linéaires d'égalité. Mémoire de Licence, Université de Béjaïa, 2015.

[9] P.D.Tao et L.T.H.An. Difference of convex fuctions optimization algorithms (DCA) for globally minimizing nonconvex quadratic forms on Euclidean balls and spheres. Operations Research Letters, (1996), Volume 19, pp :207-216 .

[10] O.Hajji. Contribution au développement des méthodes d'optimisation stochastique.Application à la conception des dispositifs électrotechniques. Thèse de Doctorat, Université de Lille, 2013.

[11] M.Moeini. La programmation DC et DCA pour l'optimisation de portefeuille. Thèse de Doctorat, Université Paul Verlaine-Metz, 2008.

[12] M.O.Bibi. Cours de programmation linéaire et quadratique, 1^{er} année Master, Université de Bejaïa, 2016.

[13] M.O.Bibi. Cours d'optimisation globale, 1^{er} année Master, Université de Bejaïa, 2015.

[14] J.T.Lapersté. Introduction à Matlab. Édition Ellipses, Paris, 2015.

Résumé

Dans ce travail, nous avons d'abord donné quelques rappels sur les concepts fondamentaux d'algèbre linéaire et de programmation mathématique. Ensuite, nous avons présenté les algorithmes des trois méthodes (gradient projeté, pénalité, DC) pour la résolution des problèmes de programmation quadratique avec des contraintes quadratiques convexes de type inégalité. La programmation des deux algorithmes (Pénalité extérieure et DCA) sous MATLAB nous a permis de faire des expérimentations numériques que nous avons illustrées par des exemples d'applications .

Mots Clés : *Programmation mathématique, Méthode du Gradient Projeté, Programmation quadratique convexe et non convexe, Optimisation globale, Optimisation DC.*

Abstract

In this work, we have recalled some fundamental concepts of linear algebra and mathematical programming, after that we have presented three methods for solving minimization quadratic problems with convex quadratic constraints. The programming of the two methods (penalty, DCA) under MATLAB has allowed us to make numerical experimentations which we have illustrated by examples of application.

Keywords : *Mathematical programming, Projected gradient method, Convex and non convex quadratic programming, Global optimization, DC optimization.*