Résolution d'un problème quadratique non convexe avec contraintes mixtes par les techniques de l'optimisation D.C.

Mira Al Kharboutly

Normandie Université

THESE

Pour obtenir le diplôme de doctorat

Spécialité : **Mathématiques Appliquées**

Préparée au sein de « Université Le Havre Normandie - LMAH »

Résolution d'un problème quadratique non convexe avec contraintes mixtes par les techniques de l'optimisation D.C.

Présentée et soutenue par
Mira AL KHARBOUTLY

Thèse soutenue publiquement le 4 avril 2018 devant le jury composé de		
M. Boubakeur BENAHMED	Professeur, Ecole Nationale Polytechnique d'Oran (ENPO) – Algérie	Rapporteur
M. Djamel BENTERKI	Professeur – Université Ferhat ABBAS Sétif 1 – Algérie	Rapporteur
M. Abdelkader SBIHI	Professeur – ESCEM Ecole de Management, Tours	Examinateur
M. Abdessamad AMIR	Professeur – Université Abdelhamid Ibn Badis (Mostaganem) - Algérie	Examinateur
M. Hassan ALABBOUD	Maître de Conférences, Université Libanaise – Tripoli - Liban	Co-encadreur de thèse
M. Adnan YASSINE	Professeur - Université Le Havre Normandie	Directeur de thèse

Thèse dirigée par Adnan YASSINE, **Laboratoire de Mathématiques Appliquées du Havre (LMAH), Université Le Havre Normandie – France.**

Etablissement **Ecole Doctorale** **Laboratoire**

UNIVERSITÉ LE HAVRE NORMANDIE

MI∫IΣ

LMAH

Remerciement

Soyons reconnaissants aux personnes qui nous donnent
du bonheur ; elles sont les charmants jardiniers
par qui nos âmes sont fleuries
Marcel Proust

Je remercie Dieu pour m'avoir armé de force, de patience, de volonté et de courage durant mes années de thèse.

Mes plus vifs remerciements s'adressent à mon Professeur, Monsieur Adnan YASSINE, pour la confiance qu'il m'a accordée en acceptant d'encadrer ce travail doctoral, de sa disponibilité et son grand soutien scientifique et humain. Sa capacité d'analyse et son enthousiasme m'ont montré que le monde de la recherche pouvait être un univers passionnant. Sa compétence, sa rigueur scientifique et sa clairvoyance m'ont beaucoup appris. Ils ont été et resteront des moteurs de mon travail de chercheuse. Ses nombreuses relectures et corrections de cette thèse ont été très appréciée. Merci d'avoir été toujours là pour me soutenir dès mon premier jour au Havre et durant les années de cette thèse. Pour tout cela merci.

Mes sincères remerciements vont également à Monsieur Hassan ALABBOUD d'avoir co-encadré ce travail de thèse. Je lui suis reconnaissante de m'avoir fait bénéficier tout au long de ce travail de sa grande compétence, de sa rigueur intellectuelle, de son dynamisme et de son efficacité certaine que je n'oublierai jamais. Soyez assuré de mon attachement et de ma profonde gratitude.

Je suis très heureuse et honorée d'exprimer ma profonde reconnaissance à :

Messieurs les Professeurs Djamel BENTERKI, Professeur à l'Université Ferhat ABBAS Sétif 1 (Algérie) et Boubakeur BENAHMED, Professeur à l'École Nationale Polytechnique d'Oran (Algérie), pour leurs remarques et l'attention toute particulière qu'ils ont accordé à ce travail et qui ont accepté d'en être les rapporteurs extérieurs.

Messieurs les Professeurs Abdessamad AMIR, Professeur à l'Université Abdelhamid Ibn Badis (Algérie) et Abdelkader SBIHI, Professeur à ESCEM École de Management (Tours), pour avoir accepté d'être membres du jury.

Je tiens également à remercier mes collègues du Laboratoire de Mathématiques appliquées du Havre (LMAH) et son directeur Professeur Aziz Alaoui qui m'a accueilli au sein de ce laboratoire durant mes années de thèse.

J'adresse aussi mes remerciements à M. Youssef TLICHE pour sa collaboration scientifique fructueuse.

Je souhaiterais aussi adresser ma gratitude à Dr. Hamdi DKHIL, Dr. Abderrahmen BELFKIH, Dr Moussaab BOUAFIA, Dr. Ibrahima DIARRASSOUBA, M. Mohamad KHORBATLY, M. Abdelrahman SAYED et M. Matthieu BRACHET pour leurs aides et leurs soutiens.

Pour tous mes amis au Havre qui m'ont apporté leur soutien moral pendant ces années d'études, je les remercie sincèrement. Tatiana FAWAZ, Mayssa YASSINE, Karima YASSINE, Joanna EID et l'équipe du Cèdre je vous remercie de tout mon cœur. Tatiana, mes remerciements ne pourront jamais égaler ton grand cœur qui m'a apporté du soutien au moment où j'avais besoin d'aide, merci pour ton amitié. Karima et Mayssa, merci d'avoir été toujours présentes à mes cotés dans les périodes les plus difficiles et de m'avoir accueilli les bras ouverts dans votre foyer où je me suis sentie comme chez moi.

Pour mes amies au Liban, Israa MOUSSA, Manar EL HASSAN et Nour SAFI, merci pour votre support et vos encouragements malgré la distance qui nous sépare.

J'adresse enfin toute mon affection à mes chers parents pour leur soutien éternel, leurs encouragements permanents pendant ces longues années d'éloignement et sans qui je ne serais pas là. À mon idole, ma mère, grâce à tes prières j'ai réussi ce travail, tu resteras toujours ma source d'inspiration, de force et de courage. À mon père pour son aide et ses conseils. À mes sœurs : Mayssa, Mariam et Jana pour leur soutien continu.

Pour mon grand amour, merci beaucoup...

Le Havre, 4 avril, 2018

Table des matières

Introduction

La programmation quadratique est une branche d'optimisation non linéaire où la fonction objectif à minimiser est une fonction quadratique et les contraintes, définissant le domaine des solutions réalisables, sont linéaires et/ou quadratiques. Son importance réside dans ses propriétés théoriques, ses applications dans différents domaines scientifiques et plusieurs disciplines telles que l'économie, la finance, la médecine, les télécommunications et les sciences de l'ingénieur. En effet, plusieurs problèmes réels et académiques peuvent se modéliser sous forme d'un programme quadratique. En raison de ses nombreuses applications, la programmation quadratique est souvent considérée comme une discipline à part entière. Les problèmes des moindres carrés en analyse numérique (régressions linéaire et non linéaire, résolution des systèmes linéaires non carrés, etc.), les problèmes de faisceaux en physique (Transmit Beamforming for Physical-Layer Multicasting, Problems in multisensor beamforming), l'optimisation de portefeuille et la gestion du risque en finance peuvent être représentés par des programmes quadratiques. Différentes variantes de ce problème, convexes et non convexes, avec des variables réelles (optimisation continue), entières ou binaires (optimisation discrète) ont été traitées et étudiées par plusieurs chercheurs.

Généralement les problèmes d'optimisation quadratique en variables binaires sont des problèmes de décisions qui sont classées NP-difficiles et NP-complet dans la théorie de la complexité. Les problèmes de bipartition de graphe, d'affectation quadratique, du sac-à-dos quadratique en sont des exemples très connus et amplement étudiés dans la littérature de l'optimisation combinatoire. Ces problèmes modélise un grand nombre de problèmes pratiques issu de l'économie, l'industrie

et la logistiques comme les choix d'investissements, la gestion de portefeuilles, la conception de réseaux, la gestion de flux, le stockage, l'ordonnancement dans les ateliers et les usines et les problèmes de transport, etc.

L'optimisation d'une fonction quadratique de variables binaires (BQP) sous contraintes linéaires ou sans contraintes a fait l'objet d'une littérature abondante, dont des travaux de synthèse réalisés par Billionnet [8] et Boros et Hammer [9] mais aussi par Hansen, Jaumard et Mathon ([28], [29]) qui ont étudié la programmation non linéaire en variables 0-1 en élargissant au cas des contraintes non linéaires. Plusieurs applications de (BQP) concernent l'ordonnancement de machines (machine scheduling), Le filtrage par motif (Pattern matching), La gestion du traffic (traffic message management), L'assignation de fréquence (frequency assignment), Les données archéologiques (archaeological Data), L'emplacement d'installations (facility location) et L'analyse des données biologiques (analysis of biological data). L'intérêt de ces problèmes et leur complexité déjà évoquées ont donc donné lieu au développement de nombreuses méthodes algorithmiques qui sont généralement des heuristiques capable de résoudre approximativement ces problèmes, souvent performantes, au moins pour certaines classes de problèmes.

Malgré des travaux pionniers datant de plus de vingt ans, les problèmes non linéaires en variables bivalentes demeurent difficiles à résoudre, et surtout quand la dimension du problème devient grande. Ceci représente donc un enjeu très important.

En programmation mathématique, la notion de convexité joue un rôle majeur. En effet, un programme convexe, i.e., la fonction objectif et le domaine réalisable sont convexes, possède des propriétés remarquables qui permettent d'en "faciliter" la résolution. Les nombreuses recherches dans le domaine de la résolution de programmes mathématiques ont montré qu'une distinction majeure doit être faite entre les problèmes convexes et les problèmes non convexes. Ceci a inspiré la réflexion de Rockafellar à ce sujet en 1993 [82] : « In fact the great watershed in optimization isn't between linearity and nonlinearity, but convexity and non

convexity ».

Ainsi, les méthodes de résolution sont généralement très différentes selon la convexité des programmes à résoudre. La convexité d'une fonction f de classe C^2 en un point x_0 de son domaine de définition s'obtient par l'étude des valeurs propres de la matrice Hessienne $H(x_0) = \nabla^2 f(x_0)$. La fonction f sera dite localement convexe si la matrice H est semi-définie positive (i.e. toutes ses valeurs propres sont positives ou nulles). De plus, f sera dite convexe si elle est localement convexe pour tout x de son domaine de définition.

Les fonctions quadratiques possèdent une propriété remarquable concernant leur convexité. En effet, toutes les dérivées secondes sont constantes. Ainsi, la matrice Hessienne d'une fonction quadratique est indépendante du point de l'ensemble de définition auquel elle est évaluée : Ainsi, la convexité du programme complet peut être évaluée à partir de sa définition et ce, indépendamment du domaine de définition. Ceci constitue un avantage, dans la mesure où le choix d'une approche de résolution basée sur les caractéristiques de convexité du problème peut être fait à partir de la simple définition de ce dernier. En revanche, lorsque le problème n'est pas globalement convexe, il ne l'est pas non plus localement, i.e., il est impossible de trouver un voisinage autour d'un point précis du domaine de définition dans lequel la fonction est localement convexe en tout point. Les propriétés associées à la convexité locale et les puissants algorithmes de résolution qui en découlent ne peuvent donc pas être utilisés.

Le but de cette thèse est de proposer une méthode originale et efficace de résolution exacte des programmes quadratiques en variables bivalentes sous contraintes mixtes, linéaires et quadratiques sous la forme :

$$(QP01) \qquad Min\left\{f(x) = \tfrac{1}{2}x^tQx + y^tx : x \in S\right\}$$

où

$$S = \left\{x \in \{0,1\}^n : A'x \leq b'; x^tL_jx + c_j^tx \leq \alpha_j;\ j = 1, ..., p\right\}$$

Q est une matrice symétrique carrée d'ordre n, $y \in \mathbb{R}^n$, A' est une matrice de type (m, n), $b' \in \mathbb{R}^m$, L_j $(j = 1, \ldots, p)$ sont des matrices carrées d'ordre n et semi-définies positives, $c_j \in \mathbb{R}^n$ et $\alpha_j \in \mathbb{R}$ pour $j = 1, \ldots, p$. Ces problèmes sont non convexes (car leur domaine de définition est non convexe et en plus leur fonction objectif f pourra être non convexe si la matrice Q n'est pas semi-définie positive). Pour pallier cette difficulté, nous montrons que dans un problème d'optimisation quadratique en 0-1, la fonction objectif peut être considérée toujours comme une fonction strictement convexe. En effet, sans perte de généralité, on peut supposer que la matrice Q de la fonction objectif est définie positive en utilisant la propriété des variables booléennes $(x_i^2 = x_i)$: soit ρ un réel strictement positif vérifiant $\rho > \|Q\|_1 > \rho(Q)$ où $\rho(Q)$ est le rayon spectral de la matrice Q $(\rho(Q) = \sup\{|\lambda_i| : i = 1, \ldots, n\}$ où les λ_i sont les valeurs propres de $Q)$ et $\|Q\|_1$ est la norme 1 de la matrice Q, définie par $\|Q\|_1 = \sup_{1 \leq i \leq n} (\sum_{j=1}^{n} |Q_{ij}|)$. La fonction f peut s'écrire sous la forme :

$$f(x) = \tfrac{1}{2}x^t Q x + y^t x = \tfrac{1}{2}x^t(Q + \rho I_n)x + (y - e)^t x = \tfrac{1}{2}x^t Q' x + y'^t x$$

où I_n est la matrice identité d'ordre n, $e = (1, \ldots, 1)^t$, $Q' = (Q + \rho I_n)$ et $y' = y - e$. On remarque facilement que la matrice Q' est définie positive et par suite la fonction f est strictement convexe.

La non convexité du domaine de définition vient de l'intégrité des variables $(x_i \in \{0, 1\}$ pour $i = 1, \ldots, n)$. Pour remédier à ce problème, notre approche consiste à résoudre le problème initial $(QP01)$ par l'algorithme DCA de l'optimisation D.C.(**D**ifférence de deux fonctions **C**onvexes) Cette approche, basée sur les outils d'analyse convexe, la dualité DC et l'optimalité locale en optimisation DC, consiste à construire deux suites $\{x_k\}$ et $\{y_k\}$ qui convergent respectivement vers la solution optimale du problème primal et le la solution optimale du problème dual. Cette approche conduit à résoudre un problème quadratique convexe $(QPC01\_D1)$ à chaque itération en minimisant la fonction quadratique f convexe sur le domaine « relaxé » convexe en utilisant la relaxation continue (consistant à remplacer $x \in \{0, 1\}^n$ par $x \in [0, 1]^n$).

De cette manière, on ramène la résolution d'un problème quadratique 0-1 non convexe avec des variables bivalentes à la résolution d'une série de problèmes quadratiques 0-1 convexes avec des variables réelles. Comme on résout le problème quadratique convexe ($QPC01\_D1$) à chaque itération, il faut choisir une méthode robuste, efficace et rapide capable de résoudre ce problème rapidement lorsque la dimension du problème devient très grande. Nous avons appliqué trois approches différentes pour résoudre ($QPC01\_D1$) :

1. La méthode de Newton non-lisse.
2. La programmation semi définie positive (SDP).
3. Les méthodes de points intérieurs (IPM).

Les résultats numériques sur des instances de grandes tailles ont montré que la première approche (méthode de Newton) est la plus efficace et la plus rapide. Les résultats théoriques et numériques montrent que l'application de l'optimisation D.C. en résolvant le problème local convexe ($QPC01\_D1$) à chaque itération avec la méthode de Newton permettra de résoudre efficacement le problème quadratique non convexe ($QP01$) et obtenir des résultats très satisfaisants lorsque la dimension de problème devient assez grande.

Cette thèse est composée de quatre chapitres et est organisée comme suit :
le chapitre 1 constitue des rappels de quelques notions de l'analyse convexe et une brève introduction sur les fondements de la programmation DC et son algorithme DCA. Le chapitre 2 est consacré à la programmation quadratique générale dont les différentes méthodes de résolution sont étudiées et présentées. Le chapitre 3 est réservé à la résolution du problème quadratique binaire non convexe avec contraintes mixtes ($QP01$) par l'algorithme DCA. Le chapitre 4 concerne la résolution du problème quadratique convexe local ($QPC01\_D1$). Nous présentons dans ce chapitre les trois approches de résolution du problème quadratique convexe local ($QPC01\_D1$). Des résultats numériques comparatifs sont également présentés en fin de ce chapitre. La thèse se termine par une conclusion générale sur les résultats obtenus ainsi que les perspectives.

Chapitre 1

Notions d'analyse convexe et optimisation D.C. et DCA

Sommaire

Ce chapitre est une brève introduction à la programmation DC (**D**ifférence de deux fonctions **C**onvexes) et l'algorithme correspondant noté dans la suite DCA (DC Algorithm). Nous détaillerons les bases théoriques et algorithmiques de la programmation DC et DCA. Nous commençons par quelques rappels d'analyse convexe, ensuite nous présenterons la programmation DC, la notion de la dualité et les conditions d'optimalité locale en programmation DC sur lesquelles est basé l'algorithme DCA. Nous terminerons par les résultats de convergence de DCA et la pénalisation exacte en programmation DC.

Pour rapel, un programme DC est formulé comme un problème de minimisation d'une fonction DC. L'ensemble des fonctions convexes joue un rôle important dans la construction des fonctions DC et la convergence de l'algorithme DCA. Une des propriétés essentielles de cet ensemble est sa stabilité par rapport aux opérations usuelles en optimisation.

la programmation DC et son algorithme DCA jouent un rôle central dans la construction des approches locales et globales qui sont basées sur l'analyse et l'optimisation convexe car la plupart des problèmes d'optimisation non convexe sont reformulés sous la forme des programmes DC.

Les fonctions DC possèdent de nombreuses propriétés importantes qui ont été établies à partir des années 50 par Alexandroff ([3], 1949) et Hartman ([32], 1959). Une des principales propriétés est leur stabilité relative aux opérations fréquemment utilisées en optimisation. En 1985, L'algorithme DCA a été introduit par P.D. Tao ([68], 1985), comme une extension des algorithmes de sous-gradients pour la programmation concave, puis largement développé depuis les années 90 par P.D. Tao ([70], [69]), A.Yassine ([101], 1988) et H. A. Le Thi ([54], [55]). On distingue deux grandes approches DC :

1. L'extension de l'approche combinatoire (cette terminologie est due au fait que les nouveaux outils introduits ont été inspirés par les concepts de

l'optimisation combinatoire) en optimisation globale continue.

2. L'extension de l'approche d'analyse convexe en optimisation non convexe.

Les algorithmes de l'approche combinatoire utilisent les techniques de l'optimisation globale (méthode de séparation et d'évaluation, techniques de coupes, méthodes d'approximation fonctionnelle et ensembliste). Ces algorithmes sont plutôt lourds à mettre en œuvre, ils doivent donc être réservés à des problèmes de dimension raisonnable possédant des structures bien adaptées aux méthodes lorsqu'il est important d'isoler l'optimum global. Actuellement on ne peut résoudre, avec ces méthodes, que des problèmes d'optimisation DC de petite taille ou alors des problèmes décomposables dans lesquels la dimension des variables non convexes est acceptable.

Concernant l'optimisation globale, H. Tuy est considéré étant le pionnier de cette approche qui remonte aux années 60 ([92],[93]). Ces deux références présentent particulièrement la théorie, les algorithmes et les applications de l'optimisation globale. Par la suite diverses contributions sont dû à l'école américaine (Pardalos, Rosen, ...), allemande (Horst, ...), française (Le Thi, Pham Dinh, ...) et vietnamienne (Phan Thien, Le Dung, ...).

L'extension de l'approche de l'analyse convexe remonte au travail de à T. Pham Dinh [72]. Cette méthode concerne le calcul des normes matricielles (problème fondamental en analyse numérique) qui est un problème de maximisation d'une fonction convexe sur un ensemble convexe. Le travail de Toland [88] sur la dualité et l'optimalité locale en optimisation DC généralise de manière élégante les résultats établis par T. Pham Dinh en optimisation convexe. La théorie de l'optimisation DC est ensuite développée notamment par Pham Dinh, H. Urruty, Penot, Yassine, Phan Thien et Le Thi. Sur le plan algorithmique dans le cadre de cette deuxième approche, on ne dispose actuellement que des DCA (DC Algorithms) introduits par T. Pham Dinh (1986), qui sont basés sur les conditions d'optimalité et de dualité en optimisation DC.

Malgré le faite que l'algorithme DCA soit un algorithme de sous-gradient, il a été

appliqué avec succès à des programmes non convexes pour lesquelles il a souvent trouvé des solutions optimales globales, notamment en grande dimension. La programmation DC et DCA ont été utilisés, avec beaucoup de succès, par de nombreux chercheurs pour modéliser et résoudre des programmes non convexes issus de différents domaines, en particulier dans le transport-Logistique, télécommunication, bioinformatique, finance, mécanique, traitement d'image, pétrochimie, Contrôle Optimal, Problèmes Inverses, etc.

La popularité de la programmation DC et les algorithmes DCA réside dans leur simplicité, flexibilité, robustesse, rapidité et performance comparées à des méthodes existantes et leur adaptation aux structures des problèmes traités et leur capacité à résoudre des problèmes industriels de grande dimension.

1.1. Éléments d'analyse convexe

L'optimisation offre un cadre de modélisation et d'algorithmique très riche pour tous les domaines de sciences appliquées. On peut distinguer deux branches de l'optimisation déterministe : la programmation convexe et la programmation non convexe.
Un programme convexe ou un problème d'optimisation convexe consiste à minimiser une fonction objectif convexe sur un domaine (ensemble des solutions réalisables) convexe. Lorsque la double convexité, de la fonction objectif et du domaine, n'est pas vérifiée, on est alors en face d'un problème d'optimisation non convexe. La double convexité d'un programme convexe permet d'établir des caractérisations (sous forme de conditions nécessaires et suffisantes) de solutions optimales et ainsi de construire des méthodes itératives convergeant vers des solutions optimales. Dans les problèmes d'optimisation, avec ou sans contraintes, une notion va jouer un rôle très important : celle de *la convexité*. En effet, pour la plupart des algorithmes que nous décrivons, la convergence vers un optimum global ne pourra être démontrée qu'avec des hypothèses fortes de la convexité.
Dans cette section, nous allons rappeler quelques définitions et théorèmes d'analyse convexe qui fondent la base de la programmation DC. Pour plus de détails

en analyse convexe, on pourra se référer aux ouvrages de Rockafellar [72] et de Hiriart-Urruty [41]. Soit X l'espace euclidien $\mathbb{R}^n$ muni du produit scalaire usuel $\langle x, y\rangle = \sum_{i=1}^{n} x_i y_i = x^t y$ et de la norme euclidienne associée $\|x\|_2 = \sqrt{\langle x, x\rangle}$, où x et y sont deux vecteurs dans $\mathbb{R}^n$ et x^t est le transposé du vecteur x. Y désignera l'espace dual de X relatif au produit scalaire qui peut être identifié à X ($X \equiv \mathbb{R}^n$ alors $Y \equiv \mathbb{R}^n$). L'analyse convexe moderne permet aux fonctions de prendre les valeurs $\pm\infty$. On notera $\overline{\mathbb{R}} = \mathbb{R} \cup \{\pm\infty\}$ qui est muni d'une structure algébrique déduite de manière naturelle, avec la convention $(+\infty) - (+\infty) = +\infty$.

Définition 1.1. ***(ensemble convexe)*** *Un ensemble $C \subset X$ est dit convexe lorsque, pour tout x et y de C, le segment $[x, y]$ est tout entier contenu dans C, c.à.d,*

$$\forall x \in C, \forall y \in C, \forall \lambda \in [0, 1] \text{ alors } \lambda x + (1 - \lambda) y \in C. \qquad \blacksquare$$

Définition 1.2. ***(projection sur un ensemble convexe)*** *Soit $C \subset X$ un ensemble convexe fermé non vide et soit $u \in X$. Il existe alors un point unique $P_C(u) \in C$ appelé projection de u sur C, défini par :*

$$d_C(u) = \|u - P_C(u)\| = \min\{\|u - v\| : v \in C\}. \qquad \blacksquare$$

Définition 1.3. ***(combinaison convexe)*** *Soient $x^1, ..., x^k$ des points de X. Un point $x \in X$ est dit combinaison convexe des points x^i $(1 \leq i \leq k)$ s'il peut s'écrire sous la forme*

$$x = \sum_{i=1}^{k} \lambda_i x^i,$$

les λ_i étant des réels vérifiant

$$\lambda_i \geq 0, (1 \leq i \leq k), \sum_{i=1}^{k} \lambda_i = 1. \qquad \blacksquare$$

On vérifie aisément que l'ensemble des combinaisons convexes des points x^i, $1 \leq i \leq k$ est convexe et fermé.

Définition 1.4. ***(enveloppe convexe)*** *Soit $S \subset X$. L'enveloppe convexe de S, notée $co(S)$, est l'intersection de tous les sous-ensembles convexes de X qui contiennent S. Elle est en effet le plus petit ensemble convexe de X qui contient S.*

Sachant que X est un espace vectoriel de dimension finie, on peut en déduire facilement que $co(S)$ est l'ensemble des points $x \in X$ qui peuvent s'écrire sous la forme de combinaison convexe finie d'éléments de S, c'est-à-dire

$$co(S) = \left\{ x = \sum_{i=1}^{m} \lambda_i x^i : x^i \in S, \lambda_i \geq 0, \forall i = 1, ..., m; \sum_{i=1}^{m} \lambda_i = 1 \right\}. \qquad \blacksquare$$

Définition 1.5. ***(variété affine)*** *Soit C un ensemble convexe. Une variété affine engendrée par C, notée $aff(C)$, est définie comme :*

$$aff(C) = \left\{ \sum_i \lambda_i x^i : x^i \in C, \sum_i \lambda_i = 1, \lambda_i \in \mathbb{R} \right\},$$

où seules les sommes finies sont prises en compte. $\blacksquare$

Définition 1.6. ***(intérieur relatif)*** *On appelle intérieur relatif d'un ensemble convexe C, noté $ir(C)$, l'intérieur de C relativement à sa variété affine, c.à.d,*

$$ir(C) = \{x \in C : \exists\, r > 0, B(x, r) \cap aff(C) \subset C\},$$

où $B(x, r) = \{y \in \mathbb{R}^n : \|y - x\| \leq r\}$ est la boule euclidienne de centre x et de rayon r. $\blacksquare$

En dimension finie, l'intérieur relatif d'un convexe C non vide n'est jamais vide et a la même dimension que C . L'intérieur relatif $ir(C)$ est vide si et seulement si C l'est aussi.

Définition 1.7. ***(domaine d'une fonction)*** *Soit $f : X \to \mathbb{R} \cup \{+\infty\}$ une fonction convexe. On appelle domaine effectif de f, noté $dom(f)$, l'ensemble*

$$dom(f) = \{x \in X, f(x) < +\infty\}. \qquad \blacksquare$$

C'est un ensemble convexe de X. Si f est concave alors son domaine effectif est défini comme celui de $-f : domf = \{x \in X : f(x) > -\infty\}$.

Définition 1.8. ***(fonction propre)*** *La fonction f est dite propre si elle ne prend jamais la valeur $-\infty$ et si elle n'est pas identiquement égale à $+\infty$.* ■

Définition 1.9. ***(fonction coercive)*** *Une fonction $f : X \to]-\infty, +\infty[$ est dite coercive si elle vérifie :*

$$\lim_{||x|| \to \infty} f(x) = +\infty. \qquad \blacksquare$$

Définition 1.10. ***(fonction convexe)*** *Soit C un ensemble convexe. La fonction $f : C \to \mathbb{R}$ est dite convexe sur C lorsque, pour tout x et y de C et tout $\lambda \in [0, 1]$,*

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda) f(y).$$

Si l'inégalité est stricte pour tout $\lambda \in]0, 1[$ et pour tout x et y de C avec $x \neq y$, alors f est dite strictement convexe. ■

Définition 1.11. ***(fonction concave)*** *Soit C un ensemble convexe. La fonction $f : C \to \mathbb{R}$ est dite concave sur C lorsque, pour tout x et y de C et tout $\lambda \in [0, 1]$,*

$$f(\lambda x + (1 - \lambda)y) \geq \lambda f(x) + (1 - \lambda) f(y). \qquad \blacksquare$$

Remarque 1.1. • *f est concave $\iff$ $-f$ est convexe.*

- *Une fonction linéaire est à la fois convexe et concave.*
- *Soit un ensemble $S \subset X$. On appelle fonction indicatrice de S, notée $\chi_S(x)$, définie par $\chi_S(x) = 0$ si $x \in S$; et $+\infty$ sinon. La fonction indicatrice d'un ensemble convexe est une fonction convexe.*
- *Soit $C \subset X$ un ensemble convexe. La fonction $f : C \to \mathbb{R}$ est convexe si et seulement si la fonction $f + \chi_C$ est convexe sur X à valeurs dans $\mathbb{R} \cup \{+\infty\}$.*

Définition 1.12. ***(fonction fortement convexe)*** *La fonction $f : C \to \mathbb{R}$ est dite fortement convexe de module ρ sur l'ensemble convexe C (autrement dit ρ-convexe) lorsqu'il existe $\rho > 0$ tel que pour tout x et y de C et tout $\lambda \in [0, 1]$, on ait :*

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y) - \lambda(1-\lambda)\frac{\rho}{2}\|x-y\|^2. \qquad \blacksquare$$

Cela revient à dire que la fonction $f - \frac{\rho}{2}\|.\|^2$ est convexe sur C.
Le module de forte convexité de f sur C, noté $\rho(f, C)$, est défini par :

$$\rho(f, C) = sup\left\{\rho > 0 : f - \frac{\rho}{2}\|.\|^2 \text{ est convexe sur } C\right\}.$$

On dit que f est fortement convexe sur C si $\rho(f, C) > 0$.

Remarque 1.2. *Toute fonction fortement convexe est strictement convexe et toute fonction strictement convexe est convexe.*

Définition 1.13. ***(épigraphe)*** *L'épigraphe de la fonction f noté $epi(f)$ est l'ensemble*

$$epi(f) = \{(x, \alpha) \in X \times \mathbb{R} : f(x) \leq \alpha\}. \qquad \blacksquare$$

La fonction f est convexe si et seulement si $epi(f)$ est un sous-ensemble convexe de $X \times \mathbb{R}$.

Définition 1.14. ***(fonction s.c.i.)*** *La fonction f est dite semi-continue inférieurement (s.c.i.) au point $x \in X$ si*

$$\liminf_{y \to x} f(y) \geq f(x). \qquad \blacksquare$$

On note $\Gamma_0(X)$ l'ensemble des fonctions convexes, s.c.i. et propres sur X.

Définition 1.15. ***(fonction conjuguée)*** *La fonction conjuguée de $f \in \Gamma_0(X)$, notée $f^* : Y \to \mathbb{R} \cup \{+\infty\}$, est définie par*

$$f^*(y) = sup\{\langle x, y\rangle - f(x) : x \in X\}. \qquad \blacksquare$$

Définition 1.16. ***(sous-gradient)*** *Soit une fonction f convexe propre sur X. Un vecteur y dans Y est appelé sous-gradient de f au point $x^0 \in dom(f)$ si pour tout $x \in X$ on a :*

$$f(x) \geq f(x^0) + \langle y, x - x^0 \rangle. \qquad \blacksquare$$

Définition 1.17. ***(sous-différentiel)*** *L'ensemble de tous les sous-gradients de f au point $x^0 \in dom(f)$ est le sous-différentiel de f au point x^0, noté $\partial f(x^0)$. Le domaine du sous-différentiel de f est $dom(\partial f) = \{x : \partial f(x) \neq \emptyset\}$.* $\blacksquare$

Proposition 1.1. • *$\partial f(x)$ est une partie convexe fermée de Y.*

• *$ir(dom(f)) \subset dom(\partial f) \subset dom(f)$.*

Définition 1.18. ***(ε-sous-gradient)*** *Soit ε un réel positif. Un vecteur $y \in Y$ est appelé ε-sous-gradient de f au point x^0 si*

$$f(x) \geq f(x^0) + \langle y, x - x^0 \rangle - \varepsilon, \forall x \in X$$

Les ε-sous-différentiel de f au point x^0, noté $\partial_\varepsilon f(x^0)$, est l'ensemble de tout les ε-sous-gradient de f au point x^0. $\blacksquare$

Proposition 1.2. • *$f \in \Gamma_0(X) \Longleftrightarrow f^* \in \Gamma_0(Y)$, et $(f^*)^* = f$.*

• *Si $f \in \Gamma_0(X)$ est différentiable en x si et seulement si $\partial f(x)$ se réduit au singleton $\{\nabla f(x)\}$.*

• *Si $f \in \Gamma_0(X)$ alors $y \in \partial f(x) \Longleftrightarrow x \in \partial f^*(y)$.*

• *$y \in \partial f(x) \Longleftrightarrow f(x) + f^*(y) = \langle x, y \rangle$.*

• *$x^0 \in argmin\{f(x) : x \in X\} \Longleftrightarrow 0 \in \partial f(x^0)$.*

Définition 1.19. ***(polyèdre convexe)*** *Un ensemble convexe C est dit polyèdre convexe s'il est l'intersection de nombre fini de demi-espaces de $\mathbb{R}^n$:*

$$C = \bigcap_{i=1}^{m} \{x \in \mathbb{R}^n : \langle a_i, x \rangle - b_i \leq 0, a_i \in \mathbb{R}^n, b_i \in \mathbb{R}\}. \qquad \blacksquare$$

Définition 1.20. ***(fonction polyédrale)*** *Une fonction $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ est dite polyédrale si son épigraphe est un polyèdre convexe de $\mathbb{R}^{n+1}$.* $\blacksquare$

Notons que toute fonction polyédrale est convexe, propre et s.c.i.

Proposition 1.3. *Soit $f \in \Gamma_0(X)$ une fonction convexe. Alors f est polyédrale si et seulement si $dom(f)$ est un polyèdre convexe et*

$$f(x) = sup\{\langle a_i, x\rangle - b_i : i = 1, ..., l\} + \chi_{dom(f)}(x)$$

Où $\chi_{dom(f)}(x)$ est la fonction indicatrice de $dom(f)$.

Proposition 1.4. *Les fonctions convexes polyédrales possèdent les propriétés intéressantes suivantes :*

- *Si f_1 et f_2 sont convexes polyédrales, alors $f_1 + f_2$, $\max\{f_1 + f_2\}$ sont convexes polyédrales.*
- *Si f est convexe polyédrale alors f^* l'est aussi et $dom(\partial f) = dom(f)$. De plus, si f est finie partout alors*

$$dom(f^*) = co\{a_i : i = 1, ..., l\},$$

$$f^*(y) = \min\left\{\sum_{i=1}^{l} \lambda_i b^i : y = \sum_{i=1}^{l} \lambda_i a^i, \sum_{i=1}^{l} \lambda_i = 1, \lambda_i \geq 0, i = 1, ..., l\right\}.$$

- *Si f est polyédrale alors $\partial f(x)$ est une partie convexe polyédrale non vide $\forall x \in dom(f)$.*
- *Soient $f_1, ..., f_l$ des fonctions convexes polyédrales sur X telles que les ensembles convexes $dom(f_i)$, $i = 1, ..., l$ aient un point commun. Alors*

$$\partial(f_1 + ... + f_l)(x) = \partial f_1(x) + ... + \partial f_l(x), \forall x.$$

1.2. Fonctions DC

Dans cette section, nous allons définir une fonction DC et des propriétés essentielles de l'ensemble des fonctions DC notamment sa stabilité par rapport aux opérations usuelles en optimisation. Pour plus de détails sur les fonctions DC, on pourra se référer à ([3], [32], [55]).

Définition 1.21. ***(fonction DC)*** *Soit C un ensemble convexe fermé non vide de X. Une fonction $f : C \to \mathbb{R} \cup \{+\infty\}$ est dite DC sur C si elle peut s'écrire sous la forme*

$$f(x) = g(x) - h(x),\ \forall x \in C,$$

où g et h sont deux fonctions convexes sur C. On dit alors que $g - h$ est une décomposition DC de f. ■

Remarque 1.3. *Si $f = g - h$ est un fonction DC sur C alors pour toute fonction convexe finie p sur C, $f = (g + p) - (h + p)$ est aussi une décomposition DC de f. Ainsi une fonction DC possède une infinité de décompositions DC.*

Définition 1.22. ***(Ensemble des fonctions DC)*** *Soit $Conv(C)$ l'ensemble des fonctions convexes sur C. L'ensemble des fonctions DC sur C noté $DC(C)$, est l'espace vectoriel engendré par $Conv(C)$ défini par*

$$DC(C) = Conv(C) - Conv(C).$$ ■

L'espace vectoriel $DC(C)$ est une classe de fonctions assez large. Il contient l'ensemble des fonctions convexes sur C, les fonctions concaves sur C, ainsi que beaucoup de fonctions ni convexes ni concaves sur C. Il contient la quasi-totalité des fonctions rencontrées dans les problèmes concrets en optimisation non convexe, et joue ainsi un rôle clé en optimisation non convexe.

Les fonctions DC possèdent beaucoup de propriétés importantes qui ont été établies à partir des années 50 par Alexandroff ([3], 1949) et Hartman ([32], 1959), etc. La classe de fonctions DC est stable par rapport à plusieurs opérations usuelles utilisées en optimisation.

Proposition 1.5. [63] *Si $f = g - h$, $f_i = g_i - h_i$, $i = 1, ..., m$, sont des fonctions dans $DC(C)$ alors*

- *Une combinaison linéaire des fonctions DC est une fonction DC, c.à.d.,*

$$\sum_{i=1}^{m} \lambda_i f_i \in DC(C), \forall \lambda_i \in \mathbb{R}.$$

- $\max_{i=1,\dots,m}(f_i) \in DC(C)$ *et* $\min_{i=1,\dots,m}(f_i) \in DC(C)$ *car*

$$\max_{i=1,\dots,m} f_i = \max_{i=1,\dots,m}[g_i + \sum_{j=1,j\neq i}^{m} h_j] - \sum_{j=1}^{m} h_j,$$

$$\min_{i=1,\dots,m} f_i = \sum_{j=1}^{m} g_j - \max_{i=1,\dots,m}[h_i + \sum_{j=1,j\neq i}^{m} g_j].$$

- $|f| \in DC(C)$, *car*

$$|f| = 2\max(g,h) - (g+h).$$

- $f^+, f^- \in DC(C)$ *où* $f^+(x) = \max(0, f(x))$ *et* $f^-(x) = \min(0, f(x))$ *car*

$$f^+ = \max(g,h) - h$$

$$f^- = g - \max(g,h)$$

- $\prod_{i=1}^{m} f_i \in DC(C)$

1.3. Programmation DC

Définition 1.23. ***(programme DC)*** *On appelle programme DC tout problème d'optimisation de la forme*

$$(P_{dc}) \qquad \alpha = inf\,\{f(x) = g(x) - h(x) : x \in \mathbb{R}^n\},$$

où g *et* $h \in \Gamma_0(\mathbb{R}^n)$ *sont appelés les composantes DC de* f *et* $g - h$ *est la décomposition DC de* f. ■

(P_{dc}) est un problème d'optimisation sans contraintes. Cependant un problème d'optimisation DC avec des contraintes convexes fermées sur C de la forme

$$inf\,\{f(x) = g(x) - h(x) : x \in C\},$$

peut être ramené sous la forme (P_{dc}) en ajoutant la fonction indicatrice de C à la première composante de f, i.e.,

$$inf\,\{F(x) = \varphi(x) - h(x) : x \in \mathbb{R}^n\},$$
$$\varphi := g(x) + \chi_C(x).$$

Des grands efforts ont été consacrés à la résolution numérique de la classe des problèmes d'optimisation non convexes suivantes :

1. $\sup\{f(x) : x \in C\}$, f et C sont convexes.
2. $\inf\{f_0(x) = g_0(x) - h_0(x) : x \in C\}$, g_0, h_0 et C sont convexes.
3. $\inf\{f_0(x) = g_0(x) - h_0(x) : x \in C, g_i(x) - h_i(x) \leq 0, i = 1, ..., m\}$, g_i, h_i, g_i, h_i, $i = 1, ..., m$ et C sont convexes.

La raison est bien simple : la majorité des problèmes d'optimisation de la vie réelle sont de nature non convexes. De plus dans les modèles mathématiques industriels les approches convexes montrent leurs limites et sont de plus en plus remplacées par des approches non convexes qui sont plus réalistes.

Le problème 1. est un cas spécial du problème 2. Il est équivalent à (P_{dc}) dans la mesure où, d'une part, on peut transformer le problème 1. en (P_{dc}) avec $g = \chi_C$ (la fonction indicatrice de C définie par χ_C si $x \in C$; $+\infty$ sinon) et $h = -f$.

Selon la même technique, le problème 2. (un problème DC sous contrainte convexe) est aussi équivalent à (P_{dc}) avec $g = g_0 + \chi_C$ et $h = h_0$.

Dans le problème 3., les contraintes $g_i(x) - h_i(x) \leq 0, i = 1, ..., m$ sont des contraintes non convexes, appelées contraintes DC, qui sont équivalentes à une seule contrainte DC, $\widehat{g} - \widehat{h} \leq 0$ du fait que $\{g_i(x) - h_i(x) \leq 0, i = 1, ..., m\} \equiv \max_{i=1,...,m}(g_i(x) - h_i(x)) \leq 0$ (voir la proposition 5). Le problème suivant est donc un problème équivalent au problème 3.

$$inf\,\Big\{f_0(x) = g_0(x) - h_0(x) : x \in C, \widehat{g}(x) - \widehat{h}(x) \leq 0\Big\}.$$

Grâce au théorème de pénalité exacte relative à la contrainte DC $\widehat{g}(x) - \widehat{h}(x)$ on peut ramener ce dernier problème à la forme (P_{dc}).

Ainsi, la résolution de (P_{dc}) entraîne celle des autres. Le problème (P_{dc}) est par conséquent un problème essentiel en programmation DC. La structure spéciale de (P_{dc}) a permis d'importants développements, tant sur le plan théorique que sur le plan algorithmique.

Sur le plan théorique, on dispose d'une élégante théorie de la dualité et de conditions d'optimalité (voir [40], [55], [88]). Du point de vue algorithmique, un algorithme remarquable "DCA", a été introduit par Pham Dinh Tao (voir [68], [69], [70]) à l'état préliminaire en 1985 et développé ensuite par Pham Dinh Tao, Le Thi Hoai An ([71], 1994). Dans la suite de ce chapitre, nous allons présenter la dualité, les conditions d'optimalité et DCA.

1.4. La dualité en optimisation DC

En analyse convexe, le concept de dualité est une notion fondamentale en mathématique. La notion de dualité dans le cas non convexe a été proposée récemment dans les travaux ([5], [66]). Plus récemment, la dualité a été introduite et développée en analyse non convexe : tout d'abord, dans le cas des programmes quasi-convexe et anti-convexe [88], puis dans le cas des programmes DC. La dualité DC introduite par Toland ([88],[89]) est une généralisation du travail de Pham Dinh Tao sur la maximisation convexe [68]. Dans cette section, nous présenterons les principaux résultats en optimisation DC concernant les conditions d'optimalité (locale et globale) et la dualité DC. Pour plus de détails, le lecteur est renvoyé à l'article par Le Thi [55].

Soient X l'espace euclidien $\mathbb{R}^n$ muni du produit scalaire $\langle ., . \rangle$ et de la norme euclidienne de $\|.\|$. Désignons par Y son espace dual ($Y \equiv \mathbb{R}^n$). Considérons le programme DC suivant :

$$(P_{dc}) \qquad \alpha = inf\,\{f(x) := g(x) - h(x) : x \in X\},$$

où $g, h \in \Gamma_0(X)$. On adopte la convention $(+\infty) - (+\infty) = (+\infty)$ comme en optimisation convexe. Puisque $h \in \Gamma_0(X)$, on a donc $h^{**} = h$ et

$$h(x) = (h^*)^*(x) = sup\{\langle x, y\rangle - h^*(y) : y \in Y\}.$$

Avec cette relation, on peut écrire la dualité DC de la manière suivante :

$$\begin{aligned}\alpha &= inf\{g(x) - h(x) : x \in X\}\\ &= inf\{g(x) - sup\{\langle x, y\rangle - h^*(y) : y \in Y\} : x \in X\}\\ &= inf\{inf\{g(x) - [\langle x, y\rangle - h^*(y)] : x \in X\} : y \in Y\}\\ &= inf\{\beta(y) : y \in Y\}\end{aligned}$$

Où

$$\begin{aligned}\beta(y) &= inf\{g(x) - [\langle x, y\rangle - h^*(y)] : x \in X\}\\ &= \begin{cases} h^*(y) - g^*(y) & \text{si } x \in dom(h^*);\\ +\infty & \text{sinon.}\end{cases}\end{aligned}$$

Grâce à la convention $(+\infty) - (+\infty) = (+\infty)$, on obtient finalement le problème dual de (P_{dc}), noté (D_{dc}) :

$$(D_{dc}) \qquad \alpha = inf\{f(x) := h^*(x) - g^*(x) : y \in Y\}.$$

Où g^* est la fonction conjuguée de g définie par :

$$g^*(y) = sup\{\langle x, y\rangle - g(x) : x \in X\}, g^* \in \Gamma_0(Y), g^{**} = g$$

Le problème (D_{dc}) est aussi un programme DC car g^* et h^* sont deux fonctions convexes dans $\Gamma_0(Y)$. On observe la parfaite symétrie entre les problèmes primal et dual.

Proposition 1.6. ***(i)*** *x^* est une solution optimale globale du problème de (P_{dc}) si et seulement si*

$$\alpha = (g - h)(x^*) \leq (h^* - g^*)(y), \forall y \in Y.$$

(ii) *y^* est une solution optimale globale du problème (D_{dc}) si et seulement si*

$$\alpha = (h^* - g^*)(y^*) \leq (g - h)(x), \forall x \in X.$$

Théorème 1.1. *Soient g, $h \in \Gamma_0(X)$. Alors*

1. $\inf\{g(x) - h(x) : x \in dom(g)\} = \inf\{h^*(x) - g^*(x) : y \in dom(h^*)\}$.
2. *Si y^0 est un minimum de $h^* - g^*$ sur Y alors chaque point $x^0 \in \partial g^*(y^0)$ est un minimum de $g - h$ sur X.*
3. *Si x^0 est un minimum de $g - h$ sur X alors chaque point $y^0 \in \partial h(x^0)$ est un minimum de $h^* - g^*$ sur Y.*

Preuve :

(1)

$$\begin{aligned}
\alpha &= inf\{g(x) - h(x) : x \in X\} \\
&= inf\{g(x) - sup\{\langle x, y\rangle - h^*(y) : y \in Y\} : x \in X\} \\
&= inf\{g(x) + inf\{h^*(y) - \langle x, y\rangle : y \in Y\} : x \in X\} \\
&= inf_x inf_y\{h^*(y) - \langle x, y\rangle + g(x)\} \\
&= inf\{h^*(y) - g^*(y) : y \in Y\}.
\end{aligned}$$

(2), (3) *cf. Toland [88].* ■

Le théorème précédent donne une relation entre les solutions du problème primal (P_{dc}) et de son dual (D_{dc}). Il montre que la résolution de l'un implique la résolution de l'autre. Ainsi pour résoudre (P_{dc}), on pourra d'abord résoudre (D_{dc}) et vice versa, suivant que l'un ou autre est plus « simple » à résoudre. Il existe même des cas particuliers où le problème dual est convexe tandis que le problème primal est non convexe. L'exemple suivant illustre un tel cas.

Exemple 1.1. *Soient $n > 1$, $g(x) = e^x$ et $h(x) = ne^{\frac{x}{n}}$, pour tout $x \in \mathbb{R}$. Le problème primal*

$$(P) \qquad \min\left\{g(x) - h(x) = e^x - ne^{\frac{x}{n}} : x \in \mathbb{R}\right\}$$

est non convexe tandis que son dual

$$(D) \qquad \min\left\{h^*(y) - g^*(y) = (n-1)g^*(y) : y \in \mathbb{R}\right\}$$

est convexe. En effet, on a $h(x) = ng(\frac{x}{n})$ *et*

$$h^*(y) = \sup\left\{\langle x, y\rangle - h(x) : x \in \mathbb{R}\right\} = \sup\left\{\langle x, y\rangle - ng(\tfrac{x}{n}) : x \in \mathbb{R}\right\}.$$

Donc

$$h^*(y) = n \sup\left\{\langle \tfrac{x}{n}, y\rangle - g(\tfrac{x}{n}) : x \in \mathbb{R}\right\} = ng^*(y).$$

1.5. Condition d'optimalité en programmation DC

Soient $\mathcal{P}$ et $\mathcal{D}$ respectivement les ensembles de solutions des problèmes (P_{dc}) et (D_{dc}).
En optimisation convexe, x^* est un minimum de la fonction convexe f si et seulement si $0 \in \partial f(x^*)$. En programmation DC, la condition d'optimalité est formulée à l'aide des $\varepsilon-$sous-différentiels de g et h.

Théorème 1.2. [101] ***(Condition d'optimalité globale)*** *Soient* $g, h \in \Gamma_0(X)$ *et* $f = g - h$. *Alors* x^* *est un minimum globale de* $g - h$ *sur* X *si et seulement si*

$$\partial_\varepsilon h(x^*) \subset \partial_\varepsilon g(x^*), \forall \varepsilon > 0$$

Remarque 1.4.

1. *Si* $f \in \Gamma_0(X)$, *on peut écrire* $g = f$ *et* $h = 0$. *Dans ce cas, La condition d'optimalité précédente est identique à celle de la programmation convexe :* $0 \in \partial f(x^*)$, *du fait que* $\partial_\varepsilon h(x^*) = \partial h(x^*) = \{0\}$, $\forall \varepsilon > 0$.

2. *D'une manière plus générale, considérons les décompositions DC de* $f \in \Gamma_0(X)$ *de la forme* $f = g - h$ *avec* $g = f + h$ *et* $h \in \Gamma_0(X)$ *finie partout sur X. Le problème DC correspondant est un "faux" problème DC car c'est*

un problème d'optimisation convexe. Dans ce cas, la condition d'optimalité $0 \in \partial f(x^*) \Leftrightarrow \partial h(x^*) \subset \partial g(x^*)$ *(Le Thi et Pham Dinh 2005). Ainsi, l'optimalité locale est suffisante pour l'optimalité globale. Par conséquent, si en plus* h *est différentiable, un point critique est également une solution globale.*

Définition 1.24. ***(point critique)*** *On appelle point critique ou point KKT généralisé, tout point* $x^* \in X$ *vérifiant*

$$\partial h(x^*) \cap \partial g(x^*) \neq \emptyset. \qquad \blacksquare$$

Définition 1.25. ***(minimum local)*** *Soient* g *et* h *deux fonctions de* $\Gamma_0(X)$*. Un point* $x^* \in dom(g) \cap dom(h)$ *est un minimum local de* $g - h$ *sur* X *si et seulement si*

$$g(x) - h(x) \geq g(x^*) - h(x^*), \forall x \in V_{x^*},$$

où V_{x^*} *est un voisinage de* x^**.* $\blacksquare$

Théorème 1.3. ***[101](condition nécessaire d'optimalité locale)*** *Si* x^* *est un minimum local de* $g - h$ *alors*

$$\partial h(x^*) \subset \partial g(x^*).$$

Preuve :

Si x^* *est un minimum local de* $g - h$*, alors il existe un voisinage* V_x *de* x *tel que*

$$g(x) - g(x^*) \geq h(x) - h(x^*), \qquad \forall x \in V_{x^*}.$$

Par suite si $y^* \in \partial h(x^*)$ *alors*

$$h(x) - h(x^*) \geq \langle x - x^*, y^* \rangle, \qquad \forall x \in V_{x^*}.$$

Ce qui est équivalent, en vertu de la convexité de g*, à* $y^* \in \partial g(x^*)$*.* $\blacksquare$

Remarque 1.5. *Si* h *est convexe polyédrale ou* f *est localement convexe au point* x^**, la condition nécessaire d'optimalité locale est également suffisante.*

Théorème 1.4. ***[101](condition suffisante d'optimalité locale)*** *Si x^* admet un voisinage V tel que*

$$\partial h(x) \cap \partial g(x) \neq \emptyset, \forall x \in V \cap dom(g),$$

Alors x^* est un minimum local de $g - h$.

Théorème 1.5. [63]

1. *$\partial h(x) \subset \partial g(x)$, $\forall x \in \mathcal{P}$ et $\partial g^*(y) \subset \partial h^*(y)$, $\forall y \in \mathcal{D}$.*

2. *Transport de minima globaux*

$$\bigcup_{x \in \mathcal{P}} \partial h(x) \subseteq \mathcal{D} \subset dom(h^*).$$

La première inclusion devient égalité si g^* est sous-différentiable dans $\mathcal{D}$ (en particulier si $\mathcal{D} \subset ir(dom(g^*))$ ou si g^* est sous-différentiable dans $dom(h^*)$) et dans ce dernier cas $\mathcal{D} \subset (dom\partial g^* \cap dom\partial h^*)$.

$$\bigcup_{y \in \mathcal{D}} \partial g^*(y) \subseteq \mathcal{P} \subset dom(g).$$

La première inclusion devient égalité si h est sous-différentiable dans $\mathcal{P}$ (en particulier si $\mathcal{P} \subset ir(dom(h))$ ou si h est sous-différentiable dans $dom(g)$) et dans ce dernier cas $\mathcal{P} \subset (dom\partial g \cap dom\partial h)$.

3. ***Transport de minima locaux :*** *Soit $x^* dom(\partial h)$ un minimum local de $g - h$. Soient $y^* \in \partial h(x^*)$ et V_{x^*} un voisinage de x^* tel que $g(x) - h(x) \geq g^*(x) - h^*(x)$, $\forall x \in V_{x^*} \cap dom(g)$. Si*

$$x^* \in int(dom(g^*)) \text{ et } \partial g^*(y^*) \subset V_{x^*},$$

alors y^* est un minimum local de $h^* - g^*$.

1.6. Algorithme d'optimisation DC (DCA)

DCA (DC Algorithm) est une méthode itérative d'optimisation locale basée sur l'optimalité locale et la dualité en programmation DC. Cet algorithme a été introduit par Pham Dinh Tao ([70], [72]) en 1985 et puis intensivement développé par Pham Dinh Tao, Le Thi Hoai An et al. ([54], [55]) depuis 1994. Cette approche est complètement différente des méthodes classiques de sous-gradient en optimisation convexe. Elle permet de construire deux suites $\{x^k\}$ et $\{y^k\}$, candidates à être solutions optimales des programmes DC primal et dual respectivement, telles que leurs limites (x^* et y^*) soient des points KKT (Karush, Kuhn et Tucker) généralisés de ces programmes, et telles que les suites $\left\{(g-h)(x^k)\right\}$ et $\left\{(h^*-g^*)(y^k)\right\}$ soient décroissantes et tendent vers la même limite $\beta = (g-h)(x^*) = (h^*-g^*)(y^*)$. On a les propriétés suivantes :

Proposition 1.7. *[101]* *1. Les suites $\left\{g(x^k)-h(x^k)\right\}$ et $\left\{h^*(y^k)-g^*(y^k)\right\}$ décroissent et tendent vers la même limite β qui est supérieure ou égale à la valeur optimale globale α.*

2. Si $(g-h)(x^{k+1}) = (g-h)(x^k)$ l'algorithme s'arrête à l'itération $k+1$, et le point x^k (resp. y^k) est un point critique de $g-h$ (resp. h^*-g^*).

3. Si la valeur optimale du problème (P_{dc}) est finie et si les suites $\{x^k\}$ et $\{y^k\}$ sont bornées, alors toute valeur d'adhérence x^* de la suite $\{x^k\}$ (resp. y^* de la suite $\{y^k\}$) est un point critique de $g-h$ (resp. de h^*-g^*).

La description de L'algorithme DC est la suivante :

Soit $x^0 \in X$ choisi à l'avance, les suites $\{x^k\}$ et $\{y^k\}$ sont définies par :

$$y^k \in \partial h(x^k), \qquad x^{k+1} \in \partial g^*(y^k).$$

Pour construire les deux suites $\{x^k\}$ et $\{y^k\}$, on définit deux programmes convexes (D_k) et (P_k), pour $k \geq 1$, comme suit :

$$(D_k) \qquad x^k \in \partial g^*(y^{k-1}) \rightarrow y^k \in$$
$$arg\min\left\{h^*(y) - [g^*(y^{k-1}) + \langle x^k, y - y^{k-1}\rangle] : y \in Y\right\} = \partial h(x^k).$$

$$(P_k) \qquad y^k \in \partial h(x^k) \rightarrow x^{k+1} \in$$
$$arg\min\left\{g(x) - [h(x^k) + \langle y^k, x - x^k\rangle] : x \in X\right\} = \partial g^*(y^k).$$

Alors le point x^{k+1} (resp. y^k) est une solution optimale du programme (P_k) (resp. (D_k)). On peut facilement comprendre que (P_k) (resp. (D_k)) est un problème d'optimisation convexe obtenu en remplaçant h (resp. g^*) de (P_{dc}) (resp. (D_{dc})) par sa minorante affine $h_k(x) = h(x^k) + \langle y^k, x - x^k\rangle$ au voisinage de x^k avec $y^k \in \partial h(x^k)$ (resp. $g_k^*(y) = g^*(y^{k-1}) + \langle x^k, y - y^{k-1}\rangle$ au voisinage de y^{k-1} avec $x^k \in \partial g^*(y^{k-1})$). On a ensuite le schéma simple suivant pour décrire L'algorithme DC :

$$\begin{array}{ccc} x^k & \rightarrow & y^k \in \partial h(x^k) \\ & \swarrow & \\ x^{k+1} \in \partial g^*(y^k) & \rightarrow & y^{k+1} \in \partial h(x^{k+1}) \end{array}$$

Grâce à la proposition (1.7), l'algorithme DC s'arrête si au moins l'une des suites $\left\{(g-h)(x^k)\right\}$, $\left\{(h^*-g^*)(y^k)\right\}$, $\{x^k\}, \{y^k\}$ converge. En pratique, nous utilisons souvent les conditions d'arrêt suivantes :

- $|(g-h)(x^{k+1}) - (g-h)(x^k)| \leq \varepsilon$.
- $\|x^{k+1} - x^k\| \leq \varepsilon$.

Pour obtenir une solution $\varepsilon-$optimale. Par conséquent, on peut décrire l'algorithme DC :

Algorithme DCA

Étape 0 : x^0 donné, $k = 0$ et soit $\varepsilon > 0$ une précision bien définie.
Étape 1 : On calcule $y^k \in \partial h(x^k)$.
Étape 2 : On détermine $x^{k+1} \in \partial g^*(y^k)$, (en général, en résolvant un problème d'optimisation convexe).
Étape 3 : Si les conditions d'arrêt sont vérifiées alors on s'arrête : $x^* = x^{k+1}$ est solution optimale du problème, sinon on pose $k = k + 1$ et on retourne à l'étape 1.

1.7. Interprétation de DCA

Dans cette section nous détaillons chaque étapes de l'algorithme DC. DCA ne fonctionne qu'avec les composantes DC, g et h. À la k^{ième} itération de DCA, on remplace la composante h par sa minorante affine $h_k(x) = h(x^k) + \langle x - x^k, y^k \rangle$ au voisinage de x^k. Puisque h est une fonction convexe, on a donc $h(x) \geq h_k(x)$, $\forall x \in X$. Par suite, $g(x) - [h(x^k) + \langle y^k, x - x^k \rangle] \geq g(x) - h(x)$, $\forall x \in X$. En d'autre terme, $g(x) - [h(x^k) + \langle y^k, x - x^k \rangle]$ est une fonction majorante de la fonction $f(x)$.

Étape 0 : Puisque nous utilisons une approche de descente, la convergence de l'algorithme est indépendante du point de départ des suites que l'algorithme crée. Ainsi, nous pouvons initialiser l'algorithme avec un choix arbitraire du point initial x_0 tant qu'il est possible.

Étape 1 : On a que $\partial h(x^k) = arg\min \left\{ h^*(y) - g^*(y^{k-1}) - (x^k)^t(y - y^{k-1}) : y \in Y \right\}$. Puisque nous minimisons par rapport à y, nous considérons que x^k et y^{k-1} sont des constantes, alors on a $\partial h(x^k) = arg\max \left\{ (x^k)^t y - h^*(y) : y \in Y \right\}$. En général, le calcul de $\partial h(x^k)$ est explicite et facile à calculer, puisque d'après les conditions d'optimalité locale, x^k est (approximativement) un sous-gradient de $h^* - g^*$, et joue ainsi un rôle similaire à celui d'un algorithme de sous-gradient, que nous résolvons rapidement et efficacement. Puisque nous maximisons étant donné x^k, nous garantissons que $(h^* - g^*)(y^k - y^{k-1}) \leq 0$, $\forall k \in \mathbb{N}$. En raison de la symétrie de la dualité, y^k converge vers un point critique de $(h^* - g^*)$, c'est à dire un minimum local.

Étape 2 : On a $x^{k+1} \in arg\min \left\{ g(x) - h(x^k) - (y^k)^t(x - x^k) : x \in X \right\}$. Identique que l'étape 1, on trouve $x^{k+1} \in arg\max \left\{ (y^k)^t x - g(x) : x \in X \right\}$. En général, pour calculer x^{k+1} il faut résoudre, à chaque itération, un problème d'optimisation convexe.

Étape 3 : Cette étape est un test d'arrêt. Même si nous puissions garantir la convergence dans la limite infinie de k, la convergence complète peut prendre beaucoup de temps, pour cela nous prenons une approximation de la solution

optimale dans une limite prédéterminée, ε. Une fois l'écart entre deux itérations consécutifs est assez petite, l'algorithme s'arrête et renvoie la solution optimale $x^* = x^{k+1}$. Lorsque DCA converge vers un point x^*, ce point doit être un point KKT généralisé. Si on lance DCA à partir d'un point KKT généralisé de f, alors DCA s'arrête immédiatement à ce point car il est aussi un minimum de f^* qui n'est autre que la fonction majorante convexe définie au point x^* par $f^*(x) = g(x) - [h(x^*) + h(x^*) + \langle x - x^*, y^* \rangle]$, $y^* \in \partial h(x^*)$.

1.8. Convergence de DCA

DCA n'est pas déterminé que si les deux suites $\{x^k\}$ et $\{y^k\}$ sont bien définies à partir d'un $x^0 \in X$ choisi à l'avance.

Lemme 1.1. ***(existence des suites)*** *Les suites $\{x^k\}$ et $\{y^k\}$ sont bien définies si et seulement si : dom $\partial g \subset$ dom ∂h et dom $\partial h^* \subset$ dom ∂g^*.*

Preuve :

Par construction, on a $x^{k+1} \in \partial g^*(y^k)$ et $y^k \in \partial h(x^k)$ $\forall k \geq 0$. D'où

$$\left\{x^k\right\} \subset R(\partial g^*) = dom(\partial g) \qquad et \qquad \left\{y^k\right\} \subset R(\partial h) = dom(\partial h^*).$$

avec $R(\partial z)$ est l'image de ∂z. ■

Théorème 1.6. ***[101](convergence de DCA)***

- ***Premier cas*** *: Supposons que les suites $\{x^k\}$ et $\{y^k\}$ sont bien définies, alors on a les propriétés suivantes :*

  1. $g(x^{k+1}) - h(x^{k+1}) \leq h^*(y^k) - g^*(y^k) \leq g(x^k) - h(x^k)$.

     L'égalité $g(x^{k+1}) - h(x^{k+1}) = g(x^k) - h(x^k)$ a lieu si et seulement si $x^k \in \partial g^*(y^k)$ et $y^k \in \partial h(x^{k+1})$.

  2. *Similairement et par dualité on a :*
     $h^*(y^{k+1}) - g^*(y^{k+1}) \leq g(x^{k+1}) - h(x^{k+1}) \leq h^*(y^k) - g^*(y^k)$.

L'égalité $h^*(y^{k+1}) - g^*(y^{k+1}) = h^*(y^k) - g^*(y^k)$ *a lieu si et seulement si* $x^{k+1} \in \partial g^*(y^{k+1})$ *et* $y^k \in \partial h(x^{k+1})$.

3. *Si la valeur optimale* α *du problème primal* (P_{dc}) *est finie alors* $\lim\limits_{k\to+\infty} \Big\{g(x^k) - h(x^k)\Big\} = \lim\limits_{k\to+\infty} \Big\{h^*(y^k) - g^*(y^k)\Big\} = \beta \geq \alpha$.
4. *Si* α *est finie et les suites* $\Big\{x^k\Big\}$ *et* $\Big\{y^k\Big\}$ *sont bornées, alors* $\forall x^* \in \Omega(x^k)$ *(resp.* $\forall y^* \in \Omega(y^k)$*) il existe un* $y^* \in \Omega(y^k)$ *(resp.*$x^* \in \Omega(x^k)$*) tels que :*

   (i) $g(x^*) - h(x^*) = g^*(y^*) - h^*(y^*) = \beta \geq \alpha$.

   (ii) $\lim\limits_{k\to+\infty} \Big\{g(x^k) + g^*(y^k)\Big\} = g(x^*) - g^*(y^*) = \langle x^*, y^*\rangle$

   (iii) $\lim\limits_{k\to+\infty} \Big\{h(x^k) + h^*(y^k)\Big\} = h(x^*) - h^*(y^*) = \langle x^*, y^*\rangle$

   Où $\Omega(z^k)$ *est l'ensemble des valeurs d'adhérence de la suite* $\{z^k\}$.

- ***Deuxième cas*** *: Supposons que* g *et* h *sont fortement convexes et les suites* $\{x^k\}$ *et* $\{y^k\}$ *sont bien définies, alors on a les propriétés suivantes :*

  1. $g(x^{k+1}) - h(x^{k+1}) \leq h^*(y^k) - g^*(y^k) - \frac{1}{2}\rho_h||x^{k+1} - x^k||^2$

     $\leq g(x^k) - h(x^k) - \frac{1}{2}(\rho_h + \rho_g)||x^{k+1} - x^k||^2$

     où ρ_h *et* ρ_g *sont respectivement les coefficients de coercivité de* h *et* g.

     L'égalité $g(x^{k+1}) - h(x^{k+1}) = g(x^k) - h(x^k)$ *a lieu si et seulement si* $y^k \in \partial h(x^{k+1})$, $x^k \in \partial g^*(y^k)$ *et* $x^{k+1} = x^k$. *Dans ce cas on a* $y^k \in \partial h(x^k) \cap \partial g(x^k)$.

  2. *Similairement et par dualité on a :*
     $h^*(y^{k+1}) - g^*(y^{k+1}) \leq g(x^{k+1}) - h(x^{k+1}) - \frac{1}{2}\rho_{h^*}||y^{k+1} - y^k||^2$

     $\leq h^*(y^k) - g^*(y^k) - \frac{1}{2}(\rho_{h^*} + \rho_{g^*})||y^{k+1} - y^k||^2$

     où ρ_{h^*} *et* ρ_{g^*} *sont respectivement les coefficients de coercivité de* h^* *et* g^*.

     L'égalité $h^*(y^{k+1}) - g^*(y^{k+1}) = h^*(y^k) - g^*(y^k)$ *a lieu si et seulement*

si $x^{k+1} \in \partial g^*(y^{k+1})$, $y^k \in \partial h(x^{k+1})$ *et* $y^{k+1} = y^k$. *Dans ce cas on a* $y^k \in \partial h(x^{k+1}) \cap \partial g(x^{k+1})$.

3. *Si la valeur optimale* α *du problème primal* (P_{dc}) *est finie alors les suites décroissantes* $\left\{(g-h)(x^k)\right\}$ *et* $\left\{(h^*-g^*)(y^k)\right\}$ *convergent vers la même limite* $\beta \geq \alpha$.

4. *Si* α *est fini et les suites* $\{x^k\}$ *et* $\{y^k\}$ *sont bornées, alors* $\forall x^* \in \Omega(x^k)$ *(resp.* $\forall y^* \in \Omega(y^k)$*) il existe un* $y^* \in \Omega(y^k)$ *(resp.*$x^* \in \Omega(x^k)$*) tels que :*

   (i) $y^* \in \partial g(x^*) \cap \partial h(x^*)$ *et* $g(x^*) - h(x^*) = \beta$.

   (ii) $x^* \in \partial g^*(y^*) \cap \partial h^*(y^*)$ *et* $h^*(y^*) - g^*(y^*) = \beta$.

   (iii) $\lim\limits_{k\to+\infty} \|x^{k+1} - x^k\| = 0 \qquad et \qquad \lim\limits_{k\to+\infty} \|y^{k+1} - y^k\| = 0.$

   Où $\Omega(z^k)$ *est l'ensemble des valeurs d'adhérence de la suite* $\{z^k\}$.

Supposons que $\{x^k\}$ et $\{y^k\}$ sont bien définis. Le théorème de convergence de DCA (Théorème 6) requiert que ces deux suites soient bornées.

Lemme 1.2. ***(bornitude des suites)*** *Si* $(g-h)$ *est coercive, i.e.,*

$$\lim_{\|x\|\to+\infty} (g-h)(x) = +\infty.$$

Alors on a

(i) La suite $\{x^k\}$ *est bornée*
(ii) Si $\{x^k\} \subset int(dom(h))$ *alors la suite* $\{y^k\}$ *est aussi bornée.*

Similairement et par dualité, si $(h^* - g^*)$ *est coercive alors on a*

(iii) La suite $\{y^k\}$ *est bornée*
(iv) Si $\{y^k\} \subset int(dom(g^*))$ *alors la suite* $\{x^k\}$ *est aussi bornée.*

Preuve :
(i) est une conséquence immédiate de la croissance de $\left\{(g-h)(x^k)\right\}$ *(Théorème 6).*

(ii) Par construction $\{y^k\} \subset (\cup\left\{\partial h(x^k) : k \geq 0\right\}$ *qui est borné car* $\left\{x^k : k \geq 0\right\}$ *est un ensemble borné de* $int(dom(h))$. ■

Remarque 1.6. *1. Le théorème de convergence (théorème 1.6) donne la convergence de la suite* $\{x^k\}$ *(resp.* $\{y^k\}$*) générée par DCA vers un point critique du programme* $\{P_{dc}\}$ *(resp.* $\{D_{dc}\}$*).*

2. Il est clair que DCA s'applique aux fonctions convexes g et h, et non à la fonction f elle-même. On voit ainsi comment le mécanisme de DCA fonctionne pour les programmes DC non différentiables (f est une fonction DC non différentiable). Et puisqu'une fonction DC admet une infinité de décompositions DC, il en aura autant de DCA.
Le choix d'une décomposition DC appropriée est crucial car il conditionne les qualités essentielles (rapidité, robustesse, globalité des solutions calculées) du DCA résultant. Théoriquement, le problème de décomposition DC optimale reste à définir et à étudier. En pratique, on cherche des décompositions DC bien adaptées à la structure spécifique du problème traité afin que les deux suites $\{x^k\}$ *et* $\{y^k\}$ *soient obtenues à moindre coût en temps de calcul, si elles ne sont pas explicites. D'autre part, un programme convexe est un programme DC pour lequel DCA converge vers une solution (locale qui est aussi globale) : de cette manière DCA permet de construire une infinité d'algorithmes pour la programmation convexe, qui pourraient être plus performants que les méthodes existantes.*

Afin d'appliquer DCA, nous devons reformuler (P) à un programme continu en utilisant des techniques de pénalité exactes dans la programmation DC. La pénalité exacte est indiquée comme suit

1.9. Principe général des techniques de pénalité

Les différentes techniques de pénalisation relèvent souvent du principe suivant. Considérons le problème d'optimisation (P) avec contraintes :

$$(P) \qquad \begin{cases} \text{Minimiser } f(x) \\ \text{s.c.} \\ g(x) \leq 0, \\ h(x) = 0, \\ x \in C \subseteq \mathbb{R}^n \end{cases}$$

où C est un sous-ensemble de $\mathbb{R}^n$ défini par des contraintes qui peuvent être facilement incorporées dans l'optimisation (par exemple contraintes d'égalité linéaire).

Une fonction $p : \mathbb{R}^n \to \mathbb{R}$ est appelée fonction de pénalisation pour (P) si elle vérifie :

- $p(x) = 0$ si $g(x) \leq 0$, $h(x) = 0$ et
- $p(x) \leq 0$ si $g(x) \not\leq 0$ ou $h(x) \neq 0$.

Le programme de pénalisation est donc

$$(P_t) \qquad \begin{cases} \text{Minimiser } f_t(x) = f(x) + tp(x) \\ \text{s.c.} \\ x \in C. \end{cases}$$

Ici, t est un scalaire strictement positif, appelé facteur de pénalisation. Le but du terme additionnel est de pénaliser la violation des contraintes (on parle alors de pénalisation extérieure) ou l'abord de la frontière du domaine admissible (on parle dans ce cas de pénalisation intérieure). La question qui se pose immédiatement est de savoir si en résolvant (P_t) on résout (P). Autrement dit, on cherche à savoir quand les ensembles de solutions de (P) et (P_t) coïncident. Cela va dépendre du choix de la fonction p et du paramètre $t > 0$.

Proposition 1.8. ***(monotonie en pénalisation)*** *Supposons que, pour tout t considéré, (P_t) ait au moins une solution, noté $\bar{x}_t$. Alors, lorsque $t > 0$ croît,*

(i) $p(\bar{x}_t)$ décroît,

(ii) $f(\bar{x}_t)$ croît,

(iii) $f_t(\bar{x}_t)$ croît (si $p(.) \geq 0$).

1.9.1. Pénalisation extérieure

On parle de *pénalisation extérieure* lorsque la fonction de pénalisation p vérifie les propriétés suivantes :

(i) p est s.c.i. sur $\mathbb{R}^n$,

(ii) $p(x) \geq 0$, $\forall x \in \mathbb{R}^n$,

(iii) $p(x) = 0 \iff x \in C$.

Le qualificatif « extérieur » vient de la propriété **(iii)**, qui exprime que f_t ne modifie f qu'à l'extérieur de l'ensemble admissible.

Proposition 1.9. ***(convergence de la pénalisation extérieure)*** *Soient C un fermé, non vide de $\mathbb{R}^n$ et $p : \mathbb{R}^n \to \mathbb{R}$ une fonction vérifiant (1.1). Supposons que f soit s.c.i. tant qu'il existe une t_0 tel que $f_{t_0}(x) \to +\infty$ quand $\|x\| \to +\infty$. Alors, on a*

(i) $\forall t \geq t_0$, (P_t) a au moins une solution $\overline{x}_t$,

(ii) la suite $\{\overline{x}_t\}_{t\uparrow\infty}$ est bornée,

(iii) tout point d'adhérence de la suite $\{\overline{x}_t\}_{t\uparrow\infty}$ est une solution de (P).

1.9.2. Pénalisation intérieure

Dans certains problèmes, le fait que les itérés $\overline{x}_t$ générés par pénalisation extérieure ne soient pas admissibles peut être un inconvénient, par exemple, parce que f n'est pas défini à l'extérieur de C. On peut introduire des méthodes de pénalisation dans lesquelles les itérés $\overline{x}_t$ restent dans C. On parle alors de pénalisation intérieure. L'idée est d'utiliser un terme de pénalisation p qui tend vers l'infini lorsque x s'approche de la frontière de C.

Supposons que l'intérieur C° de C est non vide.
La fonction de pénalisation p considérée dans ce paragraphe satisfait les conditions suivantes :

(i) p est continue sur C°,

(ii) $p(x) \geq 0$, $\forall x \in C^\circ$,

(iii) $p(x) \to +\infty$ quand $x \to \partial C, x \in C^\circ$.

Le problème de pénalisation est alors

$$(P_t) \quad \begin{cases} \text{Minimiser } f_t(x) = f(x) + tp(x) \\ \text{s.c.} \\ x \in C^\circ. \end{cases}$$

La condition **(iii)** crée une « barrière » au bord de l'ensemble admissible, si bien que f_t porte parfois le nom de *fonction barrière*.

Proposition 1.10. ***(convergence de la pénalisation intérieure)*** *[24] Supposons que f soit continue sur $\mathbb{R}^n$ et que l'ensemble admissible C non vide vérifie $C = \overline{C}_0$.*
On suppose également que C soit borné, soit $f(x) \to +\infty$ quand $\|x\| \to +\infty$. Alors, si p satisfait les conditions (1.2), on a :

(i) $\forall t \geq 0$, (P_t) *a au moins une solution* $\overline{x}_t$,

(ii) la suite $\{\overline{x}_t\}_{t\downarrow\infty}$ *est bornée,*

(iii) tout point d'adhérence de la suite $\{\overline{x}_t\}_{t\downarrow\infty}$ *est une solution de* (P).

1.9.3. Pénalisation exacte en programmation DC

Les techniques de pénalité exacte sont souvent très utiles pour transformer un problème DC sous contraintes non convexes en un problème DC sous contraintes convexes. Ce dernier problème peut être résolu par DCA. Pour plus de détails sur les techniques de pénalité exacte pour la programmation DC, on pourra se référer à ([68], [69], [70])

Considérons un problème d'optimisation

$$(P) \qquad \alpha = inf\,\{f(x) : x \in C, p(x) \leq 0\}$$

où C est un polyèdre convexe non vide et borné dans $\mathbb{R}^n$, f, p sont deux fonctions concaves finies sur C, où p est une fonction à valeurs positives ou nulles sur C. Le problème (P) est un programme non convexe. La non convexité du problème est due au fait que la fonction objectif f est concave et que la contrainte $p(x) \leq 0$ est non convexe. Une contrainte $p(x) \leq 0$ avec p concave est dite anti-convexe. L'objectif du problème (P) est de minimiser une fonction concave f sous contraintes polyédrales (convexes) et une contrainte anti-convexe. Supposons que le problème (P), dont l'ensemble de solutions est noté par $\mathcal{P}$, est réalisable. Étant donné un paramètre $t > 0$, on peut définir le problème de pénalité par :

$$(P_t) \qquad \alpha = inf\,\{f(x) + tp(x) : x \in C\}$$

L'ensemble des solutions du problème (P_t) est noté par $\mathcal{P}_t$. On obtient le théorème suivant :

Théorème 1.7. *[101]* ***(théorème de pénalité exacte)*** *Soit K un polyèdre convexe borné non vide, f et p sont deux fonctions concaves finies sur K, où p est*

une fonction à valeurs positives ou nulles sur K. Alors il existe un nombre fini $t_0 \geq 0$ tel que pour tout $t \geq t_0$, les deux problèmes (P) et (P_t) sont équivalents au sens que $\mathcal{P} = \mathcal{P}_t$ et $\alpha(t) = \alpha$. Le paramètre t_0 peut être déterminé comme suit :

- *Si l'ensemble des sommets $V(K)$ de K est contenu dans l'ensemble $\{x \in K : p(x) \leq 0\}$, alors $t_0 = 0$ et $\alpha(0) = \alpha$.*
- *Si $\alpha(0) < \alpha$ alors $t_0 = \max\left\{\frac{\alpha - f(x)}{p(x)} : x \in V(K), p(x) > 0\right\}$.*

On a aussi les propriétés suivantes :

- *$\alpha(t) = \alpha$ si et seulement si $t \geq t_0$*
- *$\mathcal{P}_t \cap \{x \in K : p(x) \leq 0\} \neq \emptyset \iff \mathcal{P}_t \subset \mathcal{P} \iff t \geq t_0$*
- *$\mathcal{P} = \mathcal{P}_t$ si $t > t_0$*

Ce théorème est très utile en programmation non convexe, notamment en programmation DC, puisque beaucoup de problèmes d'optimisation sous contraintes non convexes peuvent être transformés en problème (P) à l'aide de ce théorème.

Remarquons que l'estimation du paramètre t n'est pas toujours évidente. En pratique, on donne souvent une valeur assez grande a priori pour simplifier l'estimation de t. Pour s'assurer que la valeur de t est assez grande, on propose le théorème suivant :

Théorème 1.8. *[101] Soient C un polyèdre convexe non vide et borné, f et p deux fonctions concaves finies sur C, où p est une fonction à valeurs positives ou nulles sur C . Alors il existe un nombre fini $t^* > 0$ tel que pour tout $t \leq t^*$, les deux suites générées par DCA $\left\{f(x^k) + tp(x^k)\right\}$ et $\left\{p(x^k)\right\}$ sont décroissantes.*

Ce théorème montre qu'il existe une valeur de t assez grande telle que la suite $\left\{p(x^k)\right\}$ générée par DCA soit décroissante. C'est-à-dire, si l'on trouve que la suite $\left\{p(x^k)\right\}$ n'est pas décroissante, alors t n'est pas assez grande.

Algorithme DCA Modifié

Étape 0 : x^0 donnée, t une grande valeur donné, $k = 0$.
Étape 1 : On calcule $y^k \in \partial(-f - tp)(x^k)$.
Étape 2 : On détermine $x^{k+1} \in \arg\min\left\{-\langle x, y^k\rangle : x \in C\right\}$.
Étape 3 : Si $p(x^{k+1}) > p(x^k)$, alors on augmente la valeur de t (par exemple $t = t + 1000$).
Étape 4 : Si les conditions d'arrêt sont vérifiées alors on s'arrête : $x^* = x^{k+1}$ est solution optimale du problème, sinon on pose $k = k + 1$ et on retourne à l'étape 1.

1.10. Conclusion

Dans ce chapitre nous avons présenté quelques notations de l'optimisation D.C. et de l'analyse convexe nécessaires pour la résolution du problème d'optimisation étudié dans cette thèse. Ces notations sont utilisées dans la démonstration des résultats théoriques dans les chapitres suivantes.

Chapitre 2

La Programmation Quadratique

Sommaire

2.1. Introduction

La programmation quadratique est une branche d'optimisation non linéaire où la fonction objective à minimiser est une fonction quadratique et les contraintes sont linéaires et/ou quadratiques. Son importance réside dans ses propriétés théoriques, ses applications dans plusieurs domaines scientifiques et différentes disciplines telles que l'économie, la finance, la médecine, les télécommunications et les sciences de l'ingénieur. En fait, plusieurs problèmes réels et académiques peuvent être modélisés sous forme de programme quadratique. En raison de ses nombreuses applications, la programmation quadratique est souvent considérée comme une discipline en elle-même. C'est le cas de l'optimisation du portefeuille en finance, Transmission de faisceaux multi couches, Problèmes dans les faisceaux multi capteurs, par exemple. De nombreux problèmes financiers peuvent représenter un cas particulier de ce problème.

L'optimisation quadratique fait l'objet d'une littérature abondante, dont les travaux de synthèses réalisés par Frank et Wolfe ([22], 1956), suivis des travaux de

Boros et Hammer [9] et Billionnet [6], en optimisation quadratique en variables 0-1. Plusieurs variantes de ce problème ont été traitées par plusieurs chercheurs : cas convexes, cas non convexes avec des variables entières ou des variables binaires.

Les Programmes quadratiques binaires (BQP) apparaissent dans de nombreux domaines, y compris l'économie, la planification de la machine, la physique des solides, la gestion des messages de trafic, l'emplacement des installations, assignation de fréquence, allocation de registres, Pattern Matching, analyse des données biologiques et archéologiques, voir ([30], [49]). Un grand nombre de méthodes, heuristiques ou déterministes, ont été développées pour déterminer leurs solutions, parmi eux : Hammer-Rudeanu [31], Adams-Sherali [2], Pardalos-Rodgers [65], Jha-Pardalos [45], Adams-Dearing [1], Helmberg-Rendl [36], Glover-Kochenberger-Alidaee [25], Le Thi-Pham Dinh [54] et Billonnet-Elloumi [8].

Dans ce chapitre nous allons présenter les problèmes classiques de l'optimisation quadratique avec et sans contraintes, leurs conditions d'optimalité et définir quelques notions usuelles.

2.2. La programmation quadratique sans contraintes

Un problème de programmation quadratique sans contraintes peut s'écrire sous la forme suivante :

$$\begin{cases} \min \quad f(x) = \frac{1}{2}x^tQx + c^tx \\ x \in \mathbb{R}^n \end{cases} \tag{2.1}$$

où Q est une matrice carrée symétrique d'ordre n, c est un vecteur de $\mathbb{R}^n$ et $x \in \mathbb{R}^n$ est le vecteur inconnu à déterminer. Il s'agit donc de déterminer un point x^* de $\mathbb{R}^n$ tel que :

$$\forall x \in \mathbb{R}^n : \qquad f(x^*) \leq f(x) \tag{2.2}$$

c'est-à-dire un minimum global de f sur $\mathbb{R}^n$. Une condition suffisante pour l'existence de $x^* \in \mathbb{R}^n$ vérifiant (2.2) est que f soit continue (ou, plus généralement

semi-continue inférieurement) et possède la propriété de croissance à l'infini : $f(x) \to +\infty$ lorsque $\|x\| \to +\infty$.
Lorsque l'inégalité est stricte : $f(x^*) < f(x)$ est vérifiée $\forall x \in \mathbb{R}^n$, $x \neq x^*$, le minimum x^* est global et unique, mais éventuellement l'unicité n'est pas toujours garantie.

Pour beaucoup de problèmes d'optimisation sans contraintes, en l'absence de propriétés particulières, telles que la convexité ou la quasi-convexité, les principales méthodes connues ne permettent pas la détermination d'un minimum global : il faut alors se contenter d'un optimum local, c'est-à-dire le point qui vérifie (2.2) seulement dans un voisinage de x^*.

Dans ce qui suit, nous discuterons successivement les conditions d'optimalité puis les principales méthodes numériques (algorithmes) pour la résolution de problèmes d'optimisation quadratique sans contraintes.

Pour démontrer quelques propriétés nous avons besoin de la définition et quelques propriétés des matrices semi-définies et définies positives :

Définition 2.1. *Une matrice $X \in S_n$ (l'espace des matrices symétriques réelles carrées d'ordre n) est dite semi-définie positive (resp. définie positive), notée $X \succcurlyeq O$ (resp. $X \succ O$), si et seulement si $u^t X u \geq 0, \forall u \in \mathbb{R}^n$ (resp. $u^t X u > 0, \forall u \in \mathbb{R}^n \setminus \{0\}$), où O désigne la matrice nulle de type $n \times n$.* ■

On note l'ensemble des matrices semi-définies positives (resp. définies positives) par S_n^+ (resp. S_n^{++}). Nous avons la relation d'inclusion suivante :

$$S_n^{++} \subset S_n^+ \subset S_n \subset M_n.$$

où M_n est l'espace des matrices carrées réelles d'ordre n.

Proposition 2.1. *$X \in S_n^+$ (resp. S_n^{++}) si et seulement si l'une des conditions suivantes est vérifiée :*
(i) les valeurs propres de X sont toutes positives ou nulles (resp. sont strictement positives).
(ii) $u^t X u \geq 0, \forall u \in \mathbb{R}^n$ (resp. $u^t X u > 0 \, \forall u \neq 0$).

(iii) Les déterminants de tous les mineurs symétriques de X sont positifs ou nuls (resp. strictement positifs).

2.2.1. Conditions nécessaires d'optimalité locale

Dans cette section, on suppose que f est deux fois continûment différentiable ($f \in C^2$). Alors :

Théorème 2.1. *Une condition nécessaire pour que x^* soit un minimum (local ou global) de f est :*

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

(*ii*) *La matrice hessienne* $H(x^*) = \nabla^2 f(x^*) = \dfrac{\partial^2 f}{\partial x_i \partial x_j}(x^*)$ *est une matrice semi-définie positive.*

Preuve

Soit x^* un minimum local (ou global) de f. Comme f est deux fois continûment différentiable, le développement de *Taylor* à l'ordre 2 au voisinage de x^* donne :

$$f(x) = f(x^*) + \nabla f^t(x^*)(x - x^*) + \tfrac{1}{2}(x - x^*)^t \nabla^2 f(x^*)(x - x^*) + \|x - x^*\|^2 \epsilon(x - x^*).$$

avec $\epsilon(x - x^*) \to 0$ quand $x \to x^*$.
Si $\nabla f(x^*) \neq 0$ alors en choisissant $x = x^* - \theta \nabla f(x^*)$ on aurait, pour $\theta > 0$ suffisamment petit : $f(x) < f(x^*)$ ce qui contredirait le fait que x^* est un minimum local. Donc la condition (i) est bien nécessaire, et on peut écrire :

$$f(x) = f(x^*) + \tfrac{1}{2}(x - x^*)^t \nabla^2 f(x^*)(x - x^*) + \|x - x^*\|^2 \epsilon(x - x^*).$$

Si la matrice $\nabla^2 f(x^*)$ n'est pas semi-définie positive, c'est qu'il existe un vecteur $d \in \mathbb{R}^n$ $(d \neq 0)$ tel que :

$$d^t \nabla^2 f(x^*) d < 0.$$

Alors en choisissant $x = x^* + \theta d$, pour $\theta > 0$ suffisamment petit on aurait $f(x) < f(x^*)$ ce qui contredirait encore l'optimum locale de x^*.
La condition (ii) est donc également nécessaire. ■

Un point x^* qui vérifie la condition (i) c'est-à-dire : $\dfrac{\partial f}{\partial x_i}(x^*) = 0$ $(i = 1, ..., m)$ est appelé un point stationnaire.

2.2.2. Conditions suffisantes d'optimalité locale

Théorème 2.2. *Une condition suffisante pour que x^* soit un optimum local strict de f sur $\mathbb{R}^n$ est :*

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

(ii) *La matrice hessienne $H(x^*) = \nabla^2 f(x^*)$ est une matrice définie positive.*

Preuve
Considérons un point x^* satisfaisant les deux conditions (i) et (ii) du théorème.
Le développement de *Taylor* de f à l'ordre 2 au voisinage de x^* s'écrit alors :

$$f(x) = f(x^*) + \tfrac{1}{2}(x - x^*)^t \nabla^2 f(x^*)(x - x^*) + \|x - x^*\|^2 \epsilon(x - x^*).$$

avec $\epsilon(x - x^*) \to 0$ quand $x \to x^*$.
Pour toute direction de déplacement $d \in \mathbb{R}^n$ $(\|d\| = 1)$ on a alors :

$$f(x^* + \theta d) = f(x^*) + \frac{\theta^2}{2} d^t \nabla^2 f(x^*) d + \theta^2 \epsilon(\theta)$$

où $\epsilon(\theta) \to 0$ quand $\theta \to 0$.
D'après la condition (ii) on a : $d^t \nabla^2 f(x^*) d > 0$ et par suite, pour θ suffisamment petit, on aura : $f(x^* + \theta d) > f(x^*)$. Par suite x^* est bien un minimum local de f.
■
Remarquons que la condition (ii) du théorème 10 revient à supposer que la fonction f est strictement convexe dans un voisinage de x^*.

2.2.3. Cas des fonctions convexes : conditions nécessaires et suffisantes d'optimalité globale

Dans le cas d'une fonction convexe propre f définie sur $\mathbb{R}^n$, on a vu dans le chapitre 1 qu'une condition nécessaire et suffisante pour que x^* soit un minimum global de f est que 0 soit un sous-gradient de f en x^* ($0 \in \partial f(x^*)$).
Pour une fonction continûment différentiable, on a donc :

Théorème 2.3. [61] *Si f est une fonction convexe continûment différentiable, une condition nécessaire et suffisante pour que x soit un optimum globale de f sur $\mathbb{R}^n$ est que : $\nabla f(x^*) = 0$. Autrement dit, dans le cas convexe, la stationnarité à elle seule constitue une condition nécessaire et suffisante d'optimalité globale.*

2.2.4. Cas des fonctions quelconques : difficulté de problème général

Lorsque la fonction à minimiser est convexe, on arrive bien à se passer de la différentiabilité en tout point en utilisant la notion de sous-gradient. Ce cas particulier est cependant très important puisqu'il apparait dans la résolution de plusieurs problèmes en programmation mathématique, par exemple : en optimisation combinatoire, le problème dual d'un programme en nombres entiers consiste à optimiser une fonction convexe ou concave non partout différentiable.
Cependant, il existe beaucoup de fonctions qui, bien que continues, ne sont ni convexes, ni sous différentiables ; et il existe beaucoup de fonctions qui ne satisfont même pas l'hypothèse de continuité. Pour l'optimisation sans contraintes de telles fonctions sur $\mathbb{R}^n$ (même en supposant que les propriétés requises pour l'existence d'un optimum global sont satisfaites, par exemple : f s.c.i. et $f(x) \to +\infty$ lorsque $\|x\| \to +\infty$) on ne dispose pas d'un algorithme efficace général.

2.3. Les méthodes numériques de résolution pour la programmation quadratique sans contraintes

Nous pouvons appliquer plusieurs méthodes numériques pour résoudre un programme quadratique sans contraintes. Nous citons, par exemple, les différentes méthodes du gradient, les méthodes de directions conjuguées, la méthode de Newton et ses variantes (sécante, quasi-newton), etc.

Nous supposons dans cette section que f est continue et à dérivées premières continues. La stationnarité de f est une condition nécessaire d'optimalité. En pratique, toutes les méthode d'optimisation sans contraintes dans $\mathbb{R}^n$ consiste à chercher un point x^* stationnaire ($\nabla f(x^*) = 0$).

Ce problème est équivalent à résoudre le système d'équations non linéaires suivant :

$$\frac{\partial f}{\partial x_i}(x) = 0 \quad \forall i = 1, ..., n.$$

La résolution de ce système peut être directe, ce qui conduit à la méthode de Newton. Cependant, la méthode peut ne pas converger si le point initial est trop loin de la solution cherchée x^*. D'autre part, cette méthode nécessite le calcul des dérivées secondes en chaque point et suppose que la fonction est deux fois continûment différentiable.

C'est pourquoi les méthodes les plus couramment utilisées procèdent différemment : il s'agit de procédures itératives où l'on engendre une suite de point $x^0, x^1, ..., x^k$ convergeant vers un optimum local de f ou au moins un point stationnaire.

À chaque étape k, x^{k+1} est défini par : $x^{k+1} = x^k + \lambda_k \, d_k$ où d_k est une direction de déplacement qui peut être :

- Soit l'opposé du gradient de f en x^k : $d_k = -\nabla f(x^k)$,
- Soit calculée à partir du gradient,
- Soit choisie de façon à consister une direction de descente c'est-à-dire qu'elle vérifie la condition suivante : $\nabla f^t(x^k) d_k < 0$.

2.3.1. Méthodes de gradient. Gradient à pas prédéterminé

Les méthodes de gradient sont une famille de méthodes qui procèdent de la façon suivante :
On part d'un point initial x^0 puis on calcule le gradient $\nabla f(x^0)$ au point x^0. Puisque $\nabla f(x^0)$ indique la direction de plus grande augmentation de f, on effectue un pas de déplacement de longueur λ_0 dans la direction opposée au gradient, ce qui conduit au point :

$$x^1 = x^0 - \lambda_0 \frac{\nabla f(x^0)}{\|\nabla f(x^0)\|}$$

Cette procédure est répétée et engendre les points $x^0, x^1, ..., x^k$, suivant la relation :

$$x^{k+1} = x^k - \lambda_k \frac{\nabla f(x^k)}{\|\nabla f(x^k)\|} \text{ où, } \forall k,\ \lambda_k > 0.$$

Dans cette famille de méthodes, il convient de signaler les méthodes de *gradient prédéterminé* dans lesquelles on choisit a priori les valeurs des déplacements λ_k. Polyak (1966) a étudié la convergence de ce schéma itératif et a montré que $x^k \to x^*$ sous les seules conditions :

$$\begin{cases} \lambda_k \to 0 (k \to \infty) \\ \sum\limits_{k=0}^{\infty} \lambda_k = +\infty \end{cases}$$

(par exemple, on peut prendre $\lambda_k = \frac{1}{k}$). L'inconvénient de cette procédure est évidement la lenteur de la convergence de x^k vers x^*. le principal intérêt des méthodes de gradient à pas prédéterminé est de se généraliser au cas des fonctions non partout différentiables.

2.3.2. Méthodes de directions conjuguées

Il s'agit de méthodes itératives qui, appliquées à une fonction quadratique strictement convexe de n variables, conduisent à l'optimum exact en n étapes au plus.
Dans cette section, nous supposons que la fonction quadratique $f(x) = \frac{1}{2}x^tQx + c^tx$

est strictement convexe, où Q est une matrice symétrique, définie positive d'ordre n et $c \in \mathbb{R}^n$ est un n-vecteur $c = \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix}$.

Le principe des méthodes de gradients conjuguées consiste, en partant d'un point initial x^0, à minimiser f successivement suivant n directions $d_0, d_1, ..., d_{n-1}$ possédant la propriété d'être mutuellement conjuguées par rapport à la fonction quadratique f. Cette propriété peut s'exprimer par :

$$\left.\begin{array}{l} \forall i (0 \leq i \leq n-1) \\ \forall j (0 \leq j \leq n-1) \\ i \neq j \end{array}\right\} \Rightarrow d_i^t \, Q \, d_j = 0 \tag{2.3}$$

Supposons donc que x^{k+1} soit déterminé à partir de x^k, $\forall k = 0, 1, ..., n-2$, par :

$$x^{k+1} = x^k + \lambda_k d_k$$

où λ_k est la valeur de λ qui minimise $g(\lambda) = f(x^k + \lambda d_k)$.
En fait, dans ces conditions, le point obtenu à la n-ième étape, c'est-à-dire :

$$x^n = x^0 + \textstyle\sum_{j=0}^{n-1} \lambda_j d_j$$

est nécessairement l'optimum du problème, autrement dit, vérifie :

$$Qx^n + c = \nabla f(x^n) = 0 \tag{2.4}$$

Comme λ_k minimise f dans la direction d_k, on a, $\forall k$:

$$d_k^t \, \nabla f(x^{k+1}) = d_k^t (Qx^{k+1} + c) = 0.$$

Soit :

$$d_k^t Q(x^k + \lambda_k d_k) + d_k^t c = 0$$

D'où nous pouvons conclure :

$$\lambda_k = -\frac{d_k^t (Qx^k + c)}{d_k^t \, Q d_k} \tag{2.5}$$

Remarque 2.1. *Q étant définie positive, le dénominateur de l'expression (2.5) ne peut pas être nul que si $d_k = 0$.*

En remarquant que : $x^k = x^0 + \sum_{j=0}^{k-1} \lambda_j d_j$, on peut encore écrire :

$$d_k^t \, Q x^k = d_k^t \, Q x^0 + \sum_{j=0}^{k-1} \lambda_j d_k^t \, Q d_j = d_k^t \, Q x^0$$

et par suite λ_k est défini par l'expression suivante :

$$\lambda_k = -\frac{d_k^t (Q x^0 + c)}{d_k^t \, Q d_k} \tag{2.6}$$

Nous pouvons donc définir la propriété suivante, caractéristique de toutes les méthodes de directions conjuguées :

Proposition 2.2. *Pour tout $1 \leq k \leq n$ le point :*

$$x^k = x^0 + \sum_{j=0}^{k-1} \lambda_j d_j$$

où les valeurs de λ_j sont données par (2.6) est le minimum de f restreint à la variété affine V^k engendrée par $(d_0, d_1, ..., d_{k-1})$ et passant par x^0 (l'unicité de ce minimum découle de la stricte convexité de la fonction f).
En particulier, $x^n = x^0 + \sum_{j=0}^{n-1} \lambda_j d_j$ est l'optimum de f sur $\mathbb{R}^n$.

Ainsi, lorsque cette propriété est appliquée à une fonction quadratique strictement convexe sur $\mathbb{R}^n$, une méthode de direction conjuguées converge de façon finie et produit en au plus n étapes le minimum exact.
Comme il existe un certain nombre de degré de liberté pour le choix des directions $(d_0, d_1, ..., d_{n-1})$, on peut imaginer les nombreux algorithmes basés sur le principe général des directions conjuguées. Nous étudions plus particulièrement la méthode du *gradient conjugué* pour les fonctions quadratiques.

2.3.3. La méthode du gradient conjugué

Dans cette section, nous supposons que la fonction quadratique $f(x) = \frac{1}{2}x^tQx + c^tx$ est strictement convexe.
La méthode du gradient conjugué a été proposée à l'origine par Hestenes ([38], 1969) pour résoudre un système d'équations linéaires, et la première application en optimisation non linéaire est due à Fletcher et Reeves ([20], 1991). Elle consiste à construire progressivement des directions $d_0, d_1, ..., d_k$ mutuellement conjuguées par rapport à la matrice Q. À chaque étape k, la direction d_k est obtenue par combinaison linéaire du gradient $-\nabla f(x^k)$ en x^k, et des directions précédentes, $d_0, d_1, ..., d_{k-1}$, les coefficients de la combinaison linaire étant choisis de telle sorte que d_k soit conjuguées par rapport à toutes les directions précédentes.

En notant $g_k = \nabla f(x^k)$ le gradient de la fonction f en x^k, nous pouvons écrire l'algorithme sous la forme suivante :

Algorithm 1 Algorithme du gradient conjugué

Initialisation : Soit x^0 le point de départ, $g_0 = \nabla f(x^0) = Qx^0 + c$.
Poser $d_0 = -g_0$, $k = 0$.
Pour $(k = 0, 1, ..., n-1)$
à l'itération k on est au point x^k **faire**
$x^{k+1} = x^k + \lambda_k d_k$ avec :

$$\lambda_k = -\frac{g_k^t d_k}{d_k^t Q\, d_k} \tag{2.7}$$

Puis :

$$d_{k+1} = -g_{k+1} + \beta_k d_k \tag{2.8}$$

Avec :

$$\beta_k = \frac{g_{k+1}^t Q\, d_k}{d_k^t Q\, d_k} \tag{2.9}$$

Fin Pour

Nous remarquons que (2.7) est identique à l'expression (2.5). Pour démontrer la

validité de l'algorithme (convergence en plus n étapes), il suffit donc de vérifier que les directions engendrées par (2.8) et (2.9) sont mutuellement conjuguées.

Théorème 2.4. [61] *À une itération k quelconque de l'algorithme où l'optimum de f n'est pas encore atteint (c'est-à-dire $g_i \neq 0$, $i = 0, 1, ..., k$) on a :*
(i)

$$\lambda_k = \frac{g_k^t d_k}{d_k^t Q\, d_k} \neq 0$$

(ii)

$$\begin{aligned}\beta_k &= \frac{g_{k+1}^t\,[g_{k+1} - g_k]}{g_k^t\, g_k} \\ &= \frac{g_{k+1}^t\, g_{k+1}}{g_k^t\, g_k}\end{aligned}$$

(iii) Les directions $d_0, d_1, ..., d_{k-1}$ engendrées par l'algorithme sont mutuellement conjuguées.

2.3.4. La méthode de Newton

Le principe de la méthode de Newton pour l'optimisation est de minimiser successivement les approximations au second-ordre de la fonctionnelle f.
On suppose que la fonction f est deux fois continûment différentiable et que l'on sait calculer toutes ses dérivées secondes. L'idée consiste à remplacer, au voisinage du point courant x^k, la fonction f par son approximation quadratique :

$$q(x) = f(x^k) + \nabla f^t(x^k)(x - x^k) + \tfrac{1}{2}(x - x^k)^t \nabla^2 f(x^k)(x - x^k).$$

On prend alors comme point x^{k+1} le minimum de q lorsqu'il existe. Ceci ne peut être le cas que si $\nabla^2 f(x^k)$ est une matrice *définie positive*. La fonction q est alors strictement convexe et a un minimum unique x^{k+1} définie par : $\nabla q(x^{k+1}) = 0$.
Ce qui conduit au système linéaire :

$$\nabla f(x^k) = \nabla^2 f(x^k) \cdot (x^{k+1} - x^k).$$

D'où la formule itérative :

$$x^{k+1} = x^k - [\nabla^2 f(x^k)]^{-1} \cdot \nabla f(x^k).$$

Cette formule n'est autre que la méthode de *Newton* appliquée à la résolution du système d'équations non linéaires :

$$\frac{\partial f}{\partial x_i}(x) = 0 \quad (i = 1, \cdots, n).$$

La méthode de Newton converge en une seule itération lorsqu'elle est appliquée à une fonction quadratique strictement convexe. Cependant, lorsqu'on veut l'utiliser sur une fonction quelconque, des difficultés apparaissent, dues essentiellement au fait qu'*elle ne possède pas de propriétés de convergence globale* : si le point de départ x^0 est trop éloigné de x^* la méthode peut ne pas converger.
Puisque l'approximation de f par q n'est valable que dans un voisinage de x^k, on peut agir sur le pas de déplacement en utilisant une formule itérative du type :

$$x^{k+1} = x^k - \lambda_k [\nabla^2 f(x^k)]^{-1} \cdot \nabla f(x^k)$$

où λ_k est un scalaire choisi, par exemple, de façon à ce que la norme $\|x^{k+1} - x^k\|$ ne soit pas trop grande. On peut aussi choisir λ_k de façon à ce que x^{k+1} minimise $g(\lambda) = f(x^k + \lambda d_k)$ dans la direction

$$d_k = -[\nabla^2 f(x^k)]^{-1} \cdot \nabla f(x^k).$$

Une autre façon de déterminer le pas de déplacement est d'essayer la valeur $\lambda_k = 1$. Si $f(x^k + \lambda d_k) \leq f(x^k)$, alors λ^* minimisant $g(\lambda) = f(x^k + \lambda d_k)$ est certainement compris ente 0 et 1 (puisque si $\nabla^2 f(x^k)$ est défini positif, d_k est une direction de descente, c'est-à-dire que : $\dfrac{dg}{d\lambda}(0) < 0$).
Une autre difficulté peut apparaître lorsque le hessien $\nabla^2 f(x^k)$ n'est pas défini positif. Dans ce cas, la direction de déplacement $-[\nabla^2 f(x^k)]^{-1} \cdot \nabla f(x^k)$ peut ne pas être une direction de descente, et la convergence globale de la méthode de Newton n'est pas assurée. En effet, certains auteurs ont suggéré de perturber légèrement le hessien $\nabla^2 f(x^k)$ de façon à obtenir une matrice M_k définie positive. On aboutit ainsi à la formule itérative :

$$x^{k+1} = x^k - \lambda_k \cdot [M_k]^{-1} \cdot \nabla f(x^k)$$

où le scalaire λ_k est déterminé par l'une des techniques ci-dessus.
Nous remarquons ici que le fait que M_k soit définie positive assure que la direction de déplacement $d_k = -[M_k]^{-1} \cdot \nabla f(x^k)$ est une direction de descente. En effet,

$$\nabla f^t(x^k) \cdot d_k = -\nabla f^t(x^k) \cdot [M_k]^{-1} \cdot \nabla f(x^k) < 0.$$

Pour construire M_k à partir de $\nabla^2 f(x^k)$ on pourra, par exemple, utiliser une perturbation du type :

$$M_k = \mu_k \, I + \nabla^2 f(x^k)$$

où $\mu_k > 0$ est un scalaire choisi minimal avec la contrainte que toutes les valeurs propres de M_k soient supérieures ou égales à une constante $\gamma > 0$ donnée. La convergence globale de cette procédure est facile à démontrer.

2.3.5. Les méthodes quasi-newtoniennes

Le principe de ces méthodes consiste essentiellement en une généralisation de la formule itérative de Newton (voir 2.3.4) :

$$x^{k+1} = x^k - \lambda_k \cdot [\nabla^2 f(x^k)]^{-1} \cdot \nabla f(x^k).$$

D'après 2.3.4, une limitation importante de la méthode de Newton consistait dans la restriction : $\nabla^2 f$ définie positive.
Une extension naturelle consiste à remplacer $[\nabla^2 f(x^k)]^{-1}$ par une matrice H_k définie positive donnant la direction de déplacement à partir du gradient $\nabla f(x^k)$. D'où une formule itérative du type :

$$x^{k+1} = x^k - \lambda_k \cdot H_k \cdot \nabla f(x^k)$$

λ_k est choisi de façon à minimiser $g(\lambda) = f(x^k + \lambda d_k)$ dans la direction $d_k = -H_k \cdot \nabla f(x^k)$, ou au moins choisi de telle sorte que $g(\lambda_k) < g(0)$.
La matrice H_k est évidement modifiée à chaque itération, de telle manière que, pour une fonction quadratique convexe de la forme :

$$f(x) = \frac{1}{2}x^t Q x + c^t x$$

avec Q une matrice définie positive, H_k converge vers Q^{-1} l'inverse du hessien de f.
En fin de convergence, on retrouvera donc un comportement proche de celui de la méthode de Newton.
Lorsqu'on applique cette méthode à une fonction f quelconque, H_k peut alors être considérée, à chaque instant, comme une approximation (définie positive) de l'inverse du hessien de f.
Il existe évidemment beaucoup de variantes possibles dans le choix de la formule de mise à jour de la matrice H_k. Généralement on impose la relation :

$$x^k - x^{k-1} = H_k \left[\nabla f(x^k) - \nabla f(x^{k-1})\right]. \tag{2.10}$$

La formule de correction qui permet d'obtenir la matrice H_{k+1} à partir de la matrice H_k utilise les nouvelles informations obtenues lors de l'étape k de l'algorithme, c'est-à-dire le gradient $\nabla f(x^{k-1})$ au point x^{k-1} (généralement obtenu par recherche unidimensionnelle dans la direction $d_k = -H_k \cdot \nabla f(x^k)$ à partir de x_k).

Différentes formules de corrections du type :

$$H_{k+1} = H_k + \Delta_k$$

ont été proposées. Suivant que la matrice Δ_k est de rang 1 ou de rang 2 on parlera de correction de rang 1 ou de rang 2. Une des premières formules de correction utilisées pour construire une approximation de l'inverse du hessien consiste à choisir une matrice (de rang 1) de la forme : $\Delta_k = \alpha_k u_k u_k^t$ où u_k et α_k sont respectivement un vecteur et un scalaire choisis de telle sorte que l'on ait :

$$x^{k+1} - x^k = H_{k+1} \left[\nabla f(x^{k+1}) - \nabla f(x^k)\right].$$

On remarque que si la matrice H_0 est symétrique, la correction ci-dessus préserve la symétrie des matrices H_k. En posant :

$$\delta_k = x^{k+1} - x^k$$

$$\gamma_k = \nabla f(x^{k+1}) - \nabla f(x^k)$$

Montrons comment on peut déterminer u_k et α_k tels que : $H_{k+1} \cdot \gamma_k = \delta_k$. Soit encore :

$$[H_k + \alpha_k(u_k\, u_k^t)]\gamma_k = \delta_k.$$

En prenant le produit scalaire des deux membres avec γ_k on obtient :

$$\gamma_k^t\, H_k\, \gamma_k + \alpha_k(\gamma_k^t u_k)(u_k^t \gamma_k) = \gamma_k^t \delta_k.$$

Soit :

$$\alpha_k(u_k^t\gamma_k)^2 = \gamma_k^t(\delta_k - H_k\gamma_k).$$

En utilisant l'identité :

$$\alpha_k(u_k^t\gamma_k) = \frac{(\alpha_k u_k u_k^t \gamma_k)(\alpha_k u_k u_k^t \gamma_k)^t}{\alpha_k(u_k^t\gamma_k)^2}$$

En remplaçant :

$$\alpha_k u_k u_k^t \gamma_k \text{ par : } \delta_k - H_k\gamma_k$$

$$\alpha_k(u_k^t\gamma_k)^2 \text{ par : } \gamma_k^t(\delta_k - H_k\gamma_k)$$

On obtient la formule de correction (de rang 1) suivante :

$$H_{k+1} - H_k = \alpha_k(u_k u_k^t) = \frac{(\delta_k - H_k\gamma_k)(\delta_k - H_k\gamma_k)^t}{\gamma_k^t(\delta_k - H_k\gamma_k)} \tag{2.11}$$

La variation de cette formule provient du résultat suivant :

Théorème 2.5. [61] *Soit une fonction quadratique, A son hessien (supposé défini positif). Considérons un processus itératif qui, partant d'un point x^0, engendre successivement les points $x^1 = x^0 + \delta_1, ..., x^n = x^{n-1} + \delta_n$ par déplacement le long de n directions interdépendantes $\delta_1, \delta_2, ..., \delta_n$ successivement.*
Alors la suite des matrices H_k obtenue par :

$$\begin{cases} H_0 \text{ matrice symétrique quelconque} \\ H_{k+1} = H_k + \dfrac{(\delta_k - H_k\gamma_k)(\delta_k - H_k\gamma_k)^t}{\gamma_k^t(\delta_k - H_k\gamma_k)} \end{cases} \tag{2.12}$$

converge, en au plus n étapes, vers A^{-1}, l'inverse du hessien de f.

$$(\gamma_k = \nabla f(x^{k+1}) - \nabla f(x^k) = A(x^{k+1} - x^k) = A\,\delta_k).$$

Nous remarquons que la formule (2.11) présente l'avantage que le point x^{k+1} n'a pas besoin d'être choisi comme le minimum de la fonction f dans la direction δ_k à partir du point x^k. C'est pourquoi de nombreux auteurs ont utilisés cette formule pour construire des algorithmes ne nécessitant pas de minimisation unidimensionnelle [76].
En revanche la formule (2.11) présente l'inconvénient que, même si la fonction f est quadratique, et que H_0 et le hessien de f sont définis positifs, les matrices H_k obtenues ne sont pas nécessairement définies positives. D'autre part, la formule (2.11) peut ne pas avoir de sens, par exemple, si le terme $\gamma_k^t(\delta_k - H_k\gamma_k)$ est nul ou simplement de très faible valeur. Lorsque ces circonstances se produisent, il faut alors prévoir des procédures spéciales pour les mises à jour des matrices H_k (on peut par exemple poser $H_{k+1} = H_k$) mais alors la convergence de H_k vers l'inverse du hessien, n'est pas assurée.
Les méthodes quasi-newtoniennes qui vont être décrites par la suite, et qui sont fondées sur les formules de correction de rang 2, ne présentent pas ces inconvénients. Par contre, elles nécessitent l'utilisation d'une procédure d'optimisation unidimensionnelle exacte (algorithme de DFP) ou approchée (algorithme BFGS).

2.3.6. L'algorithme de Davidon-Fletcher-Powell (DFP)

Cet algorithme utilise la formule de correction (de rang 2) suivante :

$$H_{k+1} = H_k + \frac{\delta_k\,\delta_k^t}{\delta_k^t\,\gamma_k} - \frac{H_k\,\gamma_k\,\gamma_k^t\,H_k}{\gamma_k^t\,H_k\,\gamma_k} \tag{2.13}$$

où le point x^{k+1} est obtenu à partir de x^k par déplacement dans la direction $d_k = -H_k\,\nabla f(x^k)$ et où ;

$$\delta_k = x^{k+1} - x^k, \qquad \gamma_k = \nabla f(x^{k+1}) - \nabla f(x^k).$$

Le résultat suivant montre que, sous certaines conditions, la formule (2.13) conserve la définie-positivité des matrices H_k.

Théorème 2.6. [61] *Supposons que la matrice H_k est définie positive. Alors si la condition $\delta_k^t \gamma_k > 0$ est vérifiée, la matrice H_{k+1} donnée par (2.13) est définie positive. Cette condition est satisfaite, en particulier, si le point x^{k+1} est obtenu à partir de x^k par minimisation unidimensionnelle dans la direction $d_k = -H_k \nabla f(x^k)$.*

Remarque 2.2. *La propriété $\delta_k^t \gamma_k > 0$ est vérifiée également par des méthodes de recherche unidimensionnelle ne visant pas l'obtention de l'optimum exact dans la direction.*

La méthode utilisant la formule de correction (2.13) est alors :

Algorithm 2 Algorithme de Davidon-Fletcher-Powell

Initialisation : Soit x^0 le point de départ. Choisir H_0 définie positive quelconque (par exemple la matrice identité), $k \leftarrow 0$, $\epsilon > 0$ une précision donnée.

(a) Répéter (itération courante k)

déterminer la direction de déplacement :

$$d_k = -H_k \nabla f(x^k).$$

Déterminer θ^* solution optimale du probleme d'optimisation unidimensionnel :

$$(P_k) : \quad \min\{g(\theta) = f(x^k + \theta d_k) : \theta \geq 0\}.$$

$x^{k+1} = x^k + \theta d_k$

(b) Poser $\delta_k = x^{k+1} - x^k = \theta d_k$.

Calculer : $\gamma_k = \nabla f(x^{k+1}) - \nabla f(x^k)$ puis :

$$H_{k+1} \leftarrow H_k + \frac{\delta_k \, \delta_k^t}{\delta_k^t \, \gamma_k} - \frac{H_k \, \gamma_k \, \gamma_k^t \, H_k}{\gamma_k^t \, H_k \, \gamma_k}.$$

$k \leftarrow k + 1$.

Tant que ($\|\delta_k\| \geq \epsilon$ et $\|\nabla f(x^k)\| \geq \epsilon$).

La validité de cet algorithme découle du théorème suivant :

Théorème 2.7. [61] *Appliqué à une fonction f quadratique strictement convexe (hessien A défini positif) l'algorithme de Davidon-Fletcher-Powell engendre des*

directions $\delta_0, \delta_1, ..., \delta_k$ *vérifiant, pour tout* k *:*

$$\delta_i^t \, A \, \delta_j = 0 \quad (0 \leq i < j \leq k). \tag{2.14}$$

$$H_{k+1} \, A \, \delta_i = \delta_i \quad (0 \leq i \leq k). \tag{2.15}$$

Le théorème précédent montre que, dans le cas quadratique, les directions $\delta_0, \delta_1, ..., \delta_k$ engendrées par l'algorithme sont mutuellement conjuguées par rapport à la matrice A de la forme quadratique. Dans ce cas, l'algorithme converge en au plus n itérations.

Remarque 2.3. *Si* H_0 *prend la valeur de la matrice unité on retrouve une méthode de type gradient conjugué.*

Enfin, pour $k = n - 1$, la relation (2.16) donne :

$$H_n \, A \, \delta_i = \delta_i \quad (i = 0, 1, ..., n-1)$$

et, comme les δ_i sont linéairement indépendants on en déduit

$$H_n \, A = I \text{ donc } H_n = A^{-1}.$$

La formule de correction (2.13) permet donc de construire, en au plus n étapes dans le cas quadratique, l'inverse du hessien de f.
La convergence globale de cette méthode n'est garantie que si l'algorithme est périodiquement réinitialisé. L'une des caractéristiques de l'algorithme DFP est que ses propriétés de convergence sont assez sensibles aux imprécisions dans les sous-problèmes de minimisation unidimensionnelle. Chaque itération nécessite donc un nombre d'évaluations de la fonction assez élevé pour obtenir la precision requise.
L'algorithme décrit dans la partie suivante, et qui se rattache encore à la famille des méthodes quasi-newtoniennes, évite cet inconvénient, tout en conservant les avantages essentiels de l'algorithme DFP.

2.3.7. L'algorithme de Broyden, Fletcher, Goldfarb, Shanno (BFGS)

Cet algorithme développé indépendamment par Broyden ([10],1970), Fletcher ([19], 1970), Goldfarb ([26], 1970) et Shanno ([83], 1969) utilise, pour construire une approximation du hessien, une formule de correction de rang 2 directement dérivée de la formule (2.13) rappelée ci-dessous :

$$H_{k+1} = H_k + \frac{\delta_k\,\delta_k^t}{\delta_k^t\,\gamma_k} - \frac{H_k\,\gamma_k\,\gamma_k^t\,H_k}{\gamma_k^t\,H_k\,\gamma_k}$$

On sait que la matrice H_{k+1} obtenue par (2.13) vérifie la relation :

$$H_{k+1}\,\gamma_k = \delta_k. \tag{2.16}$$

D'après Fletcher (1970), si l'on intervertit les rôles de δ_k et de γ_k dans (2.13) et que l'on considère la suite des matrices définies par :

$$\begin{cases} G_0 \text{ symétrique définie positive quelconque} \\ G_{k+1} = G_k + \dfrac{\gamma_k\,\gamma_k^t}{\gamma_k^t\,\delta_k} - \dfrac{G_k\,\delta_k\,\delta_k^t\,G_k}{\delta_k^t\,G_k\,\delta_k} \end{cases} \tag{2.17}$$

Les matrices obtenues vérifient la relation inverse de (2.16) :

$$G_{k+1}\,\delta_k = \gamma_k. \tag{2.18}$$

La formule (2.17) permet de construire une approximation du hessien lui-même (et non pas son inverse). En plus, si nous voulons obtenir une formule de correction pour approximer l'inverse du hessien à partir de (2.17), il suffit donc de prendre l'inverse des deux membres de (2.17). Le calcul donne :

$$[G_{k+1}]^{-1} = [G_k]^{-1} + [1 + \frac{\gamma_k^t\,[G_k]^{-1}\,\gamma_k}{\delta_k^t\,\gamma_k}]\,\frac{\delta_k\,\delta_k^t}{\delta_k^t\,\gamma_k} - \frac{\delta_k\gamma_k^t[G_k]^{-1} + [G_k]^{-1}\gamma_k\,\delta_k^t}{\delta_k^t\,\gamma_k} \tag{2.19}$$

On peut voir que l'équation (2.19) permet d'exprimer directement $[G_{k+1}]^{-1}$ en fonction de $[G_k]^{-1}$, d'où la formule de correction :

$$H_{k+1} = H_k + [1 + \frac{\gamma_k^t\,H_k\,\gamma_k}{\delta_k^t\,\gamma_k}]\,\frac{\delta_k\,\delta_k^t}{\delta_k^t\,\gamma_k} - \frac{\delta_k\gamma_k^t H_k + H_k\gamma_k\,\delta_k^t}{\delta_k^t\,\gamma_k} \tag{2.20}$$

Ainsi, l'algorithme BFGS se déduit directement de l'algorithme DFP en remplaçant la formule (2.13) par la formule (2.20). La formule (2.20) a des propriétés très analogues à celle de la formule (2.13). En particulier :

(*i*) Si $\delta_k^t \gamma_k > 0$, la définie positivité des matrices H_k est préservée.

(*ii*) Appliquée à une fonction quadratique strictement convexe (hessien A défini positif) cette formule permet d'obtenir, en au plus n itérations, l'inverse A^{-1} du hessien de f. De plus, les direction δ_i successivement engendrées par l'algorithme BFGS sont mutuellement conjuguées par rapport à A^{-1}.

L'algorithme BFGS est beaucoup moins sensible que DFP aux imprécisions dans la procédure de recherche unidimensionnelle. Ceci permet l'utilisation des méthodes d'optimisation unidimensionnelles économiques qui ne nécessitent qu'un très petit nombre d'évaluation de la fonction à chaque itération sans que la vitesse de convergence de l'algorithme soit affectée.

2.4. La Programmation quadratique avec contraintes

Définition 2.2. *Un problème de programmation quadratique sous contraintes linéaires s'écrit sous la forme suivante :*

$$\begin{cases} \min \quad f(x) = \frac{1}{2}x^t Q x + c^t x \\ s.c. \\ a_i x = b_i \qquad i \in \mathcal{L} \\ a_i x \leq b_i \qquad i \in \mathcal{I} \end{cases} \tag{2.21}$$

où Q est une matrice symétrique d'ordre n, c est un vecteur de $\mathbb{R}^n$, $a_i \in \mathbb{R}^n$ et $b_i \in \mathbb{R}$ $\forall i \in \mathcal{L} \cup \mathcal{I}$ où $\mathcal{L} = \{1, ..., m\}$ et $\mathcal{I} = \{m+1, ..., m+p\}$.
Si Q est une matrice semi-définie positive (resp. définie positive), le problème (2.21) est convexe (resp. strictement convexe), et nous montrons dans ce cas que la solution optimale, si elle existe, est globale(resp. globale et unique).

2.4.1. Conditions d'optimalité

Définition 2.3. ***(optimum local)*** *Soit $x^* \in X$. On dit que x^* est un optimum local s'il existe $\epsilon > 0$ tel que*

$$f(x^*) \leq f(x); \forall x \in B_\epsilon(x^*) \cap X \tag{2.22}$$

où

$$X = \{x \in \mathbb{R}^n : a_i x = b_i \quad i \in \mathcal{L}, a_i x \leq b_i \quad i \in \mathcal{I}\}$$

et

$$B_\epsilon(x^*) = \{x \in \mathbb{R}^n : \|x - x^*\| < \epsilon\}. \qquad \blacksquare$$

Définition 2.4. ***(directions admissibles)*** *Soit $x \in \mathbb{R}^n$ un point admissible (un point vérifiant toutes les contraintes) du problème (2.21). Une direction $d \in \mathbb{R}^n$ sera dite admissible en x s'il existe $\mu > 0$ tel que $x + sd$ soit admissible pour tout $s \in]0, \mu]$.* $\blacksquare$

Dans le cas particulier où le domaine des contraintes est convexe, déterminer une direction d admissible en x revient à déterminer un point admissible $y \neq x$ tel que $d = y - x$.

Associons à chaque contrainte $i \in \mathcal{L} \cup \mathcal{I}$ un nombre réel $\lambda_i \geq 0$ appelé multiplicateur de Lagrange. La fonction de Lagrange associé au problème quadratique (2.21) est par définition la fonction :

$$L(x, \lambda) = \frac{1}{2} x^t Q x + c^t x + \sum_{i \in \mathcal{L} \cup \mathcal{I}} \lambda_i (a_i x - b_i). \tag{2.23}$$

Notons que l'ensemble actif $\mathcal{A}(x^*)$ au point x^*, est l'ensemble des indices pour lesquels la contrainte en x^* est une égalité. Soit $J(x^*) = \{1, ..., m\} \setminus \mathcal{A}(x^*)$ l'ensemble complémentaire de $\mathcal{A}(x^*)$. Ces définitions nous permettent d'écrire les conditions nécessaires d'optimalité du premier ordre. Effectivement si x^* est une solution

optimale du problème (2.21), alors il existe un vecteur $\lambda^* \in \mathbb{R}^{m+p}$ tel que

$$\begin{cases} (a) & \nabla_x L(x^*, \lambda^*) = 0 \\ (b) & a_i x^* - b_i = 0 \qquad \forall i \in \mathcal{L} \\ (c) & a_i x^* - b_i \leq 0 \qquad \forall i \in \mathcal{I} \\ (d) & \lambda_i^* \geq 0 \qquad \forall i \in \mathcal{I} \\ (e) & \lambda_i^*(a_i x^* - b_i) = 0 \qquad \forall i \in \mathcal{I} \\ (f) & \lambda_i^* \in \mathbb{R} \quad \forall i \in \mathcal{L} \end{cases} \tag{2.24}$$

où $\nabla_x L$ représente le gradient de L par rapport à x.
Ces conditions sont aussi appelées conditions d'optimalité de *Karush-Kunh-Tucker* ou (KKT). Le vecteur λ est appelé multiplicateur de Lagrange associé aux contraintes linéaires et à la solution x^*. Dans les conditions (b) et (c), on reconnait la condition d'admissibilité des contraintes en x^*. Les conditions (d) et (e) concernent uniquement les contraintes d'inégalité. La condition (f) est la condition de signe libre des multiplicateurs et la condition (e) s'appelle la condition de complémentarité qui implique que pour $i \notin \mathcal{A}(x^*)$, $\lambda_i^* = 0$ ainsi

$$\nabla_x L(x^*, \lambda^*) = Qx^* + c + A^{J^*} \lambda_{J^*} = 0 \tag{2.25}$$

où A^{J^*} est la sous-matrice de A formée par les colonnes de A dont les indices appartenant à $J(x^*)$. Une condition suffisante pour que x^* soit un minimum local est que $Z^t Q Z$ soit définie positive, où Z est la base de l'espace généré par le rayon de la matrice jacobienne $[a_i]_{i \in \mathcal{A}(x^*)}$. Si Q est semi-définie positive donc la fonction f est convexe, par contre si Q est définie positive donc la fonction f est strictement convexe. Dans ce cas, l'optimum est unique et tout minimum local est forcément global.

En plus de la difficulté d'unicité de l'optimum dans le cas non convexe, toute construction algorithmique est confrontée à la dégénérescence, qui peut être soit la dépendance linéaire des vecteurs de la matrice jacobienne, ce qui rend le calcul des éléments de Z plus complexe, soit la faiblesse d'activité d'une contrainte, dans ce cas l'algorithme aura du mal à décider si la contrainte est active ou non, par exemple on peut avoir $\lambda_i^* = 0$ pour un certain indice dans $\mathcal{A}(x^*)$.

2.5. Les méthodes numériques de résolution pour la programmation quadratique avec contraintes

Il existe plusieurs méthodes pour la résolution des problèmes quadratiques avec contraintes linéaires. Nous citons, par exemple, la méthode du gradient projeté, le lagrangien augmenté, la méthode d'UZAWA et les variantes de la méthode de Newton, etc.

2.5.1. La méthode du gradient projeté

La méthode du gradient projeté s'inspire des méthodes de gradient décrite dans 2.3.1. Pour une méthode de gradient, 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 autorisé par les contraintes. Un des premier algorithme bâtis suivant ce principe est le gradient projeté de Rosen (1960). La méthode de gradient projeté est essentiellement intéressante dans le cas des contraintes linéaires.
Supposons donc le problème donné sous la forme :

$$\begin{cases} \min \quad f(x) = \frac{1}{2}x^tQx + c^tx \\ s.c. \\ a_ix = b_i \qquad i \in I_1 \\ a_ix \leq b_i \qquad i \in I_2 \end{cases} \tag{2.26}$$

où $\forall i \in I_2 \cup I_1$, a_i désigne le vecteur-ligne correspondant aux coefficients de la i-ème contrainte.

Remarque 2.4. *S'il y a des contraintes de positivité sur les variables, celles-ci seront considérées comme inclus dans les contraintes du type I_2.*

Supposons que l'on connait une solution x (un tel point peut toujours être obtenu par programmation linéaire).
Soit $I^0(x) = \{i/i \in I_1, a_ix = b_i\} \cup I_2$ l'ensemble des indices des contraintes saturées

au point x. On va chercher au point x une direction de déplacement d ($\|d\| = 1$) qui permette de diminuer le plus possible $f(x)$ (donc qui rend $\nabla f^t(x)\, d$ minimal), mais qui, au moins pour un petit déplacement, permette de rester dans l'ensemble des solutions réalisables.
Ceci conduit à imposer à la direction d de satisfaire les relations :

$$a_i\, d = 0 \quad \forall i \in I^0(x). \tag{2.27}$$

Notons A^0 la sous-matrice de A constituée par les lignes $i \in I^0(x)$ de A. Nous ferons l'hypothèse que A^0 est de rang plein c'est-à-dire rang $(A^0) = |I^0(x)| = q$, ce qui revient à supposer qu'il n'y a pas des dégénérescences au point x_0. On voit que l'ensemble des vecteurs d satisfaisant (2.27) est le sous-espace vectoriel S^0 de dimension $n - p$ défini par :

$$S^0 = \{y / A^0\, y = 0\}.$$

Il s'agit donc de rechercher $d \in S^0$ tel que $\nabla f^t(x)\, d$ soit minimal avec la contrainte de normalisation $\|d\| = 1$.
Le résultat suivant donne la solution explicite de ce problème, obtenue en projetant le vecteur $-\nabla f(x)$ sur le sous-espace S^0.

Théorème 2.8. [61] *Si A^0 est une matrice $q \times n$ de rang plein $q \leq n$, la solution optimale du problème :*

$$\begin{cases} \min \quad \nabla f^t(x)\, d \\ s.c. \\ A^0\, d = 0 \\ \|d\| = 1 \end{cases} \tag{2.28}$$

est $d = \bar{y} = \bar{y}/\|\bar{y}\|$, où $\bar{y}$ est la projection de $-\nabla f(x)$ sur

$$S^0 = \{y / A^0\, y = 0\}.$$

donnée par

$$\bar{y} = -P^0\, \nabla f(x) = -(I - A^{0t}[A^0\, A^{0t}]^{-1}\, A^0)\nabla f(x)$$

(P^0 est appelée la matrice de projection sur S^0).

Remarquons que si $\bar{y} \neq 0$, $\bar{y}$ est une direction de descente puisque :

$$\nabla f^t(x)\,\bar{y} = -\bar{y}^t\,(\bar{y} + \bar{z}) = -\bar{y}^t\bar{y} < 0.$$

Une fois obtenue la direction de déplacement $d = \bar{y}/\|\bar{y}\|$, on détermine le déplacement maximal autorisé par les contraintes c'est-à-dire $\alpha_{\max}$ tel que :

$$\alpha_{\max} = Max\{\alpha/\alpha \geq 0; x + \alpha\bar{y} \in X\}$$

où

$$X = \{x/x \in \mathbb{R}^n; a_i x \leq b_i(\forall i \in I_1); a_i\,x = b_i(\forall i \in I_2)\}.$$

L'itéré suivant x' est alors sélectionné comme un point minimisant $f(x + \alpha\bar{y})$ sur le segment $[0, \alpha_{\max}]$.
Au point x', on devra déterminer le nouvel ensemble I^0 des contraintes saturées et la nouvelle matrice de projection P^0 permettant de calculer la nouvelle direction de déplacement $\bar{y}' = -P^0\,\nabla f(x')$, et ainsi de suite.
L'algorithme se poursuit donc de cette façon tant que :

$$\bar{y}' = -P^0\,\nabla f(x) \neq 0$$

Voyons maintenant ce qui se passe lorsque : $\bar{y}' = -P^0\,\nabla f(x) = 0$ au point courant x.
On a dans ce cas :

$$\nabla f(x) + A^{0t}\,u = 0 \tag{2.29}$$

avec :

$$u = [A^0\,A^{0t}]^{-1}\,A^0\nabla f(x). \tag{2.30}$$

Puisque les colonnes de A^{0t} sont les gradients des contraintes saturées en x, on voit que si $u \geq 0$, (2.29) exprime les conditions de (KKT) au point courant x, ce

qui veut dire que x est un optimum local du problème.
Dans le cas où $\bar{y} = 0$, mais où le vecteur u donné par (2.30) a des composantes strictement négatives, le point courant ne vérifie pas les conditions nécessaires d'optimalité et il faut déterminer une autre direction de déplacement. Pour cela, on supprime dans I^0 l'une des contraintes i pour lesquelles $u_i < 0$. On obtient alors une nouvelle matrice A'^0 et une nouvelle matrice de projection P'^0, qui permettront de déterminer une nouvelle direction de déplacement $\bar{y}' = -P'^0 \nabla f(x)$. Il est facile de démontrer que $\bar{y}'$ ainsi obtenu vérifie nécessairement :

(i) $\bar{y}' \neq 0$;

(ii) $\bar{y}'$ est une direction de descente pour f.

Les itérations peuvent alors se poursuivre avec la nouvelle direction $\bar{y}'$.
L'algorithme du gradient projeté est donc donné comme suit :

Algorithm 3 Algorithme du gradient projeté

Initialisation : choisir un point de départ $x^0 \in X$.

Répéter (itération courant k)

$x^k \in X$ est le point courant.

Soit A^0 la matrice dont les lignes correspondent aux contraintes saturées.

Calculer la matrice de projection :

$$P^0 = I - A^{0t}[A^0\, A^{0t}]^{-1}\, A^0\,;$$

Puis faire :

$$\bar{y} \leftarrow -P^0\, \nabla f(x^k)\,;$$

$$u \leftarrow -[A^0\, A^{0t}]^{-1}\, A^0\, \nabla f(x^k)\,;$$

$$u_i \leftarrow \min_{j=1,\ldots,n} \{u_j\}$$

Si ($\bar{y} = 0$ et $u_i < 0$) **alors**

Soit A'^0 la matrice obtenue à partir de A^0 en supprimant la ligne d'indice i, P'^0 la nouvelle matrice de projection correspondante et $\bar{y'} = -P'^0\, \nabla f(x^k)$.

Faire :

$\bar{y} \leftarrow \bar{y'}\,;$

Sinon

Si ($\bar{y} = 0$ et $u_i \geq 0$) **alors**

Le point courant x^k satisfait les conditions nécessaires de (KKT).

Fin des itérations (sortie de la boucle).

Fin Si

Fin Si

Déterminer $\alpha_{\max} = Max\{\alpha / x^k + \alpha\bar{y} \in X\}$ puis x^{k+1} tel que :

$$f^{k+1} = \min_{0 \leq \alpha \leq \alpha_{\max}} \{f(x^k + \alpha\bar{y})\}$$

$k \leftarrow k + 1\,;$

Tant que (condition d'arrêt non vérifiée).

En pratique, une condition d'arrêt sera d'interrompre les itérations dès que $\|y\| \leq \epsilon$ et $u_i \geq -\epsilon$ pour une tolérance $\epsilon > 0$ choisie.
Une des caractéristiques intéressantes de la méthode du gradient projeté est : entre l'itération k et l'itération $k+1$, les ensembles de contraintes saturées $I^0(x^k)$ et $I^0(x^{k+1})$ ne diffèrent généralement que par un élément ou plus. Par conséquent, les matrices A^0 ne diffèrent que par une ligne et la nouvelle matrice de projection peut être calculée directement à partir de l'ancienne, ce qui peut réduire sensiblement les calculs nécessaires à chaque itération.

2.5.2. La méthode de Newton (par résolution des équations de Karush-Kuhn-Tucker)

Nous étudions, dans cette section, une classe de méthodes, dont le principe consiste à résoudre les équations de Karush-Kuhn-Tucker par la méthode de Newton. Ces méthodes peuvent être considérées comme des méthodes primales-duales en ce sens qu'elles opèrent simultanément dans l'espace des variables primales et dans l'espace des multiplicateurs de (KKT) (variables duales). Leurs comportements asymptotique est très intéressante puisque, sous des hypothèses habituelles de régularité des fonctions f et g_i, on obtient une vitesse de convergence superlinéaire ou quadratique.
On se restreint tout d'abord au cas où les contraintes sont des contraintes d'égalité. On cherche donc à résoudre :

$$\begin{cases} \min \quad f(x) = \frac{1}{2}x^tQx + c^tx \\ s.c. \\ g_i(x) = 0 \qquad (i = 1, ..., m) \\ x \in \mathbb{R}^n. \end{cases} \tag{2.31}$$

La recherche d'un point satisfaisant (KKT) revient à résoudre le système à $m+n$ inconnues (x, λ) et $m+n$ équations :

$$\nabla f(x) + \sum_{i=1}^{m} \lambda_i \nabla g_i(x) = 0 \tag{2.32}$$

$$g_i(x) = 0 \quad (i = 1, ..., m). \tag{2.33}$$

La méthode de Newton consiste à partir d'un point (x^k, λ^k), à linéariser (2.32) et (2.33) au voisinage de x^k et à définir (x^{k+1}, λ^{k+1}) étant la solution du système linéarisé :

$$\begin{cases} \nabla f(x^k) + \sum\limits_i \lambda_i^k \nabla g_i(x^k) + \left[\nabla^2 f(x^k) + \sum\limits_i \lambda_i^k \nabla^2 g_i(x^k)\right](x^{k+1} - x^k) + \\ \sum\limits_i \lambda_i^{k+1} \nabla^2 g_i(x^k) = 0 \\ g_i(x^k) + \nabla g_i^t(x^k)\,(x^{k+1} - x^k) = 0 \quad (i = 1, ..., m) \end{cases} \tag{2.34}$$

(Les gradients et les hessiens sont relatifs aux variables x_i pour $i = 1, ..., n$)).
En faisant apparaître la fonction de Lagrange :

$$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i \nabla g_i(x)$$

On obtient le système :

$\nabla_x^2 L(x^k, \lambda^k)$	$\nabla g_1 \cdots \nabla g_m$
∇g_1^t $\vdots$ ∇g_m^t	$\mathbf{0}_{m \times m}$

$$\times \begin{bmatrix} x^{k+1} - x^k \\ \\ \lambda^{k+1} - \lambda^k \end{bmatrix} = \begin{bmatrix} -\nabla_x L(x^k, \lambda^k) \\ \\ -g(x^k) \end{bmatrix}$$

Si on note :

$$H^k = \nabla_x^2 L(x^k, \lambda^k) = \nabla^2 f(x^k) + \sum_{i=1}^{m} \lambda_i^k \nabla^2 g_i(x^k)$$

(hessien en x de la fonction de Lagrange au point (x^k, λ^k)) et

$$J^k = [\nabla g_1(x^k), \nabla g_2(x^k), ..., \nabla g_m(x^k)]^t = \frac{\partial g}{\partial x}(x^k)$$

(jacobien de g au point x^k) le système s'écrit :

$$\begin{pmatrix} H^k & (J^k)^t \\ J^k & 0 \end{pmatrix} \times \begin{pmatrix} x^{k+1} - x^k \\ \lambda^{k+1} - \lambda^k \end{pmatrix} = \begin{pmatrix} -\nabla f(x^k) - (J^k)^t \, \lambda^k \\ -g(x^k) \end{pmatrix}.$$

Si on suppose la qualification des contraintes et l'existence d'une solution (x^*, λ^*) et l'existence d'une constante positive q telle que :

$$y^t \, \nabla_x^2 L(x^*, \lambda^*) \, y \geq q \, \|y\|^2$$

$$\forall y \text{ tel que } y^t \, \nabla g_i(x^*) = 0 \ (i = 1, ..., m),$$

Alors la méthode de Newton converge vers (x^*, λ^*) ([64], [77]).
Nous constatons que pour mettre en œuvre la méthode de Newton on a besoin à chaque étape de calculer :

1. Les gradients de f et des g_i ;
2. le hessien en x de la fonction de Lagrange $L(x, \lambda)$ (mais on n'a pas besoin des dérivées secondes de f et des g_i séparément).

D'autre part, on remarque que dans les n premières équations, le terme $(J^k)^t \, \lambda^k$ s'élimine ($(J^k)^t \, \lambda^k = 0$) [61] et il reste :

$$\begin{pmatrix} H^k & (J^k)^t \\ J^k & 0 \end{pmatrix} \times \begin{pmatrix} x^{k+1} - x^k \\ \lambda^{k+1} \end{pmatrix} = \begin{pmatrix} -\nabla f(x^k) \\ -g(x^k) \end{pmatrix}. \tag{2.35}$$

Supposons que H^k inversible, la solution explicite du système (2.35) s'écrit :

$$\begin{pmatrix} x^{k+1} - x^k \\ \lambda^{k+1} \end{pmatrix} = \begin{bmatrix} L & M \\ N & O \end{bmatrix} \times \begin{bmatrix} -\nabla f(x^k) \\ -g(x^k) \end{bmatrix}$$

Avec
$L = H^{-1} - H^{-1} J^t [J \, H^{-1} J^t]^{-1} J \, H^{-1}$
$M = H^{-1} J^t [J \, H^{-1} J^t]^{-1}$
$N = [J \, H^{-1} J^t]^{-1} J \, H^{-1}$
$O = -[J \, H^{-1} J^t]^{-1}$

La méthode de Newton présente cependant quelques inconvénients, en particulier,

la séquence (x^k, λ^k) peut ne pas converger si le point de départ est trop éloigné de la solution (x^*, λ^*). Donc c'est nécessaire de partir d'un point initial proche de l'optimum qui est la plus gênante pour construire des algorithmes généraux et fiables sur le principe de la méthode de Newton.

Remarque 2.5. *La résolution des équations de Karush-Kuhn-Tucker peut se faire par des méthodes quasi newtoniennes où on utilise à chaque étape une approximation de l'inverse de* $\begin{pmatrix} H & J^t \\ J & 0 \end{pmatrix}$*, ce qui évite la nécessité de calculer les dérivées secondes.*

2.5.3. Extension de la méthode de Newton : Méthode de Wilson

Posons $y^k = x^{k+1} - x^k$. Donc le système (2.35) s'écrit comme suit :

$$\begin{cases} H^k y^k + (J^k)^t \lambda^{k+1} = -\nabla f(x^k) \\ J^k y^k = -g(x^k). \end{cases} \tag{2.36}$$

Alors si H^k est définie-positive, y^k est solution du problème d'optimisation convexe quadratique :

$$(Q) \quad \begin{cases} \min \frac{1}{2} y^t H^k y + \nabla f^t(x^k) \\ s.c. \\ J^k y + g(x^k) = 0. \end{cases}$$

et que λ^{k+1} est le vecteur dual optimal de ce problème quadratique.
En effet, au lieu d'utiliser la formule de Newton classique, on peut donc résoudre le problème (Q) à chaque itération pour obtenir x^{k+1} et λ^{k+1} à partir de x^k et λ^k. Un avantage de cette approche (voir [96]) provient du fait qu'elle utilise des informations du second ordre (hessien) du lagrangien, elle ne tendra donc pas à produire des points stationnaires qui ne sont pas des minimums locaux.
Sous l'hypothèse que H^k sont définies positives, la méthode de Wilson possède les mêmes propriétés de convergence quadratique au voisinage de l'optimum que la

méthode de Newton.
Notons que la méthode de Wilson présente un autre avantage, qui est de se généraliser très facilement au cas des contraintes d'égalité, si on a des contraintes d'inégalité du type : $g_i(x) \leq 0$, il est facile de montrer qu'il suffit de considérer dans (Q) des contraintes du type :

$$\nabla g_i^t(x^k)\, y + g_i(x^k) \leq 0.$$

Il faut alors résoudre à chaque étape le programme quadratique :

$$(Q_0) \quad \begin{cases} \min \frac{1}{2} y^t H^k y + \nabla f^t(x^k)\, y \\ s.c. \\ J^k y + g(x^k) \leq 0. \end{cases}$$

2.5.4. Algorithmes d'Uzawa et de Arrow-Hurwicz

L'algorithme d'Uzawa (1958) utilise une méthode de gradient classique (de sous gradient dans le cas non différentiable) pour résoudre le problème dual. À chaque étape la fonction de Lagrange est minimisée. Dans ses grandes lignes, la méthode est la suivante :

Algorithm 4 Algorithme d'Uzawa

Initialisation : choisir un point de départ $\lambda^0 \geq 0$.

Répéter (itération courant k)

$x^k \in X$ est le point courant.

λ^k est un point courant.

Calculer :

$$\begin{aligned} w(\lambda^k) &= \min_{x \in S} \{f(x) + \lambda_k \, g(x)\} \\ &= f(x^k) + \lambda^k \, g(x^k) \end{aligned}$$

$g(x^k) = [g_i(x^k)]_{i \in I}$ est un sous-gradient de w au point λ^k (le gradient dans le cas différentiable). Définir λ^{k+1} par :

$$\lambda^{k+1} = Proj_{\mathbb{R}^m_+}\{\lambda^k + \rho_k \, g(x^k)\}$$

où ρ_k est le pas de déplacement à l'étape k.

$k \leftarrow k + 1$;

Tant que (condition d'arrêt non vérifiée).

Dans cet algorithme, les pas de déplacement ρ_k peuvent être choisis a priori : il s'agit alors d'une méthode de gradient à pas prédéterminé ou de sous-gradient, appliquée à la fonction duale.
La méthode de Arrow-Hurwicz (voir [4]) est très voisine de la précédente. La différence réside dans le fait qu'à chaque étape k, on ne cherche pas le minimum en x de la fonction de Lagrange $L(x, \lambda^k)$ mais on se contente d'un déplacement (suivant un pas prédéterminé) dans la direction $-\nabla_x L(x^{x-1}, \lambda^k)$. La méthode consiste ainsi à se déplacer alternativement dans l'espace des variables primales et dans l'espace des variables duales avec des pas prédéterminés.

2.6. Conclusion

Dans ce chapitre nous avons présenté les problèmes classiques de l'optimisation quadratique avec et sans contraintes et les différentes méthodes de résolution de ces problèmes. Dans cette thèse nous nous intéressons au problème quadratique binaire avec contraintes mixtes, linéaires et quadratiques. L'étude, l'analyse et la résolution de ce problème seront présentées dans les deux chapitres suivants.

Chapitre 3

La programmation D.C. pour la résolution d'un problème quadratique non convexe sous contraintes mixtes

Sommaire

3.1. Introduction

L'optimisation D.C. (Différence de deux fonctions Convexes) consiste à résoudre le problème suivant :

$$(P) \qquad \alpha = inf\,\{f(x) = g(x) - h(x) : x \in \mathbb{R}^n\}$$

où g et $h \in \Gamma_0(\mathbb{R}^n)$.

Le développement des algorithmes de sous gradients a commencé dans les années 70 par Rockafellar ([82], 1970) pour résoudre des problèmes d'optimisation non linéaires et non différentiables. À la fin des années 70, Toland ([88], [89], [90]) a mis en évidence les conditions d'optimalité locale et la dualité D.C.
Ces études ont été poursuivies dans les années 80 par plusieurs chercheurs surtout Lemarechal et Hiriat-Urruty ([39], [40], [41]) qui ont démontré l'optimalité globale des solutions dans le cas où h est polyédrale. Pendant les années 90, le développement des algorithmes de sous-gradients a pris une grande valeur algorithmique et une immense importance pratique avec l'apparition d'une nouvelle génération d'algorithmes de sous-gradients appelée DCA, développés par l'équipe d'optimisation de Grenoble, pour la résolution du problème (P). Ces algorithmes sont apparus dans les travaux de Pham Dihn [70] et par suite dans ceux de Yassine ([100], [101], [102]) et An Le Thi ([53], [54], [55]) . Les algorithmes DCA ont été appliqués avec succès à de nombreux problèmes concrets de grandes dimensions ([101], [102]) et ils ont fournis des solutions optimales globales pour certains problèmes.
Dans ce chapitre, nous proposons deux différentes décompositions D.C. pour le problème quadratique non convexe sous contraintes mixtes $(QP01)$.

3.2. Résolution du problème quadratique $(QP01)$ par la programmation DC

Nous considérons le problème quadratique non convexe suivant :

$$(QP01) \qquad Min\left\{f(x) = \frac{1}{2}x^tQx + c^tx : x \in S\right\} \tag{3.1}$$

$$S = \left\{x \in \{0,1\}^n : A'x \leq b', x^tL_jx + c_j^tx \leq \alpha_j,\ j = 1, ..., p\right\} \tag{3.2}$$

Les données du problème sont :
n est la dimension du problème (le nombre de variables), m est le nombre de contraintes linéaires et p est le nombre de contraintes quadratiques.
A' est une matrice de type (m, n), $b' \in \mathbb{R}^m$.
Q est une matrice symétrique définie positive d'ordre n.
L_j $j = 1, ..., p$ sont des matrices symétriques d'ordre n.
c et c_j $j = 1, ..., p$ sont des vecteurs de $\mathbb{R}^n$.
$\alpha_j \in \mathbb{R}; \forall j = 1, ..., p$.
L'inconnu du problème est le vecteur $x \in \{0, 1\}^n$.

Dans ce travail, nous appliquons une nouvelle approche basée sur la programmation DC et de l'algorithme (DCA) pour résoudre les programmes quadratiques binaires (BQP) avec contraintes linéaires et quadratiques qui jouent un rôle important dans l'optimisation combinatoire. En utilisant une fonction de pénalité exacte, nous reformulons (BQP) sous la forme d'un programme DC qui sera ensuite résolu par l'algorithme DCA.

Première décomposition

On considère le problème pénalisé :

$$(P_t) \qquad Min \left\{ f(x) = \tfrac{1}{2} x^t Q x + c^t x + t p(x) : x \in \bar{S} \right\}$$

où $p(x) = \sum_{i=1}^{n} x_i(1 - x_i)$, $\forall x \in \bar{S}$ et t est un nombre positif assez grand.
$\bar{S} = \left\{ x \in [0, 1]^n : A'x \leq b', x^t L_j x + c_j^t x \leq \alpha_j, j = 1, ..., p \right\}$

D'après le théorème de pénalité exacte (voir chapitre 1), on a bien que les problèmes $(QP01)$ et (P_t) sont équivalents.

(P_t) peut s'écrire sous la forme suivante :

$$(P_t) \qquad Min \left\{ f(x) = \frac{1}{2} x^t Q x + c^t x + t p(x) + \chi_{\bar{S}}(x) : x \in \mathbb{R}^n \right\} \tag{3.3}$$

$\chi_{\bar{S}}$ est la fonction indicatrice de $\bar{S}$ définie par :

$$\chi_{\bar{S}}(x) = \begin{cases} 0 & \text{si } x \in \bar{S} \\ +\infty & \text{sinon} \end{cases} \tag{3.4}$$

Donc (P_t) devient :

$$(P_t): \qquad Min\left\{f(x) = \frac{1}{2}x^tQx + \chi_{\bar{S}}(x) - (-c^tx - tp(x)) : x \in \mathbb{R}^n\right\} \tag{3.5}$$

Donc on obtient la première décomposition DC avec

$$g_1(x) = \tfrac{1}{2}x^tQx + \chi_{\bar{S}}(x) \qquad et \qquad h_1(x) = -c^tx - tp(x)$$

où g_1 et h_1 sont deux fonctions convexes.

Remarque 3.1. *C'est important de noter qu'une fonction DC possède plusieurs décompositions qui ont une importante influence sur la qualité (l'accélération de la convergence, la robustesse, l'efficacité) de l'algorithme DCA. Donc il faut choisir la meilleure décomposition qui accélère la convergence.*

Appliquer DCA au problème (3.5) revient à calculer les deux suites x^k et y^k définies par :

$$y^k \in \partial h_1(x^k),\ x^{k+1} \in \partial g_1^*(y^k), \qquad \forall k \geq 0.$$

ça revient donc à calculer les sous gradients ∂h_1 et ∂g_1^*. Comme h est différentiable alors $\partial h_1(x) = \{\nabla h_1(x)\}$. Par suite ∂h_1 est explicite et facile à calculer :

$$y^k = \nabla h_1(x^k) = -c - t\sum_{i=1}^{n}(1 - 2x_i^k)e_i \tag{3.6}$$

où $e_i = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$ est le $i^{ème}$ vecteur unitaire de $\mathbb{R}^n$.

D'autre part, pour calculer x^{k+1} on a
$x^{k+1} \in argmin\left\{g_1(x) - \langle x, y^k\rangle : x \in \mathbb{R}^n\right\}$
$x^{k+1} \in argmin\left\{\frac{1}{2}x^tQx + \chi_{\bar{S}}(x) - \langle x, y^k\rangle : x \in \mathbb{R}^n\right\}$
$x^{k+1} \in argmin\left\{\frac{1}{2}x^tQx - \langle x, y^k\rangle : x \in \bar{S}\right\}$

x^{k+1} est la solution optimale du problème quadratique convexe

$$(QPC01\_D1): \qquad Min\left\{F_1(x) = \tfrac{1}{2}x^tQx - \langle x, y^k\rangle : x \in \bar{S}\right\}$$

On décrit maintenant l'algorithme DCA1 appliqué au problème (3.5) .

Algorithme DCA1

Étape 0 : x^0 donné, $k = 0$, $t > 0$ un réel strictement positif assez grand et ε_1, ε_2 des précisions bien définies positives.
Étape 1 : On calcule $y^k = \nabla h_1(x^k) = -c - t\sum_{i=1}^{n}(1 - 2x_i^k)e_i$
Étape 2 : On détermine $x^{k+1} \in \partial g_1^*(y^k)$ en résolvant le problème quadratique convexe suivant :

$$(QPC01\_D1): \qquad Min\left\{F_1(x) = \tfrac{1}{2}x^tQx - \langle x, y^k\rangle : x \in \bar{S}\right\}$$

Étape 3 : Si $|(g_1 - h_1)(x^{k+1}) - (g_1 - h_1)(x^k)| \leq \varepsilon_1$ ou $||x^{k+1} - x^k|| \leq \varepsilon_2$;
On s'arrête, $x^* = x^{k+1}$ est solution optimale de $(QPC01\_D1)$ sinon $k = k + 1$ et on retourne à l'étape 1.

Deuxième décomposition

Considérons la fonction quadratique suivante, avec ρ un nombre réel positif donné :

$$f(x) = f_\rho(x) = \tfrac{1}{2}x^t(Q - \rho I_n)x + c^tx + \tfrac{\rho}{2}\|x\|^2$$

Or si $x \in S \iff x_i \in \{0, 1\} \iff x_i^2 = x_i$ donc $\|x\|^2 = \sum_{i=1}^{n} {x_i}^2 = \sum_{i=1}^{n} x_i = x^te$.

Par suite

$$f(x) = f_\rho(x) = \tfrac{1}{2}x^t(Q - \rho I_n)x + c^t x + \tfrac{\rho}{2}x^t e$$

où $e = (1, ..., 1)^t \in \mathbb{R}^n$ et I_n est la matrice identité d'ordre n. En effet, si on choisit $\rho \geq \lambda_n(Q)$, la plus grande valeur propre de la matrice Q, donc la fonction f_ρ devient concave.

Le problème pénalisé (P_t) peut s'écrire sous la forme suivante :

$$(P_t): \qquad Min\,\{\chi_{\bar{S}}(x) - (-f_\rho(x) - tp(x)) : x \in \mathbb{R}^n\} \tag{3.7}$$

Donc on obtient la deuxième décomposition DC avec

$$g_2(x) = \chi_{\bar{S}}(x) \qquad et \qquad h_2(x) = -f_\rho(x) - tp(x)$$

où g_2 et h_2 sont deux fonctions convexes.

Appliquer DCA au problème (3.7) revient à calculer les deux suites x^k et y^k définies par :

$$y^k \in \partial h_2(x^k), x^{k+1} \in \partial g_2^*(y^k), \qquad \forall k \geq 0.$$

ça revient donc à calculer les sous gradients ∂h_2 et ∂g_2^*. $\partial h_2(x) = \{\nabla h_2(x)\}$.
Ici on a

$$y^k = \nabla h_2(x^k) = -\nabla f_\rho(x^k) - t\nabla p(x^k) = (Q - \rho I_n)x^k - c - \frac{\rho}{2}e - t\sum_{i=1}^{n}(1 - 2x_i^k)e_i \tag{3.8}$$

où $e_i = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$ est le $i^{ème}$ vecteur unitaire de $\mathbb{R}^n$.

D'autre part, pour calculer x^{k+1} on a
$x^{k+1} \in argmin\left\{g_2(x) - \langle x, y^k\rangle : x \in \mathbb{R}^n\right\}$
$x^{k+1} \in argmin\left\{\chi_{\bar{S}}(x) - \langle x, y^k\rangle : x \in \mathbb{R}^n\right\}$
$x^{k+1} \in argmin\left\{-\langle x, y^k\rangle : x \in \bar{S}\right\}$

x^{k+1} est la solution optimale du problème convexe

$$(QPC01\_D2): \qquad Min\left\{F_2(x) = -\langle x, y^k\rangle : x \in \bar{S}\right\}$$

On décrit maintenant l'algorithme DCA2 appliqué au problème (3.7)

Algorithme DCA2

Étape 0 : x^0 donné, $k = 0$ et soit ε_1, ε_2 des précisions bien définies positives.
Étape 1 : On calcule $y^k = \nabla h_2(x^k) = (Q - \rho I_n)x^k - c - \frac{\rho}{2}e - t\sum_{i=1}^{n}(1 - 2x_i^k)e_i$
Étape 2 : On détermine $x^{k+1} \in \partial g_2^*(y^k)$ en résolvant le problème convexe suivant :

$$(QPC01\_D2): \qquad min\left\{F_2(x) = -\langle x, y^k\rangle : x \in \bar{S}\right\}$$

Étape 3 : Si : $|(g_2 - h_2)(x^{k+1}) - (g_2 - h_2)(x^k)| \leq \varepsilon_1$ ou $||x^{k+1} - x^k|| \leq \varepsilon_2$; On s'arrête, $x^* = x^{k+1}$ est solution optimale de $(QPC01\_D2)$ sinon on pose $k = k + 1$ et on retourne à l'étape 1.

3.3. Conclusion

Dans ce chapitre nous avons proposé deux différentes décompositions D.C. pour résoudre le problème quadratique (QP01) non convexe sous contraintes mixtes. Ce problème nous mène à résoudre à chaque itération un problème quadratique convexe sous contraintes mixtes.
Dans ce qui suit, nous appliquons trois différentes approches pour résoudre le problème convexe $(QPC01\_D1)$. Nous exposons dans le chapitre suivant des résultats numériques comparatives des trois approches et les deux problèmes $(QPC01\_D1)$ et $(QPC01\_D2)$.

Chapitre 4

Résolution d'un programme quadratique convexe avec contraintes mixtes

Sommaire

4.1. Introduction

Nous avons montré dans le chapitre précédent que la résolution avec l'optimisation D.C. du problème quadratique non convexe avec des variables binaires et contraintes mixtes (linéaires et quadratiques), nous mène à résoudre à chaque itération un problème quadratique convexe avec contraintes mixtes et variables réelles. Dans ce chapitre, nous cherchons à résoudre un problème quadratique convexe sous contraintes mixtes par une méthode efficace capable de déterminer la solution optimale en un temps minimum. Pour résoudre ce problème nous utilisons trois approches différentes :

1. la méthode de Newton non-lisse,
2. la programmation semi définie (SDP),
3. la méthode de points intérieurs (IPM).

En comparant ces trois approches nous trouverons la meilleure méthode pour la résolution de ce programme.

Tout d'abord nous commençons par une idée générale sur chaque méthode (définitions et propriétés). Ensuite nous présentons le programme quadratique convexe avec contraintes mixtes à résoudre, puis nous décrivons en détails les trois différentes méthodes pour la résolution d'un tel programme. Nous terminons par des simulations numériques et une étude comparative de ces trois approches et les deux problèmes ($QPC01\_D1$) et ($QPC01\_D2$) et une conclusion sur les résultats.

4.2. La méthode de Newton dans le cas multidimensionnel

Cette méthode propose la résolution du système suivant :

$$\Theta(x) = 0_n \tag{4.1}$$

où $\Theta(x) = \begin{pmatrix} \Theta_1(x_1, ..., x_n) \\ \vdots \\ \Theta_n(x_1, ..., x_n) \end{pmatrix} \in \mathbb{R}^n$ et $x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \in \mathbb{R}^n$.

Nous supposons que la fonction Θ est continûment différentiable et que nous avons une estimation actuelle $x = \begin{pmatrix} x_{1,k} \\ \vdots \\ x_{n,k} \end{pmatrix}$ de la solution $x^* = \begin{pmatrix} x_1^* \\ \vdots \\ x_n^* \end{pmatrix}$.

La série de Taylor du premier ordre de la fonction Θ autour de x_k s'écrit donc sous la forme suivante :

$$\Theta(x_k + \nu) \backsimeq \hat{\Theta}(\nu) = \Theta(x_k) + (\nabla\Theta(x_k))^t \cdot \nu$$

où $\nabla\Theta(x_k) = \begin{bmatrix} \frac{\partial\Theta_1}{\partial x_{1,k}} & \cdots & \frac{\partial\Theta_1}{\partial x_{n,k}} \\ \vdots & \ddots & \vdots \\ \frac{\partial\Theta_n}{\partial x_{1,k}} & \cdots & \frac{\partial\Theta_n}{\partial x_{n,k}} \end{bmatrix}$

Si la matrice jacobienne $\nabla\Theta(x_k)$ n'est pas singulière, la méthode de Newton permet de déterminer l'approximation suivante de la solution et cela revient à résoudre le système d'équations suivant :

$$\begin{pmatrix} x_{1,k+1} \\ \vdots \\ x_{n,k+1} \end{pmatrix} = \begin{pmatrix} x_{1,k} \\ \vdots \\ x_{n,k} \end{pmatrix} + \nu = \begin{pmatrix} x_{1,k} \\ \vdots \\ x_{n,k} \end{pmatrix} - \nabla\Theta^{-1} \begin{pmatrix} x_{1,k} \\ \vdots \\ x_{n,k} \end{pmatrix} \cdot \Theta \begin{pmatrix} x_{1,k} \\ \vdots \\ x_{n,k} \end{pmatrix}$$

4.3. La programmation semi-définie (SDP)

Dans cette section, nous rappelons les notations et les définitions usuelles en programmation semi-définie. Un programme semi-défini positif (SDP) est un programme linéaire avec des contraintes matricielles. La programmation semi-définie est l'un des problèmes de l'optimisation continue considérée comme une extension de la programmation linéaire.
Elle apparait dans de nombreaux domaines et possède plusieurs applications, citons par exemple le problème de valeurs propres, l'optimisation combinatoire, les applications en statistique, etc. Cette approche consiste à représenter les fonctions quadratiques convexes et les contraintes quadratiques convexes sous la forme d'inégalités matricielles linéaires (voir [78], [97]). Depuis l'apparition des méthodes de points intérieurs initialisées par Karmarkar en 1984 [47], de nombreuses méthodes ont été proposées pour la programmation linéaire et ont ensuite été généralisées pour la programmation semi-définie.

4.3.1. Notions et définitions

Notons $M_{m,n}$ l'espace vectoriel des matrices réelles $m \times n$, M_n l'espace des matrices carrées réelles $n \times n$ et S_n l'espace des matrices symétriques réelles $n \times n$. Un programme semi-déni, dit primal, sous forme standard s'écrit comme suit :

$$(PSDP) \qquad \begin{cases} \text{Min } \langle C, X \rangle \\ \text{s.c.} \\ \mathcal{A}X = b \\ X \succeq 0 \end{cases}$$

où $b \in \mathbb{R}^m$ et $\mathcal{A}$ est un opérateur linéaire de S_n dans $\mathbb{R}^m$ tel que

$$\mathcal{A}X = \begin{pmatrix} \langle A_1, X \rangle \\ \vdots \\ \langle A_m, X \rangle \end{pmatrix}$$

C et A_i, $i = 1, ..., m$, sont des matrices de M_n. Sans perte de généralité on peut supposer que C et les A_i sont symétriques.
Le problème dual $(DSDP)$ associé au problème primal $(PSDP)$ peut s'écrire sous la forme suivante :

$$(DSDP) \qquad \begin{cases} \text{Max } \langle b, y \rangle \\ \text{s.c.} \\ \mathcal{A}^T y + Z = C \\ y \in \mathbb{R}^m, Z \succeq 0 \end{cases}$$

où $\mathcal{A}^T$ est l'opérateur adjoint de $\mathcal{A}$ défini par $\mathcal{A}^T y = \sum_{i=1}^{m} y_i A_i$.

La trace d'une matrice $A = (a_{ij}) \in M_n$ est définie par :

$$tr(A) = \sum_{i=1}^{n} a_{ii}.$$

La trace est une fonction linéaire et pour $A = (a_{ij}) \in M_n$ elle est égale à la somme des valeurs propres de A.
L'espace vectoriel $M_{m,n}$ est isomorphe à l'espace vectoriel $\mathbb{R}^{m \times n}$. Dans cet espace vectoriel le produit scalaire entre deux éléments $A = (a_{ij})$, $B = (b_{ij})$ de $M_{m,n}$ est défini par :

$$\langle A, B \rangle = A \bullet B = Tr(B^t A) = \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} b_{ij} = Tr(A^t B). \tag{4.2}$$

La norme associée à ce produit scalaire est définie par

$$\|A\| = \sqrt{\|A\|} = \sqrt{tr(A^t A)}, \forall A \in M_n.$$

En particulier, si $(A, B) \in S_n^2$, on a

$$\langle A, B \rangle = A \bullet B = Tr(B^t A) = Tr(AB) \text{ et } \|A\| = \sqrt{\sum_{i=1}^{n} \lambda_i(A)^2}$$

Théorème 4.1. *[36, 97]* ***(Complément de Schur)*** *Soit A une matrice définie positive d'ordre p, B une matrice symétrique d'ordre q et C une matrice réelle d'ordre (p, q), alors :*

$$\begin{bmatrix} A & C \\ C^t & B \end{bmatrix} \succeq 0 \Leftrightarrow B \succeq C^t A^{-1} C$$

Proposition 4.1.

1. *Toutes les valeurs propres d'une matrice symétrique $A \in S_n$ sont réelles.*
2. *Il existe une matrice orthogonale $P \in M_n$ qui diagonalise A, autrement dit $P^t A P = \Phi_A$ où Φ_A est une matrice diagonale dont les éléments diagonaux sont les valeurs propres de A. Ainsi les valeurs propres sont les solutions du polynôme caractéristique $p_n(\lambda) = det(A - \lambda I)$.*

Pour une matrice $A \in S_n$ avec $rang(A) = k$, la décomposition spectrale de A est donnée par $A = P\Phi P^t$, où $\Phi \in S_n$ est une matrice diagonale dont les éléments diagonaux sont les valeurs propres non nulles de A sur sa diagonale principale, et $P \in M_{n,k}$ telle que $P^t P = I_k$.

En effet, $M_{m,n}$ est identifiée à l'espace Euclidien $\mathbb{R}^{m \times n}$ de dimension $m \times n$ et donc M_n est identifiée à $\mathbb{R}^{n \times n}$ de dimension n^2. À la place de $M_{m,n}$, on travaillera habituellement dans S_n qui est isomorphe à $\mathbb{R}^{t(n)}$, où $t(n) = C^2_{n+1} = \frac{n(n+1)}{2}$. Il existe un opérateur linéaire qui permet de transformer une matrice $A \in M_{m,n}$ en un vecteur de $\mathbb{R}^{m \times n}$. Cet opérateur, noté *vect* est défini comme suit :

$$vect(A) = \begin{bmatrix} A_{.,1} \\ A_{.,2} \\ \vdots \\ A_{.,n} \end{bmatrix}$$

où $A_{.,i}$ est la $i^{ème}$ colonne de la matrice A. Une relation d'ordre partiel, dite l'ordre partiel Löwner, est définie sur S_n par :

$$A, B \in S_n, A \succeq B \text{ (resp. } A \succ B) \text{ si } A - B \in S_n^+ \text{ (resp. } S_n^{++}).$$

Définition 4.1. *Soient* $x = (x_1, x_2, ..., x_n)^t \in \mathbb{R}^n$ *et* $X = (x_{ij})_{i,j=1,...,n} \in M_n$. *On définit les deux opérateurs diag et Diag par :*

- $diag(X) = (x_{11}, x_{22}, ..., x_{nn})^t \in \mathbb{R}^n$.

- $Diag(x) = \begin{pmatrix} x_1 & 0 & \cdots & 0 \\ 0 & x_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & x_n \end{pmatrix} \in M_n.$ ■

Pour représenter une contrainte quadratique non convexe $x^tAx + b^tx + c \leq 0$ avec des matrices semi-définies, on introduit préalablement une nouvelle variable matricielle $W \succcurlyeq O$ telle que $W = xx^t$. On obtient alors le théorème suivant :

Théorème 4.2.

$$W = xx^t \Longleftrightarrow \begin{cases} \begin{bmatrix} 1 & x^t \\ x & W \end{bmatrix} \succcurlyeq O \\ Tr(W) - \|x\| \leq 0. \end{cases}$$

Preuve

$W = xx^t \Longleftrightarrow W - xx^t \succeq O$ et $W - xx^t \preceq O$.

Grâce au théorème de Schur, on a $W - xx^t \succeq O \Longleftrightarrow \begin{bmatrix} 1 & x^t \\ x & W \end{bmatrix} \succeq O$. De plus, $W - xx^t \preceq O \Longrightarrow Tr(W - xx^t) = Tr(W) - Tr(xx^t) = Tr(W) - \|x\|^2 \leq 0$. Par conséquent, on a démontré que $W = xx^t \Longrightarrow \begin{bmatrix} 1 & x^t \\ x & W \end{bmatrix} \succeq O$ et $Tr(W) - \|x\|^2 \leq 0$.

Inversement, $\begin{bmatrix} 1 & x^t \\ x & W \end{bmatrix} \succeq O$ et $Tr(W) - \|x\|^2 \leq 0 \Longleftrightarrow W - xx^t \succeq O$ et $Tr(W - xx^t) \leq 0$.

L'inégalité $W - xx^t \succeq O$ signifie que toutes les valeurs propres de la matrice $W - xx^t$ sont positives. De plus, $Tr(W - xx^t) \leq 0$ signifie que la somme de toutes les valeurs propres de $W - xx^t$ est négative ou nulle. Les deux formules sont vraies en même temps si et seulement si toutes les valeurs propres de $W - xx^t$ sont nulles.

Cela prouve que $W = xx^t$. ■

Le théorème précédent donne une représentation équivalente de la contrainte non convexe $W = xx^t$ dont $\begin{bmatrix} 1 & x^t \\ x & W \end{bmatrix} \succeq O$ est une contrainte convexe et $Tr(W) - \|x\|^2 \leq 0$ une contrainte anti-convexe. À l'aide du théorème (4.2), la contrainte quadratique non convexe $x^tAx+b^tx+c \leq 0$ peut être représentée via des contraintes linéaires, des contraintes matricielles linéaires, ainsi qu'une contrainte anti-convexe comme suit :

$$x^tAx + b^tx + c \leq 0 \Longleftrightarrow \begin{cases} A \bullet W + b^tx + c \leq 0, \\ \begin{bmatrix} 1 & x^t \\ x & W \end{bmatrix} \succcurlyeq O, \\ Tr(W) - \|x\|^2 \leq 0. \end{cases}$$

4.3.2. Logiciels pour résoudre le programme semi-défini

Dans cette section, on présente les outils ("solveurs") usuels et connus utilisés dans nos travaux pour la résolution numérique du programme SDP. Ces logiciels sont souvent gratuits pour les recherches académiques et téléchargeable sur le site de l'auteur.

SDPA :
SDPA [23] est une implémentation de la méthode primal-dual du point intérieur (PD-IPM) développée en langage C/C++ et MATLAB par le groupe de Kojima. Ce logiciel consiste en diverses versions selon les besoins d'utilisation. SDPA-GMP (SDPA with arbitrary precision arithmetic) ; SDPA-QD (SDPA with pseudo quaddouble precision arithmetic) ; SDPA-DD (SDPA with pseudo double-double precision arithmetic) ; SDPA-M (SDPA with MATLAB interface) ; SDPARA (SDPA parallel version) ; SDPA-C (SDPA with the positive definite matrix Completion) ; SDPARA-C (parallel version of the SDPA-C). Pour plus de détails, vous pouvez consulter le site http ://sdpa.indsys.chuo-u.ac.jp/sdpa/. Il existe aussi une version de SDPA en ligne (online solver) sur le site http ://laqua.indsys.chuo-

u.ac.jp/portal/. Il est possible de télécharger le fichier associé au problème SDP (au format SDPA) sur ce site et d'obtenir le résultat numérique immédiatement.

DSDP :
DSDP est une implémentation de l'algorithme dual de point intérieur développée en langage C par Steven J. Benson, Ye Yinyu et al. [7]. Il est plutôt conçu pour les problèmes SDP formulés avec des matrices très creuses qui ont des structures particulières comme pour l'application en optimisation combinatoire. Une interface MATLAB, une bibliothèque et sa version parallèle sont disponibles. Pour plus de détails, voir le site http ://www.mcs.anl.gov/hs/software/DSDP/.

CSDP :
CSDP est une implémentation de PD-IPM développée en langage C par Borchers [11]. Il existe également une version de MATLAB. Ce logiciel est compatible avec Linux, Windows et Mac OS. Pour plus de détails, vous pouvez se référer au site https ://projects.coin-or.org/Csdp/.

SBMethod :
C'est une implémentation de la méthode spectrale bundle développée en langage C++ par Helmberg et al. Ce logiciel a pour but de résoudre le problème de minimisation de la valeur propre maximale d'une fonction affine matricielle. Il est possible de résoudre un problème de grande dimension. Pour plus de détails, vous pouvez vous référer au site http ://www-user.tu-chemnitz.de/ helmberg/SBmethod/.

YALMIP :
C'est un solveur sur MATLAB développé par Löfberg [99]. Ce logiciel fournit une interface pour faciliter l'utilisation des solveurs externes pour résoudre des problèmes d'optimisation (tels que le programme linéaire, quadratique, conique du second ordre, semi-défini, conique mixte en variables entières, géométrique, polynomial, multiparamétrique ...). Ce logiciel est compatible avec tous les logiciels mentionnés ci-dessus, et aussi avec beaucoup d'autres logiciels gratuits ou commerciaux, comme GLPK, CLP, MOSEK, CPLEX, Xpress, GAMS etc. Pour plus de détails, vous pouvez consulter le site http ://control.ee.ethz.ch/

joloef/wiki/pmwiki.php.

4.4. Les méthodes de points intérieurs

Les méthodes des points intérieurs sont des méthodes classiques pour résoudre les problèmes d'optimisation linéaires et non linéaires. Ces méthodes sont apparues dans les années cinquante pour résoudre des problèmes d'optimisation linéaire et s'appuient sur la fonction de barrière logarithmique définie, en 1955, par Frisch. En 1984, Karmarkar a proposé un algorithme de points intérieurs avec convergence polynomiale pour résoudre des problèmes d'optimisation linéaires.

Le premier algorithme polynomial pour la programmation quadratique convexe est l'algorithme de Kozlov, Tarasov et Khachian. C'était l'extension de l'algorithme Khachian pour la programmation linéaire [48]. La seconde est l'extension de l'algorithme de Karmarkar par Ye et Tsez en 1988 [47]. Puis ont suivi d'autres extensions de méthodes de points intérieurs avec une complexité polynomiale, telle que la méthode de Goldfarb et Liu.

4.4.1. Notions générales

Considérons le problème d'optimisation suivant :

$$(P_c) \qquad \begin{cases} \min\ f(x) \\ \text{s.c.} \\ g_i(x) \leq 0, \quad i \in I = \{1, \cdots, m\} \\ h_l(x) = 0, \quad l \in L = \{1, \cdots, p\} \\ x \in \mathbb{R}^n. \end{cases} \tag{4.3}$$

où x est le vecteur des variables à optimiser, f est la fonction objectif, $h : \mathbb{R}^n \to \mathbb{R}^p$ et $g : \mathbb{R}^n \to \mathbb{R}^m$ sont des vecteurs de contraintes que les variables doivent vérifier. On notera S_{p_c} l'ensemble des solutions réalisables de (P_c) c'est-à-dire :

$$S_{p_c} = \{x \in \mathbb{R}^n : g_i(x) \leq 0, \forall i \in I, \quad h_l(x) = 0, \forall l \in L\}.$$

Les ensembles :

$$I = \{1, \cdots, m\} \quad \text{et} \quad L = \{1, \cdots, p\}$$

représentent respectivement les indices des contraintes d'égalité et d'inégalité. On suppose de plus que $p < n$ et on note par $\mathcal{I}(x)$ l'ensemble d'indices des contraintes d'inégalités actives en x, i.e.

$$\mathcal{I}(x) = \{i : g_i(x) = 0, \quad i \in I = \{1, \cdots, m\}\}$$

Le lagrangien associé au problème (P_c) est défini par :

$$L(x, \lambda^I, \lambda^L) = f(x) + \sum_{i \in I} \lambda_i^I g_i(x) + \sum_{l \in L} \lambda_l^L h_l(x). \tag{4.4}$$

où λ^I est un m-vecteur et λ^L est un p-vecteur. Dans la suite, on notera $HL = L(x, \lambda^I, \lambda^L)$.

Théorème 4.3. *[61](Karush-Kuhn-Tucker) Soit x^* un minimum local régulier du problème (P_c). Alors, il existe les multiplicateurs $\lambda^I \in \mathbb{R}^m$ et $\lambda^L \in \mathbb{R}^p$ tels que :*

$$(KKT) \qquad \begin{cases} \nabla f(x^*) + \sum\limits_{i \in I} \lambda_i^I \nabla g_i(x^*) + \sum\limits_{l \in L} \lambda_l^L \nabla h_l(x^*) = 0 \\ (\nabla_x L(x^*, \lambda^I, \lambda^L) = 0) \\ \lambda^I \geq 0 \\ \lambda_i^I g_i(x^*) = 0, \quad i \in I = \{1, \cdots, m\} \\ g_i(x^*) \leq 0 \\ h_l(x^*) = 0, \quad l \in L = \{1, \cdots, p\}. \end{cases}$$

Dans le cas convexe (c'est-à-dire lorsque $f(x)$ et $g_i(x)$ sont des fonctions convexes et $h_l(x)$ des fonctions affines), un point x^* régulier est un minimum global pour le programme (4.3) si et seulement s'il satisfait les conditions de Karush-Kuhn-Tucker.

Définition 4.2. *On appelle ξ fonction de pénalisation définie par :*

$$\begin{aligned} \xi : \quad & \mathbb{R}_+^n \to \mathbb{R} \\ & x \to -\sum_{k=1}^{n} ln(x_k) \end{aligned} \qquad \blacksquare$$

La fonction ξ s'appelle fonction barrière logarithmique (logarithmic barrier function), elle a été utilisée pour la première fois par Frish [21] en programmation convexe.
La fonction ξ pénalise, dans le cas de la minimisation, les variables du domaine réalisable, qui s'approche de la frontière.

Par conséquent, si on combine linéairement, la fonction objectif f et la fonction ξ avec un paramètre positif, alors les frontières sont évitées systématiquement lors de la minimisation.

La combinaison donne pour $\alpha > 0$ fixé et $x > 0$:

$$x \longmapsto \Upsilon_\alpha(x) = \alpha f(x) + \xi(x).$$

Une autre combinaison possible donne pour $\eta > 0$ et $x > 0$:

$$x \longmapsto \Upsilon_\eta(x) = f(x) + \eta\xi(x).$$

La fonction ξ jouit de certaines propriétés très outils dans l'étude de la convergence des algorithmes ; elle vérifie :

$$\nabla\xi(x) = -x^{-1},$$

$$\nabla^2\xi(x) = X^{-2},$$

$$\nabla\xi(e) = -e,$$

$$\nabla^2\xi(e) = I_n.$$

4.4.2. Les méthodes primales-duales de points intérieurs

Une des nombreuses approches pour résoudre le problème (P_c) consiste à utiliser une méthode de pénalité ou de barrière logarithmique. Cette approche, qui a été étudiée inversement par Fiacco et McCormick [17], consiste à minimiser une suite de problèmes de barrière de la forme :

$$\begin{cases} \min\limits_{x\in\mathbb{R}^n} \varphi_\mu(x) = f(x) - \mu \sum\limits_{i=1}^{m} ln(-g_i(x)) \\ \text{s.c.} \\ h_l(x) = 0, \quad l \in L = \{1, \cdots, p\}, \\ x \in \widehat{S}. \end{cases} \tag{4.5}$$

Pour de valeurs décroissantes du paramètre μ_k telle que : $\lim\limits_{k\to+\infty} \mu_k = 0$. L'ensemble $\widehat{S}$ représente l'intérieur relatif de $S = \{x \in \mathbb{R}^n : g(x) \leq 0\}$. Le terme de pénalisation logarithmique dans (4.5) force les itérations à rester dans l'ensemble $\widehat{S}$. Fiacco et McCormick [17] on prouvé que si l'on limite à $\widehat{S}$, le problème de barrière admet un minimum local strict $x(\mu_k)$ et la suite $(x(\mu_k))_{k\in N}$ admet une sous-suite qui converge vers la solution locale x^* du (P_c) quand $\mu \to 0$. De plus, si les conditions de qualification des contraintes sont satisfaites en x^*, alors ce point satisfait aux conditions de Kuhn-Tucker.

En pratique, pour des valeurs des paramètres μ assez grandes,la solution exacte $x(\mu_k)$ n'a pas un grand intérêt. Donc le problème de barrière est résolu avec une precision ϵ et la solution approximative du $k^{ième}$ sous-problème de barrière notée $\widehat{x}(\mu_k)$, est utilisée comme point de départ pour la résolution du $(k+1)^{ième}$ sous problème de barrière (avec $\lim\limits_{k\to\infty} \epsilon_k = 0$).

Les conditions (KKT) associées au problème de barrière (4.5) sont :

$$\begin{cases} (1) \quad \nabla\varphi_\mu(x) + \lambda^L A^L = \nabla f(x) + \sum\limits_{i=1}^{m} \dfrac{\mu}{g_i(x)} \nabla g_i(x) + \lambda^L \nabla A^L = 0 \\ (2) \quad g(x) < 0 \\ (3) \quad h(x) = 0 \\ (4) \quad \mu \geq 0 \end{cases}$$

où $A^L = (\nabla h_1(x)^t, ..., \nabla h_l(x)^t)^t$ est la matrice jacobienne de h.

En général, il y a deux méthodes pour résoudre le système d'équations (1) et (3) à savoir la méthode primale et la méthode primale-duale.

La méthode primale résout directement le système d'équations (1) et (3) par une méthode de Newton et elle met à jour à chaque itération les variables primales x et possiblement les multiplicateurs de contraintes d'égalité λ^L. Cependant, dans l'équation (1), le terme $\nabla\varphi_\mu(x)$ a une composante comprenant $\frac{\mu}{g_i(x)}$. Ainsi, les systemes (1) et (3) ne sont pas définis pour les contraintes d'inégalité actif en x^*, c'est-à-dire, pour $g_i(x^*) = 0$ pour $i \in \mathcal{I}(x)$. En plus, le rayon de convergence de Newton tend vers 0 quand $\mu \to 0$. Par conséquent, après la mise à jour du paramètre μ, la direction primale peut ne pas être efficace [98].

La méthode primale-duale a été introduite comme une alternative à la méthode primale. Le terme $\frac{\mu}{g_i(x)}$ dans l'équation (1) consiste un estimateur du multiplicateur de Lagrange qui correspond à la $j^{ième}$ contrainte d'égalité. On définit la variable duale λ^I_μ par :

$$\lambda^i_\mu = \frac{\mu}{g_i(x)}$$

de sorte que

$$\lambda^I_\mu = \mu G(x_\mu)^{-1} e,$$

où $G(x)$ est la matrice diagonale de termes $g_i(x)$ et $e = (1, ..., 1)^t \in \mathbb{R}^m$ est le vecteur unitaire.

Avec cette définition, les conditions (KKT), (1) et (3) sont équivalentes aux condition (KKT) perturbées du problème (P') :

$$\begin{cases} (i) & \nabla f(x) + \lambda^I_\mu A^I + \lambda^L A^L = 0 \\ (ii) & h(x) = 0 \\ (iii) & G(x)\lambda^I_\mu = \mu e \end{cases}$$

où $A^I = (\nabla g_1(x)^t, ..., \nabla g_m(x)^t)^t$ est la matrice jacobienne de g.

Dans une méthode primle-duale de points intérieurs, les itérés sont construits en appliquant la méthode de Newton au système d'équations (i) et (iii). La direction de Newton appelée aussi direction primale-duale $d = (d_x, d_{\lambda^L}, d_{\lambda^I_\mu})$ est alors solution du système linéaire :

$$\begin{bmatrix} HL & A^L & A^I \\ (A^L)^t & 0 & 0 \\ A^\mu_I (A^I)^t & 0 & G(x) \end{bmatrix} \begin{bmatrix} d_x \\ d_{\lambda^L} \\ d_{\lambda^I_\mu} \end{bmatrix} = - \begin{bmatrix} \nabla L(x, \lambda^L, \lambda^I_\mu) \\ h(x) \\ G(x)\lambda^L \end{bmatrix} \tag{4.6}$$

4.5. Problème quadratique convexe avec contraintes mixtes

Revenons au problème quadratique convexe avec contraintes mixtes que nous souhaitons résoudre

$$(QPC01\_D1) : \qquad \min \left\{ F_1(x) = \frac{1}{2} x^t Q x - \langle x, y^k \rangle : x \in \bar{S} \right\} \tag{4.7}$$

$$\bar{S} = \left\{ x \in [0,1]^n : A'x \leq b', x^t L_j x + c_j^t x \leq \alpha_j, j = 1, ..., p \right\} \tag{4.8}$$

où n est la dimension du problème (le nombre de variables), m est le nombre de contraintes linéaires et p est le nombre de contraintes quadratiques.
Q est une matrice symétrique, définie positive, d'ordre n.
y^k est un vecteur donné dans $\mathbb{R}^n$.
A' est une matrice de type (m, n), $b' \in \mathbb{R}^m$.
$L_j, j = 1, ..., p$ sont des matrices symétriques semi-définies positives d'ordre n.
$c_j, j = 1, ..., p$ sont des vecteurs de $\mathbb{R}^n$.
$\alpha_j \in \mathbb{R}; \forall j = 1, ..., p$.
L'inconnu du problème est le vecteur $x \in [0,1]^n$

On intègre les contraintes $x_i \in [0,1]$ dans le contraintes linéaires $A'x \leq b'$. En effet,
$x_i \in [0,1] \Rightarrow 0 \leq x_i \leq 1 \Rightarrow x_i \leq 1$ et $-x_i \leq 0$.

Donc $I_n x \leq \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = 1_n$ et $-I_n x \leq \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} = 0_n$

d'où

$$\begin{bmatrix} I_n \\ -I_n \end{bmatrix} x \leq \begin{pmatrix} 1 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

De plus on a : $A'x \leq b'$ alors $\begin{bmatrix} A' \\ I_n \\ -I_n \end{bmatrix} x \leq \begin{pmatrix} b' \\ 1_n \\ 0_n \end{pmatrix} \in \mathbb{R}^{m+2n=r}$

Soit $A = \begin{bmatrix} A' \\ I_n \\ -I_n \end{bmatrix}$ et $b = \begin{pmatrix} b' \\ 1_n \\ 0_n \end{pmatrix}$.

Le problème devient :

$$(QPC01\_D1): \qquad \min\left\{F_1(x) = \frac{1}{2}x^tQx - \langle x, y^k\rangle : x \in D\right\}. \tag{4.9}$$

où

$$D = \left\{x \in \mathbb{R}^n : Ax \leq b, \frac{1}{2}x^tL_jx + c_j^tx \leq \alpha_j, j = 1, ..., p\right\}. \tag{4.10}$$

Dans ce qui suit, nous appliquons chaque approche (La méthode de Newton, la programmation semi définie positive et la méthode des points intérieurs) sur le problème quadratique convexe sous contraintes mixtes ($QPC01\_D1$) pour trouver la plus efficace. Puis nous présentons des résultats numérique pour les deux problèmes ($QPC01\_D1$) et ($QPC01\_D2$) avec les trois approches.

4.6. Application de l'approche de Newton non-lisse

Dans cette approche, nous résolvons les equations de *Karush-Kuhn-Tucker* (KKT) par la méthode de *Newton non-lisse*. Plusieurs chercheurs ont consacré leurs travaux aux méthodes de Newton basées sur la reformulation des conditions d'optimalité du premier ordre pour résoudre certains problèmes de mathématique en se ramenant au problème ($QPC01\_D1$) ([12], [84]). Dans cette approche les conditions d'optimalité de Karush-Kuhn-Tucker sont reformulées sous forme d'un système d'équations non linéaires. L'application de la fonction de *Fischer-Burmeister* aux conditions d'optimalité permet de remplacer les contraintes de positivité et de complémentarité par un système d'équations plus simple à résoudre.

4.6.1. La Méthode de Fischer-Newton

Un problème de complémentarité non linéaire (NCP) [33] consiste à trouver un point $x \in \mathbb{R}^n$ tel que :

$$x \geq 0, \quad F(x) \geq 0, \quad \langle x, F(x) \rangle = 0,$$

où $\langle ., . \rangle$ le produit scalaire et $F = (F_1, F_2, ..., F_n)^t$ est une fonction de $\mathbb{R}^n$ dans $\mathbb{R}^n$. Nous supposons que F est continûment différentiable. Les problèmes NCP ont attiré beaucoup l'attention en raison de ses diverses applications dans la recherche opérationnelle, l'économie et l'ingénierie.

De nombreuses méthodes ont été proposées pour résoudre le problème NCP [33]. Parmi lesquels, l'une des approches les plus populaires et les plus puissantes qui a été étudié intensivement récemment est de reformuler le problème NCP en tant que système d'équations non linéaires [60] ou comme un problème de minimisation sans contraintes ([16],[18]). Une telle fonction peut constituer un problème de minimisation sans contrainte équivalent pour le NCP est appelé une fonction de mérite. En d'autres termes, une fonction de mérite est une fonction dont les minimums coïncident avec les solutions du problème NCP original.

Définition 4.3. *(**la fonction de Fischer–Burmeister**) [79] Une fonction $\varphi : \mathbb{R}^2 \to \mathbb{R}$ est dite fonction NCP si elle satisfait*

$$\varphi(a, b) = 0 \Leftrightarrow a \geq 0,\ b \geq 0,\ ab = 0 \tag{4.11}$$

où $\varphi(a, b) = \sqrt{a^2 + b^2} - (a + b)$ est la fonction de Fischer–Burmeister.

Avec la caractérisation ci-dessus de φ, le problème NCP est équivalent au système d'équations :

$$\Phi(x) = \Phi(x, F(x)) = \begin{pmatrix} \varphi(x_1, F_1(x)) \\ \vdots \\ \varphi(x_n, F_n(x)) \end{pmatrix} = 0 \tag{4.12}$$

4.6.1.1. Jacobien généralisé de Clarke

En 1975, Clarke définit des " *gradients généralisés*" pour les fonctions localement lipschitziennes à valeurs réelles. Le but est d'écrire le comportement à l'ordre 1 de fonctions non différentiables. En 1976, Clarke étend la notion de sous-différentiel généralisé à des fonctions localement lipschitziennes à valeurs vectorielles dont nous rappelons.

Sous- différentiel de Clarke

Il est claire qu'une fonction localement lipschitzienne n'est pas nécessairement différentiable. Le théorème suivant montre que l'ensemble de non différentiabilité est un ensemble de mesure nulle.

Théorème 4.4. *(**Théorème de Rademarcher**) (Vrai en dimension finie) [94]. Soient $U \subset \mathbb{R}^n$ un ouvert et $f : U \to \mathbb{R}^m$ une fonction lipschitzienne en tout point de U, alors f est différentiable presque-partout sur $\mathbb{R}^n$ (pour la mesure de Lebesgue). Toute fonction $f : \mathbb{R}^n \to \mathbb{R}^m$ localement lipschitzienne sur $\mathbb{R}^n$ est donc différentiable partout sauf sur un ensemble Ω_{nd} (Ω_{nd} est l'ensemble des points où f n'est pas différentiable) de mesure nulle, et la fonction f est dérivable en tout point arbitrairement proche de $x \in \Omega_{nd}$.*

Définition 4.4. *(**le Jacobien généralisé**) Soit $f : U \subseteq \mathbb{R}^n \to \mathbb{R}^n$ une fonction localement lipschitzienne sur U ouvert. Le B-sous-différentiel ("B" pour Bouligand) [85] de f au point $x \in U$ est définit par :*

$$\partial_B f(x) = \left\{ H \in \mathbb{R}^{n\times n} : \exists x_k \subset \mathcal{D}_f \, et \lim_{x_k \to x} J_f(x_k) = H \right\}$$

où J_f est le Jacobien de f et $\mathcal{D}_f$ désigne l'ensemble des points de différentiabilité de f.
A partir de la définition précédente on définit le sous-différentiel de Clarke (Jacobien généralisé) de la fonction f au point x comme étant l'enveloppe convexe du B-sous-différentiel :

$$\partial f(x) = conv(\partial_B f(x)). \qquad \blacksquare$$

Alors

$$\partial f(x) = conv \left\{ H \in \mathbb{R}^{n\times n} : \exists x_k \subset \mathcal{D}_f \, et \lim_{x_k \to x} J_f(x_k) = H \right\}$$

où conv désigne l'enveloppe convexe [86].

Proposition 4.2. *([13], [14]) Soit $f : U \subset \mathbb{R}^n \to \mathbb{R}^m$ une fonction localement lipschitzienne pour tout $x \in U$*

1. *$\partial f(x)$ est non vide, compact et convexe.*
2. *$\partial f(x)$ est semi-continue supérieurement c.à.d :*
   pour tout $\epsilon > 0$, $\gamma > 0$ et y tel que $\|y - x\| \leq \gamma$ alors $\partial f(y) \subset \partial f(x) + \epsilon B_{m\times n}$.
   Où $B_{m\times n}$ la boule unitaire de $\mathbb{R}^{m\times n}$.
3. *$\partial_B f(x) \subset \partial f(x)$.*
4. *Si f est différentiable sur un voisinage de x alors :*

$$\partial_B f(x) = \partial f(x) = \{f'(x)\}.$$

5. *$\partial(-f)(x) = -\partial f(x)$*

Exemples

Exemple 1 : La norme Euclidienne [13].

Soit $e_1 : \mathbb{R}^n \to \mathbb{R}$ telle que pour tout $x \in \mathbb{R}^n$, on a $e_1(x) = \|x\| =< x, x >^{\frac{1}{2}}$.
e_1 est une fonction de classe C^∞ sur $\mathbb{R}^n \backslash \{0\}$, et on a :

$$e_1'(x) = \frac{x}{\|x\|}, \quad \forall x \in \mathbb{R}^n \backslash \{0\}.$$

Alors

$$\partial e_1(x) = \partial_B e_1(x) = \left\{\frac{x}{\|x\|}\right\}, \text{ pour tout } x \neq 0.$$

Pour $x = 0$ on a :

$$\partial_B e_1(0) = \left\{v \in \mathbb{R}^n : v = \lim_{x^k \to 0} e_1'(x^k), (x^k) \subset \mathcal{D}_{e_1} \text{et } x^k \to 0\right\}$$

$$v = \lim_{x^k \to 0} e_1'(x^k) \Rightarrow v = \lim_{x^k \to 0} \frac{x^k}{\|x^k\|}$$

$$\Rightarrow \|v\| = \lim_{x^k \to 0} \frac{\|x^k\|}{\|x^k\|} \Rightarrow \|v\| = 1.$$

Alors

$$\partial_B e_1(0) = \{v : v \in \mathbb{R}^n, \|v\| = 1\}.$$

Et comme

$$\partial e_1(0) = conv\, \partial_B e_1(0)$$

Alors on obtient que

$$\partial e_1(0) = \{v : v \in \mathbb{R}^n, \|v\| \leq 1\}.$$

Exemple 2 : La fonction de Fischer-Burmeister [13].

Soit $\varphi : \mathbb{R}^2 \to \mathbb{R}$, $\varphi(a, b) = \sqrt{a^2 + b^2} - (a + b)$ la fonction de Fischer-Burmeister.
On remarque que φ est une fonction Lipschitzienne sur $\mathbb{R}^2$ car :
φ est la différence entre la fonction $g_1(a, b) = \|(a, b)^t\|$ et la fonction linéaire

$$f(a,b) = a + b.$$

On a

$$\partial_B \varphi(a,b) = \partial_B \|(a,b)\| - f'(a,b) \text{ et } \partial\varphi(a,b) = \partial\|(a,b)\| - f'(a,b).$$

Pour tout $(a,b) \neq (0,0)$ on obtient :

$$\partial\varphi(a,b) = \partial_B\varphi(a,b) = \varphi'(a,b) = \left\{\tfrac{(a,b)}{\|(a,b)\|_2} - (1,1)\right\},$$

$$\partial\varphi(a,b) = \left\{\left(\tfrac{a}{\|(a,b)\|} - 1, \tfrac{b}{\|(a,b)\|} - 1\right)\right\}$$

et pour $(a,b) = (0,0)$ on obtient

$$\partial_B\varphi(0) = \{y^t - (1,1) : \|y\| = 1\}$$

$$\Rightarrow \partial\varphi(0) = \{y^t - (1,1) : \|y\| \leq 1\}$$

On obtient alors : $\partial\varphi(a,b) = \begin{cases} \left(\tfrac{a}{\|(a,b)\|} - 1, \tfrac{b}{\|(a,b)\|} - 1\right), & \text{si } (a,b) \neq (0,0) \\ y^t - (1,1),\ y \in \mathbb{R}^2 \text{ et } \|y\| \leq 1, & \text{si } (a,b) = (0,0) \end{cases}$

Exemple 3 : La fonction min [13].

Soit la fonction $f : \mathbb{R}^2 \to \mathbb{R}$. telle que pour tout $(a,b)^t \in \mathbb{R}^2$,

$$f(a,b) = \min(a,b)$$

La fonction f est localement Lipschitzienne sur $\mathbb{R}^2$, on peut donc calculer son Jacobien généralisé :

$$f(a,b) = \min(a,b) = \begin{cases} a & \text{si} \quad a < b \\ a & \text{si} \quad a = b \\ b & \text{si} \quad a > b \end{cases}$$

f est différentiable pour tout $(a,b) \in \mathbb{R}^2$ tel que $a \neq b$, alors on a :

$$\partial f(a,b) = \partial_B f(a,b) = \begin{cases} (1,0) & \text{si} \quad a < b \\ (0,1) & \text{si} \quad a > b \end{cases}$$

Pour $(a,b)^t \in \mathbb{R}^2$ tel que $a = b$, on calcule $\partial_B f$ au point (a,a). Soit la suite $(x^k) = (x_1^k, x_2^k)^t$ converge vers (a,a) mais $(x_1^k, x_2^k) \neq (a,a)$, alors

$$\partial_B f(a,a) = \left\{ \lim_{k\to\infty} f'(x_1^k, x_2^k) \right\}$$

Si $x_1^k < x_2^k$ pour tout k, alors $\lim\limits_{k\to\infty} f'(x_1^k, x_2^k) = (1,0)$.
Si $x_1^k > x_2^k$ pour tout k, alors $\lim\limits_{k\to\infty} f'(x_1^k, x_2^k) = (0,1)$.
Pour les autres suites (x^k) qui convergent vers $(a,a)^t$ ne satisfaisant pas $x_1^k < x_2^k$ ou $x_1^k > x_2^k$ pour tout k, la suite $f'(x_1^k, x_2^k)$ n'est pas convergente.

D'après les trois cas, on obtient que :

$$\partial_B f(a,a) = \{(1,0), (0,1)\}.$$

Et comme

$$\begin{aligned} \partial f(a,a) &= conv\, \partial_B f(a,a) \\ &= \{\lambda(1,0) + (1-\lambda)(0,1), \lambda \in [0,1]\} \end{aligned}$$

On obtient

$$\partial f(a,a) = \{(\lambda, 1-\lambda), \lambda \in [0,1]\}$$

Alors le Jacobien généralisé de la fonction f est donné par :

$$\partial \min(a,b) = \begin{cases} (1,0) & \text{si} \quad a < b \\ (\lambda, 1-\lambda), \lambda \in [0,1] & \text{si} \quad a = b \\ (0,1) & \text{si} \quad a > b \end{cases}$$

En utilisant le B-sous-gradient, l'équation de Newton généralisée est,

$$H\Delta x^k = -\Phi(x^k), \tag{4.13}$$

avec Δx^k est la direction de Newton et $H \in \partial\Phi(x^k)$ est un élément du Jacobien généralisé $\partial\Phi(x^k)$. La mise à jour de Newton est alors :

$$x^{k+1} = x^k + \tau^k \Delta x^k (\text{pour la convergence locale}) \tag{4.14}$$

où τ^k est le pas de déplacement de la $k^{ième}$ itération.
Pour la convergence globale, il faut introduire un paramètre τ^k qui minimise $g(\tau) = \Phi(x^k + \tau\delta x^k)$.

Nous passons maintenant à l'application de la méthode de Newton non-lisse au problème ($QPC01\_D1$). Pour appliquer la méthode de Newton non-lisse à ce problème, il faut tout d'abord écrire les conditions nécessaires d'optimalité de KKT du premier ordre.

En effet, une condition nécessaire pour que x soit un optimum local de ($QPC01\_D1$) est qu'il existe des nombres $\lambda_i \geq 0$ et $\mu_j \geq 0$ tels que :

$$\begin{cases} (a) \quad Qx - y^k + A^t\lambda + \sum_{j=1}^{p} \mu_j(L_j x + c_j) = 0_n \\ (b) \quad \lambda_i(Ax - b)_i = 0 \qquad i = 1, ..., r \\ (c) \quad \mu_j(\frac{1}{2}x^t L_j x + c_j^t x - \alpha_j) = 0 \qquad j = 1, ..., p \\ Ax \leq b \\ \frac{1}{2}x^t L_j x + c_j^t x \leq \alpha_j \qquad j = 1, ..., p \\ \lambda_i \geq 0 \qquad i = 1, ..., r \\ \mu_j \geq 0 \qquad j = 1, ..., p \end{cases}$$

La première condition d'optimalité (a) signifie que le gradient de la fonction Lagrangienne L par rapport à x doit être nul à la solution optimale. Les conditions d'optimalités (b) et (c), appelées conditions de complémentarité, signifient que les multiplicateurs de Lagrange sont nuls lorsque les contraintes correspondantes ne sont pas saturées.

Soit :

$$G(x,\lambda,\mu) = Qx - y^k + A^t\lambda + \sum_{j=1}^{p} \mu_j(L_jx + c_j)$$

On aura :

$$\begin{cases} G(x,\lambda,\mu) = 0_n \quad \text{où } 0_n \text{ est le vecteur nul de } \mathbb{R}^n \\ \lambda_i(b-Ax)_i = 0, \qquad \lambda_i \geq 0, \qquad (b-Ax)_i \geq 0, \qquad i = 1,....,r \\ \mu_j(\frac{1}{2}x^tL_jx + c_j^tx - \alpha_j) = 0, \qquad \mu_j \geq 0, \qquad \alpha_j - \frac{1}{2}x^tL_jx - c_j^tx \geq 0, \qquad j = 1,...,p \end{cases}$$

En appliquant la proposition (4.3) aux conditions d'optimalité, on obtient :

$$\begin{cases} G(x,\lambda,\mu) = 0_n \\ \psi_i(x,\lambda_i) = \psi(\lambda_i,(b-Ax)_i) = \lambda_i + (b-Ax)_i - \sqrt{\lambda_i^2 + (b-Ax)_i^2} = 0, \qquad i = 1,....,r \\ \Gamma_j(x,\mu_j) = \Gamma(\mu_j,\alpha_j - c_j^tx - \frac{1}{2}x^tL_jx) = \mu_j + \alpha_j - c_j^tx - \frac{1}{2}x^tL_jx - \\ \sqrt{(\alpha_j - \frac{1}{2}x^tL_jx - c_j^tx)^2 + \mu_j^2} = 0, j = 1,...,p \end{cases}$$

Soit :

$$g(x,\lambda,\mu) = \begin{pmatrix} G(x,\lambda,\mu) \\ \psi(x,\lambda) \\ \Gamma(x,\mu) \end{pmatrix} = 0_{n+r+p}$$

avec

$$\psi(x,\lambda) = \begin{pmatrix} \psi_1(x,\lambda_1) \\ \vdots \\ \psi_r(x,\lambda_r) \end{pmatrix} \qquad \text{et} \qquad \Gamma(x,\mu) = \begin{pmatrix} \Gamma_1(x,\mu_1) \\ \vdots \\ \Gamma_p(x,\mu_p) \end{pmatrix}$$

On applique la méthode de Newton au système d'équations, on trouve l'équation suivante :

$$(S) : \qquad \begin{pmatrix} x_{k+1} \\ \lambda_{k+1} \\ \mu_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ \lambda_k \\ \mu_k \end{pmatrix} - \nabla g^{-1} \begin{pmatrix} x_k \\ \lambda_k \\ \mu_k \end{pmatrix} \cdot g \begin{pmatrix} x_k \\ \lambda_k \\ \mu_k \end{pmatrix}$$

Pour résoudre une telle équation, dans le cas différentiable, il faut calculer l'inverse du gradient $\nabla g^{-1}\begin{pmatrix} x_k \\ \lambda_k \\ \mu_k \end{pmatrix}$ à chaque itération, qui pourra être coûteux en termes de temps et de mémoire. Pour remédier à ce problème, nous remplaçons le calcul de l'inverse de ∇g par la résolution du système linéaire suivant :

$$\nabla g \begin{pmatrix} x_k \\ \lambda_k \\ \mu_k \end{pmatrix} \cdot \triangle u_k = -g \begin{pmatrix} x_k \\ \lambda_k \\ \mu_k \end{pmatrix}$$

avec

$$\triangle u_k = \begin{pmatrix} \triangle x_k \\ \triangle \lambda_k \\ \triangle \mu_k \end{pmatrix} = \begin{pmatrix} x_{k+1} - x_k \\ \lambda_{k+1} - \lambda_k \\ \mu_{k+1} - \mu_k \end{pmatrix} \quad , \quad \begin{pmatrix} x_k \\ \lambda_k \\ \mu_k \end{pmatrix} \text{ est l'itéré actuel,}$$

et

$$M = \nabla g \begin{pmatrix} x_k \\ \lambda_k \\ \mu_k \end{pmatrix} = \begin{bmatrix} B & A^t & C \\ D & E & \mathbf{0}^t_{(p,r)} \\ F^t & \mathbf{0}_{(p,r)} & H \end{bmatrix}_{[n+r+p,n+r+p]}$$

où
$B = Q + \sum_{j=1}^{p} \mu_j L_j$
$C = (C_1, ..., C_p)$ avec $C_i = L_i x + c_i$ colonne de C.

D est une matrice de type $r \times n$ où chaque ligne D_i $(i = 1, ..., r)$ de D est définie par :

$$D_i = -A_i + \frac{(b - Ax)_i A_i}{\sqrt{\lambda_i^2 + (b - Ax)_i^2}}$$

avec A_i est la $i^{ème}$ colonne de A pour $i = 1, ..., r$.

E est une matrice diagonale carrée d'ordre r où

$$E_{ii} = 1 - \frac{\lambda_i}{\sqrt{\lambda_i^2 + (b - Ax)_i^2}}$$

F est une matrice de type $p \times n$ où chaque ligne F_j $(j = 1, ..., p)$ de F est définie par :

$$F_j = -c_j - L_j x + \frac{(c_j + L_j x)(\alpha_j - c_j^t x - \frac{1}{2} x^t L_j x)}{\sqrt{\mu_j^2 + (\alpha_j - c_j^t x - \frac{1}{2} x^t L_j x)^2}}$$

et H est une matrice carrée d'ordre p définie par :

$$H_{jj} = 1 - \frac{\mu_j}{\sqrt{\mu_j^2 + (\alpha_j - c_j^t x - \frac{1}{2} x^t L_j x)^2}}$$

Tandis que pour le cas non-différentiable il faut remplacer par le Jacobien généralisé de ψ et Γ (voir exemple 3) :

$$\partial\psi_i(\lambda_i, (b-Ax)_i) = \begin{cases} \left(\dfrac{\lambda_i}{\|(\lambda_i, (b-Ax)_i)\|} - 1, \dfrac{(b-Ax)_i}{\|(\lambda_i, (b-Ax)_i)\|} - 1 \right), \\ \text{si } (\lambda_i, (b-Ax)_i) \neq (0,0) \\ y^t - (1,1),\ y \in \mathbb{R}^2 \text{ et } \|y\| \leq 1, \qquad \text{si } (\lambda_i, (b-Ax)_i) = (0,0) \end{cases}$$

et

$$\partial\Gamma_j(\mu_j, \alpha_j - c_j^t x - \tfrac{1}{2} x^t L_j x) = \begin{cases} \left(\dfrac{\mu_j}{\|(\mu_j, \alpha_j - c_j^t x - \frac{1}{2} x^t L_j x)\|} - 1, \dfrac{\alpha_j - c_j^t x - \frac{1}{2} x^t L_j x}{\|(\mu_j, \alpha_j - c_j^t x - \frac{1}{2} x^t L_j x)\|} - 1 \right), \\ \text{si } (\mu_j, \alpha_j - c_j^t x - \frac{1}{2} x^t L_j x) \neq (0,0) \\ y^t - (1,1),\ y \in \mathbb{R}^2 \text{ et } \|y\| \leq 1, \quad \text{si } (\mu_j, \alpha_j - c_j^t x - \frac{1}{2} x^t L_j x) = (0,0) \end{cases}$$

En fait, nous prouvons que la matrice M est inversible,

Propriétés [58] Soit A et B deux matrices carrées d'ordre m
(a) A définie positive , B semi-définie positive $\Rightarrow$ $A + B$ définie positive.
(b) A définie positive $\Rightarrow$ A inversible et A^{-1} définie positive.
(c) A définie positive $\Rightarrow$ BAB^{-1} semi-définie positive $\forall B$ inversible.

Proposition 4.3. *[46] Étant donnée une matrice en blocs*

$$\Theta = \begin{bmatrix} A^\circ & B^\circ \\ C^\circ & D^\circ \end{bmatrix}$$

Soit D° inversible, donc le complément de Schur de D° est

$$\Sigma = A^\circ - B^\circ (D^\circ)^{-1} C^\circ.$$

Θ *est inversible si et seulement si* Σ *est inversible.*

Dans notre cas on as

$$M = \begin{bmatrix} B & A^t & C \\ D & E & \mathbf{0}_{(r,p)} \\ F^t & \mathbf{0}_{(p,r)} & H \end{bmatrix}$$

une matrice de 9 blocs avec B, E et H sont définies positives.

On peut écrire,

$$M = \begin{bmatrix} A' & C' \\ F' & H \end{bmatrix}$$

avec $A' = \begin{bmatrix} B & A^t \\ D & E \end{bmatrix}$, $C' = \begin{bmatrix} C \\ \mathbf{0}_{(r,p)} \end{bmatrix}$ et $F' = \begin{bmatrix} F^t & \mathbf{0}_{(p,r)} \end{bmatrix}$

On a H inversible puisqu'elle est définie positive et le complément de Schur de H est

$$\sigma = A' - C' H^{-1} F'.$$

Donc pour monter que la matrice M est inversible, il suffit de montrer que σ est inversible.

$$\begin{aligned} \sigma &= \begin{bmatrix} B & A^t \\ D & E \end{bmatrix} - \begin{bmatrix} C \\ \mathbf{0}_{(r,p)} \end{bmatrix} - H^{-1} \begin{bmatrix} F^t & \mathbf{0}_{(p,r)} \end{bmatrix} \\ &= \begin{bmatrix} B & A^t \\ D & E \end{bmatrix} - \begin{bmatrix} C \\ \mathbf{0}_{(r,p)} \end{bmatrix} \begin{bmatrix} H^{-1} F^t & \mathbf{0}_{(p,r)} \end{bmatrix} \\ &= \begin{bmatrix} B & A^t \\ D & E \end{bmatrix} - \begin{bmatrix} C H^{-1} F^t & \mathbf{0}_{(r,p)} \\ \mathbf{0}_{(p,r)} & \mathbf{0}_{(p,r)} \end{bmatrix} \\ &= \begin{bmatrix} B - C H^{-1} F^t & A^t \\ D & E \end{bmatrix} \end{aligned}$$

On obtient alors une matrice en blocs avec E inversible puisqu'elle est définie positive et le complément de Schur de E est

$$\gamma = B - CH^{-1}F^t - A^tE^{-1}D.$$

Par conséquent, σ est inversible si et seulement si γ est inversible. On a les matrices E et H sont définies positives donc leurs inverses sont définies positives.

$$\begin{aligned} D_i &= -A + \frac{(b-Ax)_i A}{\sqrt{\lambda_i^2 + (b-Ax)_i^2}} \\ &= A[-1 + \frac{(b-Ax)_i}{\sqrt{\lambda_i^2 + (b-Ax)_i^2}}] \end{aligned}$$

avec, $|\frac{(b-Ax)_i}{\sqrt{\lambda_i^2+(b-Ax)_i^2}}| \leq 1 \Rightarrow -1 + \frac{(b-Ax)_i}{\sqrt{\lambda_i^2+(b-Ax)_i^2}} < 0.$

Donc $D_i = \beta_i A_i$ où $\beta_i = -1 + \frac{(b-Ax)_i}{\sqrt{\lambda_i^2+(b-Ax)_i^2}} < 0$, $i = 1, ..., r$.
Et,

$$\begin{aligned} F_j &= -c_j - L_j x + \frac{(c_j + L_j x)(\alpha_j - c_j^t x - \frac{1}{2}x^t L_j x)}{\sqrt{\mu_j^2 + (\alpha_j - c_j^t x - \frac{1}{2}x^t L_j x)^2}} \\ &= (c_j + L_j x)[-1 + \frac{\alpha_j - c_j^t x - \frac{1}{2}x^t L_j x}{\sqrt{\mu_j^2 + (\alpha_j - c_j^t x - \frac{1}{2}x^t L_j x)^2}}] \end{aligned}$$

avec, $|\frac{\alpha_j - c_j^t x - \frac{1}{2}x^t L_j x}{\sqrt{\mu_j^2+(\alpha_j - c_j^t x - \frac{1}{2}x^t L_j x)^2}}| \leq 1 \Rightarrow -1 + \frac{\alpha_j - c_j^t x - \frac{1}{2}x^t L_j x}{\sqrt{\mu_j^2+(\alpha_j - c_j^t x - \frac{1}{2}x^t L_j x)^2}} < 0.$

Donc $F_j = \theta_j C_j$ où $\theta_j = -1 + \frac{\alpha_j - c_j^t x - \frac{1}{2}x^t L_j x}{\sqrt{\mu_j^2+(\alpha_j - c_j^t x - \frac{1}{2}x^t L_j x)^2}} < 0$, $j = 1, ..., p$. F_j et C_j sont respectivement les lignes de F et C.

D'où,

$$\begin{aligned} \gamma &= B - CH^{-1}F^t - A^tE^{-1}D \\ &= B - \theta CH^{-1}C^t - \beta A^tEA \\ &= B + \theta' CH^{-1}C^t + \beta' A^tEA \end{aligned}$$

où $\theta' = -\theta$ et $\beta' = -\beta$, avec $\theta = \begin{bmatrix} \theta_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \theta_p \end{bmatrix}$ et $\beta = \begin{bmatrix} \beta_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \beta_r \end{bmatrix}$.

Alors, γ est inversible puisqu'on a la somme de matrices définies positives et semi-définie positive est définie positive (c.f. propriétés).

4.7. Transformation du problème $(QPC01\_D1)$ en un programme semi-défini

Dans cette section, l'espace de travail est l'espace vectoriel $S_n(\mathbb{R})$ des matrices $n \times n$ symétriques, qui est associé au produit scalaire habituel de $\mathbb{R}^n \times \mathbb{R}^n$:

$$A \bullet B = Tr(B^t A) = \sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij} B_{ij}.$$

Où $A = (A_{ij})$ et $B = (B_{ij})$.

Une relaxation semi-finie peut être construite en remplaçant xx^t par une matrice semi-définie $\overline{X}$ dont les éléments diagonaux sont dans $[0, 1]$. Soit $\overline{X} = xx^t$. La fonction objectif $F_1(x) = \frac{1}{2}x^t Q x - \langle x, y^k \rangle$ peut s'écrire sous la forme suivante :

$$w(x) = \langle \begin{pmatrix} 0 & -\frac{1}{2}(y^k)^t \\ -\frac{1}{2}y^k & \frac{1}{2}Q \end{pmatrix}, \begin{pmatrix} 1 & x^t \\ x & \overline{X} \end{pmatrix} \rangle$$

Les contraintes quadratiques $u_j(x) = \frac{1}{2}x^t L_j x + c_j^t x - \alpha_j, j = 1, ..., p$ peuvent être écrites sous la forme suivante :

$$u_j(x) = \langle \begin{pmatrix} -\alpha_j & \frac{1}{2}c_j^t \\ \frac{1}{2}c_j & \frac{1}{2}L_j \end{pmatrix}, \begin{pmatrix} 1 & x^t \\ x & \overline{X} \end{pmatrix} \rangle$$

Par contre, les contraintes linéaires $v_i(x) = A_i x - b_i, i = 1, ..., r, r = m + 2n$ peuvent être considérer comme contraintes quadratiques avec $L_i = 0$ où $A_{i,\cdot}$ désigne la $i^{ème}$ ligne de la matrice A. Nous pouvons donc écrire les contraintes linéaires sous la forme suivante

$$v_i(x) = \langle \begin{pmatrix} -b_i & \frac{1}{2}A_i \\ \frac{1}{2}A_i^t & \mathbf{0}_{n\times n} \end{pmatrix}, \begin{pmatrix} 1 & x^t \\ x & \overline{X} \end{pmatrix} \rangle$$

Cependant, les variables d'écart sont ajoutées aux contraintes d'inégalité pour travailler uniquement avec des contraintes d'égalités.
$u_j(x) \leq 0 \Leftrightarrow u_j(x) + s_j = 0$, avec $s_j \geq 0, \quad j = 1, ..., p.$
Et $v_i(x) \leq 0 \Leftrightarrow v_i(x) + s_{i+p} = 0$, avec $s_{i+p} \geq 0, \quad i = 1, ..., r.$

Cela conduit à la relaxation semi-finie pour le problème $(QPC01\_D1)$

$$(DSDP) \begin{cases} Min\langle H, X\rangle \\ s.c. \begin{cases} \langle B_j, X\rangle = 0; \quad j = 1, ..., p \\ \langle D_i, X\rangle = 0; \quad i = 1, ..., r \\ X = \begin{pmatrix} 1 & x^t & 0 \\ x & \overline{X} & 0 \\ 0 & 0 & S \end{pmatrix} \succeq 0 \end{cases} \end{cases}$$

Avec

$$H = \begin{pmatrix} 0 & -\frac{1}{2}(y^k)^t & \mathbf{0}_{n\times 1}^t \\ -\frac{1}{2}y^k & \frac{1}{2}Q & \mathbf{0}_{n\times n}^t \\ \mathbf{0}_{n\times 1} & \mathbf{0}_{n\times n} & \mathbf{0} \end{pmatrix},$$

$$B_j = \begin{pmatrix} -\alpha_j & \frac{1}{2}c_j^t & \mathbf{0}_{p\times 1}^t \\ \frac{1}{2}c_j & \frac{1}{2}L_j & \mathbf{0}_{p\times n}^t \\ \mathbf{0}_{p\times 1} & \mathbf{0}_{p\times n} & E_j \end{pmatrix} \quad \text{pour } j = 1, ..., p.$$

où

$$E_j = \begin{cases} \mathbf{1}_j & si \quad j = 1, ..., p. \\ \mathbf{0}_{p\times p} & sinon. \end{cases}$$

$\mathbf{1}_j$ est une matrice diagonale de type $p \times p$, où tous les éléments diagonaux sont égaux à 0 sauf l'élément à la position (j, j) est égal à 1.
Et

$$D_i = \begin{pmatrix} -b_i & \frac{1}{2}A_i^t & \mathbf{0}_{r\times 1}^t \\ \frac{1}{2}A_i & \mathbf{0}_{n\times n} & \mathbf{0}_{r\times n}^t \\ \mathbf{0}_{r\times 1} & \mathbf{0}_{r\times n} & E_i \end{pmatrix} \quad \text{pour } i = 1, ..., r.$$

où

$$E_i = \begin{cases} \mathbf{1}_i & si \quad i = 1, ..., r. \\ \mathbf{0}_{r\times r} & sinon. \end{cases}$$

$\mathbf{1}_i$ est une matrice diagonale de type $r \times r$, où tous les éléments diagonaux sont égaux à 0 sauf l'élément à la position (i, i) est égal à 1,
et

$$S = \begin{pmatrix} s_1 & 0 & \cdots & 0 \\ 0 & s_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & s_{r+p} \end{pmatrix} \succeq 0$$

Cet algorithme a été programmé sous Matlab avec le logiciel DSDP qui est une implémentation de l'algorithme dual de point intérieur développée en langage C par Steven J. Benson et al. (2000).

4.8. Application de la Méthode de points intérieurs

Dans le problème $(QPC01\_D1)$ nous avons :
$x_i \in [0, 1] \Leftrightarrow 0 \leq x_i \leq 1 \Leftrightarrow x_i > 0.$

alors $I_n x \leq \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = 1_n$. On a : $A'x \leq b'$, donc $\begin{bmatrix} A' \\ I_n \end{bmatrix} x \leq \begin{pmatrix} b' \\ 1_n \end{pmatrix} \in \mathbb{R}^{m+n=r'}$

Soit $A'' = \begin{bmatrix} A' \\ I_n \end{bmatrix}$ et $b'' = \begin{pmatrix} b' \\ 1_n \end{pmatrix}$

Comme dans les méthodes de points intérieurs nous avons besoin des points intérieurs ($x > 0$), nous avons préféré laisser les contraintes de positivité à part pour respecter la nature et la structure de ces méthodes. Donc le problème $(QPC01\_D1)$ peut s'écrire sou la forme suivante :

$$(QPC01\_D1): \qquad Min \quad \left\{ F_1(x) = \frac{1}{2}x^tQx - \langle x, y^k \rangle : x \in D \right\}. \tag{4.15}$$

où

$$D' = \left\{ x \geq 0, A''x \leq b'', \frac{1}{2}x^tL_jx + c_j^tx \leq \alpha_j, j = 1, ..., p \right\}. \tag{4.16}$$

La fonction de Lagrange associée à (4.15) :

$$\begin{aligned} L(x,u,\lambda,\mu) &= \frac{1}{2}x^tQx - \langle x, y^k \rangle + u^t(-x) + \lambda^t(Ax-b) + \mu^tK, \\ u \in \mathbb{R}^n_+, \lambda \in \mathbb{R}^r_+, \mu \in \mathbb{R}^p_+ & \\ \text{où } K &= \begin{pmatrix} \frac{1}{2}x^tL_1x + c_1^tx - \alpha_1 \\ \vdots \\ \frac{1}{2}x^tL_px + c_p^tx - \alpha_p \end{pmatrix} \in \mathbb{R}^p \end{aligned}$$

La première condition d'optimalité de (4.15) est la détermination d'un point stationnaire de la fonction de Lagrange L :

$$\begin{aligned} \nabla_x L(x,u,\lambda,\mu) &= Qx - y^k - u + A^t\lambda + \nabla_x K^t\mu = 0 \\ &= Qx - y^k - u + A^t\lambda + B\mu = 0 \qquad (4.17) \\ \text{où } B &= \nabla_x K^t = \nabla_x \begin{pmatrix} \frac{1}{2}x^tL_1x + c_1^tx - \alpha_1 & ... & \frac{1}{2}x^tL_px + c_p^tx - \alpha_p \end{pmatrix} \\ &= \begin{pmatrix} L_1x + c_1 & ... & L_px + c_p \end{pmatrix} \in \mathbb{R}^{n\times p} \end{aligned}$$

On considère le problème suivant :

$$\min_{x \geq 0} \quad \{L(x,\lambda,\mu) : (x,\lambda,\mu) \in D^*\} \tag{4.18}$$

où

$$D^* = \left\{ (x,\lambda,\mu) : Qx - y^k + A^tv + Bw \geq 0, x \geq 0, \lambda \in \mathbb{R}^r_+, \mu \in \mathbb{R}^p_+ \right\} \tag{4.19}$$

Soit x^* la solution optimale de (4.18) .

On peut écrire Le lagrangien l de (4.18) comme suivant :

$$\begin{aligned} l(x^*,\lambda,\mu) &= L(x^*,\lambda,\mu) + (x^*)^t\left(-Qx^* + y^k - A^t\lambda - B\mu\right) \\ &= \frac{1}{2}x^{*t}Qx^* - \langle x^*, y^k\rangle + \lambda^t\left(Ax^* - b\right) + \mu^t K \\ &- (x^*)^t Qx^* + (x^*)^t y^k - (x^*)^t A^t\lambda - (x^*)^t B\mu \\ &= -\frac{1}{2}x^{*t}Qx^* - \lambda^t b + \mu^t K - (x^*)^t B\mu \\ &= -\frac{1}{2}x^{*t}Qx^* - b^t\lambda - \mu^t\begin{pmatrix}\frac{1}{2}x^{*t}L_1x^* + \alpha_1\\ \vdots \\ \frac{1}{2}x^{*t}L_px^* + \alpha_p\end{pmatrix} \end{aligned}$$

Alors le problème dual associé à $(QPC01\_D1)$ peut s'écrire sous la forme suivante :

$$(QDC01\_D1): \qquad Max_{(\lambda,\mu)}\left\{f(\lambda,\mu) = l(x^*,\lambda,\mu) : (x^*,\lambda,\mu) \in D^*\right\} \quad (4.20)$$

Considérons la fonction barrière logarithmique $-\theta\sum_{q=1}^{k}\ln x_q$ pour remplacer la contrainte d'inégalité $x \geq 0$ dans le problème primal. Ensuite, nous combinons linéairement la fonction F et la fonction barrière logarithmique avec un paramètre positif θ dit le paramètre de perturbation, donc nous pouvons écrire :

$$\phi_B(x,\theta) := (\tfrac{1}{2}x^tQx - \langle x, y^k\rangle) - \theta\sum_{q=1}^{k}\ln x_q,$$

où $x_q > 0\ \forall q = 1,...,k$. La fonction ϕ_B est strictement convexe dont la dérivée du premier et second ordre sont :
$\nabla\phi_B(x,\theta) = (Qx - y^k) - \theta X^{-1}e$ et $\nabla^2\phi_B(x,\theta) = Q - \theta X^{-2}$

On peut facilement vérifier que, selon les hypothèses faites, que la fonction barrière logarithmique a un minimum unique. Rappelons les conditions d'optimalité nécessaires et suffisantes du premier ordre pour $(QPC01\_D1)$ de la formule (4.15)

$$\begin{cases} Qx - y^k + A^t\lambda + \sum_{j=1}^{p} \mu_j(L_jx + c_j^t) = 0_n \\ \lambda^t(Ax - b) = \sum_{j=1}^{p} \mu_j(\frac{1}{2}x^tL_jx + c_j^tx - \alpha_j) = 0 \\ -Ax + b \geq 0 \\ \frac{1}{2}x^tL_jx + c_j^tx - \alpha_j \leq 0 \qquad j = 1, ..., p \\ \lambda \geq 0 \\ \mu \geq 0 \end{cases} \tag{4.21}$$

Soient $z = b - Ax \geq 0$ et $w = \begin{pmatrix} w_1 \\ \vdots \\ w_p \end{pmatrix}$ où $w_j = \alpha_j - \frac{1}{2}x^tL_jx - c_j^tx$

Nous pouvons alors écrire :

$$\begin{cases} Qx - y^k + A^t\lambda + \sum_{j=1}^{p} \mu_j(L_jx + c_j^t) = 0_n \\ z \geq 0 \\ w \geq 0 \\ \lambda^t z = \mu^t w = 0 \\ \lambda \geq 0 \\ \mu \geq 0 \end{cases} \tag{4.22}$$

En perturbant $\lambda^t z = 0$ et $\mu^t w = 0$, on obtient le système suivant

$$\begin{cases} Qx - y^k + A^t\lambda + \sum_{j=1}^{p} \mu_j(L_jx + c_j^t) = 0_n \\ z \geq 0 \\ w \geq 0 \\ \lambda z = \theta e \quad et \quad \mu w = \theta e \\ \lambda \geq 0 \\ \mu \geq 0 \end{cases} \tag{4.23}$$

Où $e = (1, ..., 1)^t \in \mathbb{R}^{m+n}$ ou $\mathbb{R}^p$. Ce système d'équations possède une solution $(x(\theta), \lambda(\theta), \mu(\theta), z(\theta), w(\theta))$, pour tout $\theta > 0$. Le point correspondant

est appelé "θ-center". Une famille de ces points détermine une courbe continue $\{(x(\theta), \lambda(\theta), \mu(\theta), z(\theta), w(\theta)), \theta > 0\}$ appelée la trajectoire centrale primale-duale ou chemin central, pour toutes les valeurs positives de θ.

Les trois premières équations de (4.23) sont les conditions de faisabilité duale et primale. La quatrième et la cinquième équations sont des conditions de complémentarité perturbées, le paramètre θ associé à la fonction de barrière logarithmique contrôle le niveau de perturbation. La convergence vers la solution optimale est forcée en tendant le paramètre de barrière θ vers zéro.

L'approche des méthodes de points intérieurs détermine la solution approximative du système non linéaire par la méthode de Newton. Les quatre premières équations forment un système carré. Réécrivons le système (4.23) comme :

$$V_\theta(x, \lambda, \mu, z, w) = \begin{bmatrix} Qx - y^k + A^t\lambda + \sum_{j=1}^{p} \mu_j(L_j x + c_j^t) \\ -Ax - z + b \\ w_j + \frac{1}{2}x^t L_j x - \alpha_j + c_j^t x \\ \lambda z - \theta e \\ \mu w - \theta e \end{bmatrix} = 0_{2p+2m+3n}$$

La méthode de Newton forme une approximation linéaire du système non linéaire et itère selon :

$$(S') : \quad \begin{pmatrix} x_{k+1} \\ \lambda_{k+1} \\ \mu_{k+1} \\ z_{k+1} \\ w_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ \lambda_k \\ \mu_k \\ z_k \\ w_k \end{pmatrix} + \theta_k \begin{pmatrix} \triangle x_k \\ \triangle \lambda_k \\ \triangle \mu_k \\ \triangle z_k \\ \triangle w_k \end{pmatrix}$$

Où $\begin{pmatrix} \triangle x_k \\ \triangle \lambda_k \\ \triangle \mu_k \\ \triangle z_k \\ \triangle w_k \end{pmatrix}$ est une solution du système linéaire :

$$J(x_k, \lambda_k, \mu_k, z_k, w_k) \begin{pmatrix} \triangle x_k \\ \triangle \lambda_k \\ \triangle \mu_k \\ \triangle z_k \\ \triangle w_k \end{pmatrix} = -V_\theta(x_k, \lambda_k, \mu_k, z_k, w_k). \tag{4.24}$$

Où

$$J(x_k, \lambda_k, \mu_k, z_k, w_k) = \begin{bmatrix} \bar{B} & A^t & \bar{F}^t & \mathbf{0}_{n\times r'} & \mathbf{0}_{n\times p} \\ A & \mathbf{0}_{r'\times r'} & \mathbf{0}_{r'\times p} & I_{r'} & \mathbf{0}_{r'\times p} \\ \bar{F} & \mathbf{0}_{p\times r'} & \mathbf{0}_{p\times p} & \mathbf{0}_{p\times r'} & I_p \\ \mathbf{0}_{r'\times n} & Z & \mathbf{0}_{r'\times p} & \Lambda & \mathbf{0}_{r'\times p} \\ \mathbf{0}_{p\times n} & \mathbf{0}_{p\times r'} & W & \mathbf{0}_{p\times r'} & \Xi \end{bmatrix}_{[3n+2m+2p, 3n+2m+2p]}$$

est la matrice Jacobienne du système (4.24), avec

- $\bar{B} = Q + \sum_{j=1}^{p} \mu_j L_j$
- $\bar{F}$ est la matrice de type (p, n), dont les lignes sont formées par $L_j x + c_j$ pour $j = 1, ..., p$.
- Z (resp. Λ) est la matrice diagonale d'ordre r', dont les éléments diagonaux sont les z_i (resp. les λ_i) pour $i = 1, ..., r'$ et $r' = m + n$.
- W (resp. Ξ) est la matrice diagonale d'ordre p, dont les éléments diagonaux sont les w_j (resp. les μ_j) pour $j = 1, ..., p$.

Puisque toutes les variables doivent être strictement réalisables, il faut introduire dans la relation de Newton un paramètre θ_k^* pour s'assurer que toutes les variables

soient strictement positives. Donc la relation sera modifiée comme suit :

$$\begin{pmatrix} x_{k+1} \\ \lambda_{k+1} \\ \mu_{k+1} \\ z_{k+1} \\ w_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ \lambda_k \\ \mu_k \\ z_k \\ w_k \end{pmatrix} + \theta_k^* \begin{pmatrix} \triangle x_k \\ \triangle \lambda_k \\ \triangle \mu_k \\ \triangle z_k \\ \triangle w_k \end{pmatrix} \tag{4.25}$$

Donc

- $x_{k+1} = x_k + \theta_k^* \triangle x_k > 0 \Leftrightarrow \theta_k^* < min\left\{\frac{-x_{ki}}{\triangle x_{ki}} : \triangle x_{ki} < 0\right\} = \theta_{xk}^*$
- $\lambda_{k+1} = \lambda_k + \theta_k^* \triangle \lambda_k > 0 \Leftrightarrow \theta_k^* < min\left\{\frac{-\lambda_{ki}}{\triangle \lambda_{ki}} : \triangle \lambda_{ki} < 0\right\} = \theta_{\lambda k}^*$
- $\mu_{k+1} = \mu_k + \theta_k^* \triangle \mu_k > 0 \Leftrightarrow \theta_k^* < min\left\{\frac{-\mu_{ki}}{\triangle \mu_{ki}} : \triangle \mu_{ki} < 0\right\} = \theta_{\mu k}^*$
- $z_{k+1} = z_k + \theta_k^* \triangle z_k > 0 \Rightarrow \theta_k^* < min\left\{\frac{-z_{ki}}{\triangle z_{ki}} : \triangle z_{ki} < 0\right\} = \theta_{zk}^*$
- $w_{k+1} = w_k + \theta_k^* \triangle w_k > 0 \Leftrightarrow \theta_k^* < min\left\{\frac{-w_{ki}}{\triangle w_{ki}} : \triangle w_{k_i} < 0\right\} = \theta_{wk}^*$

Soit

$$\theta_k^* = min\left\{\theta_{xk}^*, \theta_{\lambda k}^*, \theta_{\mu k}^*, \theta_{zk}^*, \theta_{wk}^*\right\}. \tag{4.26}$$

Algorithm 5 Algorithme de Points intérieurs

Initialisation : Choisir $(x_0, \lambda_0, \mu_0, z_0, w_0)$ quelconque ; $k = 0$, $\epsilon > 0$ une précision donnée.

1.

(a) On calcule

$$J_k = J(x_k, \lambda_k, \mu_k, z_k, w_k)$$

(b) On résout

$$J_k \cdot \begin{pmatrix} \triangle x_k \\ \triangle \lambda_k \\ \triangle \mu_k \\ \triangle z_k \\ \triangle w_k \end{pmatrix} = -V_\theta(x_k, \lambda_k, \mu_k, z_k, w_k).$$

(c) On calcule θ_k^* suivant la formule (4.26).

2.

Si $\| \left(\triangle x_k,\ \triangle \lambda_k,\ \triangle \mu_k,\ \triangle z_k,\ \triangle w_k \right)^t \| \leq \epsilon$ **alors**

$$\text{on s'arrête et } \begin{pmatrix} x^* \\ \lambda^* \\ \mu^* \\ z^* \\ w^* \end{pmatrix} = \begin{pmatrix} x_{k+1} \\ \lambda_{k+1} \\ \mu_{k+1} \\ z_{k+1} \\ w_{k+1} \end{pmatrix}.$$

Sinon On retourne à l'étape 1.

$k \leftarrow k + 1$.

Fin Si

4.9. Résultats Numériques

Dan cette section nous présentons les résultats numériques comparatives entre les trois approches appliquées aux problèmes ($QPC01\_D1$) et ($QPC01\_D2$) afin de voir la meilleure approche avec un temps de calcul minimal.
Tout les algorithmes ont été codés dans MATLAB 7.12.0. Les résultats de calcul de la méthode de Newton et la méthode de points intérieurs ont été obtenus en implémentant les programmes sous Windows 10 sur un portable PC 2.80GH, tandis que l'algorithme SDP a été implémenté sous Windows XP.

Rappelons que :

- n est la dimension du problème (nombres de variables).
- m est le nombre des contraintes linéaires.
- p est le nombre des contraintes quadratiques.

Dans ce qui suit, on note $QPCD1\_n\_m\_p$ les problèmes testés pour ($QPC01\_D1$) et $QPCD2\_n\_m\_p$ les problèmes testés pour ($QPC01\_D2$). Le temps est exprimé en secondes.

Dans les tableaux 4.1, 4.2, 4.3 et 4.4, nous affichons les solutions optimales pour chaque méthode avec le temps d'exécution. Les meilleures solutions optimales et les meilleurs temps d'exécution sont marqués en gras. Dans le tableau 4.5 les meilleurs temps d'exécution sont marqués en gras.

problème	Newton	Temps	SDP	Temps	IPM	Temps
$QPCD1\_6\_3\_2$	**-0.026**	**0.14**	-0.022	0.73	-0.025	0.51
$QPCD1\_10\_6\_4$	-0.021	**0.085**	-0.019	6.57	**-0.022**	4.33
$QPCD1\_10\_8\_3$	**-0.0059**	**0.014**	-0.0056	3.18	-0.0058	2.51
$QPCD1\_11\_5\_3$	**-0.024**	**0.091**	-0.022	7.79	-0.023	4.18
$QPCD1\_11\_7\_2$	-0.031	**0.14**	**-0.033**	6.84	-0.029	3.27
$QPCD1\_20\_10\_2$	**-0.0068**	**0.19**	-0.0066	12.24	**-0.0068**	4.57
$QPCD1\_20\_20\_1$	-0.0064	**0.17**	-0.0065	35.29	**-0.0067**	30.94
$QPCD1\_50\_40\_3$	**-3.31**e^{-3}	**0.41**	**-3.31**e^{-3}	42.2	-3.30e^{-3}	34.75
$QPCD1\_50\_60\_3$	**-1.78**e^{-3}	**0.14**	-1.76e^{-3}	65.21	**-1.78**e^{-3}	21.64
$QPCD1\_50\_90\_4$	**-0.0032**	**0.24**	-0.0031	85.24	**-0.0032**	43.81
$QPCD1\_100\_30\_2$	**-0.0025**	**2.01**	-0.0024	66.29	-0.0022	44.48
$QPCD1\_100\_60\_15$	**-2.75**e^{-3}	**2.76**	-2.74e^{-3}	82.82	-2.732e^{-3}	53.42
$QPCD1\_100\_70\_18$	**-2.52**e^{-3}	**5.01**	-2.49e^{-3}	88.72	-2.51e^{-3}	55.12

Table 4.1. *Résultats numériques comparatifs de* ($QPC01\_D1$)

problème	Newton	Temps	SDP	Temps	IPM	Temps
$QPCD1\_200\_30\_4$	-1.246e^{-3}	**14.48**	**-1.25**e^{-3}	98.13	-1.243e^{-3}	84.11
$QPCD1\_200\_60\_5$	-1.18e^{-3}	**20.42**	-1.18e^{-3}	102.75	**-1.19**e^{-3}	55.13
$QPCD1\_200\_70\_4$	**-1.568**e^{-3}	**14.82**	-1.564e^{-3}	109.73	-1.54e^{-3}	61.37
$QPCD1\_300\_80\_4$	**-7.988**e^{-4}	**121.72**	-7.985e^{-4}	227.64	-7.97e^{-4}	178.76
$QPCD1\_300\_30\_4$	-8.55e^{-4}	**86.16**	-8.51e^{-4}	398.51	**-8.554**e^{-4}	268.46
$QPCD1\_500\_60\_5$	**-5.47**e^{-4}	**722.41**	-5.466e^{-4}	995.86	**-5.47**e^{-4}	864.12
$QPCD1\_500\_70\_10$	**-5.47**e^{-4}	**1246.92**	-5.44e^{-4}	1925.11	-5.46e^{-4}	1302.38
$QPCD1\_600\_70\_4$	**-4.35**e^{-4}	1531.77	-4.32e^{-4}	2601.57	-4.34e^{-4}	**1520.35**
$QPCD1\_600\_50\_4$	**-4.35**e^{-4}	**1605.67**	-4.34e^{-4}	2260.15	-4.34e^{-4}	1855.52
$QPCD1\_700\_50\_3$	**-3.45**e^{-4}	**2898.66**	-3.42e^{-4}	5852.65	-3.41e^{-4}	4069.64
$QPCD1\_700\_80\_4$	-3.538e^{-4}	**3434.14**	-3.53e^{-4}	9698.22	**-3.55**e^{-4}	4852.25

Table 4.2. *Résultats numériques comparatifs de* ($QPC01\_D1$)

problème	Newton	Temps	SDP	Temps	IPM	Temps
$QPCD2\_6\_3\_2$	-0.026	**0.29**	-0.026	0.84	**-0.027**	0.24
$QPCD2\_10\_6\_4$	**-0.022**	**0.032**	-0.021	5.25	**-0.022**	3.02
$QPCD2\_10\_8\_3$	**-0.0065**	**0.61**	-0.0061	2.48	-0.0062	1.88
$QPCD2\_11\_5\_3$	**-0.0238**	**0.047**	-0.0233	3.54	-0.0236	2.11
$QPCD2\_11\_7\_2$	-0.031	**0.031**	-0.031	2.69	**-0.032**	1.54
$QPCD2\_20\_10\_2$	**-0.0067**	**0.066**	-0.0062	7.98	-0.0066	4.74
$QPCD2\_20\_20\_1$	-0.0064	**2.73**	**-0.00734**	21.51	-0.00636	16.24
$QPCD2\_50\_40\_3$	$-3.32e^{-3}$	**2.41**	$-3.31e^{-3}$	32.12	$\mathbf{-3.325}e^{-3}$	36.46
$QPCD2\_50\_60\_3$	$\mathbf{-1.77}e^{-3}$	**1.29**	$-1.76e^{-3}$	66.12	$-1.76e^{-3}$	19.71
$QPCD2\_50\_90\_4$	**-0.0033**	**7.18**	-0.0032	75.52	**-0.0033**	52.96
$QPCD2\_100\_30\_2$	**-0.0025**	**9.11**	-0.0024	82.42	0.00245	53.25
$QPCD2\_100\_60\_15$	$\mathbf{-2.756}e^{-3}$	**8.79**	$-2.754e^{-3}$	81.87	$-2.75e^{-3}$	68.74
$QPCD2\_100\_70\_18$	$\mathbf{-2.531}e^{-3}$	**9.08**	$-2.529e^{-3}$	96.58	$-2.53e^{-3}$	85.71

Table 4.3. *Résultats numériques comparatifs de* ($QPC01\_D2$)

problème	Newton	Temps	SDP	Temps	IPM	Temps
$QPCD2\_200\_30\_4$	$\mathbf{-1.251}e^{-3}$	**11.54**	$-1.249e^{-3}$	84.33	$-1.246e^{-3}$	72.97
$QPCD2\_200\_60\_5$	$-1.185e^{-3}$	**25.85**	$-1.184e^{-3}$	92.22	$\mathbf{-1.187}e^{-3}$	83.41
$QPCD2\_200\_70\_4$	$\mathbf{-1.571}e^{-3}$	**18.94**	$-1.569e^{-3}$	82.21	$-1.568e^{-3}$	77.34
$QPCD2\_300\_80\_4$	$\mathbf{-7.88}e^{-4}$	**142.14**	$-7.84e^{-4}$	203.75	$-7.87e^{-4}$	184.67
$QPCD2\_300\_30\_4$	$\mathbf{-8.57}e^{-4}$	**72.11**	$-8.561e^{-4}$	256.15	$-8.563e^{-4}$	224.84
$QPCD2\_500\_60\_5$	$\mathbf{-5.488}e^{-4}$	**847.91**	$-5.484e^{-4}$	924.74	$-5.481e^{-4}$	894.27
$QPCD2\_500\_70\_10$	$\mathbf{-5.475}e^{-4}$	**1529.14**	$-5.473e^{-4}$	2081.44	$-5.473e^{-4}$	1741.74
$QPCD2\_600\_70\_4$	$-4.34e^{-4}$	**1248.47**	$-4.31e^{-4}$	2587.61	$\mathbf{-4.38}e^{-4}$	1745.12
$QPCD2\_600\_50\_4$	$\mathbf{-4.358}e^{-4}$	**1621.47**	$-4.33e^{-4}$	2199.42	$-4.33e^{-4}$	1861.74
$QPCD2\_700\_50\_3$	$-3.433e^{-4}$	**3028.57**	$-3.432e^{-4}$	5141.17	$\mathbf{-3.437}e^{-4}$	3511.34
$QPCD2\_700\_80\_4$	$\mathbf{-3.551}e^{-4}$	**3511.51**	$-3.448e^{-4}$	8898.13	$-3.542e^{-4}$	4821.98

Table 4.4. *Résultats numériques comparatifs de* ($QPC01\_D2$)

$(QPC01\_D1)$	$(QPC01\_D2)$
0.14	0.29
0.085	**0.032**
0.014	0.61
0.091	**0.047**
0.14	**0.031**
0.19	**0.066**
0.17	2.73
0.41	2.41
0.14	1.29
0.24	7.18
2.01	9.11
2.76	8.79
5.01	9.08
14.48	**11.54**
20.42	25.85
14.82	18.94
121.72	142.14
86.16	**72.11**
722.41	847.91
1246.92	1529.14
1520.35	**1248.47**
1605.67	1621.47
2898.66	3028.57
3434.14	3511.51

Table 4.5. *comparaisons du temps de calcul de* $(QPC01\_D1)$ *et* $(QPC01\_D2)$

D'après les tableaux 4.1, 4.2, 4.3, on constate que la méthode de Newton donne les meilleures solutions optimales avec un temps de calcul efficace. Notons que pour $n > 700$, on n'a pas pu obtenir de solutions (temps de calcul très grand). Nous remarquons d'après le tableau 4.5, que le temps de calcul des instances du problème ($QPC01\_D1$) est légèrement meilleur que celui du problème ($QPC01\_D2$). En comparant les problèmes ($QPC01\_D1$) et ($QPC01\_D2$), on trouve que les deux problèmes ont les mêmes valeurs objectifs optimales mais le temps de calcul de ($QPC01\_D1$) est légèrement meilleur.

4.10. Conclusion

Dans ce chapitre nous avons appliqué trois différentes approches pour résoudre le problème convexe ($QPC01\_D1$). Nous avons montré que la matrice du système linéaire de la méthode de Newton non-lisse est toujours inversible ce qui garantie la convergence de cette méthode indépendamment du choix du point initial. Les résultats numériques comparatifs ont montré la supériorité de la méthode de Newton non-lisse pour fournir rapidement une solution optimale des problèmes ($QPC01\_D1$) et ($QPC01\_D2$).

Conclusion

Dans cette thèse nous nous sommes intéressés à la résolution d'un programme quadratique non convexe ($PQ01$) avec des contraintes mixtes (linéaires et quadratiques) et des variables booléennes 0-1 s'écrivant sous la forme suivante :

$$(QP01) \qquad Min\left\{f(x) = \tfrac{1}{2}x^tQx + y^tx : x \in S\right\}$$

où

$$S = \left\{x \in \{0,1\}^n : A'x \leq b', x^tL_jx + c_j^tx \leq \alpha_j,\ j = 1, ..., p\right\}$$

L'intérêt de résoudre ce problème est double : du point de vue académique, c'est un problème important qui possède beaucoup de propriétés intéressantes mais reste un problème difficile à résoudre à cause de l'intégrité de ses variables (variables bivalentes). Du point pratique, plusieurs problèmes concrets dans plusieurs domaines tels que l'économie, la technologie de l'information, la logistique et la finance peuvent se formuler sous forme d'un cas particulier de ce problème quadratique. Les approches de l'optimisation combinatoire sont lourdes à appliquer : d'abord, les méthodes exactes sont incapables de résoudre que des instances de petites dimensions et sont inapplicables lorsque la dimension du problème devient raisonnable. Les méthodes approchées (heuristiques et méta-heuristiques) sont lourdes à mettre en œuvre et en plus elles ne garantissent pas l'optimalité de la solution obtenue. Nous avons proposé dans cette thèse l'approche de l'optimisation D.C. et plus précisément l'algorithme DCA. Cet algorithme a déjà prouvé son efficacité pour résoudre plusieurs problèmes d'optimisation non convexe de grandes dimensions avec peu d'itérations et un coût faible, en tant que nombre d'opérations arithmétiques, pour chaque itération et en fournissant des optima globaux dans la

plupart des cas. Nous avons montré que la résolution de ce problème non convexe par l'algorithme DCA est équivalente à résoudre, à chaque itération, un problème quadratique convexe $(QPC01\_D1)$:

$$(QPC01\_D1): \qquad Min\left\{F_1(x) = \tfrac{1}{2}x^tQx - \langle x, y^k\rangle : x \in \bar{S}\right\}$$

$$\bar{S} = \left\{x \in [0,1]^n : A'x \leq b', x^tL_jx + c_j^tx \leq \alpha_j, j = 1, ..., p\right\}$$

Comme nous avons une série de problèmes convexes de type $(QPC01\_D1)$ à résoudre alors il faut trouver une méthode efficace, robuste, performante et surtout rapide pour trouver une solution optimale. Nous avons développé trois approches différentes pour résoudre ce problème :

1. Nous avons montré, grâce à l'application de la proposition de Fischer-Burmeister, que les conditions d'optimalité du problème sont équivalentes à la résolution un système d'équations non linéaires qui peut-être résolu par la méthode de Newton non-lisse. Nous avons montré que la matrice du système linéaire de la méthode de Newton non-lisse est toujours inversible ce qui garantie la convergence de cette méthode indépendamment du choix du point initial.

2. Nous avons mis le programme quadratique sous forme d'un problème semi-défini positif dont les propriétés théoriques de résolution sont connues dans la littérature.

3. Nous avons appliqué une méthode de points intérieurs de type « trajectoire centrale » pour résoudre ce programme quadratique.

Les résultats numériques ont montré la supériorité de la première approche (la méthode de Newton non-lisse) pour fournir rapidement une solution optimale des problèmes $(QPC01\_D1)$ et $(QPC01\_D2)$. Nous avons développé deux compositions DC pour résoudre le problème initial $(QP01)$ qui ont ramené la résolution du problème quadratique non convexe $(QP01)$ à la résolution d'un problème quadratique convexe à chaque itération. Les résultats numériques ont montré que

le nombre d'itérations pour résoudre le problème ($QPC01\_D1$) est toujours faible ce qui conduit à une résolution efficace du problème initial ($QP01$) en temps de calcul raisonnable.

Notre perspective est de résoudre le problème avec des méthodes approchées de l'optimisation combinatoire (méthodes heuristiques, méta-heuristique, etc.) pour comparer l'efficacité de notre approche en termes de qualité de la solution optimale obtenue et le temps de calcul.

Bibliographie

[1] Adams W.P., Dearing P.M., On the equivalence between roof duality and Lagrangian duality for unconstrained $0-1$ quadratic programming problems. Discrete Appl. Math. 48 (1), pp. 1–20, (1994).

[2] Adams W.P., Sherali H.D., A tight linearization and an algorithm for 0-1 quadratic programming problems. Manag. Sci 32(10), pp. 1274–1290, (1986).

[3] Aleksandrov A.D., On surfaces represented as the difference of convex functions. Izvestiya Akad. Nauk Kazah. SSR. 60, Ser. Mat. Meh. 3, (1949).

[4] Arrow K.J., Hurwicz L., Uzawa H., Studies in linear and nonlinear programming, Stanford University Press, Stanford, USA, (1958).

[5] Atteia M., Elqortobi A., Quasi-convex duality. In A. Auslender et al. (ed.), Optimization and Optimal Control, Proc. Conference Oberwolfach March 1980, Lecture notes in Control and Inform. Sci. 30, pp. 3-8, SpringerVerlag, Berlin, (1981).

[6] Billionnet A., Optimisation quadratique en $0-1$. Optimisation combinatoire, concepts fondamentaux, pp. 191-234, Chapitre de livre Levoisier, (2005).

[7] Benson S.J., Ye Y.Y., Zhang X., Solving Large-Scale Sparse Semidefinite Programs for Combinatorial Optimization, SIAM Journal on Optimization, Vol. 10(2), pp. 443-461, (2000).

[8] Billionnet A., Elloumi S., Using a mixed integer quadratic programming solver for unconstrained quadratic $0-1$ problem. Math. Program. 109 (1, Ser.A), pp. 55–68, (2007).

[9] Boros E., Hammer P.L., Pseudo-boolean optimization. Technical report TR : 2001-33, DIMACS, (2001).

[10] Broyden C. G., The convergence of a class of double-rank minimization algorithms 2 : the new algorithm, Journal Institute of Math. and its Appl. 6, pp. 222-231, (1970).

[11] Borchers B., J. G. Young J.G., Implementation of a primal-Cdual method for SDP on a shared memory parallel architecture. Computational Optimization and Applications, Vol. 37(3), pp. 355-369, (2007).

[12] Burke J., Xu S., A non-interior predictor-corrector path-following algorithm for LCP, in Reformulation : Nonsmooth, Piecewise Smooth and and Smoothing Methods Edited by M. Fukushima, and L. Qi, Kluwer Academic publishers, Boston, pp.45-64, (1999).

[13] Buchholzer H., The Semismooth Newton Method for the Solution of Reactive Transport Problems Including Mineral Precipitation Dissolution Reactions, université de Wurzburg, 2011.

[14] Clarke H., Generalized Gradients of Lipschitz Functionals, Advances in Mathematics 40, pp. 52-67, 1981.

[15] Ellaia R., Contribution à l'analyse et l'optimisation de différences de fonctions convexes. Thèse de Doctorat troisième cycle, Université Paul Sabatier, Toulouse, (1984).

[16] Facchinei F., Soares J., A new merit function for nonlinear complementarity problems and a related algorithm. SIAM Journal on Optimization, Vol. 7, pp. 225–247, (1997).

[17] Fiacco A.V., McCormick G.P., Nonlinear programming : Sequential unconstrained minimization techniques. Republished by SIAM, John Wiley and sons, New York, (1990).

[18] Fischer A., A special Newton-type optimization methods. Optimization, Vol. 24, pp. 269–284, (1992).

[19] Fletcher R., A new approch to variable metric algorithms, The Computer Journal, Vol. 13, n° 3, pp. 317-322, (1970).

[20] Fletcher R., Practicam methods of optimization, New York, john Wiley & sons edition, (1991).

[21] Frisch K.R., The logarithmic potential method of convex programming, Technical report, Memorandum, (1955).

[22] Frank M., Wolfe P., An algorithm for quadratic programming. Naval Research Logistics Quarterly, vol. 3, pp. 95–110, (1956).

[23] Fujisawa K., Kojima M., Nakata K., SDPA (SemiDefinite Programming Algorithm) - user's manual - version 6.20. Research Report B-359, Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, Tokyo, Japan, January 2005, revised May, (2005). Available at http ://sdpa.indsys.chuo-u.ac.jp/sdpa/download.html

[24] Gilbert. J. Ch., Optimisation Différentiable, Théorie et Algorithmes . Syllabus de cours à l'ENSTA, Paris, France, (2008).

[25] Glover F., Kochenberger G.A., Alidaee B., Adaptative memory tabu search for binary quadratic programs. Manag. Sci. 44, pp. 336–345, (1998).

[26] Goldfarb D., A Family of variable matrix methods derived by variational means, Mathematics of Computation, Vol. 24, pp. 23-26, (1970).

[27] Gondran M., Minoux M., Graphes et algorithmes, pp. 250-251. Eyrolles, Paris (1995).

[28] Hammer P., Hansen P., Simeone B., Roof duality complementation and persistency in quadratic 0-1 optimization. Mathematical Programming, Vol. 28, pp. 121-155, (1984).

[29] Hammer P. Hansen P., Jaumard B., and Mathon V., Constrained nonlinear 0-1 programming. ORSA Journal on Computing, Vol. 5, pp. 97-119, (1993).

[30] Hammer, P.L., Plant location-A pseudo-Boolean approach. Isr. J. Technol.6, pp. 330–332, (1968).

[31] Hammer, P.L., Rudeanu, S., Boolean Methods in Operations Research. Springer, New York, (1968).

[32] Hartman P., On functions representable as a difference of convex functions. Pacific J. Math. 9, pp. 707-713, (1959).

[33] Harker P.T., Pang J.S., Finite dimensional variational inequality and nonlinear complementarity problem : A survey of theory, algorithms and applications. Mathematical Programming, Vol. 48, pp. 161–220, (1990).

[34] Helmberg C., Semidefinite Programming for Combinatorial Optimization. Habilitationsschrift, TU Berlin, (2000).

[35] Helmberg C., Rendl F., Weismantel R., A Semidefinite Programming Approach to the Quadratic Knapsack Problem, J. of Combinatorial Optimization, Vol. 4, No. 2, pp. 197-215, (2000).

[36] Helmberg C., Rendl F., Solving quadratic 0-1 problem by semidefinite programs and cutting planes. Math. Program. 82 (3), pp. 291–315, (1998).

[37] Henrion D., Lasserre J.B., Lefberg J., GloptiPoly 3, moments, optimization and semidefinite programming, Version 3.4, 30 september, (2008).

[38] Hestenes M.R., Multipiplier and gradient methods. Journal of optimization theory and application, pp. 303-320, (1969).

[39] Hiriart-Urruty. J. B., Subdifferential calculuc in convex analysus and optimization, Pitman, pp. 43-92, (1982).

[40] Hiriart-Urruty. J. B., Generalized differentiability, duality and optimization for problems dealing with differences of convex functions. Lecture Notes in Economics and Mathematical System 256, Springer-Verlag, pp. 37–70, (1986).

[41] Hiriart-Urruty J.B., Lemarechal C., Convex Analysis and Minimization Algorithms, Springer, Berlin, (1993).

[42] Horst R., Thoai N.V., DC Programming, Overview. Journal of Optimization Theory and Applications, Vol.103, pp. 1–43, (1999).

[43] Horst R., Pardalos P.M., Thoai N.V., Introduction to Global Optimization - Second Edition. Kluwer Academic Publishers, Netherlands, (2000).

[44] Hoang T., Convex Analysis And Global Optimization. Kluwer Academic Publishers, (1998).

[45] Jha S., Pardalos P.M., Graph separation techniques for quadratic 0˘1 programming. Comput. Math. Appl. 21 (6/7), pp. 107–113, (1991).

[46] John A. Gubner, Block matrix fotmulas, Department of electrical and computer engineering, University of Wisconsin-Madison, (2015).

[47] Karmarkar N., A new polynomial time algorithm for linear programming. Combinatorica, Vol. 4, pp. 373-395, (1984).

[48] Khachian L.G., A polynomial algorithm in linear programming, doklady akademiia nauk SSSR 244, 1093-1096. Translated into English in soviet mathematics doklady, pp. 191-194, (1979).

[49] Laughunn, D.J., Quadratic binary programming. Oper. Res, Vol.14, pp. 454–461, (1970).

[50] Le Thi H.A., Analyse numérique des algorithmes de l'optimisation d.c. approches locales et globales. Code et simulations numériques en grande dimension. Applications, Thèse de Doctorat de l'Université de Rouen, (1994).

[51] Le Thi H.A., Pham Dinh T., Solving a class of linearly constrained indefinite quadratic problems by DC Algorithms. Journal of Global Optimization, Vol. 11, pp. 253-285, (1997).

[52] Le Thi H.A., Pham Dinh T., Le Dung M. : Exact penalty in d.c. programming. Vietnam Journal of Mathematics, Vol. 27(2), pp. 169-178, (1999).

[53] Le Thi H.A., Pham Dinh T., Huynh Van N., Exact penalization and error bounds for inequality systems of concave functions and of nonconvex quadratic functions. Technical report, National Institute for Applied Sciences - Rouen, France, (2008).

[54] Le Thi, H.A., Pham Dinh, T., A continuous approach for large-scale constrained quadratic zero-one programming. (In Honor of Professor ELSTER, Founder of the Journal Optimization) Optimization 45 (3), pp. 1–28, (2001).

[55] Le Thi H.A., An efficient algorithm for globally minimizing a quadratic function under convex quadratic constraints. Mathematical Programming, Ser. A, Vol.87(3), pp. 401-426, (2000).

[56] Le Thi H.A., Moeini M., Pham Dinh T., Portfolio Selection under Downside Risk Measures and Cardinality Constraints based on DC Programming and DCA. Computational Management Science, Issue 4, (2009).

[57] Le Thi H.A., Huynh Van N., Pham Dinh T., Convergence Analysis of DC Algorithm for DC programming with subanalytic data. Technical report, National Institute for Applied Sciences - Rouen, France, (2009).

[58] Lutkepohl H., Handbook of Matrices, Humboldt-Universitat zu Berlin, Germany, pp. 113-169, (1996).

[59] Luenberger D.G., Linear and Nonlinear Programming. Second edition, pp. 366-380. Springer (2003).

[60] Mangasarian O.L., Equivalence of the complementarity problem to a system of nonlinear equations. SIAM Journal on Applied Mathematics, Vol. 31, pp. 89–92, (1976).

[61] Minoux M., Programmation mathématique, théorie et algorithmes, tome 1, pp. 112-123, (1983).

[62] Muu L.D., Phong T.Q., Pham Dinh T. : Decomposition methods for solving a class of nonconvex programming problems dealing with bilinear and quadratic function. Compuational Optimization and Applications 4, pp. 203-216, (1995).

[63] Niu Y., Programmation DC et DCA en optimisation Combinatoire et Optimisation Polynomiale via les Techniques de SDP, thèse de doctorat de l'institut national des sciences appliquées de Rouen, (2010).

[64] Ortega J.M., Rheinboldt W.C., Iterative solution of nonlinear equations in several variables, Academic Press, New York, (1973).

[65] Pardalos P.M., Rodgers G.P., Computational aspects of a branch and bound algorithm for quadratic zero-one programming. Computing 45, pp. 131–144, (1990).

[66] Penot. J.P., Duality for anticonvex programs. Journal of Global Optimization archive, Vol.19, pp. 163–182, (2001).

[67] Pham Dinh T., Contribution à la théorie de normes et ses applications à l'analyse numérique. Thèse de Doctorat d'Etat Es Science, Université Joseph Fourier-Grenoble, (1981).

[68] Pham Dinh. T., Algorithmes de calcul d'une forme quadratique sur la boule unité de la norme maximum. Numerische Mathematik, Vol.45 : pp. 377–440, (1985).

[69] Pham Dinh T., Algorithms for solving a class of non convex optimization problems. Methods of subgradients. Fermat days 85. Mathematics for Optimization, J.B. Hiriart Urruty (ed.), Elsevier Science Publishers, B.V. North-Holland, (1986).

[70] Pham Dinh T., Duality in D.C. (difference of convex functions) optimization. Subgradient methods. Trends in Mathematical Optimization, K.H. Hoffmann et al. (ed.), International Series of Numer Math. 84, Birkhauser, (1988).

[71] Pham Dinh T., Le Thi H.A., Stabilité de la dualité lagrangienne en optimisation d.c. (différence de deux fonctions convexes). C.R. Acad. Paris, t.318, Série I, (1994).

[72] Pham Dinh T., Algorithmes de calcul du maximum des formes quadratiques sur la boule unité de la norme du maximum, Séminaire d'analyse numérique, Grenoble, No 247, (1976).

[73] Pham Dinh T., Le Thi H.A., DC Programming. Theory, Algorithms, Applications, The State of the Art. First International Whorkshop on Global Constrained Optimization and Constraint Satisfaction, Nice, October 2-4, (2002).

[74] Pham Dinh T., Le Thi H.A., The DC programming and DCA Revisited with DC Models of Real World Nonconvex Optimization Problems. Annals of Operations Research, Vol.133, pp. 23-46, (2005).

[75] Pham Dinh T., Nguyen Canh N., Le Thi H.A., DC Programming and DCA for Globally Solving the Value-At-Risk, to appear in Computational Management Science Issue 4, (2009).

[76] Powell M.J.D., recent advances in unconstrained optimization, Mathematical Programming, pp. 26-57, (1971).

[77] Powell M.J.D., Algorithms for nonlinear constraints that use Lagrangian functions, Mathematical Programming, Vol 14, pp. 224-248, (1978).

[78] Poljak S., Rendl F., Wolkowicz H., A recipe for semidefinite relaxation for (01)-quadratic programming.J. Global Optim. 7, pp. 51–73, (1995).

[79] Qi L., Sun D., A Survey of Some Nonsmooth Equations and Smoothing Newton Methods, School of Mathematics, The University of New South Wales Sydney 2052, Australia, (1998).

[80] Qi L.,Jiang H., A new nonsmooth equations approach to nonlinear complementarity problems, SIAM J. CONTROL OPTIM. c 1997 Society for Industrial and Applied Mathematics Vol. 35, No. 1, pp. 178–193, (1997).

[81] Robert M. Freund, Introduction to Semidefinite Programming (SDP). Massachusetts Institute of Technology, (2004).

[82] Rockafellar R.T., Convex Analysis. Princeton University Press, N.J., (1970).

[83] Shanno D. F., Conditioning of quasi-Newton methods of function minimization, Mathematics of Computation 24, pp. 641-656, (1970).

[84] Sun D., A regularization Newton method for solving nonlinear complementarity problems, Appl. Math. Optim., Vol.36, pp. 315-339, (1999).

[85] Sun D., Han J., Newton and quasi-Newton methods for a class of nonsmooth equations and related problems, SIAM Journal of Optimization, pp. 463-480, 1997.

[86] Sun D., Han J., Newton and quasi-Newton methods for a class of nonsmooth equations and related problems, SIAM Journal of Optimization, pp. 463-480, 1997.

[87] Todd M.J., Semidefinite optimization. Acta Numerica 10, pp. 515-560, (2001).

[88] Toland J.F, Duality in nonconvex optimization Journal of Mathematical Analysis and Applications, Vol.66, pp. 399-415, (1978).

[89] Toland J.F., On Subdifferential Calculus and Duality in Nonconvex Optimization. Bull. Soc. Math. France, Mémoire 60, pp. 177–183, (1979).

[90] Toland J.F., A duality principle for non convex optimization and calculus of variations, Arch. Rational. Mech. Analysis, Vol. 71, (1979).

[91] Tuan H.D., Apkarian P., Nakashima Y. : A new Lagrangian dual global optimization algorithm for solving bilinear matrix inequalities. International Journal of Robust and Nonlinear Control, Vol. 10, pp. 561-578, (2000).

[92] Tuy H., Global Optimization, Deterministic Approaches 2nd revised edition, SpringerVerlag, Berlin, (1993).

[93] Tuy H., DC Optimisation, Theory, Methods and Algorithms, Handbook of Global Optimisation Horst and Pardalos eds, Klaver Academic Publishers, pp. 149–216, (1995).

[94] Ulbrich M., Nonsmooth Newton-like Methods for Variational Inequalities and Constrained Optimization Problems in Function Spaces, Université technique de Munchen, 2002.

[95] Vandenberghe L., Boyd S., Semidefinite Programming. SIAM Review 38, pp. 49-95, (1996).

[96] Wilson R.B., A simplicial method for convex programming, ph. D.. Thesis, Harverd University, Cambridge, Mass, (1963).

[97] Wolkowicz H., Saigal, R., Vandenberghe, L., Handbook of Semidefinite Programming-Theory, Algorithms,and Application. Kluwer, Dordrecht (2000).

[98] Wright M.H., Why a pure primal Newton barrier stepmay be infeasible, SIAM, Journal on Optimization, Vol. 41(1), pp. 25-55,(1995).

[99] YALMIP, A Toolbox for Modeling and Optimization in MATLAB, Löfberg J. In Proceedings of the CACSD Conference, Taipei, Taiwan, (2004). http ://control.ee.ethz.ch/ joloef/wiki/pmwiki.php

[100] Yassine A., Sub-gradient algorithms for solving multidimensional analysis problems od dissimilarity data, Journal of Applied Mathematics and computer science, Vol. 7, pp. 521-543, (1997).

[101] Yassine A., Méthode de région de confiance et optimisation D.C. Théorie, algorithmes et applications, HDR-Université Henri Poincaré, Nancy I, (1998).

[102] Yassine A., Études adaptatives et comparatives de certains algorithmes en optimisation. Implémentations effectives et applications, thèse de doctorat de l'Université Joseph Fourrier, Grenoble I, (1989).