APPROCHES DE POINTS INTERIEURS ET DE LA PROGRAMMATION DC EN OPTIMISATION NON CONVEXE. CODES ET SIMULATIONS NUMERIQUES INDUSTRIELLES

François Akoa

THÈSE

en vue de l'obtention du titre de

DOCTEUR DE L'INSTITUT NATIONAL
DES SCIENCES APPLIQUÉES DE ROUEN

(arrêté ministériel du 30 Mars 1992)

Spécialité Mathématiques Appliquées

présentée par

François Bertrand AKOA

Titre de la thèse :

APPROCHES DE POINTS INTÉRIEURS ET DE LA PROGRAMMATION DC EN OPTIMISATION NON CONVEXE.
CODES ET SIMULATIONS NUMÉRIQUES INDUSTRIELLES.

Date de soutenance : 27 Janvier 2005

Composition du Jury :

Président	D. Weichert	*Professeur, Directeur de l'INSA de Rouen*
Rapporteurs	M. Le Dung	*Professeur, Institut de Mathématiques de Hanoï(Vietnam)*
	A. Yassine	*Professeur, Université du Havre*
Examinateurs	S. Canu	*Professeur, INSA de Rouen*
	A. Hachemi	*Docteur-Ingénieur, IAM-RWTH Aachen(Allemagne)*
	H. A. Le Thi	*Professeur, Université de Metz, Directrice de Thèse*
	T. Pham Dinh	*Professeur, INSA de Rouen, Directeur de Thèse*

Thèse préparée au laboratoire de mathématiques de l'Institut National des Sciences Appliquées de Rouen

À François-Gabriel
que ceci te serve d'exemple et te pousse à aller plus loin.
À Moué
toi qui dirigeas mes premiers pas et me couvris de tant d'amour.

Remerciements

Cette thèse a été préparée au sein du Laboratoire LMI de l'Institut National des Sciences Appliquées de Rouen, sous la co-direction des Professeurs LE Thi Hoai An, Directrice de l'Équipe Algorithmique et Optimisation, LITA, de l'Université de Metz et Pham Dinh Tao, Directeur de l'Équipe Modélisation, Optimization et Recherche Opérationnelle.

De ces Maîtres, je garderai le dynamisme et la disponibilité. Ils m'ont encouragé et donné des conseils sur le plan scientifique et sur le plan humain. Leur sympathie, leur compréhension et leur aide permanente m'ont permis de mener à bien ce travail tout en exerçant une activité professionnelle prenante.

Je remercie Monsieur Dieter Weichert Directeur de l'INSA de Rouen, qui me fait l'honneur de présider ce jury. Il est à l'origine du projet de collaboration MOM «Modeling and Optimization in Mechanics»(entre the Institut of General Mechanics(IAM), RWTH d'Aix-la-Chapelle et notre Équipe Modélisation Optimisation et Recherche Opérationnelle). Il a suivi régulièrement et avec beaucoup d'intérêt l'évolution des traveaux menés en commun.

J'adresse mes vifs remerciements à Messieurs les Professeurs Le Dung Muu (Directeur de l'Équipe Optimisation et Recherche Opérationnelle de l'Institut de Mathématiques de Hanoi au Vietnam) et Adnan Yassine(Directeur du Laboratoire de Mathématiques Appliquées de l'Université du Havre) pour avoir acceptés de juger ce travail et d'en être rapporteurs.

Je remercie le Professeur Stéphane Canu du laboratoire PSI de l'INSA de Rouen pour sa disponibilé lors de nos discutions fructueuses en fouille de données(Data Mining) et pour avoir accepté de faire partie de ce jury.

Je remercie Monsieur Abdelkader Hachemi Docteur-Ingénieur à l'IAM pour avoir accepté de siéger au sein de ce jury. Il est le responsable scientifique de MOM à l'IAM, et avec lui des discutions en profondeur sur les modélisations mathématiques ont été très bénéfiques dans la conception et l'implémentation du code IPDCA.

Je souhaite ici témoigner mon affection à ma mère pour les sacrifices consentis pour faire de ses enfants des hommes.

Je remercie plus particulièrement Liliane mon épouse pour sa patience, sa compréhension, son amour et l'équilibre qu'elle a apporté dans ma vie en acceptant de faire de moi son époux. Ces éléments ont été des alliés de poids pendant ces années.

Merci Banga, merci Rabi, mes petites sœurs bien aimées, elles m'ont apporté tant d'amour et de soutien.

Je souhaite témoigner ma gratitude à mon beau-père, il m'a adopté comme son fils et il

m'a offert un environnement propice pour mon épanouissement intellectuel. Sa bonté et sa sagesse en font un exemple.

Je remercie mes anciens collègues de la société ILOG où pendant plus de trois ans j'ai pu profiter de leur expérience et de leur grande compétence. Le caractère multiculturel de cette entreprise m'a permis de côtoyer des nationalités diverses et d'ouvrir mon esprit à leurs coutumes et d'appréhender d'un œil nouveau les relations humaines à chaque fois enrichissantes.

Je remercie également mes anciens collègues du projet européen COCONUT. Pendant les quinze mois de travail passés avec eux, j'ai eu l'occasion de profiter de leur expérience scientifique et de me familiariser à d'autres branches de l'optimisation. Je remercie en particulier le professeur Spellucci avec qui j'ai eu des discussions intéressantes sur l'optimisation non linéaire.

Mes remerciements à Guy Moebs Ingénieur responsable du support scientifique aux utilisateurs au CRIHAN(Centre de ressources d'Informatique de Haute-Normandie) pour son aide et ses précieux conseils.

Enfin à tous ceux qui m'ont soutenu de près ou de loin et à tous ceux qui m'ont incité même involontairement à faire mieux, veuillez trouver ici le témoignage de ma très profonde gratitude.

Table des matières

Table des figures

Liste des tableaux

Introduction

Galilée [165] disait en substance que le monde est écrit en langage mathématique. Nous pourrons ajouter que les problèmes de la vie courante, des plus simples aux plus complexes se posent comme problème d'optimisation.

La modélisation de ces problèmes débouche sur deux classes de problèmes : les problèmes convexes dont on dispose aujourd'hui d'un arsenal théorique et numérique considérable pour les résoudre, et les problèmes non convexes dont l'étude est en plein essor. Cette dichotomie des problèmes d'optimisation a poussé Rokafellar à dire :

« *The great watershed in optimization isn't between linearity or non linearity but between convexity and nonconvexity*» *[159].*

L'Optimisation non convexe et l'optimisation globale connaissent, au cours de la dernière décennie, des développements spectaculaires. Il y a une raison simple et évidente : la plupart des problèmes d'optimisation de la vie courante sont de nature non convexe. Et dans les entreprises et industries, les modèles convexes simplifiés sont au fur et à mesure remplacés par des modèles non convexes plus complexe certes, mais plus proches de la réalité et surtout plus compétitifs. Il y a deux approches, on dit même deux écoles, différentes mais complémentaires concernant ces domaines :

1. Approches combinatoires globales : inspirées par les travaux d'optimisation combinatoire, on conçoit des outils théoriques et algorithmiques pour traiter des problèmes continus d'optimisation non convexes. Ici on ne s'intéresse qu'aux algorithmes globaux, i.e., des algorithmes qui permettent d'obtenir des solutions optimales globales. Parmi les plus importantes contributions de cette Ecole, il faudrait citer Hoang Tuy (reconnu comme le pionnier), R. Horst, H. Konno, H. Benson, P. Pardalos, Le Dung Muu, Le Thi Hoai An, Nguyen Van Thoai, Phan Thien Thach, et Pham Dinh Tao. Certes ces travaux ont permis une meilleure compréhension des problèmes d'optimisation non convexe et de traiter ces problèmes dans des dimensions plus importantes, surtout ceux avec des structures spécifiques. Cependant les algorithmes globaux ne sont pas applicables aux problèmes d'optimisation non convexes de taille réelle.
2. Approches d'analyse convexe à l'optimisation non convexe : ces approches, qui étaient au début moins développées, utilisent les outils d'analyse et l'optimisation convexe. Elles sont particulièrement adaptées à la programmation DC dont le premier travail semble remonter à la maximisation convexe étudiée par Pham Dinh Tao en 1975. Les travaux de dualité

en optimisation non convexe de J.F. Toland en 1979 généralisent de manière logique et élégante la maximisation convexe. Et la programmation DC et DCA sont introduits par Pham Dinh Tao en 1986 à l'état préliminaire. Ces outils théoriques et algorithmiques sont intensivement développés à partir de 1993 par Le Thi Hoai An et Pham Dinh Tao(voir [103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 95] et références incluses) pour devenir maintenant classiques et de plus en plus populaires.

La programmation DC constitue une extension assez large de la programmation convexe pour couvrir presque tous les problèmes d'optimisation non convexe de la vie courante, mais pas trop pour pouvoir utiliser l'arsenal puissant de l'analyse et l'optimisation convexes. Dans le cadre d'une approche locale, DCA est une méthode de descente sans recherche linéaire basée sur les conditions d'optimalité locales et la dualité DC On y travaille non pas avec la fonction DC f elle-même mais avec les deux fonctions convexes g et h (DC composantes de f) DCA semble être le seul algorithme disponible, à l'heure actuelle, pour l'optimisation non convexe et non différentiable et capable de les traiter dans des grandes dimensions. En pratique, on observe très souvent que les solutions optimales locales fournies par DCA sont aussi solutions optimales globales. De plus, en général, un faible nombre de redémarrages de DCA à partir des meilleures solutions permettent d'atteindre des solutions optimales globales. Ces considérations ont poussé Le THi Hoai An et Pham Dinh Tao à introduire des algorithmes locaux (en particulier DCA) dans les approches globales. Il s'est avéré que la combinaison de DCA et les techniques globales de séparation et évaluation (B&B) apporte des améliorations remarquables :

(i) Elle permet de tester la globalité des solutions locales fournies par DCA et, le cas échéant, d'offrir une meilleure solution pour redémarrer DCA.

(ii) Elle améliore les bornes supérieures fournies par DCA et ainsi accélère la convergence de B&B. Comme retombées heureuses, elle permet à B&B de résoudre globalement des problèmes d'optimisation non convexe de dimension de plus grande dimension.

Notre travail s'inscrit pleinement dans la programmation DC et DCA. Notre première et principale contribution réside dans la combinaison des méthodes de points intérieurs avec DCA afin de pouvoir utiliser avantageusement les techniques des matrices quasi-définies développées par R. Vanderbei. Il en résulte que le nouvel algorithme que nous avons nommé IPDCA, semble très performant dans le traitement numérique des problèmes d'optimisation convexe et non convexe. Nous avons appliqué IPDCA à un problème industriel en Mécanique de Structure en collaboration avec l'Institut de Mécanique d'Aachen (IAM, RWTH) d'Allemagne. C'est un problème d'optimisation convexe de dimension allant jusqu'à 320 000 contraintes et variables. Les résultats obtenus montrent la grande supériorité de IPDCA par rapport au logiciel standard LANCELOT : le rapport en temps va jusqu'à 100 fois. La deuxième contribution concerne la programmation quadratique non convexe. Nous suivons les travaux de Le Thi Hoai An et Pham Dinh Tao dans l'étude des méthodes combinées de DCA et B&B en introduisant IPDCA à la place de DCA. Les simulations numériques ont montré les avantages et désavantages d'une telle modification. Comme applications nous avons traités les problèmes de Max-Cut et les problèmes concernant la classification non linéaire de SVM (Séparateur à Vaste Marge) en fouille de données(Data Mining) en collaboration avec le Laboratoire PSI (Peception, Système d'Information) de l'INSA de Rouen avec

comme interlocuteur privilégié S. Canu.

Nos autres travaux sont consacrés à :
1. l'introduction d'un nouvel algorithme pour résoudre les problèmes non linéaires que nous avons nommé méthode de points non intérieurs en raison de la non exigence de l'intériorité des itérés à chaque itération, contrairement aux approches classiques de méthodes de points intérieurs.
2. Une technique de redémarrage de DCA basée sur les conditions globales pour les problèmes de maximisation convexe.
3. La combinaison de DCA avec l'optimisation monotone introduite par H. Tuy.

Cette thèse est divisée en trois parties :

la première partie est consacrée aux techniques d'optimisations locales et s'articule autour des méthodes de points intérieurs. Nous y présenterons deux algorithmes développés dans le cadre de notre travail ; après une présentation non exhaustive de la programmation DC, des méthodes de points intérieurs et des propriétés essentielles de la classe des matrices quasi-définies ; il s'agit d'une classe de matrices que l'on rencontre lors de la mise en œuvre des méthodes de points intérieurs. Le Chapitre 2 sera dédié à la présentation d'un nouvel algorithme basé sur une reformulation des conditions d'optimalité de Karush-Kuhn-Tucker. Le Chapitre 3 sera consacré à l'intégration des techniques d'optimisation DC dans un schéma de points intérieurs.

La seconde partie de cette thèse est consacrée aux solutions globales[1] de problèmes de programmation quadratique. Dans le premier chapitre de cette partie, nous explorerons l'intégration de l'algorithme de points intérieurs développée au Chapitre 3 dans un schéma séparation-évaluation, le second chapitre, inspiré du précédent, sera consacré à la résolution de problèmes quadratiques non convexes à variables zéro-un, une différence essentielle avec le chapitre précédent en plus du type de problème traité est l'utilisation de subdivisions entières et non rectangulaires normales comme précédemment.
Le troisième chapitre sera quant à lui consacré à l'optimisation monotone due au Professeur Tuy. Nous examinerons plus particulièrement son intégration dans une procédure séparation-évaluation dans laquelle DCA sera appelé pour améliorer la borne supérieure. Le quatrième et dernier chapitre de cette partie est consacré à une procédure permettant de redémarrer DCA à partir d'un point qui améliorera la solution retournée par DCA. Cette procédure est inspirée des conditions d'optimalité globales.

Convaincu par l'affirmation de Dantzig [193], «The final test of a theory is its capacity to solve the problems which originated it », la dernière partie de la thèse est consacrée aux applications industrielles. Nous y appliquerons les deux algorithmes présentés au Chapitre 2 et au Chapitre 3, d'abord au Chapitre 8 à un problème de mécanique de structure de grande

[1]La solution globale dans ce cadre est à distinguer de la solution globale au sens des méthodes locales qui elle signifie solution locale trouvée en partant d'un point arbitraire.

dimension, puis à un problème de séparation à vaste marge au Chapitre 9.

Tous les chapitres excepté le premier se terminent par des tests numériques.

«**Everything should be made as simple as possible, but not simpler.**»
Albert Einstein

Première partie

Méthodes de Points Intérieurs et Programmation DC

Chapitre 1

Programmation DC et méthodes de points intérieurs

1.1 Eléments d'analyse convexe et conditions d'optimalité

Dans cette section nous rappelerons brièvement quelques notions d'analyse convexe qui nous seront utiles pour la suite de notre exposé. Nous ne chercherons pas à être exhaustif. Des développements sur le sujet peuvent être trouvés dans la monographie de Rockafellar [158], et l'ouvrage de Hiriart-Urruty et Lemaréchal [177] dans lesquels la théorie est détaillée dans le cadre de la dimension finie. Le livre d'Ekeland et Temam [142] est quant à lui consacré au cadre de la dimension infinie.

Dans la suite $\mathcal{X}$ désignera l'espace euclidien $\mathbb{R}^n$, muni du produit scalaire usuel noté $\langle ., . \rangle$ et de la norme euclidienne associée $\| x \| = \sqrt{\langle x, x \rangle}$. Et $\mathcal{Y}$ désignera l'espace vectoriel dual de $\mathcal{X}$ relatif au produit scalaire, que l'on pourra identifier à $\mathcal{X}$. On notera également

$$\bar{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$$

muni d'une structure algébrique déduite de celle de $\mathbb{R}$ [158] avec la convention

$$+\infty - (+\infty) = +\infty. \tag{1.1}$$

1.1.1 Ensembles convexes

On dira qu'une partie C de $\mathcal{X}$ est *convexe* si pour tout $x, y \in C$ le segment

$$[x, y] = \{(1-t)x + ty \mid t \in [0, 1]\}$$

est contenu dans C. La Figure 1.1 illustre cette notion.
Soit $S \subset \mathcal{X}$, *l'enveloppe convexe* de S, notée $\mathrm{co}(S)$, est l'ensemble des combinaisons convexes

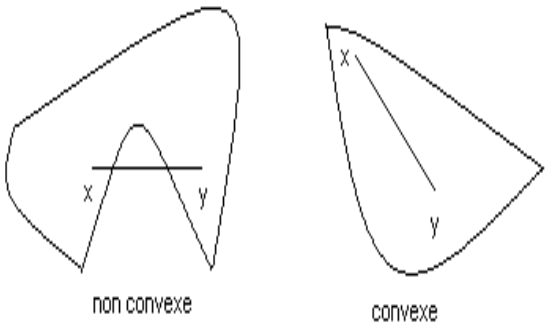


FIG. 1.1 – Un ensemble convexe.

finies d'éléments de S, c'est-à-dire

$$\text{co}(S) = \{\sum_{i=1}^{n} \lambda_i x^i \mid \lambda_i \in \mathbb{R}^+,\ x^i \in S\ \forall i = 1, 2, \cdots, n \text{ et } \sum_{i=1}^{n} \lambda_i = 1\}.$$

Soit C un ensemble convexe de $\mathbb{R}^n$ on définit la *variété linéaire* engendrée par C comme étant l'ensemble

$$\text{aff}(C) = \{\sum_{i=1}^{n} \lambda_i x^i \mid \lambda_i \in \mathbb{R},\ x^i \in C\ \forall i = 1, 2, \cdots, n \text{ et } \sum_{i=1}^{n} \lambda_i = 1\}.$$

On montre que aff(C) est le translaté d'un espace vectoriel.
On appelle *intérieur relatif* d'un ensemble convexe C, son intérieur dans aff(C), muni de la topologie induite de celle de $\mathcal{X}$. On le note

$$\text{ir}(C) = \{x \in C \mid \text{il existe } r > 0 \text{ tel que } B(x, r) \cap \text{aff}(C) \subset C\}$$

où $B(x, r)$ est la boule[1] centrée en x et de rayon r.

Remarque 1.1.1 *ir(C) est vide si et seulement si C l'est.*

1.1.2 Fonctions convexes

Soit $f : C \to \bar{\mathbb{R}}$ une fonction définie sur un ensemble convexe C de $\mathcal{X}$, on appelle *domaine* de f l'ensemble

$$\text{dom}(f) = \{x \in C \mid f(x) < +\infty\}.$$

[1] $B(x, r) = \{y\ : \parallel x - y \parallel \leq r\}$.

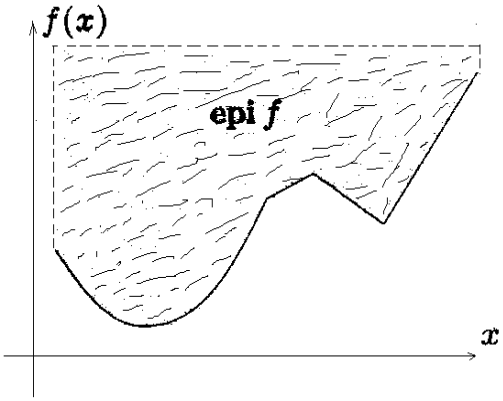


FIG. 1.2 – Epigraphe d'une fonction.

La fonction f est dite *propre* si elle ne prend jamais la valeur $-\infty$ et si elle n'est pas identiquement égale à $+\infty$. L'*épigraphe* de f est l'ensemble

$$\text{epi}(f) = \{(x, \alpha) \in C \times \mathbb{R} \mid f(x) \leq \alpha\}.$$

Un exemple d'épigraphe d'une fonction est représentée sur la Figure 1.2.
La fonction f est dite *convexe* si son épigraphe est un ensemble convexe de $\mathbb{R} \times \mathcal{X}$. Ceci équivaut à dire que l'on a l'inégalité

$$f((1-\lambda)x + \lambda y) \leq (1-\lambda)f(x) + \lambda f(y) \text{ pour tout } x, y \in C \text{ et tout } \lambda \in [0,1]. \tag{1.2}$$

Si l'inégalité est stricte dans (1.2) pour tout $\lambda \in]0,1[$ et pour tout x,$y \in C$ avec $x \neq y$ alors la fonction f est dite *strictement convexe.*
On dira que la fonction f est *fortement convexe de module* ρ sur l'ensemble convexe C s'il existe un nombre réel $\rho > 0$ tel que

$$f((1-\lambda)x+\lambda y) \leq (1-\lambda)f(x)+\lambda f(y)-(1-\lambda)\lambda\frac{\rho}{2} \parallel x-y \parallel^2 \text{ pour tout } x, y \in C \text{ et tout } \lambda \in [0,1]. \tag{1.3}$$

Plus précisément f est fortement convexe sur le convexe C si

$$\rho(f, C) = \sup\{\rho \geq 0 \mid f - \frac{\rho}{2} \parallel . \parallel \text{ est convexe sur } C\} \tag{1.4}$$

Notons que si $\rho(f, C) > 0$ alors (1.3) est vérifiée pour tout $\rho \in [0, \rho(f, C)[$. On dira que la borne supérieure est atteinte dans la définition (1.4) si $f - \frac{\rho}{2} \parallel . \parallel$ est convexe sur C tout entier. Si $C = \mathcal{X}$, alors on notera $\rho(f)$ au lieu de $\rho(f, C)$.

Remarque 1.1.2 *Toute fonction fortement convexe est strictement convexe et toute fonction strictement convexe est convexe.*

Etant donné une fonction convexe propre f sur $\mathcal{X}$, on dira que $y^o \in \mathcal{Y}$ est un *sous-gradient* de f en $x^o \in \text{dom}(f)$ si

$$\langle y^o, x - x^o \rangle + f(x^o) \leq f(x) \ \ \forall x \in \mathcal{X}.$$

L'ensemble de tous les sous-gradients de f au point x^o est le *sous-différentiel* de f au point x^o et on note $\partial f(x^o)$.
Le domaine du sous-différentiel de la fonction f est

$$\text{dom}(\partial f) = \{x \mid \partial f(x) \neq \emptyset\}. \tag{1.5}$$

Soit ε un réel positif, un élément y^o de $\mathcal{Y}$ est appelé ε-sous-différentiel de f au point x^o si

$$\langle y^o, x - x^o \rangle + f(x^o) - \varepsilon \leq f(x) \ \ \forall x \in \mathcal{X}.$$

On note $\partial_\varepsilon f(x^o)$ l'ensemble de tous les ε-sous-différentiels de f au point x^o.

La fonction f est dite *semi-continue inférieurement*(s.c.i) en un point $x \in C$ si

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

On note $\Gamma_o(\mathcal{X})$ l'ensemble des fonctions convexes s.c.i et propres sur $\mathcal{X}$.
On définit la fonction $f^\star$ *conjuguée* de f définie sur $\mathcal{Y}$ par

$$f^\star(y) = \sup\{\langle x, y \rangle - f(x) \mid x \in \mathcal{X}\}. \tag{1.6}$$

Ainsi $f^\star$ est l'enveloppe supérieure des fonctions affines sur $\mathcal{Y}$, $y \to \langle x, y \rangle - f(x)$ sur $\mathcal{Y}$.

Propriété 1.1.1 *On a les propriétés suivantes*
- $f^\star$ *est toujours convexe.*
- *Si* f *prend la valeur* $-\infty$ *alors* $f^\star$ *est identiquement égale à* $+\infty$.
- *On a* $(f^\star)^\star \leq f$

On a également la proposition suivante

Proposition 1.1.1 *Soit* $f : C \to \bar{I\!R}$ *alors :*
- $\partial f(x)$ *est une partie convexe fermée de* $\mathcal{Y}$.
- $f \in \Gamma_o(\mathcal{X}) \iff f^\star \in \Gamma_o(\mathcal{Y})$. *Dans ce cas on a* $f = f^{\star\star}$.
- $y \in \partial f(x) \iff f(x) + f^\star(y) = \langle x, y \rangle$.
- *Si* $f \in \Gamma_o(\mathcal{X})$ *alors* $y \in \partial f(x) \iff x \in \partial f^\star(y)$.
- *Si* $f \in \Gamma_o(\mathcal{X})$ *et si* $\partial f(x)$ *est réduit à un singleton* $\{y\}$, *alors* f *est différentiable en* x *et* $\nabla f(x) = y$ *et réciproquement.*
- $f(x^o) = \min\{f(x) \mid x \in \mathcal{X}\} \iff 0 \in \partial f(x^o)$.

1.1.3 Fonctions convexes polyédrales

Une partie convexe C est dite *convexe polyédrale* si

$$C = \bigcap_{i=1}^{m} \{x \mid \langle a_i, x\rangle - b_i \leq 0,\ a_i \in \mathcal{Y},\ b_i \in \mathbb{R}\}.$$

Une fonction $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ est dite polyédrale si son épigraphe est un ensemble polyédral de $\mathbb{R}^{n+1}$.
Notons que toute fonction polyédrale est propre, convexe et s.c.i.

Proposition 1.1.2 *Soit $f : \mathcal{X} \to \bar{I\!R}$ une fonction convexe. Alors f est polyédrale si et seulement si dom(f) est un ensemble convexe polyédral, et*

$$f(x) = \sup\{\langle a_i, x\rangle - b_i \mid i = 1, 2, \cdots, l\}\ \ \forall x \in dom(f). \tag{1.7}$$

Proposition 1.1.3 *[158]*

- *Si f est convexe polyédrale alors $f^\star$ l'est aussi. De plus si f est partout finie alors*

$$dom(f^\star) = co\{a_j \mid j = 1, 2, \cdots, l\},$$

$$f^\star(y) = \min\{\sum_{i=1}^{l} \lambda_i a_i \mid y = \sum_{i=1}^{l} \lambda_i a_i,\ \lambda_i \geq 0 \text{ et } \sum_{i=1}^{l} \lambda_i = 1\}.$$

- *Si f est polyédrale alors $\partial f(x)$ est une partie convexe polyédrale non vide en tout point de dom(f).*
- *Si $f_1, \cdots, f_s$ sont des fonctions convexes polyédrales sur $\mathcal{X}$ telles que les ensembles convexes $dom(f_i)$, $i = 1, 2, \cdots, s$ ont un point commun alors*

$$\partial(f_1 + \cdots + f_s)(x) = \partial f_1(x) + \cdots + \partial f_s(x),\ \ \forall x \in \mathcal{X}.$$

1.1.4 Conditions d'optimalité

Définition 1.1.1 *Soient $\mathcal{C} \subset I\!R^n$ et $x^\circ \in I\!R^n$. On dira que $d \in I\!R^n$ est tangent à $\mathcal{C}$ en x° si il existe deux suites $\{d_k\}_k \subset I\!R^n$ et $\{t_k\}_k \subset I\!R^n_+$ telles que*

$$d_k \to d,\ \ t_k \to 0^+,\ \ x^\circ + t_k d_k \in \mathcal{C}. \tag{1.8}$$

On note $\mathcal{T}_{x^\circ}\mathcal{C}$ l'ensemble des vecteurs tangents à $\mathcal{C}$ en x°, c'est le cône tangent de $\mathcal{C}$ en x°.

On a la proposition suivante.

Proposition 1.1.4 *Le cône tangent $\mathcal{T}_{x^\circ}\mathcal{C}$ est un cône fermé ; il est convexe si $\mathcal{C}$ est convexe.*

Considérons le problème d'optimisation

$$(\mathcal{P}_\mathcal{C}) \left\{ \begin{array}{l} \min f(x) \\ x \in \mathcal{C} \end{array} \right. \tag{1.9}$$

On a la condition nécessaire d'optimalité suivante

Théorème 1.1.1 *Si $x^\star$ est un minimum local de f sur $\mathcal{C}$ et si f est dérivable en $x^\star$ on a*

$$\nabla f(x^\star) \in (\mathcal{T}_{x^\star}\mathcal{C})^\star, \tag{1.10}$$

où $(\mathcal{T}_{x^\star}\mathcal{C})^\star$ désigne le dual de $\mathcal{T}_{x^\star}\mathcal{C}$.
Ce qui se traduit par

$$\nabla f(x^\star)d \geq 0, \ \ \forall d \in \mathcal{T}_{x^\star}\mathcal{C}.$$

Considérons à présent le problème d'optimisation suivant

$$(\mathcal{P}_{\mathcal{EI}}) \left\{ \begin{array}{l} \min f(x) \\ c_i(x) \ = 0, \ \ i \in \mathcal{E} \\ c_i(x) \ \geq 0, \ \ i \in \mathcal{I}, \end{array} \right. \tag{1.11}$$

avec $\mathcal{E} = \{1, 2, \cdots, m_E\}$ et $\mathcal{I} = \{1, 2, \cdots, m_I\}$ définissant une partition de $\{1, 2, \cdots, m\}$, $f : \mathbb{R}^n \to \mathbb{R}$, et $c : \mathbb{R}^n \to \mathbb{R}^m$.
On note

$$\mathcal{C} = \{x \in \mathbb{R}^n \ : \ c_\mathcal{E}(x) \ = 0, \ c_\mathcal{I}(x) \ \geq 0\}$$

l'ensemble admissible de $(\mathcal{P}_{\mathcal{EI}})$ avec $c_\mathcal{E}$(resp. $c_\mathcal{I}$) le vecteur de $\mathbb{R}^{m_E}$(resp. $\mathbb{R}^{m_I}$) formé des c_i tels que $i \in \mathcal{E}$(resp. $i \in \mathcal{I}$) et

$$\mathcal{I}^\circ(x) = \{i \in \mathcal{I} \ : \ c_i(x) = 0\}$$

l'ensemble des indices des contraintes d'inégalité *actives* en x.

Définition 1.1.2 *On appelle cône linéarisant de $\mathcal{C}$ en $\bar{x} \in \mathcal{C}$ l'ensemble*

$$\mathcal{T}'_{\bar{x}}\mathcal{C} = \{d \in I\!R^m \ : \ \nabla c_\mathcal{E}(\bar{x})d = 0, \ \nabla c_{\mathcal{I}^\circ(\bar{x})}(\bar{x})d \geq 0\}$$

On a la proposition suivante

Proposition 1.1.5 *si $\bar{x} \in \mathcal{C}$ et si $c_{\mathcal{E}\cup\mathcal{I}^\circ(\bar{x})}$ est dérivable en $\bar{x}$, alors on a l'inclusion*

$$\mathcal{T}_{\bar{x}} \subset \mathcal{T}'_{\bar{x}}. \tag{1.12}$$

Définition 1.1.3 *On parle de qualification des contraintes de $(\mathcal{P}_{\mathcal{EI}})$ en $\bar{x}$ si on a l'égalité*

$$\mathcal{T}_{\bar{x}} = \mathcal{T}'_{\bar{x}}. \tag{1.13}$$

On a le théorème suivant

Théorème 1.1.2 *(Condition nécessaire du premier ordre de Karush-Kuhn-Tucker)*
Soit $x^\star$ un minimum local de $(\mathcal{P}_{\mathcal{EI}})$. On suppose f et $c_{\mathcal{E}\cup\mathcal{I}^\circ(x^\star)}$ dérivables en $x^\star$, on suppose également que les contraintes soient qualifiées en $x^\star$. Alors il existe $\lambda^\star \in \mathbb{R}^m$ tel que

$$\begin{cases} \nabla f(x^\star) - A(x^\star)^T\lambda^\star &= 0, \\ c_{\mathcal{E}}(x^\star) &= 0, \\ c_{\mathcal{I}}(x^\star) &\leq 0, \\ \lambda^\star_{\mathcal{I}} &\geq 0, \\ (\lambda^\star_{\mathcal{I}})^T c_I(x^\star) &= 0, \end{cases}$$

où $A(x^\star)$ désigne la Jacobienne de c en $x^\star$.

Remarque 1.1.3 *Dans le cas convexe, c'est-à-dire le cas où $c_{\mathcal{E}}$ est affine $c_{\mathcal{I}}$ et f convexes, la condition nécessaire ci-dessus devient une condition suffisante pour $x^\star$ d'être minimum(global) de $(\mathcal{P}_{\mathcal{EI}})$.*

On fait l'hypothèse de la dérivabilité de $c_{\mathcal{E}\cup\mathcal{I}^\circ(\bar{x})}$ en $\bar{x}$ et de la continuité de $c_{\mathcal{I}\setminus\mathcal{I}^\circ(\bar{x})}$ en $\bar{x}$. La condition de qualification (1.13) étant difficile à vérifier, dans la pratique on utilise l'une des conditions suivantes :

(A) $c_{\mathcal{E}\cup\mathcal{I}^\circ(\bar{x})}$ est affine dans un voisinage de $\bar{x}$.

(B) **Qualification de Slater :**
- $c_{\mathcal{E}}$ est affine avec $\nabla c_{\mathcal{E}}$ surjective,
- $c_{\mathcal{I}^\circ(\bar{x})}$ est convexe,
- On peut trouver un point $\hat{x} \in \mathcal{C}$ tel que $c_{\mathcal{I}^\circ(\hat{x})<0.}$

(C) Les gradients des contraintes actives en $\bar{x}$, $\{\nabla c_i(\bar{x}) \ : \ i \in \mathcal{E}\cup\mathcal{I}^\circ(\bar{x})\}$ sont linéairement indépendants.

(D) **Qualification de Mangazarian-Fromowitz :**
si $\sum_{i\in\mathcal{E}\cup\mathcal{I}^\circ(\bar{x})} \nu_i\nabla c(\bar{x}) = 0$ avec $\nu \geq 0$ pour $i \in \mathcal{I}^\circ(\bar{x})$ alors $\nu = 0$ pour tout $i \in \mathcal{E}\cup\mathcal{I}^\circ(\bar{x})$.
On peut trouver dans [166] les diverses implications qui lient ces conditions de qualification.

1.1.5 Les fonctions DC

Afin d'étendre la programmation convexe à la résolution de la plupart des problèmes d'optimisation non convexes de la vie courante tout en continuant à utiliser son arsenal théorique et numérique, une nouvelle classe de fonctions fut introduite : la classe des fonctions DC

Soit C un ensemble convexe de $\mathcal{X}$. On note $Conv(C)$ l'ensemble des fonctions convexes sur C à valeur dans $\mathbb{R}$.
Une fonction $f : C \to \mathbb{R}\cup\{+\infty\}$ définie sur C est dite DC sur C si elle s'écrit

$$f(x) = g(x) - h(x), \text{ pour tout } x \in C \tag{1.14}$$

où g et h sont deux fonctions convexes sur C. On note $\mathcal{DC}(C)$ l'ensemble des fonctions DC sur C, c'est également l'espace vectoriel engendré par $Conv(C)$. Mise à part sa structure d'espace vectoriel, $\mathcal{DC}(C)$ est également stable par rapport aux opérations usuelles utilisées en optimisation.

Proposition 1.1.6 *[103, 104]*
- *Une combinaison linéaire de fonctions DC sur C est une fonction DC sur C.*
- *L'enveloppe supérieure(resp. inférieure) d'un nombre fini de fonctions DC à valeurs finies sur C est une fonction DC sur C.*
- *Si f est une fonction DC à valeurs finies sur C alors les fonctions $|f|$, $f^+ = \max\{f, 0\}$, $f^- = \min\{f, 0\}$ sont des fonctions DC sur C.*

Remarque 1.1.4 *Soit $f \in \mathcal{DC}(C)$ et soit $f = g - h$ sa représentation DC alors pour toute fonction convexe finie ξ sur C, $f = (g + \xi) - (h + \xi)$ définit une autre décomposition DC de f. Ainsi une fonction DC possède une infinité de décompositions DC*

Une sous-classe importante de fonctions de $Conv(C)$ est la classe de fonctions C^2 sur $\mathbb{R}^n$. Cette classe est particulièrement importante car les fonctions objectif de la plupart des problèmes d'optimisation de la vie courante appartiennent à cette sous-classe.

On note $C^{1,1}$ le sous-espace vectoriel de $C^1(C)$ des fonction dont le gradient est localement Lipschitzien sur C. $C^1(C)$ est contenu dans le cône convexe $\mathcal{LC}^2(C)$ des fonctions localement enveloppe supérieure d'une famille de fonctions de $C^2(C)$. Ces fonction sont dites $sous - C^2$.

Définition 1.1.4 *Soit C un ouvert convexe, f est localement DC sur C si tout point x° de C admet un voisinage $\mathcal{V}(x^\circ)$ noté $\mathcal{V}$, tel que pour tout x dans $\mathcal{V}$*

$$f(x) = g_{\mathcal{V}}(x) - h_{\mathcal{V}}(x) \ \text{ pour tout } \ g_{\mathcal{V}}, h_{\mathcal{V}} \in Conv(\mathcal{V}).$$

Proposition 1.1.7 *[103, 126]*
Les deux propositions suivantes sont équivalentes
(i) f est dans $\mathcal{LC}^2(C)$
(ii) f est localement DC sur C avec $h_{\mathcal{V}}$ une forme quadratique convexe.

On a le théorème suivant,

Théorème 1.1.3 *[103, 126]*
Soit C un ouvert convexe de $\mathcal{X}$.
- *Toute fonction localement DC sur C est DC sur C. En particulier toute fonction de classe C^2 sur C est DC sur C.*
- *On a la chaîne d'inclusions*

$$Conv(C) \subset \mathcal{LC}^2(C) \subset \mathcal{DC}(C).$$

- *Si C est en plus compact, alors $\mathcal{DC}(C)$ est un sous-espace vectoriel dense dans l'ensemble des fonctions continues sur C muni de la norme de la convergence uniforme sur C.*

1.1.6 Dualité en optimisation DC

Soient g et h deux fonctions convexes propres et sci sur $\mathcal{X}$. Considérons le problème d'optimisation

$$(P_{dc}) \;\; \alpha = \inf\{g(x) - h(x) \mid x \in \mathcal{X}\}$$

Puisque h est dans $\Gamma_o(\mathcal{X})$ on a $h^{\star\star} = h$, on peut donc écrire

$$h(x) = \sup\{\langle x, y\rangle - h^\star(y) \mid y \in \mathcal{Y}\},$$

et on a

$$\begin{aligned} \alpha &= \inf\{g(x) - \sup\{\langle x, y\rangle - h^\star(y) \mid y \in \mathcal{Y}\} \mid x \in \mathcal{X}\} \\ &= \inf\{\beta(y) \mid y \in \mathcal{Y}\} \end{aligned}$$

avec

$$\beta(y) = \inf\{g(x) - [\langle x, y\rangle - h^\star(y)] \mid x \in \mathcal{X}\}.$$

Il va de soi que (P_y) est un problème convexe et

$$\beta(y) = \begin{cases} h^\star(y) - g^\star(y) \text{ si } y \in \text{dom}(h^\star) \\ +\infty \text{ sinon} \end{cases}$$

On obtient finalement le problème dual de (P_{dc})

$$\alpha = \{h^\star(y) - g^\star(y) \mid y \in \text{dom}(h^\star)\}.$$

Et grâce à la convention (1.1) on peut écrire

$$(D_{dc}) \;\; \alpha = \inf\{h^\star(y) - g^\star(y) \mid y \in \mathcal{Y}\}.$$

On observe la parfaite symétrie entre le problème primal (P_{dc}) et le problème dual (D_{dc}) ; il va de soi que les résultats établis pour l'un se transposent à l'autre.

1.1.7 Conditions d'optimalité en optimisation DC

Soient $\mathcal{S}_P$ et $\mathcal{S}_D$ les ensembles de solutions des prolèmes (P_{dc}) et (D_{dc}) respectivement, et posons

$$\mathcal{P}_l = \{x^\star \in \mathcal{X} \mid \partial h(x^\star) \subset \partial g(x^\star)\} \text{ et } \mathcal{D}_l = \{y^\star \in \mathcal{Y} \mid \partial g^\star(x^\star) \subset \partial h^\star(x^\star)\}$$

Le théorème suivant est celui à partir duquel DCA est bâti.

Théorème 1.1.4 *[104, 109]*
(i) Transport de minima globaux :

$$\cup\{\partial h(x) \mid x \in \mathcal{S}_P\} \subset \mathcal{S}_D \subset dom(h^\star).$$

La première inclusion se transforme en égalité si $g^\star$ est sous-différentiable sur $\mathcal{S}_D$ (en particulier si $\mathcal{S}_D \subset ir(dom(g^\star))$ où si $g^\star$ est sous-différentiable sur $dom(h^\star)$) et dans ce dernier cas $\mathcal{S}_D \subset (dom(\partial g^\star) \cap dom(\partial h^\star))$.

(ii) Si $x^\star$ est un minimum local de $g-h$, alors $x^\star \in \mathcal{P}_l$. L'implication inverse est vraie si h est une fonction convexe polyédrale.

(iii) Soit $x^\star$ un point critique[2] de $g-h$ et $y^\star \in \partial g(x^\star) \cap \partial h(x^\star)$. Soit $\mathcal{U}$ un voisinage de $x^\star$ tel que $\mathcal{U} \cap dom(g) \subset dom(\partial h)$. Si pour tout $x \in \mathcal{U} \cap dom(g)$ il existe un $y \in \partial h(x)$ tel que $h^\star(y) - g^\star(y) \geq h^\star(y^\star) - g^\star(y^\star)$, alors $x^\star$ est un minimum local de $g-h$. Plus précisément,

$$g(x) - h(x) \geq g(x^\star) - h(x^\star), \text{ pour tout } x \in \mathcal{U} \cap dom(g). \tag{1.15}$$

(iv) Transport de minima locaux : Soit $x^\star \in dom(\partial h)$ un minimum local de $g-h$ et soit $y^\star \in \partial h(x^\star)$. Sous l'hypothèse

$$y^\star \in int(dom(g^\star)) \text{ et } \partial g^\star(y^\star) \subset \mathcal{U}, \tag{1.16}$$

par exemple si $g^\star$ est différentiable en $y^\star$, alors $y^\star$ est un minimum local de $h^\star - g^\star$.

Notons que ce théorème admet sa forme duale grâce à la symétrie observée à la section précédente entre le problème (P_{dc}) et le problème (D_{dc}).
Notons également que tous ces résultats concernent les composantes DC g et h et non la fonction f elle-même.

Dans le cas où $f = g - h$ est convexe sur $\mathcal{X}$, on parle de *faux problème DC* Ce cas nous intéressera en particulier au Chapitre 8 dans lequel le problème industriel que nous étudierons est un problème d'optimisation convexe.

Proposition 1.1.8 *[104, 109]*
Soit $f = g - h$ avec $g, h \in \Gamma_o(\mathcal{X})$ vérifiant $dom(g) \subset dom(h)$ et $ir(dom(g)) \cap ir(dom(h)) \neq \emptyset$. Soit $x_o \in dom(g)(resp.dom(h))$ un point où g (resp. h) est continue. Si f est convexe, alors
(i) h est continue en x^o.
(ii) $\partial h(x^o) = \partial g(x^o) \underline{*} \partial h(x^o)$[3]
(iii) $0 \in \partial f(x^o) \iff \partial h(x^o) \subset \partial g(x^o)$

Preuve : On a $\mathrm{dom}(f) = \mathrm{dom}(g)$ et $g = f + h$. Donc $\partial g(x^o) = \partial f(x^o) + \partial h(x^o)$ car $\mathrm{ir}(\mathrm{dom}(f)) \cap \mathrm{ir}(\mathrm{dom}(h)) \neq \emptyset$. Si g est continue en $x^o \in \mathrm{dom}(g)$, on a,

$$x^o \in \mathrm{ir}(\mathrm{dom}(g)) = int(\mathrm{dom}(g)) \subset int(\mathrm{dom}(h)). \tag{1.17}$$

Ainsi l'ensemble convexe fermé $\partial h(x^o)$ est borné et de [103] on a

$$\partial f(x^o) = \partial g(x^o) \underline{*} \partial h(x^o) = \cap\{\partial g(x^o) - v \mid v \in \partial h(x^o)\}. \tag{1.18}$$

La troisième équivalente est immédiate. □

[2] $x^\star$ est un point critique de $g-h$ si $\partial h(x^\star) \cap \partial g(x^\star)$ est non vide.
[3] $A \underline{*} B = \{x \in \mathcal{X} \mid x + B \subset A\}$ [103]

On observe donc que les problèmes d'optimisation convexes peuvent s'écrire comme problème d'optimisation DC et résolus en utilisant les techniques d'optimisation DC Soient $f, \theta \in \Gamma_o(x)$ telles que $\mathrm{dom}(f) \subset \mathrm{dom}(\theta)$ et $\mathrm{ir}(\mathrm{dom}(f)) \cap \mathrm{ir}(\mathrm{dom}(\theta)) \neq \emptyset$ alors le problème d'optimisation

$$\inf\{f(x) \mid x \in \mathcal{X}\}$$

est équivalent au faux problème d'optimisation DC

$$\inf\{(f+\theta)(x) - \theta(x) \mid x \in \mathcal{X}\}$$

dans le sens où ils ont la même valeur optimale et le même ensemble de solutions.
On a

$$0 \in \partial f(x^\star) \Rightarrow \partial\theta(x^\star) \subset \partial(f+\theta)(x^\star) = \partial f(x^\star) + \partial\theta(x^\star)$$

et l'autre implication est vraie si en plus, θ est continue en $x^\star$.

Proposition 1.1.9 *[104, 109] Soit $f = g - h$ avec $g, h \in \Gamma_o(\mathcal{X})$ vérifiant $dom(g) \subset dom(h)$ S'il existe un voisinage convexe $\mathcal{U}$ de $x^\star$ tel que f est finie et convexe sur $\mathcal{U}$, alors les propositions suivantes sont équivalentes :*
(i) $0 \in \partial(f + \mathcal{X}_{\mathcal{U}})(x^\star)$
(ii) $\partial h(x^\star) \subset \partial g(x^\star)$.
Sous les hypothèses ci-dessus, la condition nécessaire d'optimalité locale du Théorème 1.1.4

$$\partial h(x^\star) \subset \partial g(x^\star)$$

est également suffisante : $f(x^\star) \leq f(x)$ *pour tout* $x \in \mathcal{U}$.

Théorème 1.1.5 *[104, 106, 124] Soit $f = g - h$ où g, $h \in \Gamma_o(\mathcal{X})$ alors $x^\star$ est minimum global du problème (P_{dc}) si et seulement si*

$$\partial_\varepsilon h(x^\star) \subset \partial_\varepsilon g(x^\star) \text{ pour tout } \varepsilon > 0. \tag{1.19}$$

1.1.8 Algorithme d'optimisation DC : DCA

La construction des DCA repose sur la génération de deux suites $\{x^k\}_k$ et $\{y^k\}_k$ qui sont améliorées à chaque itération de sorte que leur limite respective $x^\star$ et $y^\star$ soient candidates pour être les optima locaux du problème primal et du problème dual respectivement. Ces deux suites sont liées par la dualité et vérifient les propriétés suivantes :

1. Les suites $\{g(x^k) - h(x^k)\}$ et $\{g^\star(y^k) - h^\star(y^k)\}$ décroissent à chaque itération.
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. $h^\star - g^\star$),
3. sinon toute valeur d'adhérence $x^\star$ de la suite $\{x^k\}_k$ (resp. $y^\star$ de la suite $\{y^k\}$) est un point critique de $g - h$ (resp. $h^\star - g^\star$).

Ces suites sont générées de la manière suivante : x^{k+1} (resp. y^k) est solution du problème convexe (Pk) (resp. (D_k)) défini par

$$(P_k)\ \ \alpha_k = \inf\{g(x) - [h(x^k) + \langle x - x^k, y^k\rangle] \mid x \in \mathcal{X}\} \tag{1.20}$$

$$(D_k)\ \ \inf\{h^\star(y) - [g^\star(y^{k-1}) + \langle x^k, y - y^{k-1}\rangle] \mid y \in \mathcal{Y}\} \tag{1.21}$$

Comme on peut le voir (P_k) (resp. (D_k)) est obtenu de (P_{dc}) (resp. (D_{dc})) en remplaçant h (resp. $g^\star$) par sa minorante affine définie par $y^k \in \partial h(x^k)$ (resp. $x^k \in \partial g^\star(y^{k-1})$), DCA conduit au schéma

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

Ce schéma correspond à la *forme simplifiée* de DCA, et se traduit par l'algorithme ci-dessous.

Algorithme 1.1.1 *DCA forme simplifiée*

0. x^o *donné.*
1. Pour chaque k, x^k *étant connu,* ***déterminer*** $y^k \in \partial h(x^k)$
2. ***Trouver*** $x^{k+1} \in \partial g^\star(y^k)$
3. Si test d'arrêt vérifié ***Stop*** *;* ***sinon*** $k \leftarrow k+1$ *et* ***aller*** *en 1.*

Dans la forme complète de DCA on impose le choix suivant :

$$x^{k+1} \in \arg\min\{g(x) - h(x)\ :\ x \in \partial g^\star(y^k)\} \tag{1.22}$$

et

$$y^k \in \arg\min\{h^\star(y) - g^\star(y)\ :\ y \in \partial h(x^k)\} \tag{1.23}$$

Les problèmes (1.22) et (1.23) sont équivalents aux problèmes respectifs

$$x^{k+1} \in \arg\min\{\langle x, y^k\rangle - h(x)\ :\ x \in \partial g^\star(y^k)\} \tag{1.24}$$

et

$$y^k \in \arg\min\{\langle x^k, y\rangle - g^\star(y)\ :\ y \in \partial h(x^k)\} \tag{1.25}$$

Les problèmes (1.24) et (1.25) sont des problèmes de minimisation concave. Ainsi DCA assure l'appartenance $(x^\infty, y^\infty) \in \mathcal{P}_l \times \mathcal{D}_l$.
Malgré leur apparente simplicité par rapport aux problèmes d'origine (P_{dc}) et (D_{dc}), ces problèmes restent difficiles à résoudre. En effet il s'agit de problèmes de minimisation concave comme nous l'avons déjà signalé plus haut, de plus le travail se fait sur $\partial g^\star(y^k)$ (resp. $\partial h(x^k)$). Ces difficultés rendent en pratique l'utilisation de ce schéma presque impossible, excepté pour des cas bien particuliers pour lesquels la résolution de (1.24)-(1.25) est une tâche facile.
Dans la suite DCA sera utilisé pour désigner DCA simplifié, sauf précision contraire.

1.1.9 Existence et bornitude des suites générées par DCA

On dira que DCA est bien défini si on peut construire les suites $\{x^k\}_k$ et $\{y^k\}_k$ à partir d'un point arbitraire $x^o \in \mathcal{X}$.

Lemme 1.1.1 *[104, 109] Les propositions suivantes sont équivalentes*
(i) Les suites $\{x^k\}_k$ et $\{y^k\}_k$ sont bien définies
(ii) $dom(\partial g) \subset dom(\partial h)$ et $dom(\partial h^\star) \subset dom(\partial g^\star)$

La proposition suivante établit les conditions pour lesquelles les suites générées par DCA sont bornées.

Lemme 1.1.2 *[104, 109] Si $g - h$ est coercive*[4] *alors on a*
(i) la suite $\{x^k\}_k$ est bornée ;
(ii) si $\{x^k\}_k \subset int(dom(h))$ alors la suite $\{y^k\}_k$ est aussi bornée.
Par dualité, si $(h^\star - g^\star)$ est coercive alors on a
(i) La suite $\{y^k\}_k$ est bornée
(ii) Si $\{y^k\}_k \subset int(dom(g^\star))$ alors la suite $\{x^k\}_k$ est aussi bornée.

Le théorème suivant établit la convergence de DCA simplifié.

Théorème 1.1.6 *[104, 109]*
1. On suppose les suites $\{x^k\}_k$, $\{y^k\}_k$ bien définies. Alors on a

$$(g-h)(x^{k+1}) \leq (h^\star - g^\star)(y^k) - \delta_1 \leq (g-h)(x^k) - \delta_2 \tag{1.26}$$

avec

$$\begin{aligned} \delta_1 &= \max\{\frac{\rho_2}{2} \parallel dx^k \parallel^2, \frac{\rho_2^\star}{2} \parallel dy^k \parallel^2\} \\ \delta_2 &= \max\{\frac{\rho_1+\rho_2}{2} \parallel dx^k \parallel^2, \frac{\rho_1^\star}{2} \parallel dy^{k-1} \parallel^2 + \frac{\rho_2}{2} \parallel dx^k \parallel^2, \frac{\rho_1^\star}{2} \parallel dy^{k-1} \parallel^2 + \frac{\rho_2^\star}{2} \parallel dy^k \parallel^2\}. \end{aligned}$$

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

$$x^k \in \partial g^\star(y^k),\ y^k \in \partial h(x^{k+1}) \text{ et } (\rho_1+\rho_2)dx^k = \rho_1^\star dy^{k-1} = \rho_2^\star dy^k = 0.$$

Dans ce cas on a
- *$(g-h)(x^{k+1}) = (h^\star - g^\star)(y^k)$ et x^k, x^{k+1} sont des points critiques de la fonction g-h tels que*
$$y^k \in (\partial g(x^k) \cap \partial h(x^k)) \text{ et } y^k \in (\partial g(x^{k+1}) \cap \partial h(x^{k+1})). \tag{1.27}$$
- *y^k est un point critique de $h^\star - g^\star$ tel que*
$$[x^k, x^{k+1}] \subset (\partial g^\star(y^k) \cap \partial h^\star(y^k)).$$

[4] une fonction ψ est coercive si $\lim\limits_{\|x\| \to +\infty} \psi(x) = +\infty$.

- $x^{k+1} = x^k$ *si* $\rho(g) + \rho(h) > 0$, $y^k = y^{k-1}$ *si* $\rho(g^\star) > 0$ *et* $y^k = y^{k+1}$ *si* $\rho(h^\star) > 0$.

2. *Si* α *est fini alors les suites décroissantes* $\{(g-h)(x^k)\}_k$ *et* $\{(h^\star - g^\star)(y^k)\}_k$ *sont convergentes et ont la même limite* $\beta \geq \alpha$.
   Si $\rho(g) + \rho(h) > 0$, *alors* $\lim\limits_{k\to+\infty} \{x^{k+1} - x^k\} = 0$.
   Si $\rho(g^\star) + \rho(h^\star) > 0$, *alors* $\lim\limits_{k\to+\infty} \{y^{k+1} - y^k\} = 0$.
   On a en plus

$$\begin{aligned} \lim_{k\to+\infty} \{g(x^k) + g^\star(y^k) - \langle x^k, y^k\rangle\} &= 0 \\ \lim_{k\to+\infty} \{h(x^{k+1}) + h^\star(y^k) - \langle x^{k+1}, y^k\rangle\} &= 0 \end{aligned}$$

3. *Si* α *est finie et si les suites* $\{x^k\}_k$ *et* $\{y^k\}_k$ *sont bornées alors pour toute valeur d'adhérence* $x^\star$ *de* $\{x^k\}_k$*(resp.* $y^\star$ *de* $\{y^k\}_k$*) il existe une valeur d'adhérence* $y^\star$ *de* $\{y^k\}_k$*(resp.* $x^\star$ *de* $\{x^k\}_k$*) telle que*
   (i) $y^\star \in (\partial g(x^\star) \cap \partial h(x^\star))$ *et* $g(x^\star) - h(x^\star) = \beta$;
   (ii) $x^\star \in (\partial g^\star(y^\star) \cap \partial h^\star(y^\star))$ *et* $h^\star(y^\star) - g^\star(y^\star) = \beta$.

La première partie du Théorème 1.1.6 admet une formulation similaire duale [104, 109] .

1.1.10 Optimisation DC polyédrale : Étude approfondie de DCA

On parle d'optimisation DC polyédrale lorsque l'une des composantes convexes de la décomposition DC $f = g - h$ est convexe polyédrale.
Cette classe de problèmes DC se rencontre fréquemment dans la pratique et possède des propriétés intéressantes tant sur le plan théorique que numérique. En particulier DCA a une convergence finie [104, 109]. Elle nous permet également de donner une interprétation profonde de DCA.
Notons h_k(resp. h^k) la minorante affine(resp. convexe polyédrale) de la fonction convexe h.

$$\begin{aligned} h_k(x) &= h(x^k) + \langle x - x^k, y^k\rangle \\ &= \langle x, y^k\rangle - h^\star(y^k), \ \ \forall x \in \mathcal{X} \\ h^k(x) &= \sup\{h_i(x) \mid i = 0, 1, \cdots, k\} \\ &= \sup\{\langle x, y^i\rangle - h^\star(y^i) \mid i = 0, 1, \cdots, k\}, \ \ \forall x \in \mathcal{X} \end{aligned}$$

où les suites $\{x^k\}_k$ et $\{y^k\}$ sont obtenues en suivant la procédure décrite à la Section 1.1.8.
On définit de manière duale $(g^\star)_k$(resp. $(g^\star)^k$) la minorante affine(resp. convexe polyésrale) de la fonction convexe $g^\star$:

$$\begin{aligned} (g^\star)_k(x) &= g^\star(y^{k-1}) + \langle y - y^{k-1}, x^k\rangle \\ &= \langle y, x^k\rangle - g(x^k), \ \ \forall y \in \mathcal{Y} \\ (g^\star)^k(x) &= \sup\{(g^\star)_i(y) \mid i = 0, 1, \cdots, k\} \\ &= \sup\{\langle y, x^i\rangle - g(x^i) \mid i = 0, 1, \cdots, k\}, \ \ \forall x \in \mathcal{X} \end{aligned}$$

Etant donné qu'une fonction convexe propre s.c.i se caractérise comme étant le suprémum de ses minorantes affines, il s'avère donc plus judicieux d'utiliser h^k(resp. $(g^\star)^k$) comme sous estimé la fonction convexe h(resp $g^\star$), au lieu de la minorante affine h_k(reps. $(g^\star)_k$). On obtient alors les programmes

$$\inf\{g(x) - h^k(x) \mid x \in \mathcal{X}\} \tag{1.28}$$

$$\inf\{h^\star(y) - (g^\star)^k(y) \mid y \in \mathcal{Y}\} \tag{1.29}$$

ces programmes sont non convexes en opposition aux sous-problèmes (P_k) et (D_k) qui eux l'étaient.

Lemme 1.1.3 *[104, 109]*
x^{k+1} est une solution optimale du problème (1.28).

Preuve : on commence tout d'abord par réécrire (1.28) sous la forme,

$$\inf\{g(x) - \sup\{h_i(x) \mid i = 0, 1, \cdots, k\} \mid x \in \mathcal{X}\} \tag{1.30}$$

qui peut également s'écrire

$$\inf_{x\in\mathcal{X}} \inf_{i=0,1,\cdots,k} \{g(x) - h_i(x)\} = \inf_{i=0,1,\cdots,k} \inf_{x\in\mathcal{X}} \{g(x) - h_i(x)\}. \tag{1.31}$$

Puisque x^{i+1} est la solution optimale du problème (P_i) on déduit que x^i avec

$$i \in \arg\min\{g(x^{i+1}) - h_i(x^{i+1}) \mid i = 0, 1, \cdots, k\},$$

est une solution optimale du problème (1.28).
D'autre part on a

$$g(x^{i+1}) - h(x^{i+1}) \leq g(x^{i+1}) - h_i(x^{i+1}), \quad \text{pour} \quad i = 0, 1, \cdots, k. \tag{1.32}$$

Puisque la suite $\{g(x^i) - h(x^i)\}_i$ est décroissante on a

$$g(x^{k+1}) - h(x^{k+1}) \leq \inf\{g(x^{i+1}) - h_i(x^{i+1}), \quad \text{pour} \quad i = 0, 1, \cdots, k\}.$$

Et si

$$h_k(x^{k+1}) = h(x^{k+1}), \tag{1.33}$$

alors $h^k(x^{k+1}) = h(x^{k+1})$ et on obtient l'égalité

$$g(x^{k+1}) - h(x^{k+1}) = g(x^{k+1}) - h_k(x^{k+1}) = \inf\{g(x^{i+1}) - h_i(x^{i+1}), \quad \text{pour} \quad i = 0, 1, \cdots, k\}.$$

Donc x^{k+1} est une solution optimale du problème DC polyédrale(1.28). □

La preuve du Lemme 1.1.3 éclaire sur la manière de résoudre le problème (1.28) et le lemme suivant apporte les mêmes précisions pour le problème (1.29).

Lemme 1.1.4 *[104, 109]*
y^k est une solution optimale du problème (1.29).

Preuve : Le schéma de la démonstration est similaire à celui du Lemme 1.1.3 en utilisant cette fois les expressions duales. □

Dans la démonstration du Lemme 1.1.3 nous avons fait l'hypothèse que l'égalité $h_k(x^{k+1}) = h(x^{k+1})$ est vraie. Qu'en est-il si tel n'est pas le cas ? Plus précisément que se passe-t-il si

$$h_k(x^{k+1}) < h(x^{k+1}) \text{ pour tout } k \, ? \tag{1.34}$$

Ce cas correspond à celui où k tend vers l'infini dans (1.33). Nous allons décrire dans les lignes qui suivent le comportement de DCA dans ce cas. Supposons que la valeur optimale du problème (P_{dc}) soit finie et les suites $\{x^k\}_k$ et $\{y^k\}_k$ générées par DCA. Alors

$$g(x^\infty) - h_\infty(x^\infty) = \inf\{g(x^{i+1}) - h_i(x^{i+1}) \mid i = 0, 1, \cdots, \infty\} \tag{1.35}$$

pour toute valeur d'adhérence x^∞ de la suite $\{x^k\}_k$, où h_∞ est la minorante affine de h :

$$h_\infty(x) = h(x^\infty) + \langle x - x^\infty, y^\infty\rangle = \langle x, y^\infty\rangle - h^\star(y^\infty), \ \ \forall x \in \mathcal{X} \tag{1.36}$$

avec $y^\infty \in \partial h(x^\infty)$ une valeur d'adhérence de $\{y^k\}_k$. Le point x^∞ est par conséquent une solution du programme DC

$$\inf\{g(x) - h^\infty(x) \mid x \in \mathcal{X}\} \tag{1.37}$$

où la fonction convexe h^∞ est définie par

$$h^\infty(x) = \sup\{\langle x, y^i\rangle - h^\star(y^i) \mid i = 0, 1, \cdots, \infty\}, \ \ \forall x \in \mathcal{X}$$

De manière similaire pour le problème dual DC (D_{dc}), une minorante affine de $g^\star$ est définie par

$$(g^\star)_\infty(y) = (g^\star)_\infty(y^\infty) + \langle y - y^\infty, x^\infty\rangle = \langle y, x^\infty\rangle - g(x^\infty), \ \ \forall x \in \mathcal{X}$$

et

$$(g^\star)_\infty(y) = \sup\{(g^\star)_i(y) \mid i = 1, 2, \cdots, \infty\}, \ \ \forall y \in \mathcal{Y} \tag{1.38}$$

Et comme plus haut on a

$$h^\star(y^\infty) - (g^\star)^\infty(y^\infty) = \inf\{h^\star(y^i) - (g^\star)_i(y^i) \mid i = 1, 2, \cdots, \infty\} \tag{1.39}$$

et y^∞ est une solution du problème DC

$$\inf\{h^\star(y) - (g^\star)^\infty(y) \mid y \in \mathcal{Y}\} \tag{1.40}$$

De plus les valeurs optimales des problèmes (1.37) et (1.40) sont égales

$$g(x^\infty) - h^\infty(x^\infty) = h^\star(y^\infty) - (g^\star)(y^\infty). \tag{1.41}$$

On a les théorèmes fondamentaux suivants.

Théorème 1.1.7 *[104, 109]*

(i) $g(x^{k+1}) - h(x^{k+1}) = h^\star(y^k) - g^\star(y^k)$ *si et seulement si* $h^k(x^{k+1}) = h(x^{k+1})$,
(ii) $h^\star(y^k) - g^\star(y^k) = g(x^k) - h(x^k)$ *si et seulement si* $g^\star(y^k) = (g^\star)^k(y^k)$,
(iii) $g(x^{k+1}) - h(x^{k+1}) = g(x^k) - h(x^k)$ *si et seulement si* $h^k(x^{k+1}) = h(x^{k+1})$ *et* $g^\star(y^k) = (g^\star)^k(y^k)$. *Par conséquent, si* $g(x^{k+1}) - h(x^{k+1}) = g(x^k) - h(x^k)$ *(dans ce cas la valeur commune est* $h^\star(y^k) - g^\star(y^k)$*), alors les propositions suivantes sont vraies,*
(iv) x^{k+1} *(resp.* y^k*) est une solution du problème* (1.28) *(resp.* (1.29)*).*
(v) De plus , si h *et* h^k *coïncident en une solution de* (P_{dc}) *ou* $g^\star$ *et* $(g^\star)^k$ *coïncident en une solution de* (D_{dc})*, alors* x^{k+1} *(resp.* y^k*) est également une solution de* (P_{dc}) *(resp.* (D_{dc})*).*

Théorème 1.1.8 *[104, 109]*
Si la valeur optimale du problème DC (P_{dc}) *est finie et les suites* $\{x^k\}_k$ *et* $\{y^k\}_k$ *sont bornées, alors toute valeur d'adhérence* x^∞*(resp.* y^∞*) est une solution du problème DC* (1.37)*(resp.* (1.40)*). De plus les valeurs optimales sont égales,*

$$g(x^\infty) - h^\infty(x^\infty) = h^\star(y^\infty) - (g^\star)^\infty(y^\infty).$$

Si l'une des conditions ci-dessous est vérifiée,
(i) les fonctions h^∞ *et* h *coïncident en une solution optimale de* (P_{dc}),
(ii) les fonctions $(g^\star)^\infty$ *et* $g^\star$ *coïncident en une solution optimale de* (D_{dc}),
alors x^∞ *et* y^∞ *sont également des solutions optimales de* (P_{dc}) *et* (D_{dc}) *respectivement.*

1.1.11 Un exemple test

Dans cette section pour illustrer le processus de convexification qu'entreprend DCA pendant ses itérations et l'influence qu'a le point initial et le choix de la décomposition DC sur la qualité de la solution que trouve DCA, nous considèrerons le problème d'optimisation quadratique non convexe suivant généré aléatoirement,

$$(\mathcal{T}) \begin{cases} \min f(x,y) = \frac{1}{2}\begin{pmatrix} x & y \end{pmatrix}\begin{pmatrix} -83.75 & 28.34 \\ 28.34 & -48.24 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 17.72 & 15.22 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} \\ t.q. \\ \quad 0 \le x \le 1, \;\; 0 \le y \le 1. \end{cases}$$

Ce problème a quatre solutions locales $(0,0)$, $(1,0)$, $(0,1)$ et $(1,1)$ comme le montre la Figure 1.3. La solution globale est $(1,0)$ avec comme valeur objectif -24.20.

La plus petite valeur propre du Hessien de f est -99.43. La décomposition DC de la fonction objectif considérée pour cet exemple est la suivante $f(x,y) = g(x,y) - h(x,y)$ avec

$$\begin{aligned} g(x,y) &= \frac{1}{2}\begin{pmatrix} x & y \end{pmatrix}\begin{pmatrix} 15.68 & 28.34 \\ 28.34 & 51.19 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 17.72 & 15.22 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} \\ h(x,y) &= \frac{1}{2}\begin{pmatrix} x & y \end{pmatrix}\begin{pmatrix} 99.43 & 0 \\ 0 & 99.43 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} \end{aligned}$$

Une décomposition alternative est $f(x,y) = \bar{g}(x,y) - \bar{h}(x,y)$ avec

$$\begin{aligned}
\bar{g}(x,y) &= \frac{1}{2}\begin{pmatrix} x & y \end{pmatrix}\begin{pmatrix} 99.43 & 0 \\ 0 & 99.43 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 17.72 & 15.22 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} \\
\bar{h}(x,y) &= \frac{1}{2}\begin{pmatrix} x & y \end{pmatrix}\begin{pmatrix} 183.18 & -28.34 \\ -28.34 & 147.67 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}
\end{aligned}$$

Notre choix fut de considérer que la première décomposition dans notre analyse sachant que celle-ci reste valable pour cette seconde décomposition.

1.1.11.1 Importance du choix du point initial pour DCA

Le Tableau 1.1 présente les différentes solutions trouvées par DCA en fonction des différents points initiaux. Ces points sont obtenus à partir d'une perturbation de la solution globale.

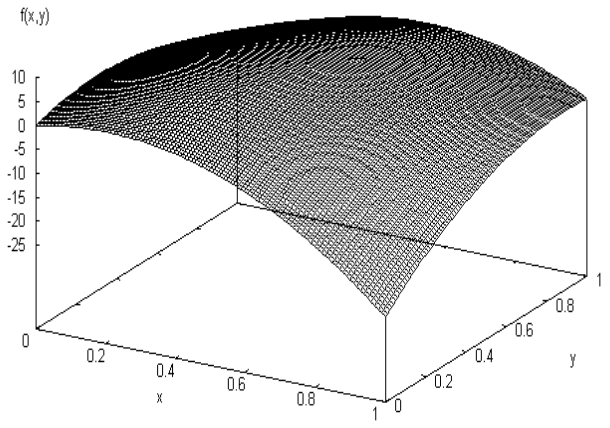


FIG. 1.3 – La surface de $f(x,y)$.

On voit sur le Tableau 1.1 l'intérêt de fournir à DCA un bon point initial. En effet en gras, la composante y du deuxième point ne diffère de la composante y du premier point que de 0.01 la composante x des deux points étant identique, on observe la grande différence des valeurs objectif trouvées par DCA dans les deux cas.
Le choix d'un bon point initial pour DCA est une tâche difficile et cette question reste ouverte. Cependant un début de réponse semble être l'adjonction à DCA d'une procédure pour calculer un bon point initial. Une telle procédure pourrait être un autre algorithme local comme cela est fait au Chapitre 3 dans lequel DCA est associé à IPDCA.

(x^0, y^0)	$iter$	$f^\star$	$(x^\star, y^\star)$
(0.75,0.25)	2	-24.20	(1.00,0.00)
(0.32,0.25)	3	-24.20	(1.00,0.00)
(0.24,0.25)	3	0.00	(0.00,0.00)
(0.24,0.27)	4	0.00	(0.00,0.00)
(0.24,0.31)	**8**	**0.00**	**(0.00,0.00)**
(0.24,0.32)	**9**	**-8.92**	**(0.00,1.00)**
(0.25,0.75)	2	-8.92	(0.00,1.00)

TAB. 1.1 – DCA avec différents points initiaux.

Notons qu'en utilisant la fonction **quadprog** de MATLAB pour résoudre le problème $(\mathcal{T})$ et en prenant comme points initiaux les mêmes que ceux indiqués sur le Tableau 1.1, la solution globale $(1, 0)$ fut trouvée à chaque fois. Aucune conclusion sur la supériorité de **quadprog** par rapport à DCA ne peut néanmoins être tirée ; cependant cette remarque ne fait que souligner l'importance particulière que revêt le choix du point initial pour DCA.

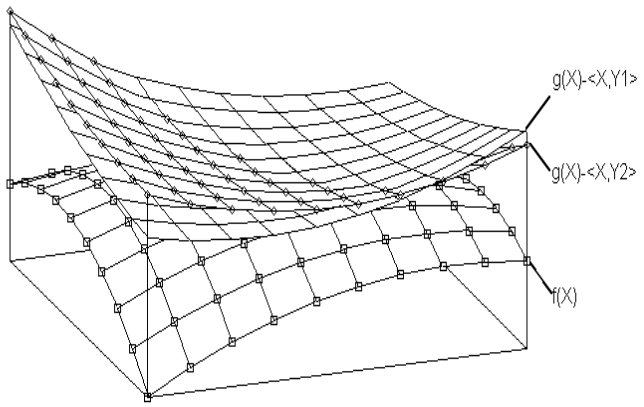


FIG. 1.4 – Processus de convexification de DCA.

1.1.11.2 Processus de convexification de DCA

La Figure 1.4 présente le processus de convexification qu'entreprend DCA. Comme on peut le voir, et comme cela a été présenté plus haut, la convexification de DCA est globale et

ξ	$iter$	$f^\star$	$(x^\star, y^\star)$
0.001	3	-24.20	(1.00,0.00)
0.01	3	-24.20	(1.00,0.00)
0.1	3	-24.20	(0.00,0.00)
1	3	-24.20	(1.00,0.00)
10	3	-24.20	(1.00,0.00)
20	4	-24.20	(1.00,0.00)
25	4	0.00	(0.00,0.00)
30	6	0.00	(0.00,0.00)

TAB. 1.2 – DCA avec différentes décompositions DC

majorante. Elle est globale car elle se fait sur tout le domaine admissible, et elle est majorante car comme le montre la Figure 1.4, la fonction DC $f = g - h$ est remplacée à chaque itération k par la fonction convexe majorante $\bar{f}_k = g - \langle ., y^k \rangle$ avec $y^k \in \partial h(x^k)$.

1.1.11.3 Importance de la décomposition DC pour DCA

La qualité de la solution retournée par DCA dépend également de la convexification qu'entreprend DCA et celle-ci dépend de la décomposition DC adoptée. Pour l'illustrer considérons une fois de plus le problème $(\mathcal{T})$ et considérons une fois de plus la décomposition DC $f(x, y) = g(x, y) - h(x, y)$ avec

$$\begin{aligned} g(x,y) &= \frac{1}{2}\begin{pmatrix} x & y \end{pmatrix}\begin{pmatrix} 15.68+\xi & 28.34 \\ 28.34 & 51.19+\xi \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 17.72 & 15.22 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} \\ h(x,y) &= \frac{1}{2}\begin{pmatrix} x & y \end{pmatrix}\begin{pmatrix} 99.43+\xi & 0 \\ 0 & 99.43+\xi \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} \end{aligned}$$

où ξ est une perturbation positive que nous faisons varier au Tableau 1.2 pour montrer qu'un choix de ξ petit entraîne un meilleur rapport qualité de la solution nombre d'itérations donc temps de calcul pour DCA. Le point initial est $(0.320, 0.25)$ dans cette analyse.

1.2 Méthodes de points intérieurs

1.2.1 Une brève présentation des méthodes de points intérieurs

La redécouverte par Karmakar [77] au début des années 1980 des méthodes de points intérieurs après leur introduction par Fiacco et McCormick [58] et leur abandon à cause de la matrice Hessienne qui devenait asymptotiquement mal conditionnée, a poussé Wright [182] à parler de la révolution des méthodes de points intérieurs. L'algorithme de Karmakar fut le premier algorithme pour la programmation linéaire à avoir une complexité polynomiale. Il s'en suivit un intérêt croissant pour l'application de ces méthodes à la résolution de

problèmes de programmation linéaire. Parmi les différentes approches étudiées, les méthodes de points intérieurs primal-dual semblèrent être les plus performantes comparées aux autres algorithmes de points intérieurs et à l'algorithme du simplex. Ceci fut vérifié numériquement [78, 82] et théoriquement [83, 84].

Le succès des méthodes de points intérieurs pour la résolution des problèmes de programmation linénaire entraîna son extension aux problèmes de programmation non linéaire et en particulier aux problèmes de programmation quadratique convexes [34, 85] ainsi qu'aux problèmes non linéaires convexes [86, 87, 88]. Ces études ne contredirent pas l'efficacité des méthodes de points intérieurs primal-dual observée pour la programmation linéaire. Il en découla l'envie de généraliser cette approche aux problèmes non linéaires(non convexes). El-Bakry *et al.* [90], McCormick et Falk [89], Akrotirianakis *et al.* [55], Nocedal *et al.* [97], Tits *et al.* [91] et Yamashita [56] ont développé des algorithmes primal-dual à convergence globale pour les problèmes non linéaires non convexes. Une comparaison numérique des différents algorithmes représentant l'état de l'art peut être trouvée dans [96].

Nonobstant leur efficacité à résoudre les problèmes linéaires et les problèmes convexes(malgré la découverte par Gilbert, Gonzagas et Karas [164] de chemins centraux à l'allure mouvementée en optimisation convexe générés par les méthodes de points intérieurs sur des problèmes indéfiniment différentiables), la gestion de la non-convexité reste une question ouverte malgré quelques avancées dans ce domaine.

Pour présenter le schéma général des méthodes de points intérieurs nous considérons le problème d'optimisation suivant,

$$\begin{cases} \min f(x) \\ c(x) \geq 0, \end{cases} \tag{1.42}$$

avec $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ et $c : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ au moins de classe $\mathcal{C}^2$.

Les méthodes de points intérieurs pour les problèmes de programmation non linéaires génèrent des pas en résolvant une suite de sous-problèmes barrière de la forme

$$\begin{cases} \min f(x) - \mu \sum_{i=1}^{m} log(\omega_i) \\ c(x) - \omega = 0, \\ \omega > 0, \end{cases} \tag{1.43}$$

pour un paramètre de barrière positif décroissant μ. Ici ω est un vecteur de variables d'écart utilisé pour transformer les contraintes d'inégalité en contraintes d'égalité. Au fur et à mesure que μ approche zéro, la solution du problème barrière (1.43) approche la solution du problème (1.42) sous certaines conditions. Il existe actuellement des implémentations robustes et efficaces des méthodes de points intérieurs(voir par exemple IPOPT [150], KNITRO [57, 98, 162], et LOQO [36]).

Comme nous l'avons mentionné plus haut, un point essentiel lors de la mise en œuvre des méthodes de points intérieurs est la gestion de la non-convexité. Pour gérer la non-convexité, et donc pour garantir que les directions de déplacement soient des directions de descente, les méthodes de points intéreurs avec recherche linéaire utilisent la technique qui conciste à perturber le Hessien du Lagrangien du problème (1.42) afin que la matrice résultante soit définie positive. En claire, le Hessien $\mathcal{H}$ est perturbé en lui ajoutant un terme ρI avec $\rho \geq 0$, de sorte que la matrice $\mathcal{H} + \rho I$ soit définie positive. Le choix de ρ est en pratique difficile car demande de déterminer la plus petite valeur propre de la matrice $\mathcal{H}$, ce qui est difficile pour des problèmes de grande taille. Si on note h le second menbre du système linéaire issu de la linéarisations des conditions nécessaires du premier ordre, on doit donc résoudre

$$(\mathcal{H} + \rho I)d = h. \tag{1.44}$$

Mais (1.44) n'est rien d'autre que la traduction des conditions d'optimalité du premier ordre du problème,

$$\min_{d \in \mathbb{R}^n} (\frac{1}{2} d^T \mathcal{H} d - h^T d) + \frac{\rho}{2} \parallel d \parallel^2 \tag{1.45}$$

Ainsi, perturber le Hessien revient à pénaliser le déplacement.
Dans le cadre de la globalisation par région de confiance, on montre que la perturbation ajoutée au Hessien du Lagrangien dans le cadre de la recherche linéaire n'est rien d'autre que le multiplicateur de Lagrange associé à la contrainte de région de confiance [163]. La globalisation par région de confiance permet donc de gérer automatiquement le paramètre ρ.

Nous avons choisi de comparer l'algorithme que nous allons présenter au Chapitre 3 au logiciel LOQO [36]. Ce choix fut motivé par deux raisons : tout d'abord LOQO est l'un des algorithmes qui représentent l'état de l'art actuel dans le domaine des méthodes de points intérieurs et le schéma algorithmique de LOQO se rapproche de celui de notre algorithme.

1.2.2 LOQO

LOQO était à l'origine un code de points intérieurs pour la programmation linéaire et quadratique convexe, puis fut étendu aux problèmes non linéaires. LOQO est un algorithme de points intérieurs primal-dual dans lequel une procédure de recherche linéaire est incorporée. Dans LOQO le problème non linéaire (1.42) est remplacé par une suite de problèmes barrière (1.42). L'algorithme calcule une direction d en factorisant[5] la matrice

$$\begin{pmatrix} \nabla^2 L(x,\lambda) & A(x)^T \\ A(x) & W\Lambda^{-1} \end{pmatrix} \tag{1.46}$$

où $A(x)$ est la Jacobienne des contraintes $c(x)$, λ le vecteur des multiplicateurs de Lagrange et $L(x,\lambda)$ le Lagrangien, $L(x,\lambda) = f(x) - \lambda^T c(x)$. On utilise la notation W pour la matrice diagonale avec $W_{ii} = \omega_i$ et Λ la matrice diagonale définie par $\Lambda_{ii} = \lambda_i$.

[5] La factorisation existe si la matrice est quasi-définie.

Une fois la direction d calculée, une recherche linéaire est entreprise pour garantir une décroissance suffisante de la fonction de mérite l_2,

$$\psi(x,\omega;\mu,\sigma) = f(x) - \mu\sum_{i=1}^{m} log(\omega_i) + \sigma \parallel c(x) - \omega \parallel_2^2, \tag{1.47}$$

avec $\sigma > 0$ le paramètre de pénalisation. Cette fonction n'est pas une fonction de pénalisation exacte puisque pour tout σ donné, un minimum de ψ n'est pas normalement un point de KKT du problème (1.42). De plus, une heuristique est utilisée pour déterminer la valeur du paramètre de pénalisation σ; à priori LOQO fixe $\sigma = 0$ puis le fait croître si nécessaire pour assurer d d'être une direction de déscente de ψ. La règle suivante est utilisée pour mettre à jour le paramètre μ à chaque itération.

$$\mu = 0.1 \min\left(0.05, \frac{1-\xi}{\xi}, 2\right)^3 \frac{\lambda^T\omega}{m}, \quad \text{avec } \xi = \frac{\min_i \lambda_i\omega_i}{\lambda^T\omega/m}. \tag{1.48}$$

Cette règle permet de faire croître le paramètre barrière à chaque itération, et parallèlement LOQO exige la décroissance de la fonction de mérite ψ à chaque itération.
Pour résoudre le système de Newton réduit dans lequel la matrice (1.46) intervient, celle-ci est factorisée en utilisant une forme de factorisation LDL^T [31].
Pour gérer la non-convexité, il est incorporé dans LOQO une procédure appelée lors de la factorisation de Cholesky de la matrice (1.46) qui teste si la matrice

$$\nabla^2(x,\lambda) + A(x)^T W^{-1}\Lambda A(x)$$

est suffisamment définie positive. Si tel n'est pas le cas un multiple de α de la matrice identité est ajouté à $\nabla^2 L$. Et puisque la valeur de α n'est pas connue à l'avance, plusieurs valeurs de α pourraient être testées et plusieurs factorisations pourraient être nécessaires.
Comme nous l'avons déjà mentionné plus haut, aucun résultat de convergence n'a été fourni pour LOQO, et il sera bien difficile d'en établir un en raison de l'utilisation de la règle dynamique 1.48 pour mettre à jour le paramètre barrière μ, et l'utilisation de la fonction de mérite non exacte (1.47).

1.3 Matrices quasi-définies

Nous allons présenter quelques résultats importants sur une classe de matrices intervenant dans le système de Newton réduit dans le contexte des méthodes de points intérieurs. Cette classe de matrices fera l'objet d'une attention particulière au Chapitre 3 dans lequel un nouveau type de régularisation sera introduit pour pouvoir exploiter ses propriétés.

Définition 1.3.1 *Une matrice symétrique indéfinie est dite symétrique quasi-définie si elle est de la forme*

$$K = \begin{pmatrix} -H & A^T \\ A & E \end{pmatrix} \tag{1.49}$$

avec H et E deux matrices symétriques définies positives.

Par extension nous dirons qu'un système linéaire est quasi-défini si la matrice du système en question est symétrique quasi-définie.
Les matrices quasi-définies possèdent des propriétés intéressantes que nous allons passer en revue dans les lignes qui suivent.

1.3.1 Inversibilité

Proposition 1.3.1 *Les matrices quasi-définies sont inversibles*

Preuve : En effet considérons le système linéaire suivant

$$\begin{pmatrix} -H & A^T \\ A & E \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix} \tag{1.50}$$

La définie positivité de H nous permet de tirer x du premier bloc d'équations et d'écrire

$$x = H^{-1}(a - A^T y). \tag{1.51}$$

En substituant cette valeur de x dans le deuxième bloc d'équations nous obtenons

$$y = (E + AH^{-1}A^T)^{-1}(b + AH^{-1}a) \tag{1.52}$$

La définie positivité des matrices H et E assure la définie positivité de la matrice

$$N = (E + AH^{-1}A^T)^{-1}(b + AH^{-1}a) \tag{1.53}$$

dans (1.52) et donc non singulière. D'où le système linéaire (1.50) admet une unique solution pour tout a et b, et donc la matrice K est non singulière. □

La preuve ci-dessus peut nous suggérer une méthode pour résoudre le système linéaire (1.50). Cette procédure pourrait s'avérer non avantageuse sur le plan numérique si la matrice K est large et creuse. En effet un point important lors de l'inversion de la matrice K est de conserver sa creusité. Ceci ne serait pas nécessairement le cas dans ce schéma dans la mesure où si l'une des colonnes de la matrice A est complètement dense, la matrice N en (1.53) est totalement dense. D'où la nécessité d'examiner d'autres approches.
La proposition suivante montre que la classe des matrices quasi-définies est stable par l'inversion.

Proposition 1.3.2 *L'inverse d'une matrice quasi-définie est quasi-définie*

Preuve : On a

$$\begin{pmatrix} -H & A^T \\ A & E \end{pmatrix}^{-1} = \begin{pmatrix} -\bar{H} & \bar{A}^T \\ \bar{A} & \bar{E} \end{pmatrix} \tag{1.54}$$

avec

$$\bar{H} = H^{-1} + H^{-1}A^T(E + AH^{-1}A^T)^{-1}AH^{-1}, \quad (1.55)$$
$$\bar{A} = (E + AH^{-1}A^T)^{-1}AH^{-1} \quad (1.56)$$
$$\bar{E} = (E + AH^{-1}A^T)^{-1}. \quad (1.57)$$

$\bar{H}$ peut se réécrire

$$\bar{H} = (H + A^T E^{-1} A)^{-1}.$$

□

1.3.2 Factorisabilité

Définition 1.3.2 *Une matrice K est factorisable s'il existe une factorisation*

$$K = LDL^T, \quad (1.58)$$

et la matrice K est fortement factorisable s'il existe une factorisation

$$PKP^T = LDL^T \text{ pour toute permutation symétrique } P, \quad (1.59)$$

avec D une matrice diagonale et L une matrice triangulaire inférieure.

On a le théorème fondamental suivant dû à Vanderbei.

Théorème 1.3.1 *[35] Toute matrice symétrique quasi-définie est fortement factorisable et cette factorisation est unique.*

Preuve : Soit P une matrice de permutation. Il suffit de montrer que les sous-matrices principales de PKP^T sont inversibles[6]. Ces sous-matrices sont de la forme $\bar{P}K_l\bar{P}^T$, où K_l est une sous-matrice principale de K et $\bar{P}$ une matrice de permutation. Une sous-matrice principale de K est de la forme

$$K_l = \begin{pmatrix} -\bar{H}_l & \bar{A}_l^T \\ \bar{A}_l & \bar{E}_l \end{pmatrix}, \quad (1.60)$$

avec $\bar{H}_l$ et $\bar{E}_l$ les sous-matrices principales des matrices H et E respectivement. D'où K_l est quasi-définie donc inversible d'après la Proposition 1.3.1.
L'unicité de la factorisation est un résultat bien connu [197]. □

[6]Etant donné une matrice carrée B la décomposition $B = LU$ existe si et seulement si les sous-matrices principales B_k de B sont inversibles [154].

Ce résultat est très important et est loin d'être évident. En effet toute permutation d'une matrice symétrique quelconque n'est pas factorisable. Pour l'illustrer considérons la matrice

$$A = \begin{pmatrix} 2 & 1 \\ 1 & 0 \end{pmatrix} \tag{1.61}$$

A n'est pas singulière et possède une factorisation LDL^T avec

$$L = \begin{pmatrix} 1 & 0 \\ \frac{1}{2} & 1 \end{pmatrix} \text{ et } D = \begin{pmatrix} 1 & 0 \\ 0 & -\frac{1}{2} \end{pmatrix}. \tag{1.62}$$

Mais la matrice permutée

$$PAP^T = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 2 & 0 \\ 0 & -\frac{1}{2} \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix} \tag{1.63}$$

ne possède pas de factorisation LDL^T [35].

1.3.3 Stabilité

Avoir une factorisation LDL^T pour toute permutation d'une matrice symétrique quasi-définie bien que for désirable ne garantit en rien la stabilité de ladite factorisation. Gill et *al.* [81] ont développé une formule qui permet de déterminer si une matrice symétrique quasi-définie aura une factorisation stable quelque soit la permutation qu'on lui applique.

Chapitre 2

Un algorithme de points non intérieurs utilisant la technique de régularisation de Chen-Harker-Kanzow-Smale et le Filtre de Markov

Dans ce chapitre, nous présenterons un algorithme de points non intérieurs non admissibles pour trouver un point satisfaisant les conditions d'optimalité du premier ordre des problèmes d'optimisation non linéaires. L'algorithme que nous allons présenter est basé sur une reformulation du système non linéaire issue de l'expression des conditions d'optimalité de karush-Kuhn-Tucker pour résoudre un problème d'optimisation non linéaire. L'algorithme utilise la technique de régularisation de Chen-Harker-Kanzow-Smale et le concept de filtre dans le cadre de la recherche linéaire. Le terme non intérieur vient du fait que dans cet algorithme la positivité des variables d'écart introduites pour transformer les contraintes d'inégalités en contraintes d'égalités n'est pas exigée à chaque itération, contrairement à l'approche classique des méthodes de points intérieurs.

2.1 Introduction

Le problème d'optimisation que nous souhaitons résoudre est le suivant,

$$(\mathcal{PG}) \begin{cases} \min f(x) \\ b \leq c(x) \leq b + r, \\ l \leq \quad x \quad \leq u, \end{cases}$$

avec $l,\ u \in \mathbb{R}^n$ et $b,\ r \in \mathbb{R}^m$. Toutes les composantes b_i de b sont des nombres réels, par contre tous les autres vecteurs sont dans l'ensemble des réels étendu, c'est-à-dire que l'on a,

$$\begin{array}{rcccl} 0 & \leq & r_i & \leq & +\infty, \\ -\infty & \leq & l_i & < & +\infty, \\ -\infty & < & u_i & \leq & +\infty. \end{array}$$

Le cas des bornes infinies sera traité explicitement plus loin dans le chapitre.

Par souci de clarté dans l'exposé, nous commencerons par la présentation de la méthode pour un problème avec uniquement des contraintes d'inégalité. Le cas général ($\mathcal{PG}$) sera abordé un peu plus loin dans le chapitre.
Le problème que nous le problème que nous coonsidérons à présent s'écrit,

$$(\mathcal{P}) \left\{ \begin{array}{l} \min f(x) \\ c(x) \geq 0, \end{array} \right.$$

avec $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ et $c : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ au moins de classe $\mathcal{C}^2$.

Nous suppososerons dans la suite disposer de la *qualification des contraintes* au sens présenté à la Section 1.1.4.

La première étape de l'algorithme consiste à transformer le problème ($\mathcal{P}$) en un problème n'ayant que des contraintes d'égalité.

$$(\mathcal{P}_\omega) \left\{ \begin{array}{l} \min f(x) \\ c(x) - \omega = 0, \\ \omega \geq 0 \end{array} \right.$$

Le vecteur $\omega \in \mathbb{R}^m$ est appelé vecteur des variables d'écart.
contrairement aux approches classiques de points intérieurs, nous n'introduirons pas de terme de pénalisation des contraintes de positivité sur les variables d'écarts dans l'objectif.
Soit

$$L(x, \omega, \lambda, \nu) = f(x) - \lambda^t(g(x) - \omega) - \nu^t\omega$$

le Lagrangien de (P_ω). Les conditions d'optimalité du premier ordre de (P_ω) s'écrivent,

$$(\mathcal{S}_\omega) \left\{ \begin{array}{rcll} \nabla f(x) - \nabla g(x)\lambda & = & 0, & \\ \lambda - \nu & = & 0, & \\ g(x) - \omega & = & 0, & \\ \omega_j\nu_j & = & 0, & \\ \omega_j & \geq & 0, & \\ \nu_j & \geq & 0, & j = 1, 2, \cdots, m. \end{array} \right.$$

où $\lambda = (\lambda_1, \cdots, \lambda_m)^T$ et $\nu = (\nu_1, \cdots, \nu_m)^T$ sont les vecteurs des multiplicateurs de Lagrange associés.
Appliquer une méthode de Newton au système non linéaire (S_ω) est une tâche ardue à cause de la présence des équations $\nu_j\omega_j = 0$, qui traduisent la relation de complémentarité.

Le principe des approches primales-duales est de remplacer le système (S_ω) par le système relaxé du type,

$$(\mathcal{S}_\omega)\begin{cases} \nabla f(x) - \nabla g(x)\lambda &= 0, \\ \omega - g(x) &= 0, \\ 2\omega_j\lambda_j &= \tau^2, \\ \omega_j &> 0, \\ \lambda_j &> 0,\ j = 1, 2, \cdots, m. \end{cases}$$

La résolution du problème (S_ω) est remplacée par la résolution d'une suite de problèmes (S_ω^τ) pour laquelle le paramètre τ tendra asymptotiquement vers 0.
Les équations $2\omega_j\lambda_j = \tau^2$ sont présentées sous cette forme et non sous la forme classique $\omega_j\lambda_j = \tau$ pour faire un lien avec la propriété principale de la fonction de Chen-Harker-Kanzow-Smale(CHKS) que nous introduirons dans la section suivante et qui nous servira à reformuler le système (S_ω^τ).

2.2 Fonction de Chen-Harker-Kanzow-Smale et reformulation

Un point essentiel dans la reformulation des conditions d'optimalité est la fonction utilisée pour réaliser cette reformulation. Plusieurs fonctions présentes dans la littérature remplissent ce rôle voir [127] pour une revue de ces différentes fonctions. Bien que l'étude de chacune d'elles fut faite indépendament des autres, Chen et Mangazarian [130] ont présenté une étude qui tente à unifier ces différentes fonctions. Ils montrent que les fonctions précédemment étudiées sont des classes de fonctions particulières d'une famille de fonctions plus générales. Plus récemment Gabriel et Moré [129] ont proposé une nouvelle famille de fonctions dont la famille de fonctions de Chen et Mangasarian est une classe particulière.

2.2.1 Fonction de Chen-Harker-Kanzow-Smale

Fisher introduisit dans [13, 14] la fonction $\phi : \mathbb{R}^2 \longrightarrow \mathbb{R}$ définie par,

$$\phi(a, b) = \sqrt{a^2 + b^2} - a - b, \tag{2.1}$$

avec a,b,c $\in \mathbb{R}$. A l'origine Burmeister [13] construisit ϕ comme exemple d'une fonction permettant de mesurer la distance au cône $\mathbb{R}_- \times \mathbb{R}_+$ et comportant certaines propriétés intéressantes, voir Fisher [19].

Définition 2.2.1 *Une fonction $F : I\!R^p \longrightarrow I\!R^q$ est Lipschitz-continue dans un voisinage de $z \in I\!R^p$, s'il existe un réel positif υ et une constante L tel que pour tout z' et z'' dans*

$$B(z, \upsilon) = \{w \in I\!R^p \mid \| w - z \| \leq L \| z' - z" \|\},$$

on a

$$\| F(z') - F(z'') \| \leq L \| z' - z" \| .$$

Proposition 2.2.1 *La fonction ϕ est sous-additive et Lipschitzienne, positivement homogène de plus elle possède la propriété suivante,*

$$\phi(a,b) \;=\; 0 \Leftrightarrow a \;\geq\; 0,\; b \;\geq\; 0,\; ab \;=\; 0.$$

Dans la littérature ϕ est appélée fonction de Burmeister-Fisher(BF). Plusieurs chercheurs ont étudié et utilisé (BF) dans leurs travaux. Elle fut par exemple utilisée dans [18, 20, 25, 28, 27, 18, 12, 26, 40]. Son grand inconvénient est de ne pas être différentiable à l'origine.
Pour rémédier à ce problème et dans l'étude des problèmes de complémentarité linéaire, Kanzow [24] introduisit la fonction
$\psi : \mathbb{R}^3 \longrightarrow \mathbb{R}$ définie par,

$$\psi(a,b,c) = \sqrt{a^2+b^2+c^2} - a - b \tag{2.2}$$

avec a,b,c $\in \mathbb{R}$. C'est la fonction de Chen-Harker-Kanzow-Smale(CHKS). Elle a la propriété suivante,

$$\psi(a,b,c) \;=\; 0 \Leftrightarrow a \;>\; 0,\; b \;>\; 0,\; 2ab \;=\; c^2. \tag{2.3}$$

En plus des autres propriétés qu'elle partage avec (BF), elle est également différentiable partout. Nous utiliserons (CHKS) pour notre reformulation dans les paragraphes qui suivent.

2.2.2 Reformulation du système de karush-Kuhn-Tucker et calcul explicite du pas

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 se ramenant au problème (P) [41, 42, 21, 23, 45, 46, 44]. Dans cette approche les conditions d'optimalité de karush-Kuhn-Tucker sont reformulées sous forme d'un système d'équations non linéaires, ce système est approximé par une suite de systèmes d'équations non linéaires en introduisant un ou plusieurs paramètres. Une méthode de type Newton leur est par la suite appliquée et sous certaines conditions les solutions des systèmes approximants convergent asymptotiquement vers la solution du problème d'origine en contrôlant de façon adéquate le ou les paramètres.

Nous suivrons l'approche initiée par Jiang [17], et qui fut par la suite reprise par Kanzow [23]. Cette approche consiste à considérer le paramètre τ comme une variable à part entière et d'ajouter une équation suplémentaire qui a pour but de faire tendre le paramètre τ effectivement vers zéro.
Le système (S_ω^τ) reformulé est,

$$(S_\omega^\tau) \begin{cases} \nabla f(x) - A^T\lambda &= 0, \\ \omega - c(x) &= 0, \\ \Theta(\omega,\lambda,\tau) &= 0, \\ \varphi(\tau) &= 0. \end{cases}$$

où $A = \nabla c(x)$, $\Theta(\omega, \lambda, \tau) = (\psi(\omega_1, \lambda_1, \tau_1), \cdots, \psi(\omega_m, \lambda_m, \tau_m))^T$ et $\varphi : \mathbb{R} \longrightarrow \mathbb{R}$ vérifie les propriétés [22] :

1. φ est continuement différentiable avec $\varphi(0) = 0$.
2. $\varphi'(\tau) > 0$ pour tout $\tau \in [0, +\infty)$.
3. $\varphi(\tau) \leq \varphi'(\tau) \cdot \tau$ pour tout $\tau \in [0, +\infty)$.
4. Pour chaque $\tau_0 > 0$, il existe une constante $\gamma > 0$(qui dépend éventuellement de τ_0) telle que $\varphi(\tau) \geq \gamma \cdot \varphi'(\tau) \cdot \tau$ pour tout $\tau \in [0, +\infty)$.

Notre choix fut de prendre $\varphi(\tau) = \tau$.
On pose

$$\Phi(x, \omega, \lambda, \tau) = \begin{pmatrix} \nabla f(x) - A^T\lambda \\ \omega - c(x) \\ \Theta(\omega, \lambda, \tau) \\ \tau \end{pmatrix}$$

Le but de la méthode est de résoudre

$$\Phi(X) = 0 \tag{2.4}$$

où $X = (x, \omega, \lambda, \tau)^T$. Appliquer la méthode de Newton pour résoudre (2.4) revient à résoudre une suite de systèmes linéaires

$$\nabla\Phi(X)\triangle X = -\Phi(X). \tag{2.5}$$

Dans notre approche le second membre du système linéaire est une généralisation de la fonction Φ. La généralisation de la fonction Φ considérée est la suivante,

$$\Phi_\varrho(x, \omega, \lambda, \tau) = \begin{pmatrix} \nabla f(x) - A^T\lambda \\ \omega - c(x) \\ \Theta(\omega, \lambda, \tau) \\ \varrho\tau \end{pmatrix}$$

avec $0 < \varrho < 1$. Ce choix permet de faire décroître modérément τ à chaque itération. Le système linéaire que nous résolvons à chaque itération est alors,

$$\nabla\Phi(X)\triangle X = -\Phi_\varrho(X), \tag{2.6}$$

soit

$$\begin{pmatrix} H & 0 & -A^T & 0 \\ -A & I & 0 & 0 \\ 0 & D_\omega & D_\lambda & d\tau \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \triangle x \\ \triangle\omega \\ \triangle\lambda \\ \triangle\tau \end{pmatrix} = - \begin{pmatrix} \sigma \\ \rho \\ \Theta \\ \varrho\tau \end{pmatrix}$$

où D_ω, D_τ et D_λ sont des matrices diagonales dont les éléménts diagonaux sont respectivement définis par,

$$\begin{aligned} D_\omega^i &= \frac{w_i}{\sqrt{\omega_i^2+\lambda_i^2+\tau^2}} - 1, \\ D_\lambda^i &= \frac{\lambda_i}{\sqrt{\omega_i^2+\lambda_i^2+\tau^2}} - 1, \\ d_\tau^i &= \frac{\tau}{\sqrt{\omega_i^2+\lambda_i^2+\tau^2}}, \end{aligned}$$

et

$$\left\{ \begin{array}{lcl} H &=& \nabla^2 f(x) - \sum_{i=1}^{m} \lambda_i \nabla^2 c_i(x), \\ \sigma &=& \nabla f(x) - A^T \lambda, \\ \rho &=& \omega - c(x), \\ \Theta &=& \Theta(\omega, \lambda, \tau). \end{array} \right.$$

Comme on peut le voir, ρ mesure l'infaisabilité primale et σ l'infaisabilité duale.

Par substitution nous allons progressivement transformer le système (2.6) en un système où la seule inconnue est $\triangle x$.
En effet la quatrième équation nous permet d'exprimer $\triangle \tau$

$$\triangle \tau = -\varrho \tau, \tag{2.7}$$

la troisième donne,

$$\triangle \omega = -W \triangle \lambda - D_\omega^{-1} \triangle \tau - D_\omega^{-1} \Theta$$

avec $W = D_\omega^{-1} D_\lambda^{-1}$.
On obtient après substitutions le système linéaire,

$$\begin{pmatrix} -H & A^T \\ A & W \end{pmatrix} \begin{pmatrix} \triangle x \\ \triangle \lambda \end{pmatrix} = \begin{pmatrix} \sigma \\ \gamma \end{pmatrix},$$

où $\gamma = D_\omega^{-1}(-\triangle \tau d\tau + \Theta) + \rho$.
Grâce à la deuxième équation on exprime $\triangle \lambda$ en fonction de $\triangle x$ on remplace $\triangle \lambda$ ainsi exprimé dans la première équation et on a plus qu'un système linéaire symétrique où $\triangle x$ est l'inconnue.
On pose

$$\begin{aligned} \mathcal{A} &= H + A^t W^{-1} A, \\ \mathcal{B} &= A^t (W^{-1} \rho + D_\lambda^{-1} (\Theta - \triangle \tau d\tau). \end{aligned}$$

On doit résoudre le système linéaire

$$\mathcal{A} \triangle x = \mathcal{B}, \tag{2.8}$$

puis successivement on obtient,

$$\begin{aligned} \triangle \omega &= A \triangle x - \rho, \\ \triangle \lambda &= -W^{-1} \triangle \omega - D_\lambda^{-1} (\Theta - \triangle \tau d\tau). \end{aligned} \tag{2.9}$$

$\triangle x$, $\triangle \omega$, $\triangle \lambda$ et $\triangle \tau$ déterminés grâce à (2.8), le nouvel itéré est

$$\begin{aligned} x_+ &= x + \alpha \triangle x, \\ \omega_+ &= \omega + \alpha \triangle \omega, \\ \lambda_+ &= \lambda + \alpha \triangle \lambda, \\ \tau_+ &= \tau + \alpha \triangle \tau, \end{aligned}$$

où α est la longueur du pas à déterminer. Pour cela nous utiliserons le concept de filtre que nous présenterons à la Section 2.4.

2.3 Solution du système linéaire

Tout le travail réside dans la résolution du système linéaire (2.8) ; dans le cas où la matrice N est définie positive nous utilisons un gradient conjugué préconditionné [196, 191]. Le préconditionneur utilisé est le préconditionneur MILU [192]. Dans le cas indéfini une matrice diagonale R est ajoutée à la matrice N de sorte que la matrice $N + R$ soit définie positive. L'approches adoptée pour le calcul de la matrice R est la suivante : nous cherchons μ tel que $R = \mu I$ où le paramètre μ est calculé en effectuant une factorisation de Cholesky LDL^T de la matrice N, puis on regarde si tous les éléments diagonaux de la matrice D sont égaux à $+1$, dans ce cas aucune perturbation de la matrice N n'est nécéssaire et μ est égale à 0, sinon on part de $\mu = 1.5$ si cette perturbation est insuffisante on multiplie μ par 2 jusqu'à avoir une perturbation qui rend $N + \mu I$ définie positive. Sinon on divise μ par deux tant que $N + \mu I$ est définie positive. Cette heuristique peut s'avérer coûteuse surtout pour des problèmes de grande taille.
Notons que d'autres approches existent dans la littérature qui permettent de rendre la matrice $N + R$ définie positive si N ne l'est pas. Citons par exemple l'algorithme de Cholesky modifié de Shnabel et Eskow [11]. Dans leur approche la matrice R vérifie :
– si N est définie positive R est la matrice nulle,
– si N es indéfinie alors $\| R \|_\infty \leq \lambda_1(N)$ où $\lambda_1(N)$ est la plus petite valeur propre de N.
– $N + R$ est assez bien conditionnée.
– Le coût de la factorisation est un petit multiple de n^2 opérations en plus des $\mathcal{O}(n^3)$ du coût d'une factorisation de Cholesky classique.

2.4 Notion de filtre en optimisation

Les méthodes de pénalité combinent deux problèmes d'optimisation, la minimisation de la fonction objectif

$$\min_{x \in \mathbb{R}^n} f(x), \tag{2.10}$$

et la minimisation de la violation des contraintes

$$\min_{x \in \mathbb{R}^n} \rho(x) \tag{2.11}$$

où $\rho(x)$ est défini par $\rho(x) = \max\{-\min_{i=1,2,\cdots,m} c_i(x), 0\}$. Cet objectif est réalisé en combinant linéairement la fonction objectif et une mesure de la violation des contraintes afin de pénaliser la violation des contraintes. Un exemple de fonction de pénalité est,

$$\Psi_\beta(x) = f(x) + \beta\rho(x), \tag{2.12}$$

où β est le paramètre de pénalité. Les méthodes de pénalité consistent à montrer que les sous-problèmes sans contrainte

$$\min_{x\in\mathbb{R}^n} \Psi_{\beta_k}(x) \tag{2.13}$$

approchent une solution $x^\star$ du problème (P) lorsque la suite des paramètres β_k tend vers l'infini.
Deux problèmes se posent pour ce type de méthodes : le premier est le choix de la suite de paramètre β_k, le second le choix du paramètre initial β_0.
Pour s'affranchir des difficultés liées au paramètre de pénalité, Fletcher et Leyffer [16] proposèrent en 1997 un nouveau concept le *filtre*. Fletcher et Leyffer définirent le filtre $\mathcal{F}$ comme un ensemble de paires de points $(f(x_i), \rho(x_i))$ telles que

$$\rho(x_i) \;\leq\; \rho(x_j) \quad ou \quad f(x_i) \;\leq\; f(x_j) \quad pour \quad i \neq j.$$

La construction du filtre repose sur un concept emprunté à l'optimisation multi-critères. On dit qu'un point x_1 *domine* un point x_2 si

$$\rho(x_1) \;\leq\; \rho(x_2) \quad et \quad f(x_1) \;\leq\; f(x_2).$$

Un nouvel itéré x_i est accepté dans le filtre s'il n'existe aucun autre point dans le filtre qui le domine.
La Figure 2.1 illustre le concept de filtre en représentant la paire $(f(x_k), \rho(x_k))$ comme des points à la base des angles droits définis par des demi-droites dans l'espace $(f(x), \rho(x))$. Chacune de ces paires est appelée paire (f, ρ) associée à x_k. Les demi-droites partant de chaque paire (f, ρ) indiquent que tout itéré dont la paire associée (f, ρ) se situe au dessus et à la droite de la paire associée d'un point du filtre est dominée par cette paire (f, ρ).
Bien que l'idée de ne pas accepter dans le filtre des points dominés soit naturelle et élégante, celle-ci fut tout de même rafinée. En effet il n'est pas souhaitable d'ajouter dans le filtre des points dont les paires associées sont proches des paires déjà présentes dans le filtre. D'où l'introduction d'une petite marge le long de la frontière définissant la zone des points dominés, tout nouvel itéré apparaissant dans cette zone est rejeté. Un point x est ajouté dans le filtre si et seulement si

$$\rho(x) \;\leq\; (1-\gamma_\rho)\rho(x_j) \quad ou \quad f(x) \;\leq\; f(x_j) - \gamma_\rho\rho(x_j) \quad pour \quad tout \quad (f(x_j), \rho(x_j)) \in \mathcal{F},$$

pour un certain $0 < \gamma_\rho < 1$. Sur la Figure 2.1 l'ensemble des points acceptables dans le filtre se trouve au dessous de la ligne en pointillés. L'on ne passe donc d'un itéré x_k à un itéré $x_k + \triangle x$ que si $x_k + \triangle x$ est accepté dans le fitre $\mathcal{F}$. Dans [16] Fletcher et Leyffer ont adopté la méthode de programmation quadratique successive(PQS) utilisant l'approche

région de confiance comme technique de globalisation pour résoudre (P). A l'itéré x_k, dans cette approche, le pas $\triangle x$ est déterminé en résolvant le problème quadratique

$$(QP_k)\begin{cases}\min \frac{1}{2}\triangle x^T W_k \triangle x + \nabla f(x_k)^T \triangle x \\ A_k \triangle x + c(x_k) \geq 0 \\ \parallel \triangle x \parallel_\infty \leq r_k,\end{cases} \tag{2.14}$$

où W_k est une matrice symétrique qui approxime le Hessien du Lagrangien associé au problème (P), A_k la jacobienne des contraintes en x_k et r_k le rayon de confiance. L'utilisation de la norme l_∞(au lieu de la norme l_2) pour définir la région de confiance dans le sous-problème (QP_k) a pour but de conserver la nature du sous-problème (QP_k) comme problème quadratique.

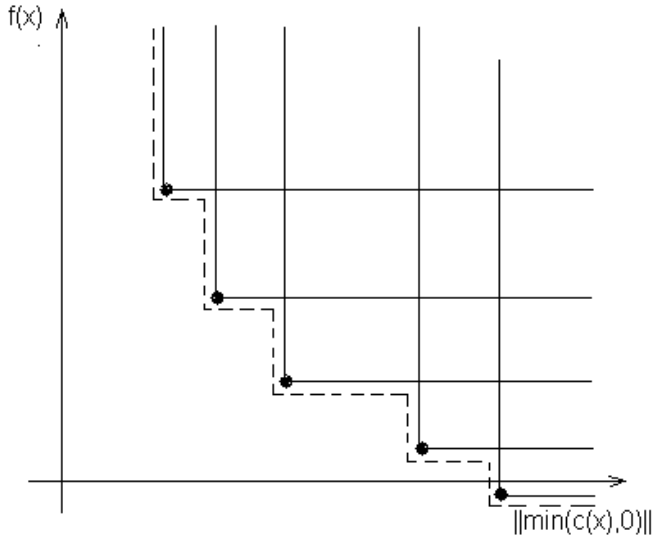


FIG. 2.1 – Exemple de filtre à cinq éléments.

Pendant le processus d'optimisation, si un itéré x_k est acceptable dans le filtre on l'ajoute en ajoutant la paire associée $(f(x_k), \rho(x_k))$ au filtre, et en y retirant toute autre paire $(f(x_j), \rho(x_j))$ telles que

$$\rho(x_j) \quad \geq \quad (1-\gamma_\rho)\rho(x_k) \quad et \quad f(x_j) \quad \geq \quad f(x_k) - \gamma_\rho \rho(x_k).$$

Cette opération s'appelle ajout de x_k dans le filtre, même si en réalité c'est le couple $(f(x_k), \rho(x_k))$ qui est ajouté dans le filtre.
Dans la première version du filtre, Fletcher et Leyffer n'ont produit aucun résultat de convergence de leur algorithme. En 1998, Fletcher Leyffer et Toint [136] prouvèrent la convergence globale de l'algorithme présenté en [16]. Cette convergence fut prouvée sous l'hypothèse entre

autres de disposer de la solution globale du problème (QP_k). Cette condition étant difficile à remplir, une version quasi-Newtonienne de l'algorithme fut suggérée. Cependant les auteurs affirment avoir obtenu des résultats au moins aussi prometteurs que ceux reportés dans [16] en utilisant le Hessien exact. Toujours la même année Fletcher, Leyffer [147] proposèrent un algorithme de type programmation linéaire successive(PLS) dans lequel la fonction objectif est remplacée par son approximation linéaire, la technique de région de confiance associée au filtre fut également utilisée et la convergence globale de l'algorithme fut établie. En 1999, Fletcher, Leyffer, Gould et Toint [137] proposèrent un algorithme toujours de type PQS avec région de confiance. Dans cette nouvelle version, le pas est décomposé en deux composantes, une normale aux contraintes et l'autre tengentielle. La convergence globale de l'algorithme fut également prouvée.

En 2003 Gould et Toint [138] ont proposé un algorithme hybride de région de confiance de type PQS-Filtre pour des problèmes non linéaires. Cet algorithme s'inspire en grande partie de celui présenté dans [137] ; la différence est l'introduction d'une alternative de pas potentiellement moins chère à calculer. Malgré cet attrait théorique aucun résultat numérique n'est fourni. Toujours dans la lignée de l'algorithme présenté dans [137], Toint et *al*. [149] ont présenté un nouvel algorithme de région de confiance utilisant la technique du filtre pour des problèmes non linéaires sans contraintes. Des résultats numériques montrent que cet algorithme ne conduit pas nécessairement à un gain important en matière de temps de calcul bien que le nombre d'évaluations de la fonction objectif et de ses dérivées soit moindre comparé aux algorithmes de région de confiance classiques. Ulbrich et Ulbrich [141] proposèrent un algorithme de région de confiance non monotone basé sur l'idée du filtre ; la convergence globale de l'algorithme fut établie. Un algorithme de points intérieurs utilisant la technique du filtre fut proposé par Ulbrich, Ulbrich et Vicente [140] dans cette approche le pas est également décomposé en une composante normale et en une autre tengentielle, seulement ici cette décomposition est faite à travers une décomposition du second membre du système linéaire issue de la linéarisation des conditions d'optimalité du premier ordre du sous-problème barrière. Aucun résultat numérique n'est reporté pour cette approche. Dans la même famille de ces méthodes Vanderbei et *al*. [37], exploitèrent l'idée du filtre et introduisirent une variante du filtre de Fletcher et Leyffer, le filtre de Markov. La difficulté essentielle rencontrée lors de l'adaptation du filtre de Fletcher et Leyffer dans le contexte des méthodes de points intérieurs réside dans la présence du paramètre barrière, car celui-ci peut être amené à varier de manière non monotone. Une solution serait d'imaginer une procédure pour faire décroître le paramètre barrière de manière monotone mais les auteurs dans [37] rapportent qu'une telle procédure conduit à un algorithme moins robuste. La seconde solution fut une modification de l'idée originale de Fletcher et Leyffer. Le filtre de Markov n'est pas vraiment un filtre au sens défini par Fletcher et Leyffer, car pour cette variante aucune information sur les itérations précédentes n'est conservée. Les résultats numériques reportés furent meilleurs que ceux de la version précédente de leur algorithme dans lequel une recherche linéaire avec fonction de mérite l_2 est utilisée, aucun résultat de convergence n'est fourni. Toujours dans la famille des méthodes de points intérieurs Wächter et Biegler ont axé leurs travaux à l'adaptation du concept de filtre à leurs algorithmes de points intérieurs [150, 151, 152]. D'autres adaptations du filtre à d'autres techniques

d'optimisation ont également été explorées [153, 155, 156].
Une analyse théorique des approches région de confiance et recherche linéaire utilisant la technique du filtre est faite dans [148].

2.5 Méthode de recherche linéaire avec filtre

Dans notre approche, la longueur du pas α acceptée est celle pour laquelle on a une réduction soit de la fonction objectif soit de la violation des contraintes exactement comme dans la version d'origine du filtre présentée par Fletcher et Leyffer. Ceci grâce à la structure de notre algorithme pour lequel garantir la positivité des variables d'écart et des variables duales du problème (P) n'est pas nécessaire à chaque itération. Cette liberté nous a affranchi plus haut de pénaliser les variables d'écart dans un terme barrière ajouté à la fonction objectif et maintenant elle nous permet d'appliquer le filtre de Fletcher et Leyffer sans le modifier. Cependant dans notre approche nous ne conservons qu'une seule information celle de l'itéré précédent comme cela est fait dans [37]. Cette variante a l'avantage d'être plus aisée à mettre en œuvre que la version d'origine de Fletcher et Leyffer. Ainsi partant de $\alpha = 1$ nous allons diviser α par deux jusqu'à avoir l'une des conditions suivantes satisfaites

$$\begin{array}{rcl} \theta(x_k + \alpha\triangle x, \omega_k + \alpha\triangle\omega, \tau_k + \alpha\triangle\tau) & \leq & (1-\gamma_\theta)\theta(x_k, \omega_k, \tau_k) \\ & ou & \\ f(x_k + \alpha\triangle x) & \leq & f(x_k) - \gamma_\theta\theta(x_k, \omega_k, \tau_k) \end{array} \tag{2.15}$$

avec γ_θ défini comme plus haut et

$$\theta(x, \omega, \tau) = \begin{pmatrix} c(x) - \omega \\ \phi(\tau) \end{pmatrix}.$$

Notons que la condition (2.15) est différente de celle utilisée dans [37] où une règle du type Armijo est adoptée.

2.6 Algorithme de points non intérieurs

Les lignes qui suivent présentent les principales étapes de l'algorithme de points non intérieurs pour résoudre le problème (P).

Algorithme 2.6.1 *Algorithme de points non intérieurs (NIPA)*

1. *Choisir x_o, w_o, τ_o, λ_o, et une tolérance ε.*
2. *Déterminer le pas $\triangle x_k, \triangle w_k, \triangle\tau_k$ et $\triangle\lambda_k$, grâce à* (2.7)-(2.9).
3. *Déterminer la longueur de pas α_k en utilisant* (2.15)*, si un pas α_k vérifiant* (2.15) *n'a pas pu être trouvé alors stop, sinon poser*

$$\begin{array}{rcl} x_{k+1} & = & x_k + \alpha_k\triangle x, \\ \omega_{k+1} & = & \omega_k + \alpha_k\triangle\omega, \\ \lambda_{k+1} & = & \lambda_k + \alpha_k\triangle\lambda, \\ \tau_{k+1} & = & \tau_k + \alpha_k\triangle\tau, \end{array}$$

4. Si $\| \Phi(x_{k+1}, w_{k+1}, \tau_{k+1}, \lambda_{k+1}) \| \leq \varepsilon$ *alors stop, sinon* $k = k + 1$ *retourner à l'étape 2.*

2.7 Traitement des bornes et des variables libres

Pour le traitement des bornes, comme pour le problème (P), la première étape de la méthode est l'introduction de variables d'écart. Le problème $(\mathcal{PG})$ s'écrit,

$$\begin{cases} \min f(x) \\ \begin{array}{lcl} g(x) - b - \omega & = & 0 \\ x - l - y & = & 0 \\ x - u + z & = & 0 \\ \omega - r + p & = & 0 \\ \omega, p, y, z, & \geq & 0, \end{array} \\ x \in \mathbb{R}^n. \end{cases} \tag{2.16}$$

On applique à ce problème la méthode décrite plus haut.
Pour le traitement des variables libres, c'est-à-dire les variables dont les bornes sont infinies, nous suivrons l'approche introduite par Vanderbei [34] et que nous rappellerons dans les lignes qui suivent.
Dans notre problème d'origine

$$\begin{cases} \min f(x) \\ b \leq g(x) \leq b + r \end{cases} \tag{2.17}$$

où toutes les variables sont libres, si l'on voit une variable libre comme une variable ayant une très grande borne supérieure S et une très petite borne inférieure $-S$, alors le problème (2.17) devient,

$$\begin{cases} \min f(x) \\ \begin{array}{lcl} g(x) - b - \omega & = & 0 \\ x + S - y & = & 0 \\ x - S + z & = & 0 \\ \omega - r + p & = & 0 \\ \omega, p, y, z & \geq & 0, \end{array} \\ x \in \mathbb{R}^n. \end{cases} \tag{2.18}$$

En additionnant et soustrayant les équations où S intervient on obtient

$$\begin{cases} \begin{array}{lcl} 2x - y + z & = & 0 \\ y + z - 2S & = & 0. \end{array} \end{cases} \tag{2.19}$$

Le but du deuxième bloc d'équations consiste à forcer z ou y à tendre vers l'infini. Si l'on abandonne ce bloc d'équations et l'on retire le 2 qui multiplie x, le problème devient,

$$\left\{\begin{array}{ll} \min f(x) & \\ g(x)-b-\omega & = 0 \\ x-y+z & = 0 \\ \omega-r+p & = 0 \\ \omega,p,y,z, & \geq 0, \\ x\in \mathbb{R}^n. & \end{array}\right. \tag{2.20}$$

On applique à ce problème la méthode décrite plus haut.

2.8 Détermination du point intial

La stratégie adoptée consiste tout d'abord à se donner un $\bar{x}^\circ$ puis à résoudre le système linéaire,

$$\begin{pmatrix} -Q(\bar{x}^\circ) & A(\bar{x}^\circ)^T \\ A(\bar{x}^\circ) & \delta I_m \end{pmatrix} \begin{pmatrix} x \\ \lambda \end{pmatrix} = \begin{pmatrix} f(\bar{x}^\circ) \\ g(\bar{x}^\circ) \end{pmatrix} \tag{2.21}$$

où $Q(\bar{x}^\circ) = \nabla^2 f(\bar{x}^\circ)$ et δ un nombre positif. Le choix de δ est déterminant pour la performance de la méthode $\delta = 1$ nous est apparu comme étant un bon choix lors des tests numériques. Le système (2.21) est par la suite réduit en tirant λ du second bloc d'équations puis remplaçant cette valeur dans le premier bloc, le système résultant nous permet de déterminer x. On pose (x°, λ°) la solution du système (2.21), alors w° est déterminé en posant $w^\circ = \max\{g(x^\circ), \eta\}$ et τ° en posant $\tau^\circ = \sqrt{\max\{2w^T\lambda, \eta\}}$, où η est choisi dans $\{0.1, 1, 10, 100\}$.

2.9 Résultats numériques

L'algorithme présenté dans ce chapitre fut implémenté en C et en algèbre dense. Dans les tests numériques qui suivent les dérivées premières et secondes sont calculées en utilisant des techniques de différences divisées.

2.9.1 Etude de l'exemple de Wächter et Biegler

Le premier exemple que nous traitons est un exemple donné par Wächter et Biegler [39] pour lequel les méthodes de points intérieurs classiques [31, 61, 56, 31] échouent.

$$\left\{\begin{array}{ll} \max f(x) & \\ x^2+a & \geq 0 \\ x-b & \geq 0 \end{array}\right. \tag{2.22}$$

avec a, $b \in \mathbb{R}$ et $b \geq 0$. La fonction f est quelconque mais supposée suffisamment régulière. Nous prendrons $f(x) = x$ et $a = -1$ et $b = 2$. Ce problème est bien posé et pas pathologique.

Il n'est pas pathologique au sens où la jacobienne des contraintes d'égalité du problème

$$\begin{cases} \max\ f(x) \\ x^2 - \omega_1 - 1 = 0 \\ x - \omega_2 - 2 = 0 \end{cases} \tag{2.23}$$

est surjective quelque soit le point $X = (x, \omega_1, \omega_2)$. La solution $x^\star = 2$ de ce problème est l'optimum global. De plus pour $a \neq -b^2$, les conditions suffisantes du second ordre et la stricte complémentarité sont vérifiés en $x^\star$. En appliquant une méthode de point intérieurs s'inspirant de l'une des approches [31, 61, 56] avec point initial $(-2, 1, 1)$, on montre que la première direction primale de la recherche linéaire, qui satisfait la linéarisation des contraintes d'égalité de (2.23) peut être exprimée paramétriquement de manière suivante

$$\begin{pmatrix} \triangle x \\ \triangle \omega_1 \\ \triangle \omega_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 2 \\ -5 \end{pmatrix} + \gamma \begin{pmatrix} 1 \\ -4 \\ 1 \end{pmatrix}$$

avec $\gamma \in \mathbb{R}$. Dans ce cas le pas unité de la recherche linéaire ne peut jamais être accepté pour tout γ, car le point

$$X_0 + \triangle X = \begin{pmatrix} -2 \\ 1 \\ 1 \end{pmatrix} + \begin{pmatrix} 0 \\ 2 \\ -5 \end{pmatrix} + \gamma \begin{pmatrix} 1 \\ -4 \\ 1 \end{pmatrix}$$

violant inévitablement les contraintes de positivité des variables d'écart $\omega_1 \geq 0$ et $\omega_2 \geq 0$ et ceci pour toute valeur du paramètre γ. Ainsi le pas sera choisi dans]0,1[. Et on montre que le nouvel itéré restera toujours dans l'ensemble

$$\mathcal{S} = \{(x, \omega_1, \omega_2)\ :\ x \leq -\sqrt{\omega_1 + 1},\ \omega_1 \geq 0,\ \omega_2 \geq 0\}$$

et ne sera donc pas admissible. Cet argument pourra être repris pour les itérés suivants. Le pas unité ne sera jamais accepté et les itérés jamais admissibles. De plus Wächter et Biegler montrent pour le problème (2.22) que la suite générée convergera vers un point $(x^\star, \omega_1^\star, \omega_2^\star) = (-\nu, 0, 0)$, avec $\nu > 0$.

Cet exemple tente de prouver que toutes les méthodes primales-duales calculant une direction satisfaisant les contraintes linéarisées et un pas α assurant la stricte positivité des variables d'écart, produiront une suite de points non admissibles. Wächter et Biegler remarquent que le choix d'une fonction de mérite et celle dans ces approches de la gestion du paramètre barrière n'ont aucune influence sur le phénomène. Ils montrent également que le point $(-\nu, 0, 0)$, avec $\nu > 0$ ne satisfait jamais l'optimalité d'une mesure d'admissibilité de la forme $\| c(x) \|_p$, $1 \leq p \leq \infty$.
Notre algorithme, s'il s'inspire du schéma d'un algorithme de points intérieurs classique n'est pas un algorithme de points intérieurs. En effet comme on a pu le voir plus haut aucune exigence n'est faite sur la positivité des variables d'écart à chaque itération, ceci

iter	f(x)	$(\omega_1\ ,\ \omega_2)$	$(\lambda_1,\ \lambda_2)$
1	-2.00e+000	(1.00e+000, 1.00e+000)	(5.00e-001, 5.00e-001)
2	-2.00e+000	(1.00e+000, 1.00e+000)	(5.00e-001, 5.00e-001)
3	-1.81e+000	(7.10e-001, 0.00e+000)	(4.38e-001, 6.59e-001)
4	-1.58e+000	(0.00e+000, -4.75e-002)	(6.57e-001, 1.39e+000)
5	-4.59e-001	(-2.31e+000, -1.85e+000)	(1.32e+002, 4.15e+002)
6	1.67e+000	(-2.85e+000, -2.84e-001)	(1.78e+004, 1.58e+004)
7	1.67e+000	(-2.85e+000, -2.83e-001)	(0.00e+000, 7.55e+004)
8	1.84e+000	(0.00e+000, -1.43e-001)	(-2.87e+000, 3.80e+004)
9	2.00e+000	(2.97e+000, 0.00e+000)	(-1.49e+000, 7.40e+000)
10	2.00e+000	(3.00e+000, 7.70e-007)	(-2.19e-001, 1.87e+000)
11	2.00e+000	(3.00e+000, 4.48e-007)	(-2.78e-017, 1.00e+000)
12	2.00e+000	(3.00e+000, 4.48e-007)	(0.00e+000, 1.00e+000)
13	2.00e+000	(3.00e+000, 1.74e-009)	(2.25e-014, 1.00e+000)

TAB. 2.1 – Résultats numériques pour le problème de Wächter et Biegler.

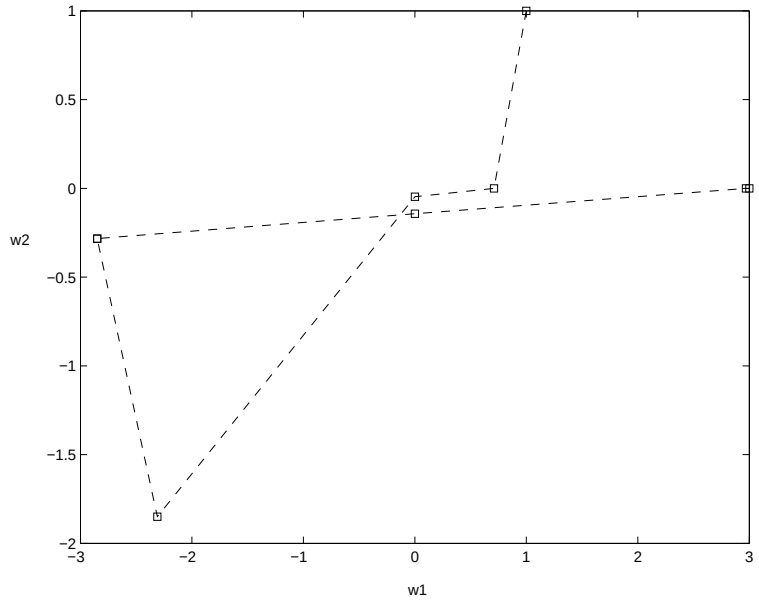


FIG. 2.2 – Progression des variables d'écart pour le problème de Wächter et Biegler.

peut être vu sur la Figure 2.2. Cependant nous savons que la propriété (2.3) est vérifiée à l'optimum ce qui nous garantit cette positivité. La Figure 2.3 montre l'évolution des relations de complémentarité $\omega_i\lambda_i$, $i = 1, 2$ en fonction des itérations. NIPA résout le problème de Wächter et Biegler en treize itérations, le Tableau 2.1 présente l'historique de ces itérations.

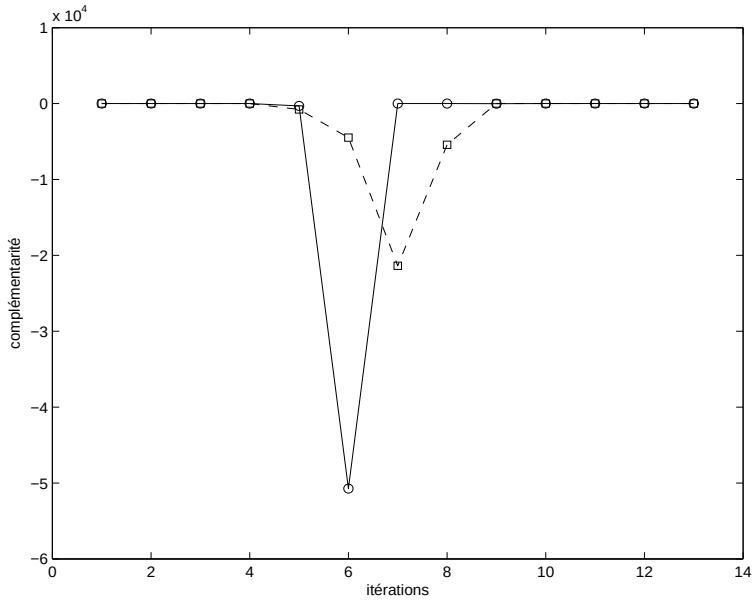


FIG. 2.3 – Progression de la relation de complémentarité en fonction du nombre d'itérations pour le problème de Wächter et Biegler. En pointillés $\omega_1\lambda_1$ et en continu $\omega_2\lambda_2$.

2.9.2 Autres problèmes test.

Les autres problèmes tests sont tirés de la collection Hock-Schittkowski [38] et de la collection CUTE [160]. Dans le tableau ci-dessous la colonne HS comporte les numéros des problèmes comme dans [38]. Les points initiaux sont ceux donnés dans [38].
Dans les tableaux nous avons les notations suivantes

- *iter* le nombre d'itérations,
- $kkt = \max(\| \sigma \|, \| \rho \|)$. Avec σ et ρ comme définis au début du chapitre,
- $f(x^\star)$ la valeur de la fonction objectif,
- n et m le nombre de variables et le nombre de contraintes respectivement.

Les durées d'optimisation ont été volontairement omises en raison de leur trop petites valeurs vu les dimensions des problèmes traités.
Les résultats reportés sont comparables aux résultats de référence dans [38].

Les résultats des problèmes de la collection CUTE sont reportés au Tableau 2.3.

Nous considérons la famille de problèmes de contrôle optimal OPTCNTRL [161]. Les inconnues du problème sont les variables d'état x et y, et la variable de contrôle u, chacune

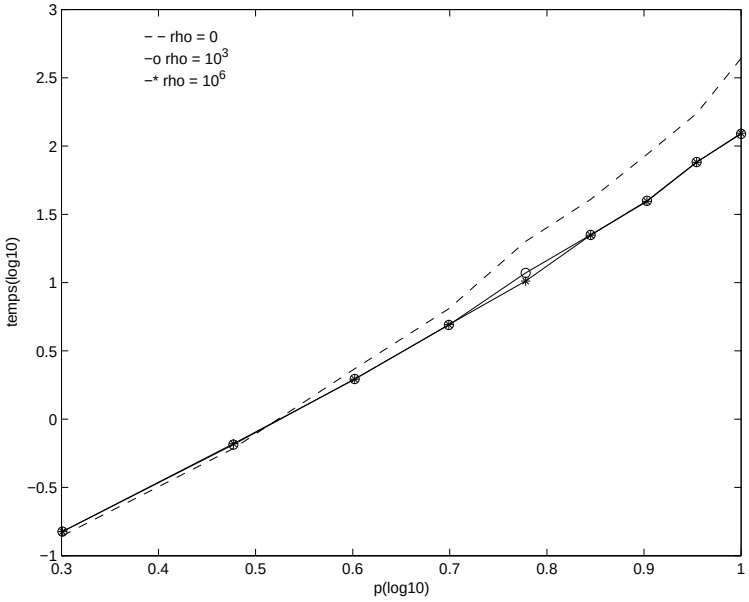


FIG. 2.4 – Temps en fonction de la dimension.

discrétisée en p points.

$$\left\{\begin{array}{l} \min \frac{1}{2}\sum_{i=0}^{p} x_i^2 + \rho \sum_{i=1}^{p-1} (u_i - 0.2)^2 \\ \qquad\begin{array}{rcll} x_{i+1} & = & x_i + 0.2y_i, & \text{pour } \ i = 0, 1, \cdots, p-1, \\ y_{i+1} & = & y_i - 0.01y_i^2 - 0.004x_i + 0.2u_i, & \text{pour } \ i = 0, 1, \cdots, p-1, \\ x_0 & = & 10, & \\ y_0 & = & 0, & \\ y_p & = & 0. & \end{array} \end{array}\right.$$

avec $\rho = 0, 10^3, 10^6$. Ce problème a $n = 3p+2$ variables et $m = 2p+3$ contraintes. Nous nous attarderons sur l'évolution du temps de calcul en fonction de la dimension du problème ; la Figure 2.4 représente cette évolution. On y observe que le temps de résolution est plus grand pour $\rho = 0$ comparé aux deux autres valeurs de ρ, nous n'avons pas d'explication pour le moment sur ce fait ; on se serait attendu à l'inverse car plus ρ est grand, plus le Hessien de l'objectif est mal conditionné. On observe également un temps de calcul moyen assez élevé pour chacun des trois problèmes ; ceci est dû d'une part à l'utilisation des différences divisées pour le calcul des dérivées et d'autre part à la procédure de résolution du système linéaire (2.8) qui peut s'avérer coûteuse dès que la dimension du problème devient grande.

2.10 Conclusion

Nous avons présenté un nouvel algorithme de points non intérieurs. Les deux principaux ingrédients de cet algorithme sont la fonction de Chen-Harker-Kanzow-Smale qui sert à la

reformulation des conditions d'optimalité et le concept de filtre intégré dans une recherche linéaire. L'attrait essentiel de cet algorithme est la limitation des paramètres à gérer, nous n'avons par exemple pas de paramètre barrière à gérer dans notre algorithme car ce paramètre est considéré comme une variable à part entière. De plus, l'algorithme peut générer des itérés négatifs pour les variables d'écart ce qui, combiné au concept de filtre, permet potentiellement de prendre des pas plus grands à chaque itération, comparé aux méthodes de points intérieurs classiques où la longueur du pas doit être prise pour garantir la positivité des variables d'écart à chaque itération. Ce travail est une première étape des recherches restent à faire sur le plan théorique en particulier au niveau de sa convergence.

HS	n	m	$iter$	$f(x^\star)$	kkt
1	2	0	15	6.40e-011	4.87e-013
2	2	0	9	4.94e+000	1.48e-009
3	2	0	7	7.27e-005	4.64e-007
4	2	0	6	2.67e+000	3.98e-007
5	2	0	21	-1.91e+000	1.57e-007
6	2	1	8	4.68e-013	7.06e-007
7	2	1	18	-1.73e+000	4.67e-008
8	2	2	7	-1.00e+000	9.12e-007
9	2	1	9	-5.00e-001	7.32e-007
10	2	1	18	-1.00e+000	1.96e-009
11	2	1	7	-8.50e+000	6.21e-007
12	2	1	14	-3.00e+001	3.55e-010
13	2	1	12	1.98e-001	7.08e-008
14	2	2	9	1.39e+000	3.03e-008
15	2	2	13	3.07e+000	6.16e-007
16	2	2	10	2.50e-001	1.16e-008
17	2	2	13	1.00e-002	1.35e-007
18	2	2	9	5.00e+000	7.99e-008
19	2	2	17	-6.96e-001	1.02e-007
20	2	3	9	4.02e-001	8.74e-007
21	2	1	2	-1.00e+002	2.35e-008
22	2	2	8	1.00e+000	1.77e-007
23	2	5	10	2.00e+000	8.65e-009
24	2	3	10	-1.00e+000	3.6e-008
25	3	0	2	7.68e-029	2.15e-014
26	3	1	18	3.13e-012	9.98e-007
27	3	1	14	4.00e+000	4.08e-009
28	3	1	9	3.64e-019	2.31e-009
29	3	1	10	-2.26e+001	6.61e-008
30	3	1	7	1.00e+000	1.24e-007

Suite du tableau à la page suivante...

HS	n	m	$iter$	$f(x^\star)$	kkt
31	3	1	9	6.00e+000	6.77e-007
32	3	2	12	1.00e+000	3.59e-007
33	3	2	20	-4.59e+000	2.07e-007
34	3	2	11	-8.34e-001	2.65e-008
35	3	1	6	1.11e-001	5.79e-011
36	3	1	7	-3.30e+003	1.12e-008
37	3	2	15	-1.49e-013	4.31e-010
38	4	2	28	1.90e-012	1.67e-009
39	4	2	15	-1.00e+000	6.96e-008
40	4	3	17	-2.50e-001	5.51e-007
41	4	1	15	2.00e+000	1.51e-008
42	4	1	7	1.39e+001	3.15e-009
43	4	3	23	-4.19e+001	6.38e-007
44	4	6	8	-1.30e+001	4.09e-011
45	5	1	8	1.00e+000	8.24e-008
46	5	2	20	6.20e-012	5.82e-008
47	5	3	18	3.17e-011	6.79e-007
48	5	2	16	1.18e-011	3.90e-007
49	5	2	17	1.39e-009	9.14e-007
50	5	3	10	1.06e-015	1.06e-007
51	5	3	6	4.09e+000	1.83e-007
52	5	3	5	4.82e+000	7.21e-007
53	5	3	5	3.82e+000	3.31e-007
54	6	1	5	1.94e-001	2.58e-007
55	6	6	8	6.33e+000	2.24e-007
56	7	4	19	-3.83e-017	9.23e-008
57	5	3	5	3.82e+000	3.31e-007
60	3	1	14	1.37e-001	6.35e-007
61	3	2	21	-1.07e+002	9.68e-007
64	3	1	17	6.30e+003	4.57e-007
65	3	1	14	9.54e-001	3.30e-008
66	3	2	35	5.18e-001	6.82e-007
73	4	3	12	2.99e+001	1.74e-008
74	4	4	19	5.17e+003	2.05e-007
76	4	3	4	-4.13e+000	5.27e-007
77	4	3	14	2.42e-001	3.62e-007
78	5	3	51	-1.30e-013	2.05e-007
79	5	3	7	7.88e-002	4.47e-007

noms	n	m	*iter*	$f(x^\star)$	*kkt*
aircrfta	8	8	9	0.00e+000	5.00e-007
booth	2	2	11	0.00e+000	5.16e-007
byrdsphr	3	2	12	-4.68e+000	1.73e-010
bt1	2	1	11	-1.00e+000	2.27e-007
bt2	3	1	13	3.26e-002	1.35e-007
bt3	5	3	13	4.09e+000	5.89e-007
bt4	3	2	25	-4.55e+001	4.92e-007
bt5	3	2	18	9.62e+002	7.61e-007
bt6	5	2	15	2.77e-001	4.12e-007
bt7	5	3	23	3.60e+002	5.94e-009
genhs28	30	28	8	3.15e+000	2.16e-008
gottfr	2	2	8	0.00e+000	6.31e-008
hatflda	4	0	12	9.17e-016	4.81e-008
hatfldb	4	0	9	4.00e-002	7.05e-008
hatfldc	4	0	20	7.17e-016	4.29e-008
hatfldd	4	0	8	3.15e-015	4.82e-007
hatflde	4	0	13	2.38e-016	9.21e-007
hatfldf	3	3	9	0.00e+000	5.25e-009
maratos	2	1	16	-1.00e+000	5.25e-009
optcntrl1	32	23	37	9.63e-014	6.42e-006
optcntrl3	32	23	9	1.83e-001	4.72e-007
optcntrl6	32	23	9	1.83e-001	4.73e-007
zangwil2	2	0	8	-1.82e+001	1.01e-007
zangwil3	3	3	7	0.00e+000	3.39e-007

TAB. 2.3 – Résultats numériques pour des problèmes de la collection CUTE.

Chapitre 3

Un algorithme de points intérieurs avec régularisation DC(IPDCA) pour les problèmes non linéaires

Un algorithme de points intérieurs primal-dual pour résoudre des problèmes non linéaires est proposé. Cet algorithme est l'extension au cas non linéaire de l'algorithme pour la programmation quadratique non convexe présenté dans [8]. Le nouvel algorithme résout de façon séquentielle le problème d'optimisation non linéaire grâce aux conditions d'optimalité perturbées du premier ordre d'un problème convexe. Ce problème convexe est le sous-problème convexe généré par l'agorithme de programmation DC(DCA) [105, 106, 107, 108, 109, 110]. Une fonction de mérite L_1 non différentiable utilisée dans une procédure de recherche linéaire est incorporée dans l'algorithme pour garantir sa convergence globale. La procédure de recherche linéaire est utilisée pour modifier la longueur du pas afin d'assurer la décroissance de la fonction de mérite à chaque itération. Des longueurs de pas différentes sont utilisées pour les variables primales et duales.
Les directions générées par l'algorithme sont assurées d'être des directions de descente pour la fonction de mérite qui est employée pour guider l'algorithme vers une solution locale du problème d'optimisation. L'algorithme assure la convergence partant d'un point quelconque grâce à la décroissance monotone de la fonction de mérite à chaque itération.

3.1 Introduction

Le but principal de l'algorithme que nous allons présenter dans les lignes qui suivent est de trouver un point de karush-Khun-Tucker du problème de programmation non linéaire suivant

$$(\mathcal{P}_{\mathcal{EI}}) \begin{cases} \min f(x) & \\ Ax - b & = \ 0, \\ c(x) & \geq \ 0, \\ x \in \mathbb{R}^n, & \end{cases}$$

où $f : \mathbb{R}^n \to \mathbb{R}$ et $c : \mathbb{R}^n \to \mathbb{R}^m$ sont au moins de classe C^2, et c supposé *concave.* $A \in \mathbb{R}^{p\times n}$ une matrice *surjective*, et $b \in \mathbb{R}^p$ un vecteur.

Pour simplifier les notations, tout au long de ce chapitre nous considérerons le problème

$$(\mathcal{P}) \begin{cases} \min f(x) \\ c(x) \quad \geq \quad 0, \\ x \in \mathbb{R}^n, \end{cases}$$

la manière dont les contraintes d'égalité et les variables libres x sont prises en compte dans l'algorithme seront discutées dans l'Annexe A.

Nous faisons l'hypothèse que l'on a *qualification des contraintes* au sens présenté à la Section 1.1.4.

La section qui suit sera consacrée à l'intégration de la régularisation DC dans une méthode de points intérieurs. Dans la dernière partie du chapitre des résultats numériques comparatifs seront présentés.

3.2 Méthode de points intérieurs primal-dual

La première étape dans l'approche de méthode de points intérieurs considérée dans ce chapitre consiste à introduire les variables d'écart pour transformer d'une part les contraintes d'inégalités en contraintes d'égalités et d'autre part, prendre en compte le fait que les variables primales x sont libres, c'est-à-dire sans bornes. Pour les gérer nous avons suivi l'approche décrite dans [31] et que nous avons rappelée au chapitre précédent. Le problème transformé est,

$$\begin{cases} \min f(x) \\ c(x) - w \quad = \quad 0, \\ w \geq 0 \end{cases}$$

avec $w \in \mathbb{R}^m$. Le problème est par la suite transformé en une suite de problèmes en ajoutant des termes barrière à la fonction f. Le problème transformé s'écrit,

$$(P_\mu) \begin{cases} \min f_\mu(w, x) \\ c(x) - w \quad = \quad 0, \end{cases}$$

où $f_\mu(w, x) = f(x) - \mu \sum_{i=1}^{m} log(w_i)$.
Le terme logarithmique barrière est utilisé pour assurer implicitement les inégalités $w \geq 0$. La fonction f_μ est la fonction logarithmique barrière de Fiacco-McCormick [58].
Nous supposons pour toute la suite disposer d'une décomposition DC de la fonction objectif $f = g - h$, avec g et h convexes. Une telle décomposition de la fonction f s'appelle une régularisation DC de la fonction f.

Notons que l'on peut toujours définir une telle décomposition en posant $g = f + \rho \parallel x \parallel^2$ et $h = \rho \parallel x \parallel^2$ où ρ est un réel tel que g est convexe.
Posons,

$$\begin{aligned} g_\mu(w,x) &= g(x) - \mu\sum_{i=1}^{m_I} log(w_i), \\ h_\mu(w,x) &= h(x). \end{aligned}$$

A l'itération k on suppose disposer de x_k entre autres. Dans notre approche on linéairise la partie concave de la décomposition de l'objectif au point x_k. Puis on résoudra approximativement à chaque itération le problème,

$$(\mathcal{DC}_\mu) \left\{ \begin{array}{rcl} \min g_\mu(w,x) & - & \nabla^T h(x_k)x \\ c(x) - w & = & 0, \end{array} \right.$$

Cette approche est semblable à celle du schéma DCA, mais la différence réside sur le fait que le sous-problème $(\mathcal{DC}_\mu)$ n'est pas convexe. Une approche alternative mais équivalente à la nôtre aurait été de suivre exactement le schéma de DCA. Dans ce cas, nous aurions obtenu le sous-problème convexe.

$$\left\{ \begin{array}{rcl} \min g(x) & - & \nabla^T h(x_k)x \\ c(x) & \geq & 0, \end{array} \right.$$

Nous l'aurions résolu grâce au schéma classique de points intérieurs. En suivant le schéma décrit plus haut nous obtiendrions le problème $(\mathcal{DC}_\mu)$.
Soit

$$L_\mu = g_\mu(w,x) - \nabla^T h(x_k)x - \lambda^T(c(x) - w) \tag{3.1}$$

le Lagrangien associé au problème $(\mathcal{DC}_\mu)$. Les conditions d'optimalité du premier ordre s'écrivent,

$$(KKT_\mu) \left\{ \begin{array}{rcl} \nabla_x L_\mu = \nabla_x g_\mu(w,x) - \nabla h(x_k) - A(x)^T\lambda & = & 0, \\ \nabla_w L_\mu = -\mu W^{-1}e + \lambda & = & 0, \\ \nabla_\lambda L_\mu = -c(x) + w & = & 0, \end{array} \right.$$

avec $A(x) := \nabla c(x)$, $W = diag(w_i : i = 1, 2, \cdots, m_I)$. En remarquant que $\nabla_x g_\mu(w,x) = \nabla g(x)$, après avoir prémultiplié la seconde équation de (KKT_μ) par W.
En posant

$$F_\mu(w,x;\lambda) = \begin{pmatrix} \nabla g(x) - \nabla h(x_k) - A(x)^T\lambda \\ -\mu e + W\lambda \\ -c(x) + w \end{pmatrix}$$

on doit résoudre le système d'équations non-linéaires,

$$F_\mu(w,x;\lambda) = 0. \tag{3.2}$$

Remarquons qu'au point $\Pi_k = (x_k, w_k; \lambda_k)$ ce système traduit les conditions d'optimalité du problème (P_E) en Π_k, car on note que $\nabla g(x_k) - \nabla h(x_k) = \nabla f(x_k)$.

On utilise la méthode de Newton pour résoudre le système non linéaire (3.2). A la k-ème itération et pour μ fixé le système linéaire à résoudre est,

$$J(\Pi_k)\triangle\Pi = -F_\mu(\Pi_k), \tag{3.3}$$

où $J(\Pi_k)$ est la matrice Jacobienne de $F_\mu(\Pi_k)$ et $\triangle\Pi = (\triangle x, \triangle w; \triangle\lambda)$. Le calcul explicite de la matrice $J(\Pi_k)$ donne,

$$J(\Pi_k) = \begin{pmatrix} G_k & 0 & -A_k^T \\ 0 & \Lambda_k & W_k \\ -A_k & I_m & 0 \end{pmatrix} \tag{3.4}$$

avec $G_k = G(x_k)$, le Hessien du Lagrangien au point x_k, $A_k = A(x_k)$, $\Lambda_k = diag(\lambda_k^i : i = 1, 2, \cdots, m)$ et I_n, I_m les matrices identité d'ordre n et m.
Par souci de clarté dans notre exposé nous allons momentannément abandonner l'indexation par k.
On pose,

$$\begin{aligned} \sigma &= \nabla f(x) - A(x)^T\lambda, \\ \gamma &= \mu e - W\lambda, \\ \rho &= c(x) - w, \end{aligned}$$

alors le système linéaire s'écrit,

$$\begin{pmatrix} G(x) & 0 & -A(x)^T \\ 0 & \Lambda_k & W \\ -A(x) & I_m & 0 \end{pmatrix} \begin{pmatrix} \triangle x \\ \triangle w \\ \triangle\lambda \end{pmatrix} = \begin{pmatrix} -\sigma \\ \gamma \\ \rho \end{pmatrix}. \tag{3.5}$$

La deuxième équation du système (3.5) nous donne,

$$\triangle w = -D\triangle\lambda + \Lambda^{-1}\gamma, \tag{3.6}$$

où $D = \Lambda^{-1}W$. On aboutit après prémultiplication des équations par -1 le système réduit,

$$\begin{pmatrix} -G(x) & A(x)^T \\ A(x) & D \end{pmatrix} \begin{pmatrix} \triangle x \\ \triangle\lambda \end{pmatrix} = \begin{pmatrix} \sigma \\ -\rho - \Lambda^{-1}\gamma \end{pmatrix}. \tag{3.7}$$

On observe que la matrice intervenant dans le système linéaire (3.7) est symétrique quasi-définie donc fortement factorisable [35].

L'approche adoptée ici peut être vue comme l'application d'une régularisation à la matrice $Q(x)$ représentant le Hessien du Lagrangien du problème (P_E) au point x.
En effet en suivant la décomposition de la fonction objectif et le schéma décrit dans cette section, le Hessien du Lagrangien du problème (P_E) au point x s'écrit, $Q(x) = G(x) - H(x)$ avec $H(x) = \nabla^2 h(x)$ matrice définie positive par hypothèse. Ainsi grâce à la régularisation DC nous obtenons un système linéaire dont la matrice est symétrique quasi-définie à chaque itération.
Dans [34], pour les problèmes non convexes, une perturbation est appliquée au Hessien du

Lagrangien $Q(x)$. Notre approche diffère de cette dernière sur deux points essentiels. Tout d'abord dans [34] un terme αI_n est ajouté à la matrice $Q(x)$ de sorte que la matrice

$$N(x) = (Q(x) + \alpha I_n) + A(x)^T W^{-1} A(x)$$

soit définie positive. Pour obtenir ce résultat les auteurs dans [34] calculent le paramètre α de la manière suivante : ils effectuent une factorisation de Cholesky LDL^T de la matrice $N(x)$, puis ils testent si tous les éléments diagonaux de la matrice D sont égaux à $+1$. Si tel est le cas, aucune perturbation de la matrice $Q(x)$ n'est nécessaire et α est égale à 0. Sinon ils partent de $\alpha = 1.5$ si cette perturbation est insuffisante ils multiplient α par 2 jusqu'à avoir une perturbation qui rende $N(x)$ définie positive. Sinon ils divisent α par deux tant que $N(x)$ est définie positive. Comme on peut le voir cette heuristique peut s'avérer coûteuse surtout pour des problèmes de grande dimension.
Le deuxième point sur lequel porte la différence de nos deux approches est comprise dans le premier mais mérite à notre avis d'être signalé. Dans [34] il est demandé que la matrice $N(x)$ soit définie positive et non la matrice $Q(x)$. Ce point est essentiel car même si le terme αI_n est ajouté à la matrice $Q(x)$, la matrice résultante $Q(x) + \alpha I_n$ peut ne pas être définie positive alors que la matrice $N(x) = (Q(x) + \alpha I_n) + A(x)^T W^{-1} A(x)$ l'est. Il suffit pour cela que α soit supérieur à la plus petite valeur propre de $Q(x)$. La conséquence est que la matrice intervenant dans le système linéaire (3.7) après l'ajout de cette perturbation n'est pas nécessairement quasi-définie ; ce que garantit systématiquement notre régularisation.

Nous supposons la direction de Newton $\triangle\Pi_k$ calculée, nous pouvons déterminer l'itéré suivant,

$$\Pi_{k+1} = \Pi_k + \Upsilon_k \triangle\Pi_k,$$

où $\Upsilon_k = diag\{\alpha_{x_k} I_n, \alpha_{w_k} I_m, \alpha_{\lambda_k} I_m\}$. Les longueurs de pas $\alpha_{x_k}, \alpha_{w_k}, \alpha_{\lambda_k}$ sont déterminées dans $(0, 1]$ et pourraient éventuellement être égales ou différentes les unes des autres. En outre étant donné que les variables d'écart, w_k, et la variable duale λ_k, sont toutes positives à la solution du problème barrière $(\mathcal{DC}_\mu)$, cette propriété est maintenue pour toutes les itérations. Pour cela on définit,

$$\begin{aligned} \alpha_{w_k}^{\max} &= \gamma \max_{1 \le j \le m} \{-\tfrac{w_k^{(j)}}{\triangle w_k^{(j)}} : \triangle w_k^{(j)} < 0\}, \\ \alpha_{\lambda_k}^{\max} &= \gamma \max_{1 \le j \le m} \{-\tfrac{\lambda_k^{(j)}}{\triangle \lambda_k^{(j)}} : \triangle \lambda_k^{(j)} < 0\}, \end{aligned}$$

qui représentent les longueurs de pas maximales autorisées et qui garantiraient la positivité des variables respectives, avec $\gamma \in (0, 1]$.
La distance du point Π_k au chemin central est mesurée en utilisant la norme euclidienne des conditions d'optimalité perturbées, soit $\| F_\mu(\Pi_k) \|$ une fois que celle-ci est plus petite qu'un ε_μ préfixé ; on fait par la suite décroître le paramètre μ et le processus est répété jusqu'à ce que μ devienne zéro.

3.3 Fonction de mérite

Le but de la fonction de mérite est de fournir une mesure du progrès à la fois vers la solution du problème (P_μ) et du problème (P). Ceci est réalisé en s'assurant de sa décroissance à chaque itération de l'algorithme. Une procédure pour ajuster les longueurs des pas des variables est utilisée afin de garantir la décroissance de la fonction de mérite. Les méthodes de points intérieurs utilisent la fonction logarithmique barrière pour éliminer les contraintes d'inégalité du problème initial. Cependant elles ne fournissent aucune méthode pour gérer les contraintes d'égalité qui sont directement prises en compte dans le problème barrière. Alors la fonction de mérite est une combinaison de deux termes, la fonction barrière et les contraintes d'égalité.
Nous adopterons dans la suite les notations suivantes, $X = (w, x)$, $\triangle X = (\triangle w, \triangle x)$ $f_k = f(x_k)$ et $\rho_k = c(x_k) - w_k$

Nous ferons les hypothèses suivantes :
(A1) - la suite $\{X_k\}_k$ est bornée.

Pour un paramètre μ fixé on considère la *fonction de mérite*

$$\mathcal{M}_{\sigma\mu}(w, x) = f_\mu(w, x) + \sigma \| \, c(x) - w \, \|_P. \tag{3.8}$$

où $\| \, . \, \|_P$ désigne une norme sur $\mathbb{R}$ et σ un paramètre de pénalisation positif. On note $\| \, . \, \|_D$ la norme duale de $\| \, . \, \|_P$, définie par

$$\| \, y \, \|_D = \sup_{\|x\|_P \leq 1} x^T y, \tag{3.9}$$

et $|x^T y| \leq \| \, x \, \|_P \| \, y \, \|_D$.
$\mathcal{M}_{\sigma\mu}$ est une fonction de mérite exacte [57], c'est-à-dire que si $(w(\mu), x(\mu))$ est une solution de $(\mathcal{DC}_\mu)$ et si σ satisfait une condition donnée $(w(\mu), x(\mu))$ est un minimum local strict de $\mathcal{M}_{\sigma\mu}$. Cette condition est précisée dans la Proposition 3.3.1.

Notons que l'Hypothèse (A1) faite, la suite $\{\mathcal{M}_{\sigma\mu}(X_k)\}_k$ est bornée car $\mathcal{M}_{\sigma\mu}$ est continue.

Proposition 3.3.1 *$\mathcal{M}_{\sigma\mu}$ a une dérivée directionnelle en $X_k = (w_k, x_k)$ avec $w_k > 0$. Sa valeur dans la direction $\triangle X$ donnée par* (3.6)-(3.7) *est,*

$$\mathcal{M}'_{\sigma\mu}(X_k; \triangle X) = -\triangle x^T G_k \triangle x - \triangle w^T D_k \triangle w - \tilde{\lambda}^T \rho_k.$$

Elle est négative si $\sigma > \| \, \tilde{\lambda}_k \, \|_D$, où $\tilde{\lambda}_k = \lambda_k + \triangle\lambda$.

Preuve : Nous abandonnerons momentanément l'indexation par k pour des besoins de clarté dans notre exposé. Puisque $w > 0$, alors f_μ a des dérivées directionnelles au point (x, w). La norme étant continuement lipschitziène, elle a des dérivées directionnelles dans

toutes les directions. $\| \ . \ \|_P \circ\rho$ a des dérivées directionnelles et en appliquant la règle de calcul des dérivées composées on obtient successivement,

$$\begin{aligned}(\| \ . \ \|_P \circ\rho)'(X;\triangle X) &= (\| \ . \ \|_P)'(\rho;\nabla\rho\triangle X)\\ &= (\| \ . \ \|_P)'(\rho;A\triangle x-\triangle w)\\ &= -\| \ \rho \ \|_P \ .\end{aligned}$$

La dernière égalité est obtenue du fait que $\triangle X$ satisfait les contraintes linéarisées (3.5) et la définition des dérivées directionnelles.
Puisque $\triangle X$ est supposé avoir été obtenu à partir du système (3.5), de la première équation du système linéaire (3.5) nous obtenons après prémultiplication par $\triangle x$

$$\nabla^T f(x)\triangle x = -\triangle x^T G\triangle x + \tilde{\lambda}^T(\triangle w - c) \tag{3.10}$$

de la deuxième équation de (3.6) on a

$$\tilde{\lambda} = -D\triangle w + \mu W^{-1}e \tag{3.11}$$

donc

$$\nabla f^T(x)\triangle x - \mu W^{-1}\triangle w - \sigma \| \ \rho \ \|_P = -\triangle x^T G\triangle x - \triangle w^T D\triangle w - \tilde{\lambda}^T\rho - \sigma \| \ \rho \ \|_P \ .$$

Si $\sigma > \| \ \tilde{\lambda} \ \|_D$, puisque $|\tilde{\lambda}^T\rho| \leq \| \ \tilde{\lambda} \ \|_D \| \ \rho \ \|_P$, on obtient

$$\mathcal{M}'_{\sigma\mu}(X_k;\triangle X) \leq -\triangle x^T G\triangle x - \triangle w^T D\triangle w + (\| \ \tilde{\lambda} \ \|_D - \sigma) \| \ c \ \|_P \ .$$

et puisque G et D sont des matrices définies positives, on en déduit le résultat. □
La conséquence est que le pas obtenu grâce à (3.6)-(3.7) est une direction de descente de la fonction de mérite $\mathcal{M}_{\sigma\mu}$ sous la condition que le paramètre de pénalisation σ_k est supérieur à $\| \ \tilde{\lambda}_k \ \|_D$, ceci sera utilisé plus loin dans une procédure de recherche linéaire dans laquelle $\mathcal{M}_{\sigma\mu}$ décroît à chaque itération.

Dans la suite la norme adoptée sera la norme L_1.

3.3.1 Mise à jour du paramètre de pénalisation

Nous adopterons les hypothèses faites dans [99].

i) Pour tout k, $\sigma_k \geq \| \ \tilde{\lambda}_k \ \| + \bar{\sigma}$;
ii) il existe un k_o tel que si $k \geq k_o$ et $\sigma_{k-1} \geq \| \ \tilde{\lambda}_k \ \| + \bar{\sigma}$, alors $\sigma_k = \sigma_{k-1}$,
iii) si la suite $\{\sigma_k\}$ est bornée alors σ_k est modifié un nombre fini de fois.

Nous adoptons également la règle de mise à jour suivante,

si $\sigma_{k-1} \geq \| \ \tilde{\lambda}_k \ \| + \bar{\sigma}$ **alors**
  $\sigma_k = \sigma_{k-1}$
sinon
  $\sigma_k = \max\{\zeta\sigma_{k-1}, \| \ \tilde{\lambda}_k \ \| + \bar{\sigma}\}$
fin si

où $\zeta > 1$.

3.4 Détermination des longueurs de pas

3.4.1 Longueur des pas primale

La longueur de pas primale est déterminée en calculant tout d'abord un α^P_{max} qui garantirait la non négativité des variables qui doivent rester non négatives. Ceci signifie pour notre problème $(\mathcal{P}_\mu)$ que $w_j + \alpha\triangle w$ doit demeurer strictement positif. On prend donc

$$\alpha^P_{\max} = \{\max\{\frac{\epsilon^j_w - \triangle w_j}{w_j}, j = 1, 2, \cdots, m\}\}^{-1}, \tag{3.12}$$

avec ϵ_w un vecteur de $\mathbb{R}^m$. Dans notre implémentation $\epsilon_w = \xi e$, avec ξ un réel positif. Ceci permet de garantir que w reste strictement borné inférieurement par zéro. Alors le pas α_k est déterminé dans $(0, \alpha^P_{\max}]$ afin de vérifier la condition d'Armijo

$$\mathcal{M}_{\sigma\mu}(X_k + \alpha_k\triangle X) \leq \mathcal{M}_{\sigma\mu}(X_k) + \beta\alpha_k\mathcal{M}'_{\sigma\mu}(X_k; \triangle X) \tag{3.13}$$

où β est une constante dans $(0, \frac{1}{2})$. Si la condition n'est pas satisfaite la longueur de pas est réduite en choisissant la nouvelle longueur de pas dans $[\beta_1\alpha_k, \beta_2\alpha_k]$, avec $\beta_1, \beta_2 \in (0, 1)$ et $\beta_1 < \beta_2$. La procédure est répétée jusqu'à ce que (3.13) soit satisfait, dans ce cas on pose $\alpha^P_k = \alpha_k$ et le nouvel itéré primal est $X_{k+1} = X_k + \alpha^P_k\triangle X$.

3.4.2 Longueur des pas duale

Nous avons adopté la modification de la stratégie de Yamashita [56] présentée dans [55]. L'idée ici est d'utiliser les informations fournies par l'itéré primal X_{k+1} pour déterminer la longueur de pas duale α^D_k. Pour le paramètre μ fixé, la longueur de pas duale $\alpha^j_{\lambda_k}$ pour toutes les variables duales λ^j_k, $j = 1, 2, \cdots, m$, est déterminée afin que les contraintes

$$\alpha^j_{\lambda_k} = \max\{\alpha > 0 : l^j_k \leq (w^j_k + \alpha^P_k\triangle w^j_k)(\lambda^j_k + \alpha\triangle\lambda^j_k) \leq u^j_k\} \tag{3.14}$$

soient satisfaites, avec l^j_k et u^j_k, $j = 1, 2, \cdots, m$ déterminées par,

$$l^j_k = \min\{\ \frac{1}{2}\underline{m}\mu, (w^j_k + \alpha^P_k\triangle w^j_k)\lambda^j_k\} \tag{3.15}$$

$$u^j_k = \max\{\ 2\bar{m}\mu, (w^j_k + \alpha^P_k\triangle w^j_k)\lambda^j_k\} \tag{3.16}$$

et

$$0 \leq \underline{m} \leq \min_j\{1, \frac{(1-\eta)(1-\frac{\eta}{\Pi^{\mu+1}})\min_j\{w^j_k\lambda^j_k\}}{\mu}\} \tag{3.17}$$

$$0 < \bar{m} \leq \max_j\{1, \frac{\max_j\{w^j_k\lambda^j_k\}}{\mu}\} \tag{3.18}$$

avec $\eta \in (0,1)$ et Π un réel positif. La longueur de pas duale commune est alors,

$$\alpha_k^D = \min\{1, \min_j\{\alpha_{\lambda_k}^j\}\}. \tag{3.19}$$

Notons que

$$\lim_{\mu\to 0}(1 - \frac{\eta}{\Pi^{\mu+1}}) = 1 - \frac{\eta}{\Pi} \tag{3.20}$$

et pour tout $j = 1, 2, \cdots, m$ on a

$$\lambda_k^j + \alpha_k^D \triangle\lambda_k^j \geq 1 - \frac{\eta}{\Pi} > 0. \tag{3.21}$$

La suite $\{X_k\}_k$ est bornée par hypothèse et $\{w_k\}_k$ est strictement bornée inférieurement par zéro par construction des itérés. La proposition suivante due à Yamashita [56] nous dit que ceci vaut également pour $\{\lambda_k\}_k$.

Proposition 3.4.1 *Pour μ fixé les bornes inférieures(resp. supérieures) l_k^j(resp. u_k^j), $j = 1, 2, \cdots, m$, intervenant dans les contraintes permettant de calculer la longueur de pas duale, sont strictement bornées inférieurement par zéro(resp. supérieurement) si les composantes correspondantes w_k^j, sont également strictement bornés inférieurement par zéro et supérieurement.*

Preuve : La démonstration de cette proposition est faite dans [56]. □
La conséquence de cette proposition est que les éléments de la matrice diagonale D_k sont strictement bornés inférieurement par zéro.

3.5 Solution du problème Barrière ($\mathcal{P}_\mu$)

Nous allons présenter dans les lignes qui suivent les principales étapes de la procédure pour résoudre la problème barrière ($\mathcal{P}_\mu$).

Algorithme 3.5.1 *Solution du problème ($\mathcal{P}_\mu$).*
Nous supposons disposer des données suivantes :
le paramètre $\mu > 0$, un point $\Pi(\mu) = (x(\mu), w(\mu); \lambda(\mu))$ avec $w(\mu) > \epsilon_w$ et $\lambda(\mu) > 0$, et les paramètres $\sigma_0 > 0$, $\vartheta, \rho > 0$, $\beta_1, \beta_2 \in (0,1)$ et $\beta_1 \leq \beta_2$.
Répéter jusqu'à $\| F_\mu(x_k, w_k, \lambda_k) \| \leq \vartheta\mu$
(i) Calculer les pas
calculer la direction de Newton $(\triangle x_k, \triangle w_k, \triangle\lambda_k)$ en résolvant (3.6)- (3.7).
(ii) Calculer les longueurs de pas
- Calculer la longueur de pas primale
Suivre la procédure décrite en 3.4.1 pour calculer α_k^P.

- Calculer la longueur de pas duale
Suivre la procédure décrite en 3.4.2 pour déterminer α_k^D.
(iii) Nouvel itéré

$$\begin{aligned} x_{k+1} &= x_k + \alpha_k^P \triangle x_k \\ w_{k+1} &= w_k + \alpha_k^P \triangle w_k \\ \lambda_{k+1} &= \lambda_k + \alpha_k^D \triangle \lambda_k \end{aligned}$$

(iv) $k = k+1$ *mettre à jour* σ_k *grâce à la règle de mise à jour 3.3.1 et retourner en (i) Fin.*

Nous allons énoncer et démontrer la convergence de l'Algorithme 3.5.1 mais avant nous devons énoncer quelques résultats préliminaires.

Proposition 3.5.1 *Soit* $\Pi_k = (x_k, w_k; \lambda_k)$ *la suite générée par l'Algorithme 7.2.1 pour* μ *fixé. Alors la suite de matrices* $\{K_k^{-1}\}_k$ *est bornée, où*

$$K_k := \begin{pmatrix} -G_k & A_k^T \\ A_k & D_k \end{pmatrix}.$$

Preuve : La matrice K_k est symétrique quasi-définie elle est donc inversible [35], et sa matrice inverse est,

$$K_k^{-1} := \begin{pmatrix} -\bar{G}_k & \bar{A}_k^T \\ \bar{A}_k & \bar{D}_k \end{pmatrix},$$

où

$$\begin{aligned} \bar{G}_k &= G_k^{-1} + G_k^{-1} A_k^T (D_k + A G_k^{-1} A_k^T)^{-1} A_k G_k^{-1}, \\ \bar{A}_k &= (D_k + A_k G_k^{-1} A_k^T)^{-1} A_k G_k^{-1}, \\ \bar{D}_k &= (D_k + A_k G^{-1} A_k^T)^{-1}. \end{aligned}$$

La formule de Sheman-Morrison-Woodbury appliquée à $\bar{G}_k$ donne,

$$\bar{G}_k = (G_k + A_k^T D_k^{-1} A_k)^{-1}. \tag{3.22}$$

De la Proposition 3.3.1 on sait que D_k est bornée. D'où K_k^{-1} est bornée, puisque toutes les matrices qui y sont engagées sont bornées. □

Proposition 3.5.2 *Soit* $\Pi_k = (x_k, w_k; \lambda_k)$ *la suite générée par l'Algorithme 3.5.1 pour* μ *fixé. Alors la suite de vecteurs* $\{(\triangle x_k, \triangle w_k, \lambda_k + \triangle\lambda)\}_k$ *est bornée.*

Preuve : Nous pouvons réécrire le système (3.7) comme suit,

$$\begin{pmatrix} -G_k & A_k^T \\ A_k & D_k \end{pmatrix} \begin{pmatrix} \triangle x \\ \lambda_k + \triangle\lambda \end{pmatrix} = \begin{pmatrix} \nabla f(x_k) \\ -c(x_k) + w_k + \mu \Lambda_k^{-1} e \end{pmatrix} \tag{3.23}$$

Nous avons montré à la Proposition 3.5.1 que la matrice inverse de la matrice intervenant dans ce système existe et est bornée : donc les suites $\{\triangle x_k\}_k$, $\{\lambda_k + \triangle\lambda_k\}_k$sont bornées. De l'expression (3.6) on déduit que $\{\triangle w_k\}_k$ est aussi bornée. □

Proposition 3.5.3 *La suite de paramètres de pénalisation $\{\sigma_k\}_k$ est stationnaire pour k assez grand.*

Preuve : Puisque la suite $\{\tilde{\lambda}_k\}_k$ est bornée d'après la Proposition 3.5.2, et la règle de mise à jour du paramètre de pénalisation dans l'Algorithme 3.5.1, on déduit que $\{\sigma_k\}_k$ est bornée donc stationnaire pour k assez grand. □

Proposition 3.5.4 *Supposons que l'hypothèse (A1) est faite, alors le pas $\triangle X_k = (\triangle x_k, \triangle w_k)$ obtenu grâce à* (3.6)-(3.7) *dans l'Algorithme 3.5.1 vérifie :*

$$\lim_{k\to\infty} \triangle X_k = 0. \tag{3.24}$$

Preuve : Nous avons vu que pour k assez grand la suite de paramètres de pénalisation $\{\sigma_k\}_k$ est constante. Soit σ cette constante, la condition d'Armijo montre que pour k fixé la suite $\{\mathcal{M}_{\sigma\mu}(X_k)\}_k$ décroît et cette suite est bornée puisque l'on a fait l'hypothèse (A1) et $\mathcal{M}_{\sigma\mu}$ est continue, donc la suite $\{\mathcal{M}_{\sigma\mu}(X_k)\}_k$ est bornée. D'où la suite $\{\mathcal{M}_{\sigma\mu}(X_k)\}_k$ converge. Ceci implique que la suite $\{\alpha_k^P \mathcal{M}'_{\sigma\mu}(X_k; \triangle X_k)\}_k$ converge vers zéro, donc elle est bornée. Et par conséquence il existe $M > 0$ tel que

$$|\alpha_k^P \mathcal{M}'_{\sigma\mu}(X_k; \triangle X_k)| \leq M. \tag{3.25}$$

pour tout $k \in \mathbb{N}$. Si nous supposons que $\| \triangle X_k \| > 0$ nous allons montrer que l'Algorithme 3.5.1 va générer un nouveau point tel que la fonction de mérite décroît. Du développement de Taylor au premier ordre on a,

$$\mathcal{M}_{\sigma\mu}(X_{k+1}) = \mathcal{M}_{\sigma\mu}(X_k) + \alpha_k^P \mathcal{M}'_{\sigma\mu}(X_k; \triangle X_k) + \mathcal{O}(\alpha_k^{P^2} \| \triangle X_k \|^2). \tag{3.26}$$

Si nous posons $\phi_k = \mathcal{O}(\alpha_k^{P^2} \| \triangle X_k \|^2)$, alors on a,

$$\mathcal{M}_{\sigma\mu}(X_{k+1}) - \mathcal{M}_{\sigma\mu}(X_k) - \rho\alpha_k^P \mathcal{M}'_{\sigma\mu}(X_k; \triangle X_k) = (1-\rho)\alpha_k^P \mathcal{M}'_{\sigma\mu}(X_k; \triangle X_k) + \phi_k \tag{3.27}$$

Nous voyons qu'il existe une constante $\epsilon > 0$, telle que si $\alpha_k^P < \epsilon$ alors le membre de droite de l'équation (3.27) est négatif c'est-à-dire que

$$\phi_k < -(1-\rho)\alpha_k^P \mathcal{M}'_{\sigma\mu}(X_k; \triangle X_k). \tag{3.28}$$

Puisque $\{\alpha_k^P \mathcal{M}'_{\sigma\mu}(X_k; \triangle X_k)\}_k$ est bornée on a,

$$\phi_k < (1-\rho)M. \tag{3.29}$$

Donc à l'itération k la longueur de pas est au moins $\beta_1\epsilon$ et de (3.25)- (3.29) nous avons

$$\mathcal{M}_{\sigma\mu}(X_{k+1}) - \mathcal{M}_{\sigma\mu}(X_k) \leq -\rho M < 0. \tag{3.30}$$

□

Théorème 3.5.1 *On suppose l'hypothèse* $(A1)$ *faite et* μ *fixé. Alors l'Algorithme 3.5.1 se termine en un point satisfaisant les conditions d'optimalité du premier ordre du problème* $(\mathcal{P}_\mu)$.

Preuve : Nous allons d'abord montrer qu'asymptotiquement α_k^D devient un. Pour cela nous allons montrer que

$$\lim_{k\to\infty} \parallel \lambda_k + \triangle\lambda_k - \mu W_{k+1}e \parallel = 0. \tag{3.31}$$

En ajoutant $\mu W_{k+1}^{-1}e$ des deux côtés de la seconde équation de (3.5) on a,

$$\parallel \lambda_k + \triangle\lambda_k - \mu W_{k+1}e \parallel = \parallel \mu W_k^{-1}e - D_k\triangle w_k - \mu W_{k+1}^{-1}e \parallel \tag{3.32}$$

Alors on a

$$\parallel \lambda_k + \triangle\lambda_k - \mu W_{k+1}e \parallel \leq \parallel D_k \parallel\parallel \triangle w \parallel + \mu \parallel W_k^{-1} - W_{k+1}^{-1} \parallel \tag{3.33}$$

où $D_k = W_k^{-1}\Lambda_k$ et

$$\parallel W_k^{-1} - W_{k+1}^{-1} \parallel^2 \leq m \max_{1\leq j\leq m}\{(\frac{1}{w_k^j} - \frac{1}{w_{k+1}^j})^2\} = m \max_{1\leq j\leq m}\{(\frac{\alpha_k^P \triangle w_k^j}{w_k^j w_{k+1}^j})^2\}. \tag{3.34}$$

puisque $\alpha_k^P \in (0,1]$, $(\triangle w_k^j)^2 \leq \parallel \triangle w_k \parallel^2$ $j = 1, 2, \cdots, m$, et puisque la suite $\{w_k\}_k$ est strictement bornée inférieurement par zéro de la Proposition 3.5.4 on a

$$\lim_{k\to\infty} \parallel W_k^{-1} - W_{k+1}^{-1} \parallel^2 \leq m \lim_{k\to\infty} \max_{1\leq j\leq m}\{(\frac{\parallel \triangle w_k^j \parallel}{w_k^j w_{k+1}^j})^2\} = 0. \tag{3.35}$$

ainsi quand $k \to \infty$ in (3.35) nous avons le résultat. Pour k suffisamment grand on a

$$\lambda_{k+1} = \lambda_k + \triangle\lambda_k. \tag{3.36}$$

Pour démontrer les conditions de complémentarité, on réécrit,

$$W_{k+1}\lambda_{k+1} = W_{k+1}(\lambda_k + \triangle\lambda_k)$$

de la seconde équation de (3.5) nous sommes en mesure d'écrire,

$$W_{k+1}\lambda_{k+1} = W_{k+1}W_k^{-1}(-\Lambda_k\triangle w_k + \mu e)$$

mais les éléments de la matrice diagonale $W_{k+1}W_k^{-1}$ peuvent s'écrire

$$\frac{w_{k+1}^j}{w_k^j} = 1 + \alpha_k^P \frac{\triangle w_k^j}{w_k^j}, j = 1, 2, \cdots, m.$$

En utilisant la Proposition 3.5.4 on a,

$$\lim_{k\to\infty} W_{k+1}W_k^{-1} = I_m$$

Donc on a,

$$\lim_{k\to\infty} W_{k+1}\lambda_{k+1} = \mu e.$$

La troisième équation de (3.5) et la Proposition 3.5.4 donnent,

$$\lim_{k\to\infty} \parallel c(x_k) - w_k \parallel = 0.$$

De la première équation de (3.5) on a,

$$G_k \triangle x = A_k^T \triangle\lambda - \nabla f_k + A_k^T \lambda_k.$$

Pour k suffisamment grand $\triangle\lambda_k = \lambda_{k+1} - \lambda_k$ et $\nabla f_k = \nabla f_{k+1} + \mathcal{O}(\parallel \triangle x \parallel^2)$ car f est supposé au moins de classe C^2. On a donc pour k assez grand

$$A_k^T\lambda_{k+1} - \nabla f_{k+1} = G_k\triangle x + \mathcal{O}(\parallel \triangle x \parallel^2)$$

d'où grâce à la Proposition 3.5.4,

$$\lim_{k\to\infty} \parallel A_k^T\lambda_{k+1} - \triangle f_{k+1} \parallel = 0.$$

□

Proposition 3.5.5 *Pour μ et k fixés la suite $\{\lambda_k\}_k$ générée par Algorithme 3.5.1 est bornée.*

Preuve : Du Théorème 3.5.1 la suite $\{W_k\lambda_k\}_k$ est convergente et $\{W_k\}_k$ est bornée par hypothèse, on en déduit que la suite $\{y_k\}_k$ converge et donc elle est bornée. □

3.6 Convergence globale

Nous allons présenter dans les lignes qui suivent les principales étapes de la procédure pour résoudre le problème $(\mathcal{P})$.

Algorithme 3.6.1 *Algorithme IPDCA pour résoudre le problème* $(\mathcal{P})$
Choisir un point initial $(x_o, w_o; \lambda_o)$ *tel que* $w_o > \varepsilon_w, \lambda_o > 0$.
Choisir une valeur initiale pour le paramètre de pénalisation $\sigma > 0$ *et une valeur initiale pour le paramètre barrière* $\mu > 0$.
Choisir les paramètres ϑ, ϵ, $\delta, r \in (0,1)$.
Répéter jusqu'à $\| F(x_k, w_k, y_k) \| \leq \epsilon$
Appliquer l'Algorithme 7.2.1 pour trouver $(x(\mu), w(\mu); \lambda(\mu))$, *tel que*
$\| F_\mu(x(\mu), w(\mu), y(\mu)) \| \leq \vartheta \mu_k$
Poser $k = k + 1$ *et* $(x_k, w_k; \lambda_k) = (x(\mu), w(\mu); \lambda(\mu))$
et calculer $\delta = w_k^T y_k$ *et* $\mu_k = r\frac{\delta}{m}$.
Fin.

Théorème 3.6.1 *Posons* $\Pi(\mu) = (x(\mu), w(\mu); \lambda(\mu))$ *et* $\{\Pi(\mu)\}_{\mu \geq 0}$ *la suite approximant les points centraux déterminés par l'Algorithme 3.5.1 pour diverses valeurs de* μ. *La suite* $\{\Pi(\mu)\}_{\mu \geq 0}$ *est bornée et sa valeur d'adhérence* $(x_\star, w_\star, y_\star)$ *vérifie les conditions de karush-Kuhn-Tucker du problème* $(\mathcal{P})$.

Preuve : De l'hypothèse (A1) et de la Proposition 3.5.4 $\{\Pi(\mu)\}_{\mu \geq 0}$ est bornée donc admet des valeurs d'adhérence. Soit $(x_\star, w_\star; \lambda_\star)$ une de ces valeurs, du Théorème 3.5.1 et du fait que $\mu \to 0$ nous avons successivement,

$$\begin{array}{lcllcl} W_\star y_\star & = & \lim\limits_{\mu \to 0} W(\mu) y(\mu) & = & \lim\limits_{\mu \to 0} \mu e & = & 0, \\ c(x_\star) - w_\star & = & \lim\limits_{\mu \to 0} c(x(\mu)) - w(\mu) & = & 0, & & \\ \nabla f(x_\star) - A^T y_\star & = & \lim\limits_{\mu \to 0} \nabla f(x(\mu)) - A^T y(\mu) & = & 0. & & \end{array}$$

Le théorème est prouvé. □

3.7 Choix du point initial

Le choix du point initial est une étape crucial dans la mise en œuvre des méthodes de points intérieurs. Plusieurs heuristiques furent présentées. La seule étude faite sur le sujet à notre connaissance est celle de Nocedal et *al.* [135], mais la stratégie utilisée reste encore à parfaire. Dans notre approche nous commençons par résoudre le système linéaire,

$$\begin{pmatrix} -Q(x_0) & A^T(x_0) \\ A(x_0) & \delta I_m \end{pmatrix} \begin{pmatrix} x \\ \lambda \end{pmatrix} = \begin{pmatrix} f(x_0) \\ c(x_0) \end{pmatrix} \tag{3.37}$$

où $Q(x_0) = \nabla^2 f(x_0)$, $x_0 \in \mathbb{R}^n$ préalablement choisi et δ un réel positif. Le choix de δ est déterminant pour la performance de la méthode $\delta = 10e^{-8}$ qui nous est apparu comme étant un bon choix lors des tests numériques. Par la suite w est déterminé en posant $w = \max\{c(x), \eta\}$ et $\lambda = \max\{\lambda, \eta\}$ où η est choisi dans $\{0.1, 1, 10, 100\}$.

3.8 Résultats Numériques

L'Algorithme présenté dans ce chapitre a été codé en C et testé sur une machine AMD, de 1300 MHz, avec 512 Mo de RAM, sur Linux Redhad9.0. Le test d'arrêt utilisé pour DCA et IPDCA sera vérifié s'il est inférieur ou égal à $\varepsilon = 10^{-6}$, et celui par défaut pour LOQO [36] version 6.06. Le test d'arrêt de DCA pour ces tests est :

$$\| x_{k+1} - x_k \| / (1 + \| x_k \|) \leq \varepsilon. \tag{3.38}$$

La décomposition DC utilisée par DCA et IPDCA est obtenue en calculant la valeur approximative de la plus petite valeur propre λ_{min} du Hessien du Lagrangien Q en utilisant ARPACK [70] puis on pose $\rho = \min\{0, \lambda_{min} - 0.001\}$ et $G = Q + \rho I_n$ et $H = \rho I_n$. L'appel des routines en Fortran d'ARPACK dans notre code C est faite en utilisant l'approche décrite dans [73]. Les sous-problèmes quadratiques convexes de DCA sont résolus grâce à CPLEX 8.0 [71]. Dans notre implémentation ϑ est adapté en fonction du problème de manière à ne faire qu'une itération dans Algorithme 3.5.1. Le temps est reporté en secondes.

Dans les tableaux ci-dessous,
n désigne le nombre de variables du problème,
m désigne le nombre de contraintes du problème,
iter le nombre d'itérations,
temps le temps d'exécution,
" − " signifie que l'algorithme n'a pas pu résoudre le problème.

Pour résoudre les systèmes linéaires quasi-définis nous avons adapté le code accompagnant [32] à l'origine destiné au cadre de la programmation linéaire, au cadre non linéaire.

3.8.1 Problèmes de la collection CUTE

Les premiers problèmes testés sont de la collection CUTE [160], il s'agit de problèmes quadratiques non convexes.
Les points initiaux sont ceux calculés par chaque algorithme excepté pour DCA, où $x_0 = (0, 0, \cdots, 0)^T$ fut choisi comme point initial pour tous les problèmes. Les résultats sont reportés dans le Tableau 3.1. Les problèmes que nous souhaitons résoudre peuvent s'écrire

$$(\mathcal{QP}) \begin{cases} \min f(x) & = & \frac{1}{2}\langle Q, x\rangle + \langle q, x\rangle \\ Ax - b & \geq & 0, \\ x \in \mathbb{R}^n, \end{cases}$$

où $Q = Q^T \in \mathbb{R}^{n\times n}$ et $A \in \mathbb{R}^{m\times n}$ sont deux matrices, $q \in \mathbb{R}^n$ et $b \in \mathbb{R}^m$ sont deux vecteurs. On pose

$$g(x) = \frac{1}{2}\langle (Q + \rho I_n)x, x\rangle + \langle q, x\rangle \tag{3.39}$$

$$h(x) = \frac{1}{2}\rho\langle x, x\rangle \tag{3.40}$$

On a l'algorithme suivant

Algorithme 3.8.1 *DCA pour résoudre* $(\mathcal{QP})$
1. *Soit* $x_0 \in I\!R^n$ *un point initial. On pose* $k = 1$.
2. *Calculer* $y_k = \rho x_k$.
3. *Calculer* x_{k+1} *solution du problème quadratique convexe suivant :*

$$(\mathcal{QPC}) \begin{cases} \min \frac{1}{2}\langle (Q + \rho I_n)x, x\rangle + \langle q - y_k, x\rangle \\ Ax - b \ \geq 0, \\ x \in I\!R^n, \end{cases}$$

4. *Si le test d'arrêt* (3.38) *est vérifié alors stop ; sinon poser* $k = k + 1$ *et aller en 2. Sortir avec* x_k *et* $f(x_k)$ *comme meilleure solution et meilleure valeur de la fonction objectif.*

Remarque 3.8.1 *Une décomposition DC alternative pour DCA et pour IPDCA aurait été de prendre,*

$$g(x) = \frac{1}{2}\bar{\rho}\langle x, x\rangle + \langle q, x\rangle \tag{3.41}$$

$$h(x) = \frac{1}{2}\langle (-Q + \bar{\rho} I_n)x, x\rangle \tag{3.42}$$

avec $\bar{\rho}$ *un réel te que la matrice* $-Q+\bar{\rho}$ *est définie positive. Mais ce choix a conduit lors de nos tests numériques à un algorithme IPDCA moins stable sur la majorité des problèmes, et pour DCA la qualité de la solution était nettement moins bonne qu'avec la décomposition* (3.39). □

Dans le Tableau 3.1 on voit que IPDCA résout tous les problèmes avec les paramètres par défaut en des temps raisonnables, comparé aux deux autres algorithmes. On note également que LOQO trouve des résultats moins bons pour les problèmes blockqp2 et blockqp4 quand on demande à LOQO de calculer son propre point initial, alors que IPDCA trouve une solution proche du meilleur résultat trouvé par DCA. Pour les problèmes ncvxbqp1 et ncvxbqp3 LOQO n'a pas réussi à résoudre ces problèmes avec les paramètres par défaut et DCA n'a pas convergé après plus de trois heures. Pour des problèmes de dimensions plus importantes IPDCA est le seul des trois algorithmes à avoir résolu les problèmes en temps raisonnables.

3.8.2 Problèmes de complémentarité linéaire

Les seconds problèmes testés sont les problèmes de complémentarité linéaire(LCP). Ces problèmes s'écrivent

$$(LCP) \begin{cases} x^T(Mx - d) = 0 \\ Mx - d \geq 0 \\ x \geq 0, \end{cases}$$

où M est une matrice $n \times n$ et d un vecteur de $\mathbb{R}^n$. Pour tester le code nous allons considérer trois formulations du problème (LCP). Les données ont été tirées de [74]. Pour cette classe de problèmes le point initial de DCA est $x_0 = (0, 0, \cdots, 0)^T$. Ce choix se justifie par le fait que la valeur optimale de (LCP) est zéro qui est la valeur de la fonction objectif en en x_0. Par suite si $d \leq 0$, alors x_0 est une solution de (LCP), sinon x_0 est un bon point initial pour DCA car DCA est une méthode de descente qui génère une suite $\{x_k\}_{k\geq 1}$ contenue dans $\{x \in \mathbb{R}^n : Mx - d \geq 0, \quad x \geq 0\}$. Pour les deux autres codes les points initiaux sont ceux calculés par chacun d'eux.
Ces problèmes sont traités dans le cadre de la programmation DC avec différentes reformulations comme programmes DC en suivant [105, 116].

3.8.2.1 Première formulation du problème (LCP)

La première formulation est le problème quadratique non convexe,

$$(QP1) \begin{cases} \min f_1(x,y) = \frac{1}{2}\langle Q \begin{pmatrix} x \\ y \end{pmatrix}, \begin{pmatrix} x \\ y \end{pmatrix} \rangle \\ Mx - y - d = 0 \\ x \in \mathbb{R}^n_+, y \in \mathbb{R}^n_+. \end{cases}$$

Avec

$$Q = \begin{pmatrix} 0 & I_n \\ I_n & 0 \end{pmatrix}.$$

Lemme 3.8.1 *Les propositions suivantes sont équivalentes*
(i) $x^\star \in \mathbb{R}^n$ est solution du problème (LCP) ;
(i) $(x^\star, y^\star = Mx^\star - d) \in \mathbb{R}^n \times \mathbb{R}^n$ est solution de (QP1) et $f_1(x^\star, y^\star) = 0$.

On pose $z = (x, y) \in \mathbb{R}^n \times \mathbb{R}^n$. On pose

$$g_1(z) = \frac{1}{2}\langle (Q + \rho I_{2n})z, z\rangle \tag{3.43}$$

$$h_1(z) = \frac{1}{2}\rho\langle z, z\rangle. \tag{3.44}$$

On peut écrire

$$f_1(z) = g_1(z) - h_1(z).$$

Lemme 3.8.2 *Il existe un nombre non négatif $\rho_{\min}$ tel que pour tout $\rho > \rho_{\min}$ les fonctions f_1 et g_1 sont convexes.*

Preuve : Il est aisé de voir que les valeurs propres de la matrice Q sont -1 et $+1$ ansi si l'on pose $\rho_{\min} = 1$ ceci prouve le lemme. □

Grâce à ce lemme le problème (LCP) est équivalent au problème

$$(QPC1)\begin{cases} \min g_1(z) - h_1(z) \\ \begin{pmatrix} M \\ -I_{2n} \end{pmatrix} z - b = 0, \\ z \geq 0 \\ z \in \mathbb{R}^{2n}, \end{cases}$$

On a $\partial h_1(z) = \rho z$.

Algorithme 3.8.2 *DCA pour résoudre (LCP) : Première formulation*

1. *Soit $z_0 \in I\!R^n \times I\!R^n$ un point initial. On pose $k = 1$.*
2. *Calculer $w_k = \rho z_k$.*
3. *Calculer x_{k+1} solution du problème quadratique convexe suivant :*

$$(QPC1)\begin{cases} \min \frac{1}{2}\langle (Q + \rho I_{2n})z, z\rangle - \langle w_k, z\rangle \\ \begin{pmatrix} M \\ -I_{2n} \end{pmatrix} z - b = 0, \\ z \geq 0, \\ z \in I\!R^{2n}, \end{cases}$$

4. *Si le test d'arrêt* (3.38) *est vérifié alors stop ; sinon poser $k = k + 1$ et aller en 2. Sortir avec x_k et $f(x_k)$ comme meilleure solution et meilleure valeur de la fonction objectif.*

On observe au Tableau 3.2 qu'alors que LOQO converge plus rapidement que IPDCA et DCA pour la plupart des problèmes avec cette formulation, IPDCA et DCA trouve de meilleures solutions. En particulier toutes les valeurs objectif trouvées par ces deux algorithmes sont inférieures à $10e^{-6}$. Les résultats numériques du Tableau 3.2 montrent également que IPDCA et DCA trouvent des résultats comparables même ci DCA trouve toujours la solution globale.

3.8.2.2 Seconde formulation du problème (LCP)

La seconde formulation du problème (LCP) considérée est la suivante,

$$(QP2)\begin{cases} \min f_2(x) = \frac{1}{2}\langle x, (M + M^T)x\rangle + \langle d, x\rangle \\ Mx - d \geq 0, \\ x \in \mathbb{R}^n_+. \end{cases}$$

On peut écrire,

$$\begin{aligned} f_2(x) &= \frac{1}{2}\langle x, (M + M^T)x\rangle + \langle d, x\rangle \\ &= \frac{1}{2}\langle x, (\rho I_n + M + M^T)x\rangle + \langle d, x\rangle - \frac{1}{2}\rho\langle x, x\rangle \\ &= g_2(x) - h_2(x), \end{aligned}$$

avec

$$g_2(x) = \frac{1}{2}\langle x, (\rho I_n + M + M^T)x\rangle + \langle d, x\rangle \quad (3.45)$$

$$h_2(x) = \frac{1}{2}\rho\langle x, x\rangle. \quad (3.46)$$

Lemme 3.8.3 *Il existe un nombre non négatif $\rho_{\min}$ tel que pour tout $\rho > \rho_{\min}$ les fonctions f_2 et g_2 sont convexes.*

Preuve : En effet il suffit de prendre $\rho_{\min} = \min\{0, \lambda_{min} - 0.001\}$ où λ_{min} est la plus petite valeur propre de la matrice $M + M^T$ □
Grâce à ce lemme le problème (LCP) est équivalent à,

$$(QP2)\begin{cases} \min g_2(x) - h_2(x) \\ Mx - d \geq 0, \\ x \in \mathbb{R}^n_+. \end{cases}$$

On a $\partial h_2(x) = \rho x$.

Algorithme 3.8.3 *DCA pour résoudre (LCP) : Première formulation*
1. Soit $x_0 \in I\!R^n \times I\!R^n$ un point initial. On pose $k = 1$.
2. Calculer $y_k = \rho x_k$.
3. Calculer x_{k+1} solution du problème quadratique convexe suivant :

$$(QPC1)\begin{cases} \min \frac{1}{2}\langle (M + M^T + \rho I_n)x, x\rangle - \langle y_k, x\rangle \\ Mx - d \geq 0, \\ x \in I\!R^n_+. \end{cases}$$

4. Si le test d'arrêt (3.38) est vérifié alors stop ; sinon poser $k = k + 1$ et aller en 2. Sortir avec x_k et $f(x_k)$ comme meilleure solution et meilleure valeur de la fonction objectif.

Les résultats numériques du Tableau 3.3 montrent que LOQO est plus rapide que les deux autres codes comme cela était déjà le cas pour la formulation (QP1), mais la valeur de la fonction objectif est en général moins bonne que celle trouvée par IPDCA et DCA. Pour cette formulation, IPDCA est plus rapide que DCA sur cinq problèmes, mais la solution trouvée par DCA est globale.
Alors que la formulation (QP2) demande plus d'espace mémoire que la formulation (QP1), IPDCA et DCA semblent mieux se comporter pour cette formulation. L'avantage non négli geable de la formulation (QP1) est que les valeurs propres de la matrice Q sont connues de manière explicite, ce qui n'est pas le cas pour la formulation (QP2). Un des gros inconvénients de la formulation (QP1) est de pouvoir transformer un problème convexe en un problème non convexe potentiellement plus difficile à résoudre. En effet si la matrice M est symétrique définie positive, la formulation (QP2) est sans doute la plus adaptée.

3.8.2.3 Troisième formulation du problème (LCP)

La troisième formulation du problème (LCP) considérée n'est pas de classe C^2, seul DCA sera donc testé pour cette formulation.

$$(QP3)\begin{cases}\min f_3(x)=\sum\limits_{i=1}^{n}\min\{x_i,(Mx-d)_i\}\\ Mx-d\geq 0,\\ x\in\mathbb{R}^n_+.\end{cases}$$

On pose

$$C=\{x\in\mathbb{R}^n\mid Mx-d\geq 0,\ x\geq 0\}.$$

On a la proposition suivant,

Lemme 3.8.4 *Les propositions suivantes sont équivalentes*
(i) $x^\star\in\mathbb{R}^n$ est solution du problème (LCP) ;
(i) $x^\star\in\mathbb{R}^n$ est solution de (QP3) et $f_3(x^\star)=0$.

Soit

$$\mathcal{X}_C(x)=\begin{cases}0 & si\ x\in C\\ \infty & si\ x\notin C\end{cases}$$

la fonction indicatrice de C. Puisque C est un polyèdre convexe, $\mathcal{X}_C$ est une fonction convexe et on pose $g_3(x)=\mathcal{X}_C(x)$. On peut écrire

$$f_3(x)=\mathcal{X}_C(x)-\sum_{i=1}^{n}\max\{-x_i,-(Mx-d)_i\}.$$

Notons que la fonction $h_3(x)=\sum\limits_{i=1}^{n}\max\{-x_i,-(Mx-d)_i\}$ est convexe sur $\mathbb{R}^n$. Le problème (LCP) est alors équivalent au problème

$$\min_{x\in\mathbb{R}^n} g_3(x)-h_3(x).$$

Calcul explicite de $\partial h_3(x)$

Par définition on a,

$$\partial h_3(x) = \sum_{i=1}^{n}\partial\max\{-x_i,-(Mx-d)_i\} \tag{3.47}$$

$$= -\sum_{i=1}^{n}\begin{cases}e_i & si\quad x_i<(Mx-d)_i\\ M_i & si\quad x_i>(Mx-d)_i\\ [e_i,M_i] & si\quad x_i=(Mx-d)_i\end{cases} \tag{3.48}$$

où $e_i \in \mathbb{R}^n$ est le vecteur dont toutes les composantes sont nulles exceptée la i^{eme} et M_i la i^{eme} ligne de la matrice M.

Algorithme 3.8.4 *DCA pour résoudre (LCP) : troisième formulation*
1. *Soit $x_0 \in I\!R^n \times I\!R^n$ un point initial. On pose $k = 1$.*
2. *Calculer $y_k \in \partial h_3(x_k)$ grâce à la formule* (3.47).
3. *Calculer x_{k+1} solution du problème linéaire suivant :*

$$\begin{cases} \min -\langle x, y_k \rangle \\ Mx - d \geq 0, \\ x \in I\!R^n_+. \end{cases}$$

4. *Si le test d'arrêt* (3.38) *est vérifié alors stop ; sinon poser $k = k + 1$ et aller en 2. Sortir avec x_k et $f(x_k)$ comme meilleure solution et meilleure valeur de la fonction objectif.*

Les résultats de DCA pour cette formulation sont reportés au Tableau 3.4. Comme on pouvait s'y attendre, cette formulation est celle pour laquelle DCA est plus performant en terme de temps écoulé pour résoudre les problèmes et en terme de qualité de la solution ; avec cette formulation DCA trouve la solution globale. La rapidité de DCA avec cette formulation est normale car seul un système linéaire est résolu à chaque itération dans l'Algorithme 3.8.4.

3.8.3 Comparaison des algorithmes IPDCA et NIPA

Dans cette partie nous allons comparer deux algorithmes : l'algorithme NIPA présenté au Chapitre 2 et l'algorithme IPDCA. Cette comparaison est faite en utilisant les problèmes LCP. La première comparaison est faite sur le chemin suivi par chacun des deux algorithmes. Pour cette comparaison nous considérons le problème (LCP) avec les données,

$$M = \begin{pmatrix} 1 & 2 \\ 2 & 5 \end{pmatrix}, \quad q = \begin{pmatrix} -1 \\ -1 \end{pmatrix} \tag{3.49}$$

L'unique solution de ce problème est $(1, 0)$. La Figure 3.1 présente le chemin central suivi par IPDCA ainsi que le chemin suivi par NIPA. Les autres comparaisons portent sur certains des problèmes (LCP) déjà utilisés plus haut. Pour ces tests nous nous sommes limités à des petites dimensions ; NIPA ayant été codé en algèbre dense. La formulation utilisée est la formulation (QP1) et les résultats sont reportés au Tableau 3.5. On observe sur le Tableau 3.5 que NIPA est plus lent que IPDCA ; cependant la solution trouvée est comparable à celle retournée par IPDCA dans la plupart des cas et le nombre d'itérations effectuées par NIPA est inférieur à celui d'IPDCA.

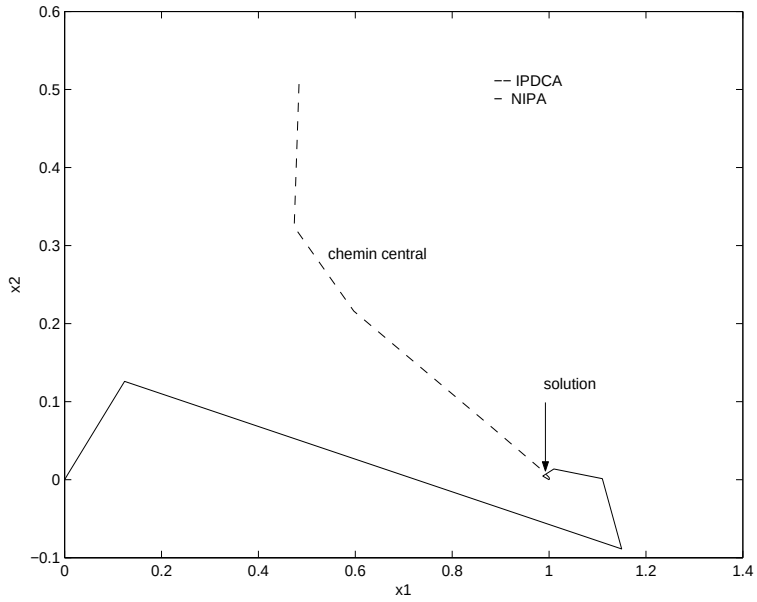


FIG. 3.1 – Comparaison des chemins intérieurs de IPDCA et de NIPA.

3.9 Un algorithme à deux phases IPDCA-DCA

Les résultats numériques reportés plus haut montrent que IPDCA est très souvent plus rapide que DCA pour résoudre les problèmes. La qualité de la solution trouvée par DCA dépendant du point initial, nous avons imaginé une procédure à deux phases dans laquelle IPDCA est utilisé comme sous-procédure de DCA pour le calcul d'un bon point initial.

Algorithme 3.9.1 *Un algorithme à deux phases IPDCA-DCA*

(i) - Lancer IPDCA et poser x_0 la solution obtenue.
(ii) - Lancer DCA en prenant x_0 comme point initial.

Pour tester cette approche, nous avons considéré deux types de tests d'arrêt pour DCA, pour illustrer l'importrance sur la performance de DCA.

Les résultats numériques pour certains problèmes de la collection CUTE sont reportés au Tableau 3.6. Ils montrent qu'utiliser IPDCA pour le calcul d'un point initial pour DCA améliore considérablement la qualité de la solution retournée par DCA.

Comme on peut le voir sur le Tableau 3.6, utiliser le test d'arrêt avec erreur relative conduit à une solution moins bonne que l'utilisation du test d'arrêt avec erreur absolue. Cependant le temps de résolution de DCA pour ce dernier test est en moyenne plus important.

3.10 Conclusion

Nous avons présenté un algorithme de point intérieur pour des problèmes non linéaires. La particularité essentielle de cet algorithme est l'intégration des techniques d'optimisation DC dans le cadre des méthodes de points intérieurs afin d'obtenir un système de Newton quasi-défini. La convergence globale de l'algorithme est prouvée. Des résultats comparatifs sur des problèmes quadratiques non convexes sont reportés et montrent la robustesse de notre algorithme. Un algorithme combinant IPDCA et DCA dans une procédure à deux phases fut également présentée et des résultats numériques encourageants reportés.
Le nouvel algorithme hérite de DCA d'une grande flexibilité sur la décomposition DC à adopter, et comme DCA certaines décompositions sont plus adaptées que d'autres pour un problème donné.

noms	*dim*		*iter*			*obj*			*temps*		
	n	m	LOQO	IPDCA	DCA	LOQO	IPDCA	DCA	LOQO	IPDCA	DCA
blockqp1	2005	1001	11	22	3	2.50	2.50	2.50	0.34	1.50	17.17
blockqp2	2005	1001	11	25	23	2.49	-995.02	-996.10	0.34	1.64	162.70
blockqp3	2005	1001	11	20	59	2.49	2.50	2.50	0.35	1.43	443.91
blockqp4	2005	1001	12	25	45	2.50	-495.57	-498.10	0.35	1.64	335.42
blockqp5	2005	1001	5	31	59	2.49	2.49	2.49	0.35	1.74	216.6
bloweya	2002	1002	13	26	152	-0.04540810	-0.04524207	-0.03999393	449.9	37.33	448.08
bloweyb	2002	1002	13	37	64	-0.03039616	-0.03033312	-0.02774721	456.20	32.04	208.80
bloweyc	2002	1002	13	34	148	-0.03029363	-0.03011640	-0.02651008	455.61	37.29	603.3
ncvxqp1	1000	500	415	50	76	-71591471.38	-71637668.93	-71591692.69	104.42	0.62	170.44
ncvxqp2	1000	500	202	51	96	-57813785.8	-57812673.26	-57809401.89	49.82	0.64	209.25
ncvxqp3	1000	500	144	82	520	-31185146.45	-31412647.51	-31215041.07	35.75	1.19	1198.54
ncvxqp4	1000	250	393	46	94	-93978733.69	-94014749.48	-93972214.99	54.58	0.38	98.45
ncvxqp5	1000	250	205	56	401	-66292380.71	-66378347.61	-66249398.83	27.88	0.52	333.68
ncvxqp6	1000	250	117	70	894	-35154566.11	-35438954.63	-34220883.59	16.01	0.67	593.64
ncvxqp7	1000	750	454	57	59	-43524289.14	-43544910.62	-43419827.58	132.66	1.14	268.28
ncvxqp8	1000	750	185	56	192	-30457300.77	-30497179.22	-30438817.48	51.33	1.14	958.96
ncvxqp9	1000	750	110	88	581	-21570321.62	-21575500.75	-21090198.97	29.78	1.70	1430.01
qpnband	1000	500	77	25	4	-1124.24705	-1124.23616	-1124.250	12.83	0.28	1.35
ncvxbqp1	10000	0	-	20	-	-	-19855401647.27	-	-	7.04	-
ncvxbqp3	10000	0	-	45	-	-	-6554273184.65	-	-	8.90	-
ncvxqp1	10000	5000	-	55	-	-	-7.513264094e+09	-	-	18.22	-
ncvxqp4	10000	2500	-	46	-	-	-9.388710506e+09	-	-	11.58	-
ncvxqp6	10000	2500	-	77	-	-	-3.539778813e+09	-	-	15.20	-
ncvxqp7	10000	7500	-	45	-	-	-5.219739526e+09	-	-	24.98	
ncvxbqp1	100000	50000	-	99	-	-	-7.448920999e+11	-	-	2137.58	-
ncvxbqp4	100000	25000	-	60	-	-	-9.388324212e+11	-	-	1186.58	-

TAB. 3.1 – Résultats numériques pour des problèmes non convexes de la collection CUTE

noms	*dim*	*iter*			*obj*			*temps*		
		LOQO	IPDCA	DCA	LOQO	IPDCA	DCA	LOQO	IPDCA	DCA
LCP6	200	13	13	2	1.757949e-06	2.58382e-08	0.00000e+00	2.7	5.38	1.29
LCP6	600	14	14	2	2.022318e-6	2.54344e-08	0.00000e+00	74.14	133.92	16.52
LCP6	1000	13	15	2	9.093290e-06	6.65979e-11	0.00000e+00	318.52	634.43	75.74
LCP7	200	13	10	2	1.020821e-06	6.80039e-10	0.00000e+00	0.04	0.06	0.11
LCP7	600	13	10	2	3.030998e-06	5.44985e-10	0.00000e+00	0.12	0.23	0.57
LCP7	1000	13	10	2	5.041174e-06	3.25158e-10	0.00000e+00	0.23	0.45	1.36
LCP8	200	13	9	2	8.452518e-07	2.76540e-12	0.00000e+00	0.04	0.06	0.12
LCP8	600	13	9	2	2.525208e-06	7.71044e-12	0.00000e+00	0.12	0.22	0.50
LCP8	1000	13	9	3	4.205165e-06	1.27629e-11	0.00000e+00	0.24	0.43	1.04
LCP9	200	13	18	2	6.299152e-07	9.76059e-15	0.00000e+00	0.56	2.45	0.75
LCP9	600	13	19	2	2.321204e-06	8.56726e-14	0.00000e+00	11.38	65.63	11.87
LCP9	1000	13	19	2	4.547452e-06	1.55742e-13	0.00000e+00	58.17	286.96	46.37
LCP10	200	13	42	3	4.456435e-07	3.22597e-07	0.00000e+00	0.04	0.07	0.08
LCP10	600	13	38	3	2.474782e-06	4.98873e-08	0.00000e+00	0.09	0.28	0.26
LCP10	1000	13	33	3	4.869774e-06	7.61082e-07	0.00000e+00	0.16	0.60	0.55
LCP11	200	36	11	2	5.301906e-21	1.24804e-09	0.00000e+00	0.10	0.04	0.15
LCP11	600	36	11	2	1.591248e-20	3.17033e-09	0.00000e+00	0.36	0.17	0.6
LCP11	1000	36	11	2	2.652306e-20	5.11765e-09	0.00000e+00	0.80	0.39	1.15
LCP12	200	32	11	2	7.333264e-16	2.81398e-09	0.00000e+00	0.09	0.03	0.13
LCP12	600	32	11	2	2.205046e-15	7.29728e-09	0.00000e+00	0.33	0.18	0.52
LCP12	1000	32	11	2	3.676774e-15	1.17430e-08	0.00000e+00	0.71	0.37	1.05
LCP13	200	31	14	2	7.286676e-07	2.05470e-11	0.00000e+00	1.30	1.87	0.81
LCP13	600	25	15	2	8.616030e-07	1.32207e-13	0.00000e+00	22.19	51.78	11.78
LCP13	1000	14	15	2	2.150484e-06	2.09175e-14	0.00000e+00	66.04	227.08	47.88

TAB. 3.2 – Résultats numériques pour la formulation (QP1).

noms	*dim*	NbIter			*obj*			*temps*		
		LOQO	IPDCA	DCA	LOQO	IPDCA	DCA	LOQO	IPDCA	DCA
LCP6	200	14	24	1	-8.73082e-07	-5.93882e-07	0.00000e+00	2.81	4.25	0.62
LCP6	600	14	25	1	-5.11382e-07	-2.69843e-07	0.00000e+00	66.91	122.21	6.51
LCP6	1000	14	21	1	-7.25144e-07	-4.855446e-07	0.00000e+00	316.15	553.42	24.19
LCP7	200	13	9	1	1.31821e-07	8.23192e-10	2.84217e-14	0.01	0.01	0.05
LCP7	600	13	9	1	3.94506e-07	2.49104e-09	-8.52651e-14	0.06	0.04	0.29
LCP7	1000	13	9	1	6.57192e-07	4.15690e-09	5.68434e-14	0.09	0.09	0.60
LCP8	200	13	8	1	1.22828e-07	3.38972e-09	0.00000e+00	0.01	0.00	0.06
LCP8	600	13	8	1	3.67951e-07	7.16517e-09	0.00000e+00	0.07	0.04	0.25
LCP8	1000	13	8	1	6.13064e-07	1.09395e-08	0.00000e+00	0.10	0.08	0.53
LCP9	200	13	6	1	2.10815e-07	4.27413e-12	0.00000e+00	0.92	0.31	0.14
LCP9	600	13	6	1	6.77150e-07	1.19548e-11	0.00000e+00	23.99	8.84	1.14
LCP9	1000	13	6	1	1.34700e-06	2.04096e-11	0.00000e+00	126.15	39.06	3.56
LCP10	200	13	11	1	4.25555e-07	1.62392e-08	0.00000e+00	0.01	0.01	0.02
LCP10	600	13	11	1	1.10547e-07	1.364242e-11	0.00000e+00	0.04	0.04	0.08
LCP10	1000	13	11	2	4.20640e-06	-5.20611e-06	0.00000e+00	0.07	0.08	0.19
LCP11	200	41	11	1	1.42339e-06	2.12007e-08	0.00000e+00	0.04	0.01	0.02
LCP11	600	40	11	1	9.41877e-07	3.58400e-08	0.00000e+00	0.17	0.06	0.10
LCP11	1000	39	11	1	8.42141e-06	5.04793e-08	0.00000e+00	0.30	0.10	1.16
LCP12	200	44	10	1	1.40377e-07	1.12434e-07	0.00000e+00	0.05	0.01	0.02
LCP12	600	42	10	1	7.91198e-07	3.60798e-12	0.00000e+00	0.15	0.04	0.08
LCP12	1000	43	10	1	5.45247e-07	1.12434e-07	0.00000e+00	0.31	0.09	0.16
LCP13	200	13	10	1	6.26356e-07	1.506243e-08	0.00000e+00	0.88	0.49	0.44
LCP13	600	13	10	1	2.31637e-06	2.220446e-16	0.00000e+00	24.34	14.11	5.72
LCP13	1000	13	10	1	3.67533e-06	2.220446e-16	0.00000e+00	117.02	62.70	20.06

TAB. 3.3 – Résultats numériques pour la formulation (QP2).

noms	*dim*	*iter*	*obj*	*temps*
LCP6	200	1	0.00000e+00	0.32
LCP6	600	1	0.00000e+00	2.74
LCP6	1000	1	0.00000e+00	9.82
LCP7	200	1	0.00000e+00	0.07
LCP7	600	1	0.00000e+00	0.39
LCP7	1000	1	0.00000e+00	1.63
LCP8	200	1	0.00000e+00	0.05
LCP8	600	1	0.00000e+00	0.32
LCP8	1000	1	0.00000e+00	1.55
LCP9	200	1	0.00000e+00	0.18
LCP9	600	1	0.00000e+00	2.20
LCP9	1000	1	0.00000e+00	8.43
LCP10	200	1	0.00000e+00	0.05
LCP10	600	1	0.00000e+00	0.31
LCP10	1000	2	0.00000e+00	1.92
LCP11	200	1	0.00000e+00	0.07
LCP11	600	1	0.00000e+00	0.35
LCP11	1000	1	0.00000e+00	1.57
LCP12	200	1	0.00000e+00	0.05
LCP12	600	1	0.00000e+00	0.33
LCP12	1000	1	0.00000e+00	1.54
LCP13	200	1	0.00000e+00	0.18
LCP13	600	1	0.00000e+00	2.19
LCP13	1000	1	0.00000e+00	8.45

TAB. 3.4 – Résultats numériques de DCA pour la formulation (QP3).

noms	*dim*	NIPA			IPDCA		
		iter	*obj*	*temps*	*iter*	*obj*	*temps*
LCP6	20	12	-2.80e-009	0.36	20	5.01e-008	0.02
LCP6	60	14	-1.76e-007	12.72	22	1.75e-009	0.30
LCP6	100	13	-1.64e-006	69.84	24	4.57e-009	1.23
LCP7	20	10	-2.23e-009	0.31	25	1.40e-007	0.02
LCP7	60	10	-1.00e-008	8.10	37	1.15e-007	0.04
LCP7	100	10	-1.78e-008	52.95	33	1.94e-007	0.08
LCP8	20	10	1.91e-006	0.32	36	2.50e-007	0.03
LCP8	60	10	1.90e-006	7.93	34	5.11e-007	0.05
LCP8	100	10	1.89e-006	52.77	36	1.10e-007	0.07
LCP9	20	11	-4.46e-008	0.35	19	6.10e-007	0.03
LCP9	60	11	-1.70e-007	8.91	23	9.51e-007	0.16
LCP9	100	12	-3.41e-006	63.12	25	3.74e-007	0.50
LCP10	20	11	-1.24e-005	0.34	19	3.62e-007	0.04
LCP10	60	13	-4.06e-007	10.15	23	1.84e-007	0.03
LCP10	100	13	-7.33e-006	68.33	25	1.70e-007	0.04

TAB. 3.5 – Comparaison des algorithmes IPDCA et NIPA.

noms	IPDCA			DCA					
	iter	*obj*	*temps*	$\parallel x_{k+1} - x_k \parallel \leq \varepsilon$			$\frac{\|x_{k+1}-x_k\|}{1+\|x_k\|} \leq \varepsilon$		
				iter	*obj*	*temps*	*iter*	*obj*	*temps*
ncvxqp1	50	-7.16376e+07	0.62	2	-7.16376e+07	4.31	1	-7.16376e+07	2.15
ncvxqp2	51	-5.78126e+07	0.64	2	-5.78127e+07	3.95	1	-5.78127e+07	2.00
ncvxqp3	82	-3.14126e+07	1.19	441	-3.14275e+07	988.00	1	-3.14252e+07	2.32
ncvxqp4	46	-9.40147e+07	0.38	2	-9.40148e+07	1.96	1	-9.40148e+07	0.99
ncvxqp5	56	-6.63783e+07	0.52	146	-6.63855e+07	128.00	1	-6.63783e+07	0.91
ncvxqp6	70	-3.54491e+07	0.67	835	-3.54898e+07	546.80	1	-3.54860e+07	0.74
ncvxqp7	57	-4.35242e+07	1.14	2	-4.35245e+07	8.58	1	-4.35245e+07	4.29
ncvxqp8	56	-3.04971e+07	1.14	126	-3.05067e+07	656.59	1	-3.05010e+07	5.24
ncvxqp9	88	-2.15750e+07	1.70	1960	-2.15784e+07	2860.00	1	-2.15762e+07	4.45

TAB. 3.6 – Résultats numériques pour l'algorithme combiné IPDCA-DCA

Deuxième partie

Globalisation

Chapitre 4

Combiné IPDCA-DCA dans un schéma séparation-évaluation pour résoudre les problèmes quadratiques non convexes

4.1 Introduction

Considérons le problème quadratique

$$(\mathcal{P}) \left\{ \begin{array}{ll} \min f(x) & = \frac{1}{2}x^TQx + q^Tx \\ x \in D, & \end{array} \right.$$

où $D = \{x \in \mathbb{R}^n : Ax - b \geq 0\}$, $Q = Q^T \in \mathbb{R}^{n\times n}$ et $A \in \mathbb{R}^{m\times n}$ sont deux matrices données, $q \in \mathbb{R}^n$ et $b \in \mathbb{R}^m$ deux vecteurs. On suppose que la matrice Q est indéfinie. Du fait que Q n'est pas définie positive, $(\mathcal{P})$ est un problème non convexe dont les solutions locales ne sont pas nécessairement globales.
L'approche classique utilisée pour résoudre globalement le problème $(\mathcal{P})$ sont les méthodes séparation-évaluation ou Branch and Bound. La différence entre ces méthodes réside dans la stratégie d'évaluation et de séparation utilisée. Pour une présentation de ces méthodes voir [119, 94, 123, 92].
Plus récemment des méthodes basées sur les conditions nécessaires et suffisantes d'optimalité globales [124, 48] ont été développées [47, 49, 50, 53]. Ces algorithmes semblent prometteurs pour des problèmes de petites dimensions. La vérification de l'optimalité reste néanmoins une tâche ardue.
L'algorithme que nous nous proposons de présenter dans les lignes qui suivent est particulier dans l'intégration au schéma séparation-évaluation de la procédure IPDCA-DCA introduite au Chapitre 3 pour le calcul de la borne supérieure, au lieu de l'utilisation de DCA seul comme cela a déjà été fait avec succès [103, 104, 105, 111, 112, 113, 114, 115] pour l'amélioration de la borne supérieure. Une autre particularité de notre algorithme est le calcul systématique

de la borne supérieure sur les deux rectangles obtenus à l'issue de l'étape de séparation du schéma de séparation-évaluation. Cette évaluation systématique se justifie par le fait qu'une telle stratégie nous évite de rester cantonné dans une portion du domaine alors que la solution se trouve dans l'autre. Et ce d'autant plus que l'amélioration de la borne supérieure que ce soit par DCA ou par IPDCA-DCA se fera sur le rectangle courant issu de la subdivision.

Dans la section qui suit nous exposerons en détail l'algorithme pour résoudre le problème $(\mathcal{P})$, nous présenterons également par souci de complétude les techniques de subdivisions à la section 3. Des résultats numériques comparatifs sont présentés à la fin du chapitre.

4.2 Description de l'algorithme

Supposons que l'on dispose de la décomposition $Q = G + H$, avec G et H des matrices définies positive et négative respectivement. Notre choix fut de prendre $H = -\rho I$ avec ρ tel que $Q - H$ est définie positive. Cette décomposition induit une décomposition DC de la fonction objective $f(x) = g(x) + h(x)$, avec $g(x) = \frac{1}{2}x^T Gx + q^T x$ et $h(x) = -\frac{1}{2}\rho\sum_{i=1}^{n} x_i^2$.
La première étape de l'algorithme est la construction d'un rectangle $R_o \subset \mathbb{R}^n$ qui contient D. Pour cela on résout n programmes linéaires

$$\max\{x_i \ \ t.q \ \ x \in D\}, \ \ i = 1, 2, \cdots, n \tag{4.1}$$

pour obtenir les valeurs optimales $L_i^o, i = 1, 2, \cdots, n$ et

$$\min\{x_i \ \ t.q \ \ x \in D\}, \ \ i = 1, 2, \cdots, n \tag{4.2}$$

pour obtenir les valeurs optimales $l_i^o, i = 1, 2, \cdots, n$. Le rectangle peut alors s'écrire,

$$R^o = \{x : l_i^o \leq x_i \leq L_i^o, i = 1, 2, \cdots, n\}.$$

4.2.1 Borne inférieure

Soit $R = \{x : l_i \leq x_i \leq L_i, i = 1, 2, \cdots, n\}$ un rectangle dans R^n. Nous adopterons la convention classique qui est de prendre $+\infty$ comme borne inférieure d'un ensemble vide.
Dans les méthodes de Branch and Bound, l'approche classique pour calculer la borne inférieure est d'utiliser un sous-estimateur convexe de la fonction objectif. Quand elle est facile à calculer, un sous-estimateur convexe de choix pour une fonction est son enveloppe convexe.

Définition 4.2.1 *L'enveloppe convexe d'une fonction h sur un convexe D est une fonction h_{conv} telle que :*
(i) h_{conv} est convexe sur D.
(ii) $h_{conv}(x) \leq h(x)\ \forall x \in D$.
(iii) Si $\tilde{h}$ est une fonction vérifiant (i) et (ii), alors $\tilde{h}(x) \leq h_{conv}(x)\ \forall x \in D$.

On a le théorème suivant dû à Falk et à Hofmann.

Théorème 4.2.1 *[76] Soit S un polytope tel que $S = conv\{s^1, \cdots, s^n\}$ et soit h une fonction concave sur D. Alors l'enveloppe convexe de h sur D est définie par :*

$$h_{conv}(x) = \min\{\sum_{i=1}^{n}\lambda_i h(s^i) : \sum_{i=1}^{n}\lambda_i s^i = x, \sum_{i=1}^{n}\lambda_i = 1, \lambda_i \geq 0, pour i = 1, 2, \cdots, n\}$$

D'après le Théorème 4.2.1 en dimension finie on peut toujours calculer l'enveloppe convexe d'une fonction concave sur un polyèdre. Puisque la fonction concave h est séparable, son enveloppe convexe sur le rectangle R est simplement la somme de fonctions affines $h^{R_i}_{conv}(x_i)$ qui coïncident avec h_i aux extrémités du segment $[l_i, Li]$, c'est-à-dire la fonction

$$h^R_{conv}(x) = \sum_{i=1}^{n} h^{R_i}_{conv}(x_i)$$

où

$$h^{R_i}_{conv}(x_i) = -\frac{1}{2}\rho(l_i + L_i)x_i + \frac{1}{2}\rho l_i L_i.$$

Ainsi $g(x) + h^R_{conv}(x)$ est un sous-estimateur de f sur le domaine $\mathcal{K} = D \cap R$. La solution du problème convexe

$$(LB) \qquad \min\{g(x) + h^R_{conv}(x) : x \in \mathcal{K}\}$$

nous fournit un point x^R tel que,

$$g(x^R) + h^R_{conv}(x^R) \leq \min\{f(x) : x \in \mathcal{K}\} \leq f(x^R).$$

Donc $\beta(R) = g(x^R) + h^R_{conv}(x^R)$ est une borne inférieure pour f sur R et $f(x^R)$ est une borne supérieure de la valeur optimale $f_\star$.

4.2.2 Borne supérieure

La borne supérieure est calculée grâce à l'algorithme IPDCA-DCA présenté dans la partie précédente. Plus concrètement la borne supérieure est calculée en deux étapes. Lors de la première étape IPDCA est appliqué au problème

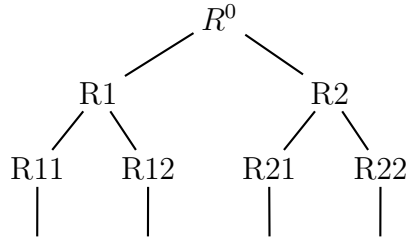
$$(UB) \min\{f(x) : x \in \mathcal{K}\}.$$

Puis DCA est appliqué au même problème en partant du point retourné par IPDCA.

4.2.3 Subdivision rectangulaire normale

Rappelons d'abord le concept de subdivision rectangulaire normale due à Tuy [123][Définition VII.7].
Soit $R = \{x : l_i \leq x_i \leq L_i, i = 1, 2, \cdots, n\}$ un rectangle et h^R_{conv} un sous-estimateur de h sur R défini plus haut. Notons x^R et $\beta(R)$ la solution optimale et la valeur optimale respectivement du problème (LB). Soit le processus de subdivision rectangulaire dans lequel

un rectangle est subdivisé par le billet d'hyperplans parallèles à certaines facettes de l'ortan $\mathbb{R}^n_+$. Ce processus génère une famille de rectangles qui peuvent être représenté par un arbre dont la racine est R^o et tel que un nœud est le successeur d'un autre si et seulement s'il représente un élément de la partition du rectangle correspondant au dernier nœud. Un chemin infini dans cet arbre correspond alors à une suite infinie de rectangles R_k, $k = 1, 2, \cdots$



Un exemple d'arbre de subdivisions.

Définition 4.2.2 *Une suite de rectangles $\{R_k\}_k$ est dite normale si*

$$\underline{\lim}_{k\to+\infty} |h(x^{R_k}) - h_{conv}(x^{R_k})| = 0.$$

Une subdivision rectangulaire est dite normale si toute suite infinie de rectangles qu'elle génère est normale.

Supposons que l'on dispose d'une procédure de subdivision rectangulaire. Nous pouvons présenter deux algorithmes de séparation-évaluation : Dans le premier l'algorithme IPDCA est appelé en premier, le point obtenu est ensuite utilisé par DCA comme point initial et dans le second algorithme DCA utilise le point admissible fourni par le calcul de la borne inférieure comme point initial. Dans les deux cas le calcul de la borne supérieure est systématique sur les deux rectangles issus de la subdivision.

Algorithme 4.2.1 *Algorithme BBALG1*

Initialisation On pose $R_0 = [l_0, L_0]$ où l_0 et L_0 sont obtenus en résolvant les programmes linéaires (4.2) *et* (4.1).
Calculer $h^{R_0}_{conv}$ et résoudre le programme quadratique convexe

$$(R_0LB)\, \min\{g(x) + h^{R_0}_{conv}(x) : x \in \mathcal{K}_0\}$$

pour obtenir la solution optimale x^{R_0} et la valeur optimale correspondante β_0, appliquer IPDCA au problème

$$(R_0UB)\, \min\{f(x) : x \in \mathcal{K}_\prime\}.$$

Soit $\tilde{x_0}$ la solution trouvée ; puis appliquer DCA au problème (R_0UB) en prenant $\tilde{x_0}$ comme point initial. Soit x_0 la solution trouvée par DCA et $\gamma_0 = f(x_0)$.
Si $\gamma - \beta_0 \leq \varepsilon$, alors stop←vrai, x_0 est une $\varepsilon-$solution pour $(\mathcal{P})$.
Sinon stop← faux finsi.
Poser $\mathcal{R} \leftarrow R_0$, $k \leftarrow 0$.
Tant que stop=faux faire,
- *sélectionner un rectangle $R_k \in \mathcal{R}$ tel que*

$$\beta_k = \beta(R_k) = \min\{\beta(R) : R \in \mathcal{R}\}.$$

- *Diviser $R_k = \{x : l_i^k \leq x_i \leq L_i^k, i = 1, 2, \cdots, n\}$ en deux sous-rectangles R_{k1} et R_{k2} en utilisant une subdivision rectangulaire normale choisie.*

- *Pour R_{k1} et R_{k2} calculer $h_{conv}^{R_{ki}}$ et résoudre*

$$(R_{ki}LB) \min\{g(x) + h_{conv}^{R_{ki}}(x) : x \in \mathcal{K}_{ki}\}$$

  pour obtenir $x^{R_{ki}}$ et $\beta(R_{ki})$.
- *Pour R_{k1} et R_{k2} résoudre*

$$(R_{ki}UB) \min\{f(x) : x \in \mathcal{K}_{ki}\}$$

  en utilisant IPDCA pour obtenir $\tilde{x}_{ki}$. Résoudre $(R_{ki}UB)$ avec DCA en utilisant comme point initial $\tilde{x}_{ki}$ retourné par IPDCA et soient $x_{R_{ki}}^{DCA}$ les solutions trouvées.
- *Si l'on a $f(x_{R_{ki}}^{DCA}) < \gamma_{k-1}$ alors poser $x_k = x_{R_{ki}}^{DCA}$ et $\gamma_k = f(x_k)$.*
- *Poser $\mathcal{R} \leftarrow (\mathcal{R} \backslash R_k) \cup \{R_{ki} : \beta(R_{ki}) < \gamma_k - \varepsilon, i = 1, 2\}$*
- *Si $\mathcal{R} = \varnothing$, alors stop←vrai, x_k est une $\varepsilon-$solution, sinon $k \leftarrow k + 1$ finsi.*

Fin Tant que

Algorithme 4.2.2 *Algorithme BBALG2*

Initialisation : on pose $R_0 = [l_0, L_0]$ où l_0 et L_0 sont obtenus en résolvant les programmes linéaires (4.2) et (4.1).
Calculer $h_{conv}^{R_0}$ et résoudre le programme quadratique convexe

$$(R_0LB) \min\{g(x) + h_{conv}^{R_0}(x) : x \in \mathcal{K}_0\}$$

pour obtenir la solution optimale x^{R_0} et la valeur optimale correspondante β_0. Appliquer DCA au problème

$$(R_0UB) \min\{f(x) : x \in \mathcal{K}_0\}.$$

en prenant x_0^R comme point initial. Soit x_0 la solution trouvée par DCA et $\gamma_0 = f(x_0)$.
Si $\gamma - \beta_0 \leq \varepsilon$, alors stop←vrai, x_0 est une $\varepsilon-$solution pour $(\mathcal{P})$.
Sinon stop← faux finsi.
Poser $\mathcal{R} \leftarrow R_0$, $k \leftarrow 0$.
Tant que stop=faux faire,

– *sélectionner un rectangle* $R_k \in \mathcal{R}$ *tel que*

$$\beta_k = \beta(R_k) = \min\{\beta(R) : R \in \mathcal{R}\}.$$

– *Diviser* $R_k = \{x : l_i^k \leq x_i \leq L_i^k, i = 1, 2, \cdots, n\}$ *en deux sous- rectangles* R_{k1} *et* R_{k2} *en utilisant une subdivision rectangulaire normale choisie.*

– *Pour* R_{k1} *et* R_{k2} *calculer* $h_{conv}^{R_{ki}}$ *et résoudre*

$$(R_{ki}LB)\min\{g(x) + h_{conv}^{R_{ki}}(x) : x \in \mathcal{K}_{ki}\}$$

pour obtenir $x^{R_{ki}}$ *et* $\beta(R_{ki})$.
– *Pour* R_{k1} *et* R_{k2} *résoudre*

$$(R_{ki}UB)\min\{f(x) : x \in \mathcal{K}_{ki}\}$$

avec DCA en utilisant comme point initial $x^{R_{ki}}$. *Soient* $x_{R_{ki}}^{DCA}$ *les solutions trouvées.*
– *Si l'on a* $f(x_{R_{ki}}^{DCA}) < \gamma_{k-1}$ *alors poser* $x_k = x_{R_{ki}}^{DCA}$ *et* $\gamma_k = f(x_k)$.
– *Poser* $\mathcal{R} \leftarrow (\mathcal{R} \backslash R_k) \cup \{R_{ki} : \beta(R_{ki}) < \gamma_k - \varepsilon, i = 1, 2\}$
– *Si* $\mathcal{R} = \varnothing$, *alors stop←vrai,* x_k *est une* $\varepsilon-$*solution, sinon* $k \leftarrow k + 1$ *finsi.*
Fin Tant que

Le théorème suivant s'applique aux algorithmes précédemment présentés.

Théorème 4.2.2 *[111]*
– *Si l'algorithme se termine à l'itération* k, *alors* x_k *est une solution globale du problème* $(\mathcal{P})$.
– *Si l'algorithme est infini, alors il génère une suite bornée* $\{x_k\}_k$ *dont tout point d'accumulation est un optimum global de* $(\mathcal{P})$, *et*

$$f(x_k^{DCA}) \searrow f_\star, \quad f(x^{R_k}) \nearrow f_\star.$$

Algorithme 4.2.3 *Algorithme BBALG3*

Initialisation : on pose $R_0 = [l_0, L_0]$ *où* l_0 *et* L_0 *sont obtenus en résolvant les programmes linéaires* (4.2) *et* (4.1).
Calculer $h_{conv}^{R_0}$ *et résoudre le programme quadratique convexe*

$$(R_0LB)\min\{g(x) + h_{conv}^{R_0}(x) : x \in \mathcal{K}_0\}$$

pour obtenir la solution optimale x^{R_0} *et la valeur optimale correspondante* β_0. *et* $\gamma_0 = f(x^{R_0})$.
Si $\gamma - \beta_0 \leq \varepsilon$, *alors stop←vrai,* x_0 *est une* $\varepsilon-$*solution pour* $(\mathcal{P})$.
Sinon stop← faux finsi.
Poser $\mathcal{R} \leftarrow R_0$, $k \leftarrow 0$.
Tant que stop=faux faire,

– *sélectionner un rectangle* $R_k \in \mathcal{R}$ *tel que*

$$\beta_k = \beta(R_k) = \min\{\beta(R) : R \in \mathcal{R}\}.$$

– *Diviser* $R_k = \{x : l_i^k \leq x_i \leq L_i^k, i = 1, 2, \cdots, n\}$ *en deux sous- rectangles* R_{k1} *et* R_{k2} *en utilisant une subdivision rectangulaire normale choisie.*

– *Pour* R_{k1} *et* R_{k2} *calculer* $h_{conv}^{R_{ki}}$ *et résoudre*

$$(R_{ki}LB)\min\{g(x) + h_{conv}^{R_{ki}}(x) : x \in \mathcal{K}_{ki}\}$$

pour obtenir $x^{R_{ki}}$ *et* $\beta(R_{ki})$.
– *Si l'on a* $f(x_{R_{ki}}) < \gamma_{k-1}$ *alors poser* $x_k = x^{R_{ki}}$ *et* $\gamma_k = f(x_k)$.
– *Poser* $\mathcal{R} \leftarrow (\mathcal{R} \backslash R_k) \cup \{R_{ki} : \beta(R_{ki}) < \gamma_k - \varepsilon, i = 1, 2\}$
– *Si* $\mathcal{R} = \varnothing$, *alors stop*$\leftarrow$*vrai,* x_k *est une* $\varepsilon-$*solution, sinon* $k \leftarrow k + 1$ *finsi.*
Fin Tant que

4.3 Construction de subdivisions rectangulaires normales

La règle suivante de subdivision de R_k fut introduite dans [100, 122].

subdivision exhaustive :

De manière générale i_k est déterminé comme étant l'indice de la plus longue arrête de R_k, c'est-à-dire, tel que

$$(L_{i_k}^k - l_{i_k}^k)^2 = \max\{(L_i^k - l_i^k)^2, i = 1, 2, \cdots, n\}.$$

soit

$$\alpha = \frac{1}{2}(L_{i_k}^k + l_{i_k}^k).$$

Alors R_k est subdivisé en deux sous-rectangles,

$$R_{k1} = \{x \in R_k : x_{i_k} \leq \alpha\}, \;\; R_{k2} = \{x \in R_k : x_{i_k} \geq \alpha\}.$$

ω-subdivision :

Pour un rectangle sélectionné R_k, $\beta(R_k) < f(x_k)$, donc,

$$h(x^{R_k}) - h_{conv}(x^{R_k}) > 0.$$

Déterminer l'indice i_k tel que

$$i_k \in \arg\max_i\{h^i(x_i^{R_k}) - h_{conv}^i(x_i^{R_k})\}.$$

Puis on subdivise R_k en deux sous-rectangles,

$$R_{k1} = \{x \in R_k : x_{i_k} \leq x_{i_k}^{R_k}\}, \;\; R_{k2} = \{x \in R_k : x_{i_k} \geq x_{i_k}^{R_k}\}.$$

subdivision adaptive :

Pour tout rectangle R_k choisi, deux points sont pris en compte. Le premier est x_k^R et le second v^k est tel que

$$v_i^k \in \arg\min_i\{h^i(l_i^k), h^i(L_i^k)\},$$

qui n'est rien d'autre que

$$v^k \in \arg\min_{R_k}\{h_{conv}^{R_k}(y)\}.$$

Ces points se nomment points de bisection. Soit i_k un indice tel que

$$|(v^k - x^{R_k})_{i_k}| = \max_i |(v^k - x^{R_k})_i|$$

et soit

$$\alpha = \frac{1}{2}(v^k + x^{R_k})_{i_k}.$$

Alors les sous-rectangles obtenus par bisection de R_k sont,

$$R_{k1} = \{x \in R_k : x_{i_k} \leq \alpha\}, \;\; R_{k2} = \{x \in R_k : x_{i_k} \geq \alpha\}.$$

La proposition suivante est établie dans [[94] Chapitre 4].

Proposition 4.3.1 *Soit $R_1 \supset R_2 \supset \cdots$ une suite infinie de rectangles telle que R_{k+1} est obtenue de R_k via la subdivision adaptive, et soient $\{v^k\}_k$ et $\{x^{R_k}\}_k$ suites de points de bisection. Alors il existe des sous suites $\{v_l^k\}_l$ et $\{x_l^{R_k}\}_l$ telles que,*

$$\lim_{l \to +\infty} \| v_l^k - x_l^{R_k} \| = 0.$$

Il fut prouvé dans [123] que la subdivision exhaustive ainsi que la $\omega-$subdivision en conjonction avec la procédure de calcul de la borne inférieure utilisant l'enveloppe convexe h_{conv} génère un procéssus de subdivision rectangulaire normale. Il est prouvé dans [93] que la subdivision adaptive elle aussi génère une subdivision rectangulaire normale.

La $\omega-$subdivision due à Falk et Soland [125] fut reconnue dans [93] comme étant la plus efficace de ces trois subdivisions à l'issue des tests numériques. Nous suivrons cette observation et nous adopterons la $\omega-$subdivision pour tester notre approche.

4.4 Résultats numériques

Les tests numériques qui suivent sont effectués sur deux types de problèmes quadratiques. Les premiers sont générés de manière aléatoire en suivant le schéma présenté dans [120] et que nous rappelerons dans la suite. Les résultats numériques relatifs à ce type de problèmes sont reportés dans le Tableau 4.1. Le deuxième type de problèmes quadratiques est tiré de [121]. Dans les tableaux, la colonne *iter* représente le nombre d'itérations total de l'algorithme , la colonne *NbRest* le nombre de fois que DCA a redémarré à partir des meilleures solutions

admissibles courantes et en travaillant sur le rectangle initial avant d'avoir la ε-solution, la colonne *temps* le temps total. Dans ces tests $\varepsilon = 10e^{-2}$.

A partir d'une matrice $m \times n$ générée aléatoirement on commence par calculer sa décomposition en valeurs singulières dans le but d'obtenir des matrices unitaires U et V de dimension $n \times n$ et $m \times m$. On pose alors

$$A = U \begin{pmatrix} \Sigma \\ \Omega \end{pmatrix} V^T,$$

où $\Sigma = diag(\sigma_i)$ avec les σ_i choisis dans $[-1, 1]$ et $\Omega = diag(\omega_i)$ les ω_i choisis dans $[-1, 1]$. On génère par la suite un $x^\star$ tel que $x_i^\star \in [-10, 10]$. Puis on calcule

$$b = -H^T x^\star.$$

La matrice Hessienne est déterminée en calculant,

$$Q = UDU^T$$

avec $D = diag(\delta_i)$ où $\delta_i \in [-1, 1]$. On sélectionne μ_j et λ_j dans $[-1, 1]$ puis on pose

$$q = -Qx^\star + A\mu + \lambda.$$

Le Tableau 4.1 nous fournit deux enseignements :

1. utiliser la combinaison IPDCA-DCA conduit à moins de redémarrages de DCA que lorsque DCA est utilisé seul pour améliorer la borne supérieure. Ceci confirme les observations faites dans la partie précédente lors de l'introduction de l'algorithme à deux phases IPDCA-DCA. On observait alors que la combinaison IPDCA-DCA donnait de meilleurs résultats que l'utilisation de chacun des deux algorithmes indépendamment.
2. Les bornes inférieure et supérieure obtenues en utilisant BBALG1 sont comparables à celles obtenues avec BBALG2. Ce constat mérite une attention particuliére dans la mesure où dans BBALG1 le point initial transmis à DCA pour calculer la borne supérieure est calculé par IPDCA qui n'utilise pas comme point initial le point obtenu par le calcul de la borne inférieure, mais détermine son propre point initiale en se servant du rectangle qui lui est transmis comme bornes de la variable.

dim		BBALG1					BBALG2				
n	m	*iter*	*NbRest*	*LB*	*UB*	*temps*	*iter*	*NbRest*	*LB*	*UB*	*temps*
10	1	46	1	-3.90	-3.86	4.08	46	1	-3.90	-3.86	5.85
10	5	9	1	-129.02	-122.75	2.63	9	1	-129.02	-122.75	2.69
20	2	73	1	-2.43	-2.37	23.63	73	1	-2.43	-2.37	24.85
20	10	15	1	-421.90	-413.20	18.16	15	2	-441.71	-413.20	10.71
30	15	41	1	-615.03	-608.61	34.47	37	1	-615.68	-608.61	35.21
30	21	37	1	-497.20	-490.32	34.16	37	1	-497.21	-490.32	36.01
40	20	25	1	-1173.19	-1157.30	55.96	23	4	-1199.01	-1157.30	48.07
50	25	36	1	-1154.40	-1123.86	137.51	36	1	-1154.39	-1123.86	131.52
50	35	37	3	-1110.75	-1097.21	251.49	49	4	-1041.86	-1025.77	205.09
60	30	50	1	-1224.61	-1203.18	333.00	50	1	-1224.61	-1203.18	323.93
60	42	41	1	-1350.84	-1306.56	366.9	41	1	-1350.84	-1306.56	360.84
70	7	54	1	-1765.04	-1740.69	56.23	57	1	-1759.07	-1740.69	68.99
70	35	34	1	-2140.41	-2078.40	70.05	33	3	-2137.84	-2078.40	305.7
80	8	97	1	-1339.04	-1321.72	598.13	65	7	-1361.70	-1312.19	724.38
80	40	57	3	-1657.83	-1625.61	903.2	57	1	-1646.67	-1625.61	1081.27
90	45	65	1	-1935.84	-1895.95	1245.51	65	1	-1935.84	-1895.95	1382.26
100	10	44	1	-3395.14	-3345.54	63.30	44	1	-3395.14	-3345.54	52.45
100	50	77	1	-1907.19	-1883.44	2300.10	76	3	-1930.36	-1888.77	1854.05
250	25	183	1	-5676.94	-5609.18	8235.00	183	1	-5678.93	-5609.18	8103.46
300	30	109	1	-13658.75	-13523.21	3231.44	109	1	-13658.75	-13523.21	2516

TAB. 4.1 – Résultats numériques pour les algorithmes BBALG1 et BBALG2 sur le premier type de problèmes.

Chapitre 5

Combiné IPDCA-DCA dans un schéma séparation-évaluation pour résoudre un problème quadratique à variables zéro-un

Dans ce chapitre nous proposons une algorithme de séparation évaluation pour résoudre le problème

$$(\mathcal{P}) \begin{cases} \min \frac{1}{2} x^T Q x \\ x \in \{0,1\}^n, \end{cases}$$

où $Q \in \mathbb{R}^{n \times n}$ une matrice symétrique.

Ce problème englobe une grande famille de problèmes d'optimisation combinatoire en particulier elle englobe les problèmes de coupe maximale qui retiennent notre attention dans ce chapitre.

Plusieurs algorithmes ont été proposés pour résoudre ce type de problème [111, 179, 180, 189]. L'algorithme de séparation-évaluation que nous proposons ici diffère de celui présenté dans [111] sur deux points essentiels :

(i) la borne inférieure est calculée à partir d'un problème relaxé classique ;
(ii) le branchement est fait par subdivisions entières comme dans [117], et non par subdivisions rectangulaires normales.

5.1 Formulation équivalente simple

La première étape dans notre démarche consiste à transformer le problème $(\mathcal{P})$ en un problème

$$(\mathcal{Q}) \begin{cases} \min \frac{1}{2}u^T Au + a^T u + c \\ u \in \{-1,1\}^n, \end{cases}$$

On pose $f(u) := \frac{1}{2}u^T Au + a^T u + c$, $A = \frac{1}{4}Q$ et α la valeur optimale de ce problème.
Cette transformation est faite en utilisant le changement de variable $u_i = \phi(x_i)$ avec

$$\phi \ : \ \mathbb{R} \to \mathbb{R}, \ \ \phi(t) = 2t - 1,$$

et dans ce cas $a = Ae$ et $c = \frac{1}{2}a^T e$.
Nous justifirons ce changement de variable plus loin dans le chapitre. Le changement de variable effectué, on considère le problème

$$(\mathcal{Q}1) \begin{cases} \min \frac{1}{2}u^T (A + \rho I)u + a^T u + c \\ u \in [-1,1]^n, \end{cases}$$

où I est la matrice identité d'ordre n et ρ est choisi de sorte que la matrice $A + \rho I$ soit définie négative, on pourra prendre $\rho < -\lambda_n$ où λ_n est la plus grande valeur propre de la matrice A.

Le problème $(\mathcal{Q}1)$ est un problème de minimisation d'une fonction quadratique strictement concave sur un hyper-rectangle la solution est donc atteinte sur les sommets c'est-à-dire sur $\{-1,1\}^n$. Donc $(\mathcal{Q}1)$ est équivalent au problème

$$\begin{cases} \min \frac{1}{2}u^T (A + \rho I)u + a^T u + c \\ u \in \{-1,1\}^n, \end{cases}$$

qui n'est rien d'autre que le problème

$$\begin{cases} \min \frac{1}{2}u^T Au + a^T u + c \\ u \in \{-1,1\}^n, \end{cases}$$

c'est à dire le problème $(\mathcal{Q})$, car sur $\{-1,1\}^n$ on a $u^T u = n$.
Ainsi à ce niveau on dispose de l'équivalence

$$(\mathcal{Q}) \iff (\mathcal{Q}1)$$

En considérant le problème

$$(\mathcal{Q}2) \begin{cases} \min \frac{1}{2}u^T Au + a^T u + c \\ s.t \\ \quad p(u) \leq 0, \\ \quad u \in [-1,1]^n, \end{cases}$$

avec

$$p(u) = \sum_{i=1}^{n}(u_i + 1)(1 - u_i), \tag{5.1}$$

qui est une fonction concave.

On a l'équivalence imédiate

$$(\mathcal{Q}) \iff (\mathcal{Q}2)$$

et donc la chaîne d'équivalences

$$(\mathcal{Q}1) \iff (\mathcal{Q}) \iff (\mathcal{Q}2). \tag{5.2}$$

L'équivalence entre le problème $(\mathcal{Q}1)$ et le problème $(\mathcal{Q}2)$ provient d'un résultat plus générale énoncé dans le théorème suivant.

Théorème 5.1.1 *[118]*
Soit le problème

$$(\mathcal{P}) \min\{f(x) \ : \ p(x) \leq 0, \ x \in \mathcal{C}\}$$

avec $\mathcal{C}$ un polyèdre convexe borné. On suppose f Lipschitz sur $\mathcal{C}$ et p concave avec $p(x) \geq 0$ pour tout $x \in \mathcal{C}$. Alors il existe $\tau_0 > 0$ tel que pour tout $\tau \geq \tau_0$, le problème $(\mathcal{P})$ est équivalent[1] *au problème*

$$(\mathcal{P}_\tau) \min\{f(x) + \tau p(x) \ : \ x \in \mathcal{C}\}$$

Dans le cas général le paramètre τ_0 est difficile voir impossible à determiner. Mais pour le problème qui nous intéresse dans ce chapitre nous avons vu plus haut que le choix $\tau_0 = \lambda_n$ nous garantissait l'équivalence entre ces deux problèmes.
Grâce l'équivalence établie plus haut, nous allons travailler sur le problème $(\mathcal{Q}1)$.

Remarque 5.1.1 *Un choix alternatif pour la fonction p aurait pu être [112],*

$$p(u) = \sum_{i=1}^{n} \min(u_i, 1 - u_i). \tag{5.3}$$

Ce choix aurait conduit à un problème DC polyédral pour lequel on sait que DCA converge en un nombre fini d'étapes ; mais le fait d'utiliser IPDCA dans la procédure IPDCA-DCA pour le calcul de la borne supérieure, nécessite d'avoir des fonctions de classe C^2.

5.2 Procédure de séparation-évaluation

Nous allons présenter les principales étapes de la procédure de séparation-évaluation.

[1] l'équivalence ici signifie que les deux problèmes ont le même ensemble de solutions globales.

5.2.1 Borne supérieure

La borne supérieure se calcule en appliquant IPDCA-DCA au problème ($\mathcal{Q}1$). La structure du problème ($\mathcal{Q}1$) nous fournit une décomposition DC naturelle. En effet le problème ($\mathcal{Q}1$) peut s'écrire

$$(\mathcal{DC}) \min_{u\in\mathbb{R}^n}\{\mathcal{X}_{[-1,1]^n}(u) - (-\frac{1}{2}u^T(A+\rho I)u - a^Tu - c)\}$$

qui se met ainsi sous la forme d'un problème DC Nous rappellons que ρ est choisi pour que la matrice $(A+\rho I)$ soit définie négative, nous rappelons également que $\mathcal{X}_{[-1,1]^n}$ désigne la fonction indicatrice de $[-1,1]^n$. Pour la suite on pose $\underline{f}(u) = \frac{1}{2}u^T(A+\rho I)u + a^Tu + c$.

L'algorithme DCA pour résoudre ($\mathcal{DC}$) est le suivant :

Algorithme 5.2.1 *IPDCA-DCA pour résoudre le problème* ($\mathcal{DC}$)

0. Soient x_0 un point initial fourni par IPDCA appliqué au problème ($\mathcal{DC}$) *et ε une précision choisie, poser $k = 0$.*
1. A l'étape k résoudre le problème de programmation linéaire

$$(\mathcal{DC}_k) \min_{u\in[-1,1]^n} u_k^T(A+\rho I)u + a^Tu$$

on pose u_{k+1} la solution de ce problème.
2. si $\| u_{k+1} - u_k \| /(1+ \| u_k \|_k) \leq \varepsilon$ alors fin, sinon $k \leftarrow k+1$ retourner en 1.

Notons qu'une autre décomposition DC aurait conduit au problème DC,

$$\min_{u\in\mathbb{R}^n}\{\mathcal{X}_{[-1,1]^n}(u) + (\frac{1}{2}u^T(A+(\rho+\lambda)I)u + a^Tu + c) - \frac{1}{2}\lambda u^Tu\}$$

où λ est déterminé pour que la matrice $A+(\rho+\lambda)I$ soit définie positive. L'algorithme IPDCA-DCA qui en découlerait aurait été

Algorithme 5.2.2 *IPDCA-DCA pour résoudre le problème* ($\mathcal{DC}$)

0. Soient x_0 un point initial fourni par IPDCA appliqué au problème ($\mathcal{DC}$) *et ε une précision choisie, poser $k = 0$.*
1. A l'étape k résoudre le problème de programmation quadratique convexe

$$\min_{u\in[-1,1]^n} \frac{1}{2}u^T(A+(\rho+\lambda)I)u + a^Tu - \frac{1}{4}\lambda u_k^Tu$$

on pose u_{k+1} la solution de ce problème.
2. si $\| u_{k+1} - u_k \| /(1+ \| u_k \|_k) \leq \varepsilon$ alors fin, sinon $k \leftarrow k+1$ retourner en 1.

Cette seconde décomposition conduit à un DCA potentiellement plus complexe car dans ce cas il nous faut résoudre un sous-problème quadratique convexe à chaque itération, contre un problème de programmation linéaire pour la première décomosition DC dont on calcule explicitement la solution.
En effet si on pose $v_k = (A + \rho I)u_k + a$ on doit résoudre,

$$(\mathcal{DC}_k) \quad \min_{u\in[-1,1]^n} \langle v_k, u\rangle.$$

Soit u_{k+1} la solution de $(\mathcal{DC}_k)$, alors pour $i = 1, 2, \cdots, n$
si $v_k^i < 0$ alors $u_{k+1}^i = 1$,
si $v_k^i > 0$ alors $u_{k+1}^i = -1$,
si $v_k^i = 0$ alors $u_{k+1}^i \in [-1, 1]$.
Le second intérêt qu'il y a à considérer la première décomposition est que pour celle-ci, la suite de points générés par DCA est toujours dans $\{-1, 1\}^n$. Le troisième intérêt de la première décomposition est que dans ce cas on a un programme DC polyédral pour lequel DCA a une convergence finie.

Remarque 5.2.1 *Plusieurs stratégies d'évaluation de la borne supérieure seront examinés :*
- *une évéluation systématique de la borne supérieure sur les deux rectangles issus de la subdivision entière ;*
- *une évaluation de la borne supérieure par DCA ou IPDCA-DCA que dans le cas où la borne inférieure calculée sur le rectangle en question est plus petite que la meilleure borne supérieure connue.*

Chacune de ces stratégies a ses avantages et ses inconvénients comme nous allons le voir à travers les tests numériques.

Nous allons également explorer plusieurs choix de domaines pour l'amélioration de la borne supérieure aussi bien en utilisant DCA tout seul ou IPDCA-DCA. La première consistera à travailler sur le rectangle courant ; le domaine des variables est dans ce cas le rectangle issu de la subdivision entière. La seconde stratégie consistera à calculer la borne supérieure par DCA sur le domaine initial tout entier.
La première stratégie à pour avantage de réduire le domaine des variables, car à l'étape k, $k-1$ variables auront des valeurs fixées à -1 ou à +1. Mais dans le cas de la non évaluation systématique de la borne supérieure sur tous les deux rectangles issus de la subdivision, il y a un risque de rester cantonner dans une patie du domaine alors que la solution se trouve dans l'autre. Ce problème ne se pose pas pour la seconde stratégie mais dans ce cas on travail sur le domaine initial tout entier sans exploiter les informations des itérations précédentes. □

5.2.2 Borne inférieure

Pour calculer la borne supérieure dans la procédure séparation-évaluation, nous considérons le problème relaxé,

$$\begin{cases} \min f(u) \\ \quad u \in [-1, 1]^n, \end{cases}$$

Soit β la valeur optimale de ce problème. On a toujours $\alpha \geq \beta$.

Considérons la décomposition DC de la fonction objectif suivante $f(u) = g(u) - h(u)$ avec

$$g(u) = \frac{1}{2}u^T(A - \mu I)u + a^T u + c \text{ et } h(u) = -\frac{1}{2}\mu u^T u$$

avec μ un réel strictement négatif tel que la matrice $A - \mu I$ est définie positive.
Soit ψ le sous-estimateur de la fonction concave $-h$ sur $R = \prod_{i=1}^{n}[t_i,\ T_i]$. Puisque h est séparable,

$$\psi = \sum_{i=1}^{n}\psi_i \quad \text{avec} \quad \psi_i(u_i) = \frac{\mu}{2}(t_i + T_i)u_i - \frac{\mu}{2}t_i T_i.$$

Dans notre procédure de séparation-évaluation, l'hyper-rectangle R sera toujours symétrique et vaudra toujours $R = [-1,\ 1]^n$, ce qui entraîne que le sous-estimateur ψ sera constant pendant tout le processus d'optimisation ceci grâce à la stratégie de branchement que nous allons préciser plus loin dans le chapitre. Le problème que nous considérons pour le calcul de la borne inférieure est donc

$$(\mathcal{LB}) \begin{cases} \min \bar{f}(u) = g(u) + \frac{\mu}{2}n \\ u \in [-1, 1]^n, \end{cases}$$

car $n = -\sum_{i=1}^{n} t_i T_i$. Le problème quadratique convexe $(\mathcal{LB})$ est résolu grâce à CPLEX.

Proposition 5.2.1 *Soit $u^\star$ la solution du problème convexe $(\mathcal{LB})$, si $u^\star \in \{-1, 1\}^n$, alors $u^\star$ est solution du problème $(\mathcal{P})$.*

Preuve : Pour que $u^\star$ soit solution du problème $(\mathcal{P})$ il faut que $u^\star$ soit dans $\{-1, 1\}^n$ et que $\bar{f}(u^s) = f(u^\star)$. On a déjà $u^\star \in \{-1, 1\}^n$ par hypothèse, de plus on a l'équivalence,

$$\bar{f}(u^\star) = f(u^\star) \iff g(u^\star) - h(u^\star) = g(u^\star) + \text{conv}(-h)(u^\star),$$

soit

$$h(u^\star) = -\text{conv}(-h)(u^\star). \tag{5.4}$$

Or on a bien $h(u^\star) = \frac{\mu}{2}n = -\text{conv}(-h)(u^\star)$ car $u^\star \in \{-1, 1\}^n$. □

La Proposition 5.2.1 nous fournit un test d'arrêt pour l'algorithme de séparation-évaluation que nous allons présenter.

5.2.3 Stratégie de subdivision

La stratégie de subdivision entière intervient quand la solution $u^\star$ du problème $(\mathcal{LB})$ n'est pas dans $\{-1, 1\}^n$. Le choix de la variable sur laquelle on branche est fait en déterminant son indice par la résolution du problème,

$$i_c \in \arg\max_{i=1,2,\cdots,n}\{p_i(u_i) \ :\ u_i \notin \{-1, 1\}\} \tag{5.5}$$

où $p_i(u_i) = (1 + u_i)(1 - u_i)$. L'indice i_c déterminé, on considère la disjonction

$$u_{i_c} = 1 \vee u_{i_c} = -1 \tag{5.6}$$

Pour chaque valeur de u_{i_c} dans la disjonction (5.6) résultera deux domaines dans lesquels la valeur de u_{i_c} sera fixée à -1 et à 1 respectivement. Et le calcul de la borne supérieure et de la borne inférieure sera fait sur des problèmes ayant un nombre de variables inférieure à n car u_{i_c} sera déja fixé. Ainsi à l'étape k de la procédure de séparation-évaluation notre problème ne comportera plus que $n - k$ variables les k autres ayant déjà été fixées lors des itérations antérieures. Cette stratégie qui a montré son efficacité dans dans [117], diffère de celle présentée dans [111] où le nombre de variables reste constant fixé à n jusqu'à la fin du processus de séparation évaluation. Par contre la stratégie de subdivision rectangulaire normale qui y est présentée garantit la décroissance vers zéro de $T_i - t_i$; alors que dans notre cas on a toujours $T_i - t_i = 2$. La stratégie de subdivision entière que nous considérons peut être vue comme un cas particulier de la subdivision rectangulaire normale présentée dans [111].

5.3 Description de l'algorithme de séparation-évaluation

Algorithme 5.3.1 *Algorithme BBA01*

Initialisation

Résoudre le problème quadratique convexe $(\mathcal{LP})$ *pour obtenir la solution* x^0 *et la valeur optimale correspondante* β_0, *si* $x^0 \in \{-1, 1\}^n$ *alors* x^0 *est solution globale du problème* $(\mathcal{Q})$, *sinon appliquer DCA au problème* $(\mathcal{DC})$ *en prenant* x^0 *comme point initial Soit* $\tilde{x_0}$ *la solution trouvée par DCA et* $\gamma_0 = f(x_0)$.
Si $\gamma - \beta_0 \leq \varepsilon$, *alors stop*$\leftarrow$*vrai,* x_0 *est une* $\varepsilon-$*solution pour* $(\mathcal{P})$.
Sinon stop$\leftarrow$ *faux finsi.*
Poser $\mathcal{K}_0 = [-1, 1]^n$ *et* $\mathcal{L} = \mathcal{K}_0$, $k \leftarrow 0$.

Tant que *stop=faux* ***faire***,

1. *Sélectionner un point* x_k *tel que*

$$f(x^k) = \min\{f(x) \ : \ x \in \mathcal{L}\}.$$

soit $\mathcal{K}_k$ *son domaine.*

2. *Déterminer l'indice* i_c^k *en utilisant la formule de subdivision entière 5.5.*
Soient $\mathcal{K}_{-1}^k$ *et* $\mathcal{K}_1^k$ *les domaines résultant après avoir fixé* $x_{i_c^k} = -1$ *et* $x_{i_c^k} = 1$ *respectivement.*

3. *Pour* $\mathcal{K}_{-1}^k$ *et* $\mathcal{K}_1^k$ *résoudre*

$$(\mathcal{LB}_{-1}) \ \min\{g(x) \ : \ x \in \mathcal{K}_{-1}^k\} \ \ et \ \ (\mathcal{LB}_1) \ \min\{g(x) \ : \ x \in \mathcal{K}_1^k\}$$

pour obtenir respectivement les couples $(x^{\mathcal{K}_{-1}^k}, \beta(\mathcal{K}_{-1}^k))$ *et* $(x^{\mathcal{K}_1^k}, \beta(\mathcal{K}_1^k))$.

4. *Si* $x^{\mathcal{K}_i^k} \in \{-1,1\}^n$ *alors stop,* $x^{\mathcal{K}_i^k}$ *est solution globale du problème* $(\mathcal{Q})$, *avec* $i=-1$ *et* $i=1$.
5. *En prenant comme point de départ* $\bar{x}^{\mathcal{K}_{-1}^k}$ *et* $\bar{x}^{\mathcal{K}_1^k}$ *respectivement fournis par IPDCA, résoudre les problèmes*

$$(\mathcal{DC}_{-1})\ \min\{f(x)\ :\ x \in \mathcal{K}_{-1}^k\}\ \text{ et }\ (\mathcal{DC}_1)\ \min\{f(x)\ :\ x \in \mathcal{K}_1^k\}$$

grâce à l'Algorithme 5.2.1 poser $(\tilde{x}_{\mathcal{K}_{-1}^k}, f(\tilde{x}_{\mathcal{K}_{-1}^k}))$ *et* $(\tilde{x}_{\mathcal{K}_1^k}, f(\tilde{x}_{\mathcal{K}_1^k}))$ *les solutions et les valeurs objectif respectives trouvées.*

6. *Si l'on a* $f(\tilde{x}_{\mathcal{K}_i^k}) < \gamma_{k-1}$ *alors poser* $x_k = \tilde{x}_{\mathcal{K}_i^k}$ *et* $\gamma_k = f(\tilde{x}_{\mathcal{K}_i^k})$ *pour* $i=-1$ *et* $i=1$.
7. *Poser* $\mathcal{L} \leftarrow (\mathcal{L}\backslash\mathcal{K}_k) \cup \{\mathcal{K}_i^k : \beta(\mathcal{K}_i^k) < \gamma_k - \varepsilon, i=-1,1\}$.
8. *Si* $\mathcal{L} = \varnothing$, *alors stop←vrai,* x_k *est une* $\varepsilon-$*solution, sinon* $k \leftarrow k+1$ *finsi.*

Fin Tant que

Nous considérons l'algorithme ci-dessous dans lequel la borne supérieure est calculée en utilisant DCA tout seul.

Algorithme 5.3.2 *Algorithme BBA02*

Initialisation

Résoudre le problème quadratique convexe $(\mathcal{LP})$ *pour obtenir la solution* x^0 *et la valeur optimale correspondante* β_0, *si* $x^0 \in \{-1,1\}^n$ *alors* x^0 *est solution globale du problème* $(\mathcal{Q})$, *sinon appliquer DCA au problème* $(\mathcal{DC})$ *en prenant* x^0 *comme point initial Soit* $\tilde{x_0}$ *la solution trouvée par DCA et* $\gamma_0 = f(x_0)$.
Si $\gamma - \beta_0 \leq \varepsilon$, *alors stop←vrai,* x_0 *est une* $\varepsilon-$*solution pour* $(\mathcal{P})$.
Sinon stop← faux finsi.
Poser $\mathcal{K}_0 = [-1,1]^n$ *et* $\mathcal{L} = \mathcal{K}_0$, $k \leftarrow 0$.

Tant que *stop=faux* ***faire***,

1. *Sélectionner un point* x_k *tel que*

$$f(x^k) = \min\{f(x)\ :\ x \in \mathcal{L}\}.$$

soit $\mathcal{K}_k$ *son domaine.*

2. *Déterminer l'indice* i_c^k *en utilisant la formule de subdivision entière 5.5.*
Soient $\mathcal{K}_{-1}^k$ *et* $\mathcal{K}_1^k$ *les domaines résultant après avoir fixé* $x_{i_c^k} = -1$ *et* $x_{i_c^k} = 1$ *respectivement.*
3. *Pour* $\mathcal{K}_{-1}^k$ *et* $\mathcal{K}_1^k$ *résoudre*

$$(\mathcal{LB}_{-1})\ \min\{g(x)\ :\ x \in \mathcal{K}_{-1}^k\}\ \text{ et }\ (\mathcal{LB}_1)\ \min\{g(x)\ :\ x \in \mathcal{K}_1^k\}$$

pour obtenir respectivement les couples $(x^{\mathcal{K}_{-1}^k}, \beta(\mathcal{K}_{-1}^k))$ *et* $(x^{\mathcal{K}_1^k}, \beta(\mathcal{K}_1^k))$.

4. Si $x^{\mathcal{K}_i^k} \in \{-1,1\}^n$ alors stop, $x^{\mathcal{K}_i^k}$ est solution globale du problème $(\mathcal{Q})$, avec $i=-1$ et $i=1$.
5. En prenant comme point de départ $x^{\mathcal{K}_{-1}^k}$ et $x^{\mathcal{K}_1^k}$ respectivement, résoudre les problèmes

$$(\mathcal{DC}_{-1})\ \min\{\underline{f}(x)\ :\ x\in\mathcal{K}_{-1}^k\}\ \text{ et }\ (\mathcal{DC}_1)\ \min\{\underline{f}(x)\ :\ x\in\mathcal{K}_1^k\}$$

grâce à l'Algorithme 5.2.1 poser $(\tilde{x}_{\mathcal{K}_{-1}^k}, f(\tilde{x}_{\mathcal{K}_{-1}^k}))$ et $(\tilde{x}_{\mathcal{K}_1^k}, f(\tilde{x}_{\mathcal{K}_1^k}))$ les solutions et les valeurs objectif respectives trouvées.
6. Si l'on a $f(\tilde{x}_{\mathcal{K}_i^k}) < \gamma_{k-1}$ alors poser $x_k = \tilde{x}_{\mathcal{K}_i^k}$ et $\gamma_k = f(\tilde{x}_{\mathcal{K}_i^k})$ pour $i=-1$ et $i=1$.
7. Poser $\mathcal{L} \leftarrow (\mathcal{L}\backslash\mathcal{K}_k)\cup\{\mathcal{K}_i^k : \beta(\mathcal{K}_i^k) < \gamma_k - \varepsilon, i=-1,1\}$.
8. Si $\mathcal{L}=\varnothing$, alors stop←vrai, x_k est une $\varepsilon-$solution, sinon $k\leftarrow k+1$ finsi.

Fin Tant que

Dans le dernier algorithme DCA est appelé sur le rectangle actuel uniquement lorsque la borne inférieure du sous-problème en question est inférieure à la meilleure borne supérieure connue.

Algorithme 5.3.3 *Algorithme BBA03*

Initialisation

Résoudre le problème quadratique convexe $(\mathcal{LP})$ pour obtenir la solution x^0 et la valeur optimale correspondante β_0, si $x^0 \in \{-1,1\}^n$ alors x^0 est solution globale du problème $(\mathcal{Q})$, sinon appliquer DCA au problème $(\mathcal{DC})$ en prenant x^0 comme point initial Soit $\tilde{x_0}$ la solution trouvée par DCA et $\gamma_0 = f(x_0)$.
Si $\gamma - \beta_0 \le \varepsilon$, alors stop←vrai, x_0 est une $\varepsilon-$solution pour $(\mathcal{P})$.
Sinon stop← faux finsi.
Poser $\mathcal{K}_0 = [-1,1]^n$ et $\mathcal{L}=\mathcal{K}_0$, $k\leftarrow 0$.

***Tant que** stop=faux **faire**,*
1. Sélectionner un point x_k tel que

$$f(x^k) = \min\{f(x)\ :\ x\in\mathcal{L}\}.$$

soit $\mathcal{K}_k$ son domaine.
2. Déterminer l'indice i_c^k en utilisant la formule de subdivision entière 5.5.
Soient $\mathcal{K}_{-1}^k$ et $\mathcal{K}_1^k$ les domaines résultant après avoir fixé $x_{i_c^k} = -1$ et $x_{i_c^k} = 1$ respectivement.
3. Pour $\mathcal{K}_{-1}^k$ et $\mathcal{K}_1^k$ résoudre

$$(\mathcal{LB}_{-1})\ \min\{g(x)\ :\ x\in\mathcal{K}_{-1}^k\}\ \text{ et }\ (\mathcal{LB}_1)\ \min\{g(x)\ :\ x\in\mathcal{K}_1^k\}$$

pour obtenir respectivement les couples $(x^{\mathcal{K}_{-1}^k}, \beta(\mathcal{K}_{-1}^k))$ et $(x^{\mathcal{K}_1^k}, \beta(\mathcal{K}_1^k))$.

4. Si $x^{\mathcal{K}_i^k} \in \{-1,1\}^n$ *alors stop,* $x^{\mathcal{K}_i^k}$ *est solution globale du problème* $(\mathcal{Q})$*, avec* $i=-1$ *et* $i=1$.

5. En prenant comme point de départ $x^{\mathcal{K}_{-1}^k}$*, résoudre*

$$(\mathcal{DC}_{-1})\ \min\{\underline{f}(x)\ :\ x\in\mathcal{K}_{-1}^k\}$$

grâce à l'Algorithme 5.2.1 si $f(x^{\mathcal{K}_{-1}^k})<\gamma_{k-1}$ *et poser* $(\tilde{x}_{\mathcal{K}_{-1}^k}, f(\tilde{x}_{\mathcal{K}_{-1}^k}))$ *les solutions et les valeurs objectif trouvées. De même si* $f(x^{\mathcal{K}_1^k})<\gamma_{k-1}$ *résoudre*

$$(\mathcal{DC}_{1})\ \min\{\underline{f}(x)\ :\ x\in\mathcal{K}_{1}^k\}$$

grâce à l'Algorithme 5.2.1 en prenant comme point de départ $x^{\mathcal{K}_1^k}$ *on pose* $(\tilde{x}_{\mathcal{K}_1^k}, f(\tilde{x}_{\mathcal{K}_1^k}))$ *la solution et la valeur objectif trouvées.*

6. Si l'on a $f(\tilde{x}_{\mathcal{K}_i^k})<\gamma_{k-1}$ *alors poser* $x_k=\tilde{x}_{\mathcal{K}_i^k}$ *et* $\gamma_k=f(\tilde{x}_{\mathcal{K}_i^k})$ *pour* $i=-1$ *et* $i=1$.

7. Poser $\mathcal{L}\leftarrow(\mathcal{L}\backslash\mathcal{K}_k)\cup\{\mathcal{K}_i^k : \beta(\mathcal{K}_i^k)<\gamma_k-\varepsilon, i=-1,1\}$.

8. Si $\mathcal{L}=\varnothing$*, alors stop*$\leftarrow$*vrai,* x_k *est une* $\varepsilon-$*solution, sinon* $k\leftarrow k+1$ *finsi.*

Fin Tant que

Le théorème suivant établit la convergence des algorithmes ci-dessus.

Théorème 5.3.1 *[111]*

– *Si l'algorithme se termine à l'itération* k*, alors* x_k *est une solution globale du problème* $(\mathcal{P})$.

– *Si l'algorithme est infini, alors il génère une suite bornée* $\{x_k\}_k$ *dont tout point d'accumulation est un optimum global de* $(\mathcal{P})$*, et*

$$f(x_k^{DCA})\searrow f_\star,\quad f(x^{R_k})\nearrow f_\star.$$

5.4 Résultats numériques

Les tests numériques ont été tirés de la collection de graphes G-set due à Giovanni Rinaldi et utilisée par Benson et *al.* [178]. Ces graphes sont générés aléatoirement. Les problèmes sont traités comme problèmes de coupe maximale.

Un problème de coupe maximale se formule comme suit, étant donné un graphe $G=(V,E)$, non dirigé et connexe avec $V=\{1,2,\cdots,n\}$ et $E\subset\{(i,j)\ :\ 1\le i<j\le n\}$ donnés. Soit $\omega_{ij}=\omega_{ji}$, le poids associé à l'arrête (i,j) donné tel que $\omega_{ij}=0$ pour $(i,j)\in E$; en particulier on pose $\omega_{ii}=0$. Le problème de coupe maximale ou max-cut en anglais consiste à trouver une partition (V_1,V_2) de V telle que la somme des poids des arrêtes entre V_1 et V_2 est maximale. Le problème de coupe maximale s'écrit alors,

$$\begin{cases}\min \frac{1}{2}\sum\limits_{1\le i<j\le n}\omega_{ij}(1-u_iu_j)\\ |u_i|=1,\quad i=1,2,\cdots,n.\end{cases}$$

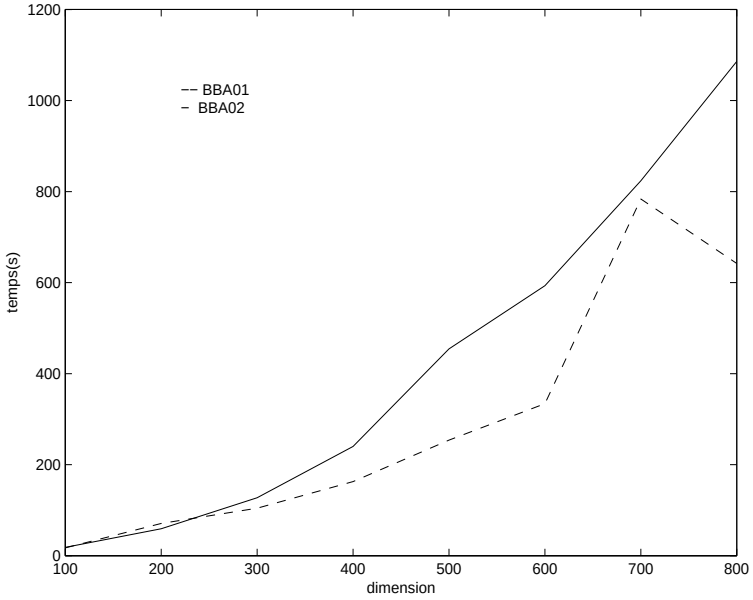


FIG. 5.1 – Temps en fonction de la dimension.

Vu la complexité de ce problème, plusieurs heuristiques et algorithmes d'approximation ont été proposés dans la littérature voir [179, 180] et références incluses.

Pour nos tests numériques le système d'exploitation est Windows XP, la machine un PC de 1300 Mhz de vistesse processeur avec 512K de RAM, les programmes ont été écrits en C, la précision choisie dans BB01, BB03 est $\varepsilon = 10e^{-6}$. Le calcul de la plus petite valeur propre et de la plus grande valeur propre a été effectué par MATLAB.
Dans le Tableau 5.1,
$-Nom$ désigne le nom du graphe,
$-dim$ la dimension du problème,
$-iter$ le nombre d'itération du processus de séparation-évaluation,
$-LB$ la borne inférieure à l'issue du processus de séparation-évaluation,
$-UB$ la borne supérieure à l'issue du processus de séparation-évaluation,
$-NbRed$ le nombre de redémarrages de DCA,
$-temps$ la durée du processus de séparation-évaluation.

Un point important que l'on observe sur le Tableau 5.1 est la constance du nombre d'itérations en fonction de la dimension ; le nombre d'itérations y est toujours en effet égale à la dimension du problème. Nous observons sur la Figure 5.2 que le nombre de redémarrages de DCA dans BB01 est en moyenne supérieur à celui de BB02 ceci est confirmé sur la Figure 5.2, on y voit que la même observation s'applique au niveau du temps écoulé.
Notons la brusque cassure sur la Figure 5.2 pour la courbe correspondant à BB02, avec un jeu de données plus important on aurait eu une remontée de celle-ci.

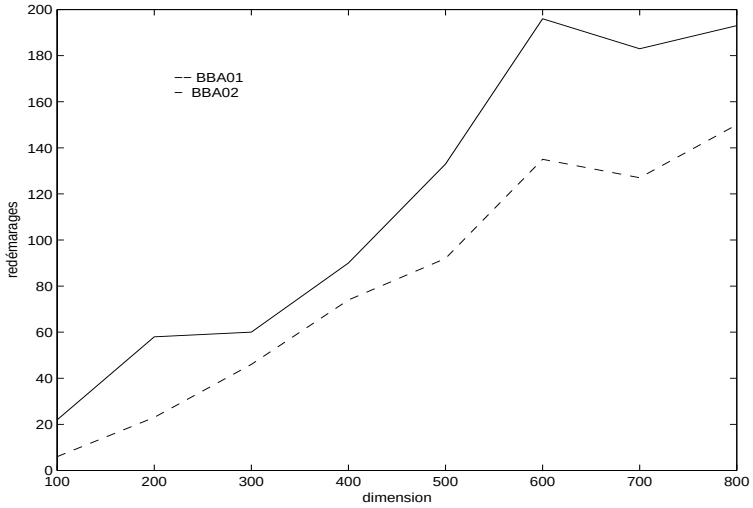


FIG. 5.2 – Nombre de redémarrages de DCA.

Signalons que pour les deux algorithmes, pour ce jeu de données les solutions trouvées par DCA sont toutes dans $\{-1, 1\}^n$.
Dans le Tableau 5.2 les algorithmes BB02 et BB03 sont comparés, et comme on pouvait s'y attendre, le temps de calcul est inférieur pour BB03 mais le nombre de redémarrages de DCA est identique pour les deux algorithmes.

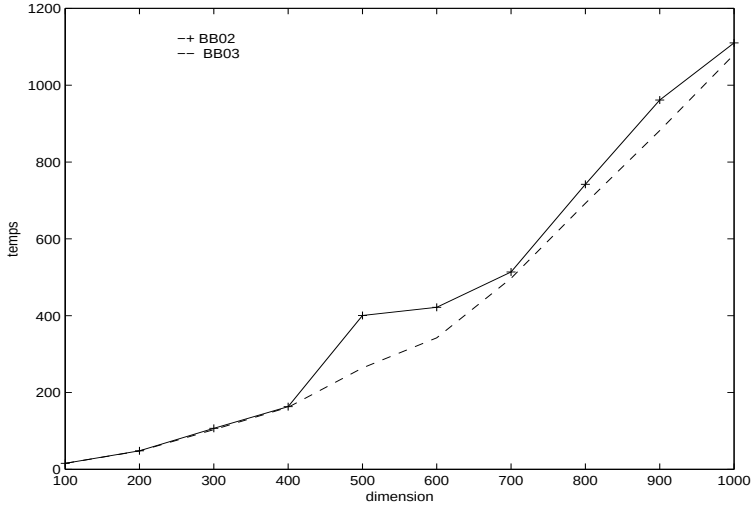


FIG. 5.3 – Temps en fonction de la dimension pour BB02 et BB03.

Nom	*dim*	BBA01					BBA02				
		iter	*NbRed*	*LB*	*UB*	*temps*	*iter*	*NbRed*	*LB*	*UB*	*temps*
G10	100	100	22	-135.47	-113.86	18.36	100	6	-135.47	-123.29	17.20
G20	200	200	58	-250.49	-218.32	59.05	200	23	-250.49	-232.75	70.88
G30	300	300	60	-371.30	-350.05	127.27	300	46	-371.30	-352.48	104.25
G40	400	400	90	-487.20	-465.83	239.99	400	74	-487.20	-467.81	162.76
G50	500	500	133	-601.86	-580.84	454.46	500	92	-601.86	-582.33	253.98
G60	600	600	196	-736.12	-714.61	593.16	600	135	-736.12	-715.57	334.11
G70	700	700	183	-774.41	-753.08	823.80	700	127	-774.41	-755.65	783.62
G80	800	800	193	-876.75	-857.13	1085.92	800	150	-876.75	-863.47	642.06

TAB. 5.1 – Résultats numériques pour les algorithmes BBA01 et BBA02.

Nom	*dim*	BBA02					BBA03				
		iter	*NbRed*	*LB*	*UB*	*temps*	*iter*	*NbRed*	*LB*	*UB*	*temps*
Gg10	100	84	10	-120.95	-120.68	15.44	84	10	-120.95	-120.68	15.42
Gg20	200	181	18	-246.80	-246.70	48.16	181	18	-246.80	-246.70	47.43
Gg30	300	368	34	-379.48	-379.28	106.48	368	34	-379.48	-379.28	103.26
Gg40	400	256	19	-498.50	-498.30	163.00	256	19	-498.50	-498.30	161.41
Gg50	500	446	60	-631.60	-631.10	400.29	446	60	-631.60	-631.10	263.96
Gg60	600	543	70	-757.03	-756.08	421.60	543	70	-757.03	-756.08	342.26
Gg70	700	636	86	-858.58	-858.24	513.56	636	86	-858.58	-858.24	497.30
Gg80	800	753	121	-960.16	-960.00	741.85	753	121	-960.16	-960.00	692.68
Gg90	900	843	125	-1090.90	-1090.80	961.50	843	125	-1090.90	-1090.80	881.90
Gg100	1000	926	164	-1219.75	-1219.50	1110.25	926	164	-1219.75	-1219.50	1079.92

TAB. 5.2 – Résultats numériques pour les algorithmes BBA02 et BBA03.

Chapitre 6

Optimisation Monotone. Approche séparation-évaluation pour résoudre un problème quadratique avec contraintes linéaires

6.1 Introduction

Les problèmes d'optimisation convexes et en particulier linéaires ont été largement étudiés. Il existe dans la littérature des algorithmes permettant de résoudre des problèmes de tailles moyennes et de grandes tailles de ce type. Or, les problèmes rencontrés dans la vie courante sont souvent non convexes, et comportent de nombreux maxima locaux. Les méthodes classiques de l'optimisation convexe ne fournissent en principe que des minima locaux souvent éloignés de l'optimum global. D'où le développement ces dernières années de nouvelles approches qui même si elles ne garantissent pas toujours la globalité de l'optimum, permettent de résoudre une classe de problèmes plus larges.

Dans ce chapitre nous présenterons en première partie la méthode monotone introduite par Hoang Tuy dans [143]. Nous y rappellerons les principales définitions et propositions, puis une approche séparation-évaluation sera présentée pour la résolution globale d'un problème quadratique ; dans cette approche la borne inférieure est calculée grâce à DCA et la borne supérieure par la méthode monotone. Le chapitre se termine par des tests numériques.

6.2 La Méthode d'optimisation monotone

6.2.1 Fonction croissante et ensembles normaux

Dans cette partie nous vous présenterons quelques définitions qui nous seront indispensables pour la suite de notre exposé.

Définition 6.2.1 *x' domine x*
Soient x et x' dans $\mathbb{R}^n$, on dira que x' domine x si,

$$x'_i \geqslant x_i, \forall i = 1, 2, \cdots, n.$$

On définit la *dominance stricte* en considérant des inégalités strictes dans la définition précédente.
On note

$$\mathbb{R}^n_+ = \{x \in \mathbb{R}^n | x \geqslant 0\}$$

et

$$\mathbb{R}^n_{++} = \{x \in \mathbb{R}^n \mid x > 0\}.$$

Pour x dans $\mathbb{R}^n_+$ on note $I(x) = \{i \mid x_i = 0\}$ et

$$\mathsf{K}_x = \{x' \in \mathbb{R}^n_+ \mid x'_i > x_i, \forall i \notin I(x)\},$$

$$cl\mathsf{K}_x = \{x' \in \mathbb{R}^n_+ \mid x' \geq x\}.$$

Soient a et b dans $\mathbb{R}^n_+$, si $a \leqslant b$ on définit l'*hyperrectangle* $[a, b]$ comme étant l'ensemble des x tels que $a \leqslant x \leqslant b$.

Définition 6.2.2 *fonction croissante*
Une fonction $f : \mathbb{R}^n \to \mathbb{R}$ est dite croissante sur $\mathbb{R}^n_+$ si $f(x) \leqslant f(x')$ si $0 \leqslant x \leqslant x'$; et f est croissante sur l'hyperrectangle $[a, b] \subset \mathbb{R}^n_+$ si $f(x) \leqslant f(x')$ pour $a \leqslant x \leqslant x' \leqslant b$.

Définition 6.2.3 *Ensemble normal*
Un ensemble $G \subset \mathbb{R}^n_+ (resp. H \subset \mathbb{R}^n_+)$ est dit normal (resp. anti − normal) si pour tout point x, x' dans $\mathbb{R}^n_+$ tels que $x \leqslant x'$, si $x' \in G$ (resp. $x \in H$) alors $x \in G$ (resp. $x' \in H$).

Proposition 6.2.1 *Etant donné un ensemble $D \subset \mathbb{R}^n_+$ l'ensemble $N[D] = (D - \mathbb{R}^n_+) \cap \mathbb{R}^n_+$ est le plus petit ensemble normal contenant D. Si D est compact alors $N[D]$ est compact.*

Proposition 6.2.2 *L'intersection et l'union d'une famille d'ensembles normaux sont des ensembles normaux.*

Proposition 6.2.3 *Tout ensemble normal est connexe. Un ensemble normal G est d'intérieur non vide si et seulement si G contient un élément u de $\mathbb{R}^n_{++}$. La fermeture d'un ensemble normal est un ensemble normal.*

La proposition suivante montre qu'un ensemble normal borné est essentiellement un ensemble de niveau d'une fonction croissante.

Proposition 6.2.4 *[143]*
- *Pour toute fonction croissante g dans $\mathbb{R}^n_+$ l'ensemble $G = \{x' \in \mathbb{R}^n_+ \mid g(x) \leqslant 1\}$ est normal et est fermé si g(x) est s.c.i.*

- *Réciproquement, pour tout ensemble fermé normal $G \subset \mathbb{R}^n_+$ d'intérieur non vide il existe une fonction s.c.i croissante $g : \mathbb{R}^n \to \mathbb{R}$ telle que $G = \{x' \in \mathbb{R}^n_+ \mid g(x) \leqslant 1\}$.*

Preuve : La première assertion est évidente nous ne démontrerons que la seconde.
La démonstration de cette proposition se subdivise en quatre étapes.
On pose $g(x) = \mathcal{I}nf\{\alpha \geq 0 \mid x \in \alpha G\}$.
– Montrons que $0 \leqslant g(x) < +\infty$.
En effet G est d'intérieur non vide et normal de la proposition précédente il existe un u dans $\mathbb{R}^n_{++}$ tel que $u \in G$, et G est connexe donc $[0, u] \subset G$. De plus pour tout $x \in \mathbb{R}^n_+$ le segment de droite $\{\alpha x \mid \alpha \geq 0\}$ rencontre $(0, u] \subset G$.
– Montrons que g est croissante.
Puisque pour tout $\lambda \geq 0$ l'ensemble λG est normal, alors pour $x \leq x'$, si $x' \in \lambda G$, alors $x \in \lambda G$ et donc $g(x) \leq g(x')$.
– Montrons que $G = \{x \in \mathbb{R}^n_+ \mid g(x) \leq 1\}$.

1. Montrons que $G \subset \{x \in \mathbb{R}^n_+ \mid g(x) \leq 1\}$.
En effet il est évident que si $x \in G$ alors $g(x) \leq 1$.
2. Montrons que $G \supset \{x \in \mathbb{R}^n_+ \mid g(x) \leq 1\}$.
Par l'absurde, soit $x \notin G$ montrons que $x \notin \{x \in \mathbb{R}^n_+ \mid g(x) \leq 1\}$. En effet puisque G est fermé il existe $\alpha > 1$ tel que $x \notin \alpha G$ et puisque G est un ensemble normal, $\forall \lambda \leq \alpha$ on a $x \notin \lambda G$, et donc $g(x) \geq \alpha > 1$, c'est-à-dire $x \notin \{x \in \mathbb{R}^n_+ \mid g(x) \leq 1\}$.

– Montrons que g est une fonction s.c.i.
Il suffit de montrer que pour $\alpha \in \mathbb{R}_+$ l'ensemble $\Omega_\alpha = \{x \in \mathbb{R}^n_+ \mid g(x) \leq \alpha\}$ est fermé. Soit $\{x_n\}_n \subset \Omega_\alpha$ telle que $\{x_n\}_n$ converge vers x_o. On a $\forall n$, $g(x_n) \leq \alpha$. Pour tout $\alpha' > \alpha$ on a $g(x_n) < \alpha'$ $\forall n$, donc $x_n \in \alpha' G$. Mais puisque $\alpha' G$ est fermé, on a $x_o \in \alpha' G$ et puisque α' peut être pris aussi proche de α, que l'on veut, on a $g(x_o) \leq \alpha$. Ainsi Ω_α est fermé.

□

Définition 6.2.4 *Borne supérieure d'un ensemble*
On dira qu'un point $y \in \mathbb{R}^n_+$ appartient à la borne supérieure d'un ensemble normal et borné G si $y \in clG$ avec $K_y \subset \mathbb{R}^n_+ \backslash G$. On notera $\partial^+ G$ la borne supérieure de G.

Proposition 6.2.5 *[123] Soit $G \subset [0, b]$ un ensemble compact et normal d'intérieur non vide. Pour tout point $z \in \mathbb{R}^n_+ \backslash \{0\}$ la demi-droite allant de 0 à z rencontre $\partial^+ G$ en un unique point $\pi_G(z)$, défini par :*

$$\pi_G(z) = \lambda z, \lambda = \max\{\alpha > 0 \mid \alpha z \in G\}. \tag{6.1}$$

Preuve : Puisque G est d'intérieur non vide, il existe un $\alpha > 0$ tel que $\alpha z \in G$. La compacité de G nous assure que λ défini dans (6.1) par la formule ci-dessus vérifie $0 < \lambda < +\infty$ et que $y = \lambda y \in G$.
De plus si $x \in K_y$, puisque pour tout $z' = \alpha z$ avec $\alpha > \lambda$ on a $z' \in K_y$, il existe donc $\alpha > \lambda$ vérifiant $y \leq z' \leq x$, donc par normalité de G, $z' \in G$, ce qui contredit la définition de λ. Ainsi $K_y \cap G = \emptyset$ et donc $y \in \partial^+ G$.
Démontrons l'unicité. Si $y' = \lambda' z$ avec $\lambda' > \lambda$ alors $y' \in K_y$, donc $y' \notin G$. Ce qui signifie que si $y' = \lambda' z \in \partial^+ G$ alors nécessairement $\lambda' \leq \lambda$. Et en interchangeant le rôle de λ et λ' on

aura $\lambda \leq \lambda'$, d'où l'unicité. □

6.2.2 Maximisation d'une fonction croissante

Nous dirons qu'un problème est un problème d'optimisation monotone s'il s'agit d'une maximisation ou d'une minimisation d'une fonction croissante sur l'intersection d'un ensemble normal et d'un ensemble anti-normal.
Considérons le problème suivant

$$\max\{f(x) \mid x \in G \cap H\} \tag{6.2}$$

où $G \subset [0, b] \subset \mathbb{R}^n_+$ est un ensemble compact d'intérieur non vide, H un ensemble anti-normal et fermé, et f une fonction croissante sur $[0, b]$. On pose $\Omega = G \cap H$.

Proposition 6.2.6 *[143] Le maximum de f sur $G \cap H$ s'il existe est atteint sur $(\partial^+ G) \cap H$.*

Preuve : Puisque G est d'intérieur non vide, si $G \cap H \neq \emptyset$ alors on a nécessairement $(G \cap H)\backslash\{0\} \neq \emptyset$ et donc puisque f est croissante, si f atteint son maximum sur $G \cap H$, il existe un $z \neq 0$ dans $G \cap H \subset \mathbb{R}^n_+$ tel que f y atteint son maximum. Alors on sait que la demi-droite allant de 0 à z rencontre $\partial^+ G$ en un unique point $y = \pi_G(z)$. Puisque $z \in G$ nous devons donc avoir $z \leq y$, ce qui implique que $y \in H$ car $z \in H$ et H normal. Ainsi, $y \in (\partial^+ G) \cap H$. Et d'autre part, puisque f est croissante, et $y \geq z$, il s'en suit que $f(y) \geq f(z)$, c'est-à-dire que f atteint son maximum en y sur $G \cap H$. □
Notez que très souvent les ensembles H et G se présentent sous la forme

$$G = \{x \in \mathbb{R}^n_+ \mid g_i(x) \leq 1, \;\; i = 1, 2, \cdots, m_1\} \tag{6.3}$$

et

$$H = \{x \in \mathbb{R}^n_+ \mid h_j(x) \geq 1, \;\; j = m_1 + 1, \cdots, m\} \tag{6.4}$$

En posant $g(x) = \max\{g_i(x), i = 1, 2, \cdots, m_1\}$ et $h(x) = min\{h_j(x), j = m_1 + 1, \cdots, m\}$ on peut alors écrire

$$G = \{x \in \mathbb{R}^n_+ \mid g(x) \leq 1\} \tag{6.5}$$

$$H = \{x \in \mathbb{R}^n_+ \mid h(x) \geq 1\} \tag{6.6}$$

On a donc le corollaire suivant

Corollaire 6.2.1 *Si les ensembles G et H sont définis comme en* (6.3) *et* (6.4) *respectivement où g et h sont des fonctions continues et croissantes et telle que $g(0) < 1$, alors le problème* (6.2) *est équivalent à*

$$\max\{f(x) \mid g(x) = 1, \; h(x) \geq 1, \; x \geq 0\}$$

Proposition 6.2.7 *Si Ω est un ensemble compact de $\mathbb{R}^n_+$ alors le problème* $\max\{f(x) \mid x \in \Omega\}$ *où f est une fonction croissante est équivalent à* $\max\{f(x) \mid x \in N[\Omega]\}$

Preuve : Soit z le point de Ω où f atteint son maximum sur Ω. Alors pour tout $x \in \Omega$, $f(x) \leq f(z)$ et donc $f(z) \geq f(x')\ \forall x' \in (x - \mathbb{R}^n_+) \cap \mathbb{R}^n_+$. Donc $f(x) \leq f(z)\ \forall x \in N[\Omega]$. □

6.2.3 Approximation des ensembles normaux par des polyblocs

Définition 6.2.5 *Polybloc*
Un ensemble P de $\mathbb{R}^n_+$ est un polybloc dans $[a,b] \subset \mathbb{R}^n_+$ si P est une union d'un nombre fini d'hyperrectangles $[0,z]$, $z \in T \subset [a,b], (|T| < +\infty)$; T est l'ensemble des sommets du plybloc P.
Un sommet $z \in T$ est dit sommet propre s'il n'est dominé par aucun autre élément de $z' \in T$. Ce qui signifie que $z \notin [0,z'], \forall z' \in T \backslash \{z\}$.

Proposition 6.2.8 *Tout polybloc est fermé et normal. L'intersection d'un nombre fini de polyblocs est un polybloc.*

Proposition 6.2.9 *Soit $G \subset [0,b]$ un ensemble compact et normal. Pour tout $z \in [0,b] \backslash G$ il existe un point x tel que $z \in K_x$ et $K_x \subset \mathbb{R}^n_+ \backslash G$.*

Proposition 6.2.10 *Si $\bar{x} \in [0,b]$ et $\bar{z} \in [0,b] \cap K_{\bar{x}}$ alors $P = [0,\bar{z}] \backslash K_{\bar{x}}$ est un polybloc dans $[0,b]$ de sommets*

$$z^i = \bar{z} - (\bar{z}_i - \bar{x}_i)e^i, i \notin I(\bar{x}).$$

Preuve : Soit $K_i = \{x \in \mathbb{R}^n_+ \mid \bar{x}_i < x_i\}$. Puisque $K_{\bar{x}} = \cap_{i \notin I(\bar{x})} K_i$, on a

$$\begin{aligned} P &= [0,\bar{z}] \backslash K_{\bar{x}} \\ &= \cup_{i \notin I(\bar{x})} [0,\bar{z}] \backslash K_i. \end{aligned}$$

Mais

$$[0,\bar{z}] \backslash K_i = \{x \mid 0 \leq x_i \leq \bar{x}_i,\ 0 \leq x_j \leq \bar{z}_j\ \ \forall j \neq i\} = [0,z^i]$$

avec z^i le vecteur défini par :

$$z^i_j = \bar{z}_j \ \text{ et } \ z^i_i = \bar{x}_i,\ \forall j \neq i,$$

c'est-à-dire

$$z^i = \bar{z} - (\bar{z}_i - \bar{x}_i)e^i,\ i \notin I(\bar{x}).$$

□

Théorème 6.2.1 *Soit G un ensemble compact de $[0,b] \subset \mathbb{R}^n_+$. Alors les assertions suivantes sont équivalentes :*
(i) G est normal ;
(ii) Pour tout point $z \in G^b = [0,b] \backslash G$ il existe un polybloc dans $[0,b]$ qui sépare z de G. (i.e qui contient G mais pas z.)
(iii) G est une intersection d'une famille de polyblocs dans $[0,b]$.

Preuve :
- $(i) \Rightarrow (ii)$ On sait de la proposition 8 que si $z \in G^b$, alors il existe un x tel que $z \in K_x$ avec $K_x \cap G = \emptyset$ i.e $P = [0,b]\backslash K_x$contient G et non z. D'autre part P est un polybloc d'après la Proposition 6.10.
- $(ii) \Rightarrow (iii)$ Soit E l'intersection de tous les polyblocs contenant G. On a donc $G \subset E$ si (i) est vraie, alors pour tout $z \in E\backslash G$ il existe un polybloc contenant G mais pas z, donc nécessairement $E \subset G$.
- $(iii) \Rightarrow (i)$ En effet tout polybloc est un ensemble fermé et normal.

□

Conséquence :
Un ensemble compact normal(rep. convexe) peut être approcher aussi proche que l'on veut par un polybloc(resp. par un polyèdre). La Figure 6.1 illustre cette approximation.
De plus le maximum d'une fonction croissante(resp. convexe) sur un polybloc sera atteint sur un sommet du polybloc(polyèdre).

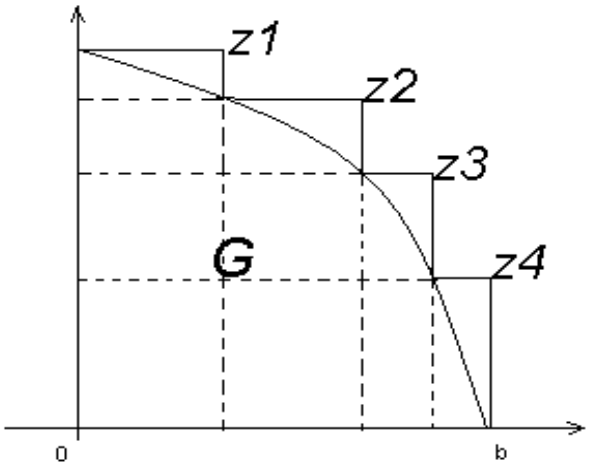


FIG. 6.1 – Approximation d'un ensemble normal par des polyblocs.

Proposition 6.2.11 *Soit $x^k = \pi_G(z^k)$. Alors le polybloc P_{k+1} obtenu de P_k en remplaçant $[0,z^k]$ par $[0,z^k]\backslash K_{x^k}$ vérifie $\Omega \subset P_{k+1} \subset P_k\backslash\{z^k\}$.*

Le polybloc P_{k+1} est généré par,

$$V_{k+1} = (T_k\backslash\{z^k\}) \cup \{z^{k,1}, \cdots, z^{k,n}\}$$

avec $z^{k,i} = z^k - (z_i^k - x_i^k)e^i$.

Proposition 6.2.12 *L'ensemble des sommets propres de P_{k+1} obtenu de V_{k+1} en éliminant les sommets impropres selon la règle :*
Pour tout $z \in T_k \backslash \{z^k\}$: si $z \geq x^k$, et si $z_i < z_i^k$ pour exactement un $i \in \{1, 2, \cdots, n\}$ alors éliminer $z^{k,i}$.

6.2.4 Algorithme d'approximation extérieure par des polyblocs

On suppose qu'il existe un $a \in]0, b[$ tel que $0 < a \leq x$, $\forall x \in H$.

Définition 6.2.6 *Un point réalisable $\bar{x}$ est dit solution ϵ-optimale si*

$$f(\bar{x}) \geq f(x) - \epsilon, \ \forall x \in G \cap H.$$

Algorithme 6.2.1

1. *Initialisation. Choisir $\epsilon > 0$(la tolérance). Si $a \notin G$ le problème est irréalisable. Sinon on pose $T_1 = \{b\}$. Soit $\bar{x}^1$ la meilleure solution réalisable disponible on pose $CBV = f(\bar{x}^1)$, sinon on prend $CBV = -\infty$ si l'on ne dispose pas d'un tel $\bar{x}^1$.*
2. *Etape 1. Retirer de T_k tous les z tels que $z < a$ et tous les z tels que $f(z) \leq CBV + \epsilon$ Notons $\tilde{T}_k$ l'ensemble des éléments restant.*
3. *Etape 2. Si $\tilde{T}_k = \emptyset$ alors Stop : si $CBV = -\infty$ le problème est irréalisable ; si $CBV > -\infty$ la meilleure solution actuelle est acceptée comme solution ϵ-optimale pour le problème.*
4. *Etape 3. $\tilde{T}_k \neq \emptyset$, choisir $z^k \in \arg\max\{f(z) \mid z \in \tilde{T}_k\}$. Calculer $x^k = \pi_G(z^k)$. Si $x^k = z^k$ $(i.e z^k \in G)$ alors z^k est une solution optimale. Sinon déterminer la nouvelle meilleure solution réalisable $\bar{x}^{k+1}$ et la nouvelle valeur de CBV*
5. *Etape 4. De $V_{k+1} = (\tilde{T}_k \backslash \{z^k\}) \cup \{z^{k,1}, \cdots, z^{k,n}\}$ éliminer tous les éléments impropres en utilisant la règle de la(proposition 12) on note T_{k+1} l'ensemble résultant.*
6. *Etape 5. Poser $k \leftarrow k + 1$ et retourner en Etape 1.*

Théorème 6.2.2 *[143] Si l'algorithme 1 est infini, chacune des suites générées $\{z^k\}_k$, $\{x^k\}_k$ admet une sous-suite qui converge vers la solution optimale.*

6.2.5 Optimisation de la différence de fonctions croissantes

Considérons le problème

$$\max\{f(x) - g(x) \mid x \in G \cap H\}$$

où G est un ensemble normal dans $[0, b] \subset \mathbb{R}^n_+$, H un ensemble anti-normal dans $[0, b]$ f et g des fonctions croissantes sur $[0, b]$.
Pour tout $x \in [0, b]$ puisque $g(x) \leq g(b)$, on a $g(x) + t = g(b)$ pour $t = g(b) - g(x) \geq 0$. On peut donc réécrire le problème comme suit :

$$\max\{f(x) + t - g(b) \mid x \in G \cap H, t = g(b) - g(x)\}$$

et en ajoutant la constante $g(b)$ à l'objectif on obtient le problème

$$\max\{f(x)+t \mid x \in G \cap H, 0 \leq t \leq g(b)-g(x)\}$$

soit

$$\max\{f(x)+t \mid (x,t) \in D \cap E\}$$

avec

$$E = \{(x,t) \mid x \in G, t+g(x) \leq g(b), 0 \leq t \leq g(b)-g(0)\}$$

et

$$D = \{(x,t) \mid x \in H, 0 \leq t \leq g(b)-g(0)\}$$

On a clairement E qui est anti-normal dans$[0,b] \times [0, g(b)-g(0)]$. De même puisque $t+g(x)$ est croissante sur $[0,b] \times [0, g(b)-g(0)]$, D est un ensemble normal. Et enfin la fonction

$$F(x,t) = f(x)+t$$

est croissante sur $[0,b] \times [0, g(b)-g(0)]$.

6.3 Approche séparation-évaluation pour la résolution d'un problème quadratique

On considère le problème suivant

$$(QP) \quad \max\{\frac{1}{2}\langle x, Qx\rangle + \langle c, x\rangle \mid Ax \leq d, x \geq 0\}$$

où le polyèdre $Ax \leq d$ est contenu dans le rectangle $[0,b] \subset \mathbb{R}^n_+$

La réduction au problème d'optimisation monotone que nous allons entreprendre dans la section qui suit est différente de celle présentée dans [144] où le problème (QP) est reformulé comme problème multiplicatif généralisé, et résolu comme problème d'optimisation monotone à l variables où l est le rang de la forme quadratique.

6.3.1 Réduction à un problème d'optimisation monotone

On définit $x^+ = \max(x,0)$ et $x^- = \max(-x,0)$. On pose

$$Q = Q^+ - Q^- \text{ et } c = c^+ - c^- \tag{6.7}$$

et

$$\frac{1}{2}\langle x, Qx\rangle + \langle c, x\rangle = f_1(x) - f_2(x)$$

avec

$$f_1(x) = \frac{1}{2}\langle x, Q^+x\rangle + \langle c^+, x\rangle \tag{6.8}$$

et

$$f_2(x) = \frac{1}{2}\langle x, Q^- x\rangle + \langle c^-, x\rangle. \tag{6.9}$$

On note que f_1 et f_2 sont des fonctions croissantes et positives dans $\mathbb{R}^n_+$. Comme f_2 est croissante, $f_2(x) \leq f(b) =: \gamma \ \forall x \in [0, b]$. On pose $t = \gamma - f_2(x)$. Alors $0 \leq t \leq \gamma \ \forall x \in [0, b]$. et notre problème s'écrit,

$$\max\{f_1(x) + t - \gamma \mid Ax \leq d, 0 \leq x \leq b, f_2(x) + t \leq \gamma, 0 \leq t \leq \gamma\}$$

On adopte la notation $x_{n+1} = t$ et $b_{n+1} = \gamma$ on a le problème suivant dans $\mathbb{R}^{n+1}_+$

$$\max\{F(x) \mid Ax \leq d, 0 \leq x \leq b, g(x) \leq 1\}.$$

Et finalement on obtient la forme canonique

$$\max\{F(x) \mid x \in G \cap H\}$$

avec $G = \{x \in \mathbb{R}^{n+1}_+ \mid g(x) \leq 1, x \leq y, Ay \leq d, 0 \leq y \leq b\}$ qui est un ensemble normal et $H = \mathbb{R}^{n+1}_+$ qui est à l'évidence anti-normal.

Proposition 6.3.1 *Si $\bar{z} \in [0, b]\backslash G$, on a $\pi(\bar{z}) = \mu\bar{z}$, avec*

$$\mu = \max\{\alpha > 0 \mid g(\alpha\bar{z}) \leq 1, \alpha\bar{z} \leq y, Ay \leq d, 0 \leq y \leq b\}$$

Il est clair que $\mu = min(\mu_1, \mu_2)$ avec μ_1 solution de l'équation du second dégré $g(\alpha\bar{z}) = 1$ et μ_2 valeur optimale du problème linéaire

$$\max\{\alpha \mid \alpha\bar{z} \leq y, Ay \leq d, 0 \leq y \leq b\}$$

6.3.2 La procédure d'évaluation

Nous donnons en entrée à la sous-routine : $R = [0, q] \subset [0, b]$, γ la meilleure valeur de la fonction objectif actuelle ; la tolérance $\epsilon > 0$ nous donnons également le nombre maximum d'itérations à exécuter dans la procédure, on le notera *nbiter*. En sortie de la procédure : $\beta(R)$ la borne supérieure de $\max\{f(x) \mid x \in R \cap G \cap H\}$, puis $z(R)$ tel que $\beta(R) = f(z(R))$ si $\beta(R) \geq \gamma + \epsilon$ et $x(R)$ la projection de $z(R)$ sur G.

Algorithme 6.3.1 – *Si $q \in G$ on sort de la procédure avec $\beta(R) = f(q)$ et γ est mis à jour si nécessaire. Sinon on pose $\tau = 1$(itérations internes).*

1. *Etape 1. Calculer x^τ la projection de z^τ sur G. Calculer*

$$z^{\tau,i} = z^\tau - (z^\tau_i - x^\tau_i)e^i, i = 1, 2, \cdots, n$$

et poser $T_{\tau+1} = (W_\tau\backslash\{z^\tau\}) \cup \{z^{\tau,1}, \cdots, z^{\tau,n}\}$.
Si $x^\tau \in H$(i.e x^τ réalisable) alors $\gamma \leftarrow \max(\gamma, f(x^\tau))$.

2. *Etape 2.* $W_{\tau+1}$ *est l'ensemble issu de* $T_{\tau+1}$ *après avoir retiré tous les éléments impropres. et tous les* $z \in T_{\tau+1}$ *tels que :* $f(z) \leq \gamma + \epsilon$. *Si* $W_{\tau+1} = \emptyset$, *alors on prend* $\beta(R) = \gamma + \epsilon$. *Sinon calculer*

$$z^{\tau+1} \in \arg\max\{f(z) \mid z \in \tilde{W_{\tau+1}}\}.$$

Si $\nu = nbiter$, *alors Stop, avec* $\beta(R) = f(z^{\tau+1})$, $z(R) = z^{\tau+1}$, $x(R) = x^{\tau+1}$.
Sinon $\tau \leftarrow \tau + 1$ *et retourner en Etape1.*

6.3.3 L'algorithme de séparation-évaluation

Initialisation $\bar{x}^o$ meilleure solution réalisable actuelle.
$\mathcal{P}_1 = \mathcal{S}_1 = \{R_o = [0, b]\}$
k = 1 ;

– Etape 1(Evaluation)
Pour tout $R = [0, q] \in \mathcal{P}_k$ calculer $\beta(R)$ et $x(R)$ grâce à la procédure d'évaluation. En appliquant cette procédure à tous les éléments de $\mathcal{P}_k$ mettre à jour γ et la valeur de f en la meilleure solution courante grâce à la procédure DCA.
– Etape 2(Elagage)
Effacer tous les éléments de $\mathcal{S}_k$ tels que

$$\beta(R) \leq \gamma + \epsilon$$

On note $\mathcal{R}_k$ l'ensemble des éléments de $\mathcal{S}_k$ restant.
– Etape 3(Test d'arrêt)
Si $\mathcal{S}_k = \emptyset$, alors Stop : $\bar{x}^k$ meilleure solution actuelle est solution $\epsilon - optimale$ du problème.
– Etape 4(Séparation)
Sélectionner $R_k \in \arg\max\{\beta(R) \mid R \in \mathcal{R}_k\}$, puis calculer

$$j_k \in \arg\max\{x_i(R) - z_i(R) \mid i = 1, 2, \cdots, n\}$$

où $x_i(R)$ et $z_i(R)$ sont fournis par la procédure d'évaluation.
Subdiviser R_k en deux sous-rectangles $R_{k_1} = [0, q_1]$ et $R_{k_2} = [0, q_2]$ tels que les composantes de q_1(resp. q_2) soient égales à celles de $z(R_k)$ (resp. $x(R_k)$) sauf la j_k qui est égale à celle de $x(R_k)$(resp. $z(R_k)$).
On note $\mathcal{P}_{k+1}$ la collection de ces deux sous-hyperrectangles de R_k
– Etape 5 $\mathcal{S}_{k+1} = (\mathcal{R}_k \backslash \{R_k\}) \cup \mathcal{P}_{k+1}$.$k \leftarrow k + 1$ retourner en Etape 1.

Notons que la subdivision adoptée couvre tout le domaine et présente l'avantage de ne pas prendre en compte $K_{x(R_k)}$.

6.4 Résultats numériques

Les éléments des matrices Q, A et des vecteurs c, b sont générés avec leur signe de sorte que l'ensemble d'admissibilé soit non vide. La matrice symétrique Q est construite en suivant la

méthodologie dans [15]. Plus précisément la matrice Q est construite de manière suivante : on pose $Q = PDP^T$ où P est une matrice orthogonale[1] et D une matrice diagonale. La matrice orthogonale P est de la forme $Q^1Q^2Q^3$ où

$$Q^j = I - 2\frac{w^j {w^j}^T}{\| w^j \|^2},\ j = 1, 2, 3.$$

Et les composantes du vecteur w^j sont choisies aléatoirement dans $(-1, 1)$, les éléments diagonaux de la matrice D sont choisis aléatoirement dans $(-5, 5)$.
Le point initial de l'algorithme est pris comme étant,

$$x_i^\circ = 0,\ i = 1, 2, \cdots, n.$$

Le test d'arrêt pour DCA est $err \leq 10^{-6}$ où

$$err = \| x^{k+1} - x^k \| / (1 + \| x_k \|).$$

Dans le Tableau 6.1, LB_o et UB_o représentent respectivement la borne inférieure et la borne supérieure au début de l'optimisation et LB_f et UB_f représentent respectivement la borne inférieure finale et la borne supérieure à la fin de l'optimisation. *it* représente le nombre d'itérations et *rect* le nombre de rectangles générés pendant le processus d'optimisation.
On observe le nombre important de rectangles générés pendant le processus d'optimisation et le nombre important d'itérations. Ces deux points sont les principales limites à l'utilisation de la méthode pour des problèmes de grandes dimensions.

[1] Une matrice P est dite orthogonale si $P^TP = I$, où I est la matrice identité.

dim		BB							BB-DCA						
m	n	LB_0	UB_0	LB_f	UB_f	it	$temps$	$rect$	LB_0	UB_0	LB_f	UB_f	it	$temps$	$rect$
11	10	-10.18	434	75.81	79.59	4728	2639	1935	76.17	434	76.17	79.97	4394	2468	1742
9	6	-1e-3	552	115.8	121.5	386	150	105	117.9	552	117.9	123.8	311	132	93
8	8	1.026	113.3	33.97	35.67	1326	469	839	34.13	113.3	34.14	35.84	1202	427	736
7	7	-8.417	109	21.3	22.24	525	161	152	17.36	109	21.43	22.5	476	146	157
6	6	3.56	111	43.78	45.97	307	86	193	43.86	111	43.86	46.04	300	82	187

TAB. 6.1 – Résultats numériques

Chapitre 7

Une procédure de redémarrage de DCA pour la résolution d'un problème de maximisation d'une fonction quadratique convexe sous des contraintes linéaires

Dans ce chapitre nous nous intéressons à la maximisation d'une fonction quadratique convexe sous des contraintes linéaires. Basée sur les conditions d'optimalité locales et globales nous proposons une méthode de redémarrage de DCA pour résoudre globalement sous certaines conditions le problème en présence.
La méthode proposée n'utilise que des sous-problèmes linéaires et génère une suite de maxima locaux ou points stationnaires. L'algorithme converge vers une ε-solution.
Une combinaison de la procédure de redémarrage et de la technique de séparation-évaluation est également présentée. Des tests numériques sur des problèmes de grandes dimensions sont reportés.

7.1 Introduction

Considérons le problème d'optimisation suivant,

$$(\mathcal{P}) \quad \max\{f(x) : x \in D\}, \tag{7.1}$$

où $f(x) = \frac{1}{2}x^T Cx + d^T x$, C est une matrice $n \times n$, supposée symétrique et définie positive, $d \in \mathbb{R}^n$, $D = \{x : Ax \leq b\}$ borné, A une matrice $m \times n$ et $b \in \mathbb{R}^m$.
Le problème $(\mathcal{P})$ est un problème multi-extrêma. Du point de vue de la complexité $(\mathcal{P})$ appartient à la classe des problèmes NP-difficile.
Les conditions d'optimalité globale obtenues par Strekalovsky [47, 48] pour le problème $(\mathcal{P})$ peuvent se formuler comme suit

Théorème 7.1.1 *[53]*
Un point $z \in D$ vérifiant $\nabla f(z) \neq 0$ est solution globale du problème $(\mathcal{P})$ si et seulement si

$$\langle \nabla f(y), x - y \rangle \leq 0, \text{ pour tout } y \in E_{f(z)}(f) \text{ et } x \in D, \tag{7.2}$$

où $E_{f(z)}(f) = \{y \in I\!R^n : f(y) = f(z)\}$ est l'ensemble de niveau de f au point z.

Lorsque D a une structure particulière telle qu'un rectangle, un simplex, ou une boule, une procédure basée sur la notion d'ensemble de résolution et le Théorème 7.1.1 est présentée dans [49] pour résoudre globalement $(\mathcal{P})$.
Comme on le voit, pour vérifier la condition (7.2) du Théorème 7.1.1 on doit résoudre

$$\max\{\langle \nabla f(y), x - y \rangle : Ax \leq b\}$$

pour tout $y \in E_{f(z)}(f)$. Résoudre ce problème est une tâche très difficile. D'où la nécessité de trouver un ensemble approximant l'ensemble de niveau $E_{f(z)}(f)$, et ainsi vérifier la condition sur un nombre fini de points. Le Lemme 2.1 dans [53] stipule que la détermination d'un tel ensemble approximant $E_{f(z)}(f)$ est numériquement possible.

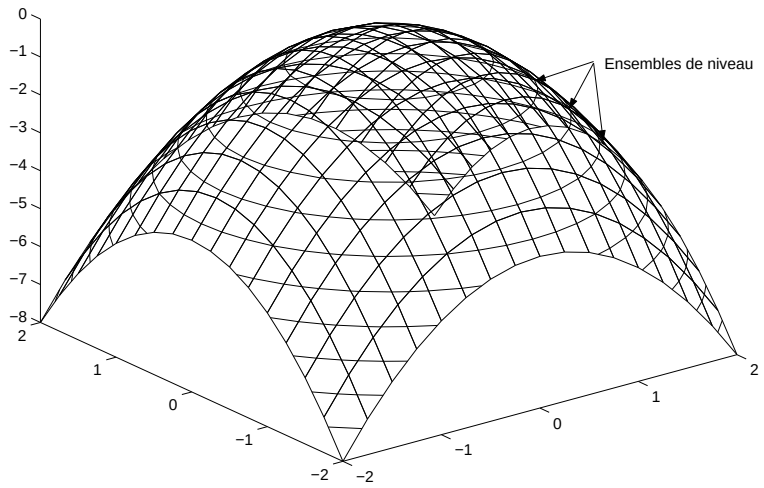


FIG. 7.1 – Ensembles de niveau d'une fonction concave.

7.2 Techniques d'approximation de l'ensemble de niveau

Dans notre approche nous partons de l'approximation à $2n$ points de l'ensemble de niveau $E_{f(z)}(f)$ proposée dans [53].

$$A_z^{2n} = \{y^1, y^2, \cdots, y^{2n} \in \mathbb{R}^n : y^i = x^\star + \alpha_i l^i,\ y^{n+i} = x^\star - \alpha_i l^i,\ i = 1, 2, \cdots, n\} \qquad (7.3)$$

avec $f(x^\star) = \min\{f(x) : x \in \mathbb{R}^n\}$ et $\alpha_i = \frac{\sqrt{2}}{\|l^i\|}\sqrt{\frac{f(z)-f(x^\star)}{\lambda^i}}$, où $l^i,\ \lambda^i,\ i = 1, 2, \cdots, n$ sont respectivement les vecteurs et les valeurs propres de la matrice C.
Une question légitime mais non explorée par les auteurs dans [53] est celle de savoir de combien cette approximation est proche de $E_{f(z)}(f)$. En d'autres termes à quelle distance ce trouve l'ensemble A^{2n} ?
Pour répondre à cette question nous devons définir la distance entre l'ensemble discret A^{2n} et l'ensemble continu $E_{f(z)}(f)$.

Définition 7.2.1 *soit A_i et A_{i+1} deux points de A^{2n} on définit la distance entre le segment $[A_i, A_{i+1}]$ et l'ensemble $E_{f(z)}(f)$, comme étant la distance entre le point M_i milieu du segment $[A_i, A_{i+1}]$ et l'ensemble $E_{f(z)}(f)$. Notons $d(M_i, E_{f(z)}(f))$ cette distance.*
La distance entre l'ensemble A^{2n} et $E_{f(z)}(f)$ est alors définie par,

$$(Dist) \quad d(A^{2n}, E_{f(z)}(f)) = \max_{i=1,2,\cdots,2n} d(M_i, E_{f(z)}(f)).$$

Un point important est à préciser dans cette définition. Comment définit-on le prédécesseur ou le suivant d'un point de A^{2n} dans $\mathbb{R}^n$ quand $n \geq 2$? Notre choix fut de ranger les points par l'ordre des valeurs algébriques des α_i intervenant dans la définition des points y_i de l'ensemble A^{2n} tout en faisant l'hypothèse qu'elles sont toutes distinctes.
L'ordre des éléments définis, on remarque que le problème $(Dist)$ a un intérêt double.

- Il nous fournit une estimation de la distance entre A^{2n} et $E_{f(z)}(f)$.
- Il nous fournit $2n$ points suplémentaires, car calculer $d(M_i, E_{f(z)}(f))$ revient à déterminer la projection de M_i sur $E_{f(z)}(f)$. Le problème $(Dist)$ résolu, les $2n$ points obtenus associés aux $2n$ points de l'ensemble A^{2n} nous fournissent un nouvel ensemble avec plus de points qui approximent mieux que A^{2n} l'ensemble $E_{f(z)}(f)$.

Cette idée est généralisée dans la procédure suivante

Algorithme 7.2.1 *Approximation de l'ensemble de niveau*
Initialisation : L'approximation A^{2n} de $E_{f(z)}(f)$ donné, on définit l'ensemble $\mathcal{B}_z = A^{2n}$.
Tant que Etiquette $\leftarrow$ faux faire
$\mathcal{F} = \varnothing$

1. *Pour $i = 1, 2, \cdots, |\mathcal{B}_z| - 1$ faire*
   (a) *Calculer M_i milieu du segment $[A_i, A_{i+1}]$ pour $A_i,\ A_{i+1} \in \mathcal{B}_z$.*
   (b) *Calculer la projection*

$$(\mathcal{D}) \quad P_i = Proj_{E_{f(z)}(f)}(M_i),$$

   puis poser $\mathcal{F} = \mathcal{F} \cup P_i$.
2. *Fin Pour.*

3. $\mathcal{B}_z = \mathcal{B}_z \cup \mathcal{F}$.

Fin Tant que.

A la fin de cet algorithme, $\mathcal{B}_z$ est le nouvel ensemble approximant $E_{f(z)}(f)$.
Deux points sont à préciser dans cet algorithme, le test d'arrêt et la résolution du problème $(\mathcal{D})$.
Pour le premier point deux tests d'arrêts furent examinés :

1. Considérer comme test d'arrêt la vérification du test
$$d(\mathcal{B}_z, E_{f(z)}(f)) \leq \varepsilon, \tag{7.4}$$
où ε est un réel positif.
2. Considérer comme test d'arrêt
$$|\mathcal{B}_z| \geq dim, \tag{7.5}$$
où dim est la dimension maximale prédéfinie de l'ensemble approximant $E_{f(z)}(f)$. Notons que $dim \geq 4n$.

Dans nos tests numériques nous avons considéré le second test d'arrêt car celui-ci nous permet de gérer la mémoire à alouer pour déterminer l'ensemble approximant. En pratique nous avons pris $dim = 8n$.
Pour le second point résoudre $(\mathcal{D})$ revient à résoudre un problème de la forme

$$(CQP) \begin{cases} \min\limits_{x \in \mathbb{R}^n} \frac{1}{2} \parallel x - y \parallel^2 \\ \quad \frac{1}{2} x^T C x + d^T x = \quad r, \end{cases}$$

Ce problème est non convexe malgré la définie positivité de la matrice C.

7.2.1 Solution du problème (CQP)

La première étape pour la résolution de (CQP) est de transformer ce problème pour l'écrire sous une forme plus standard d'un problème de minimisation d'une fonction quadratique convexe sous la contrainte que le point soit dans une boule, puis appliquer DCA pour résoudre globalement le problème transformé.

On définit la factorisation de Cholesky

$$C = R^T R$$

puis on adopte le changement de variables,

$$y = R(x - x^\star)$$

où $x^\star$ est le centre de la forme quadratique convexe définie par $\frac{1}{2} x^T C x + d^T x$. c'est-à-dire $x^\star = -C^{-1} d$.
On peut donc réécrire le problème (CQP) sous la forme

$$(BEQP) \begin{cases} \min \frac{1}{2} y^T Q y + q^T y \\ \quad \parallel y \parallel = s, \end{cases}$$

où

$$\begin{array}{rcl} Q & = & R^{-T}R \\ q & = & R^{-1}(x^\star + d), \\ s & = & \sqrt{2r^2 - d^T x^\star}. \end{array}$$

Ce problème reste non convexe cependant il est équivalent [104] au problème

$$(BIQP)\left\{ \begin{array}{c} \min \frac{1}{2} y^T (Q+\gamma I) y + q^T y \\ \| y \| \leq s, \end{array} \right.$$

où γ est choisi tel que la matrice $Q + \gamma I$ n'est pas sémi-définie positive.

Algorithme 7.2.2 *Solution du problème (CQP)*
On définit $y^\star = -Q^{-1}q$.

1. *Si* $\| y^\star \| < s$ *appliquer l'algorithme GDCA [104] au problème (BIQP).*
2. *Sinon appliquer l'algorithme GDCA [104] au problème*

$$(BINQP)\left\{ \begin{array}{c} \min \frac{1}{2} y^T Q y + q^T y \\ \| y \| \leq s, \end{array} \right.$$

   car dans ce cas les problèmes (BEQP) et (BINQP) sont équivalents.
3. *Soit* $\tilde{y}$ *la solution obtenue,* $\tilde{x} = R^{-1}\tilde{y} + x^\star$ *est solution du problème (CQP).*

7.3 Procédure de redémarrage de DCA

Avant de présenter la procédure, nous énoncerons deux propositions. Tout d'abord on définit l'ensemble

$$\mathcal{M}(z, \gamma, l) = \{x \in \mathbb{R}^n : \langle \nabla f(y^i), x \rangle \leq \gamma \text{ pour tout } y^i \in \mathcal{B}_z,\ i = 1, 2, \cdots, l\}. \tag{7.6}$$

où $\mathcal{B}_z$ est l'ensemble approximant $E_{f(z)}(f)$ obtenu en utilisant Algorithme 7.2.1, et $l = |\mathcal{B}_z|$. On définit également

$$\theta_l = \max_{i=1,2,\cdots,l} \langle \nabla f(y^i), u^i - y^i \rangle \tag{7.7}$$

où les u^i, $i = 1, 2, \cdots, l$ sont les solutions des problèmes de programmation linéaire

$$\max_{x \in D} \langle \nabla f(y^i), x \rangle.$$

On a la proposition fondamentale suivante,

Proposition 7.3.1 *s'il existe un* $y^i \in \mathcal{B}_z$ *pour* $z \in D$ *tel que* $\theta_l = \langle \nabla f(y^i), u^i - y^i \rangle > 0$ *alors*

$$f(u^i) > f(z).$$

Preuve : Par la définition de u^i, on a

$$\max_{x\in D}\langle \nabla f(y^i), x - y^i\rangle \;=\; \langle \nabla f(y^i), u^i - y^i\rangle.$$

Puisque f est convexe,

$$f(u) - f(v) \geq \langle \nabla f(v), u - v\rangle \textit{ pour tout } u,\ v \textit{ dans } \mathbb{R}^n.$$

Alors l'hypothèse de la proposition implique,

$$f(u^i) - f(z) = f(u^i) - f(y^i) \geq \langle \nabla f(y^i), u^i - y^i\rangle > 0.$$

□

Cette proposition nous fournit donc un nouveau point initial u^i à même d'améliorer la valeur de l'objectif trouvé par DCA.

Proposition 7.3.2 *Si* $\theta_l \leq 0$, *alors* $D \subset \mathcal{M}(z, \gamma, l)$ *avec* $\gamma = 2(f(z) - t)$, *pour* $z \in D$ *et* $t = \frac{1}{2}\min_{1,2,\cdots,l}\langle d, y^i\rangle$.

Preuve : De l'inégalité $\theta_l \leq 0$ on a,

$$\langle \nabla f(y^i), u^i - y^i\rangle \leq 0 \textit{ pour } i = 1, 2, \cdots, n.$$

Donc la série d'inégalités,

$$\langle \nabla f(y^i), x\rangle \leq \langle \nabla f(y^i), u^i\rangle \leq \langle \nabla f(y^i), y\rangle.$$

Et,

$$\begin{array}{rcl} \langle \nabla f(y^i), y\rangle & = & \langle \nabla C y^i + d, y^i\rangle \\ & = & 2(f(z) - \frac{1}{2}\langle d, y^i\rangle) \\ & \leq & 2(f(z) - \frac{1}{2}\min_{1,2,\cdots,l}\langle d, y^i\rangle). \end{array}$$

□

Ainsi si l'on définit

$$\begin{array}{ll} \alpha & = \max\{f(x) : x \in D\} \\ \alpha' & = \max\{f(x) : x \in \mathcal{M}(z, \gamma, l)\}. \end{array}$$

Sous les hypothèses de la Proposition 7.3.2, on aura

$$\alpha \leq \alpha'.$$

Cette inégalité nous donne une borne supérieure sur α, celle-ci pouvant être grossière. Seulement utiliser α' comme borne supérieure pour α dans notre approche est inappliquable. En effet contrairement à l'approche dans [53] où le calcul de α' est explicite, dans notre cas il nous faudrait résoudre à nouveau un problème d'optimisation globale équivalent à celui qui nous intéresse pour déterminer α'.

Algorithme 7.3.1 *RDCA1 : Première procédure de redémarrage de DCA.*

Entrée : *Une fonction quadratique convexe f un polyèdre convexe D, et le l pour le nombre maximal de points prédéfinis pour approximer l'ensemble de niveau.*
Sortie : *Une approximation du maximum global de f sur D.*
Etape 1. *Choisir un point $x^\circ \in D$ et poser $k = 0$*
Etape 2. *Utiliser DCA pour trouver un maximum local $z^k \in D$ en partant du point x^k.*
Step 3. *Construire l'ensemble $A^{2n}_{z^k}$ au point z^k en utilisant (7.3).*
Step 4. *Construire l'ensemble $\mathcal{B}_{z^k}$ au point z^k en utilisant Algorithme 7.2.1.*
Step 5. *Pour tout $y^i \in \mathcal{B}_{z^k}$, $i = 1, 2, \cdots, |\mathcal{B}_{z^k}|$, résoudre le problème*

$$\max_{x\in D}\langle \nabla f(y^i), x\rangle.$$

Soient u^i, $i = 1, 2, \cdots, |\mathcal{B}_{z^k}|$, les solutions.
Etape 6. *Trouver $i_{\max} \in \{1, 2, \cdots, |\mathcal{B}_{z^k}|\}$ tel que*

$$\theta^k_{|\mathcal{B}_{z_k}|} = \langle \nabla f(y^{i_{\max}}), u^{i_{\max}} - y^{i_{\max}}\rangle = \max_{i=1,2,\cdots,|\mathcal{B}_{z^k}|,} \langle \nabla f(y^i), u^i - y^i\rangle$$

Etape 7. *Si $\theta^k_{|\mathcal{B}_{z_k}|} \leq 0$, stop. Dans ce cas, z^k est une $\varepsilon = \varepsilon_k$ solution globale du problème $(\mathcal{P})$, où $\epsilon_k = \alpha'_k - f(z^k)$ et*

$$\alpha'_k = \max\{f(x) : x \in \mathcal{M}(z_k, \gamma_k, |\mathcal{B}_k|)\},$$

$\gamma_k = 2(f(z_k) - t)$, *où* $t = \frac{1}{2}\min_{1,2,\cdots,l}\langle d, y^i\rangle$.
Etape 8. *Poser $x^{k+1} = u^{i_{\max}}$, $k \leftarrow k + 1$ retourner à Etape 2.*

A l'étape 2 de l'algorithme nous devons utiliser DCA pour trouver une solution locale de $(\mathcal{P})$ partant d'un point x_k. Explicitons cette procédure.
La décomposition DC de $-f$ choisie pour l'algorithme DCA est la suivante $-f = g - h$ avec

$$\begin{array}{rcl} g(x) & = & -d^T x \\ h(x) & = & \frac{1}{2}x^T C x. \end{array}$$

Algorithme 7.3.2 *DCA1*
Tant que $err > \varepsilon$ faire
résoudre

$$\max_{x\in D}\langle y^k + d, x\rangle$$

où $y^k = Cx^k$.
Fin Tant que

Le test d'arrêt de l'algorithme sera précisé plus loin. On remarque qu'avec cette décomposition tous les sous-problèmes de l'algorithme RDCA se ramènent à la résolution de problèmes linéaires.

Théorème 7.3.1 *La suite $\{z^k\}_k$ générée par l'Algorithme 3.9.1 soit converge vers une solution du problème $(\mathcal{P})$ en un nombre fini d'étapes , soit trouve une ε solution approchée pour une certaine valeur de $\varepsilon > 0$ calculée à une étape de l'algorithme.*

Preuve : Deux cas sont à considérer. Le premier si $\theta^k_{|\mathcal{B}_{z_k}|} > 0$ pour tout k, alors la suite $\{z^k\}_k$ converge vers une solution globale de $(\mathcal{P})$ en un nombre fini d'étapes.
En effet soit $i_o \in \{1, 2, \cdots, n\}$ tel que,

$$\theta^k_{|\mathcal{B}_{z_k}|} = \langle \nabla f(y^{i_o}), u^{i_o} - y^{i_o} \rangle > 0.$$

De la convexité de f et du fait que $y^{i_o} \in E_{f(z^k)}(f)$, on peut écrire,

$$f(u^{i_o}) - f(z^k) = f(u^{i_o}) - f(y^{i_o}) \geq \langle \nabla f(y^{i_o}), u^{i_o} - y^{i_o} \rangle > 0.$$

Ainsi $f(u^{i_o}) > f(z^k)$, et u^{i_o} peut être considéré comme nouveau point initial pour redémarrer DCA. Ainsi, $f(z^{k+1}) \geq f(u^{i_o})$. On obtient donc la relation,

$$f(z^{k+1}) \geq f(z^k), \text{ pour tout } k = 0, 1, 2, \cdots .$$

Et puisque le nombre de maxima z^k est fini, cette suite croissante et majorée converge en un nombre fini d'étapes.
Le second cas, si $\theta^k_{|\mathcal{B}_{z_k}|} \leq 0$ pour un certain k, alors de la Proposition 7.3.2 on a $D \subset \mathcal{M}(z^k, \gamma_k, |\mathcal{B}_{z_k}|)$ pour $\gamma_k = 2(f(z_k) - t)$, où $t = \frac{1}{2} \min_{1,2,\cdots,l} \langle d, y^i \rangle$. Et donc,

$$f(z^k) \leq \max_{x \in D} f(x) \leq \max_{x \in D} f(x) \leq \max_{x \in \mathcal{M}(z^k, \gamma_k, |\mathcal{B}_{z_k}|)} f(x).$$

□

A l'étape 7 de l'algorithme, le calcul de $\alpha_k^{'}$ est en général impossible car revenant à résoudre globalement un problème équivalent à celui que l'on souhaite résoudre. La partie portant sur le calcul de $\alpha_k^{'}$ à l'étape 7 est donc présentée à titre indicatif.

La procédure ci-dessous est semblable à l'algorithme 3.9.1 ; la seule différence est que dans l'algorithme ci-dessous, l'étape 3 n'existe plus. Cet algorithme est identique à l'algorithme 1 dans [53], où DCA est pris comme algorithme local.

Algorithme 7.3.3 *RDCA2 : Deuxième procédure de redémarrage de DCA.*

Entrée : *Une fonction quadratique convexe f un polyèdre convexe D, et le l pour le nombre maximal de points prédéfinis pour approximer l'ensemble de niveau.*
Sortie : *Une approximation du maximum globale de f sur D.*
Etape 1. *Choisir un point $x^\circ \in D$ et poser $k = 0$*
Etape 2. *Utiliser DCA pour trouver un maximum local $z^k \in D$ en partant du point x^k.*
Step 3. *Construire l'ensemble $A^{2n}_{z^k}$ au point z^k en utilisant* (7.3).

Step 4. *Pour tout* $y^i \in A^{2n}_{z^k}$, $i = 1, 2, \cdots, 2n$, *résoudre le problème*

$$\max_{x \in D} \langle \nabla f(y^i), x \rangle.$$

Soient u^i, $i = 1, 2, \cdots, 2n$, *les solutions.*

Etape 5. *Trouver* $i_{\max} \in \{1, 2, \cdots, 2n, \}$ *tel que*

$$\theta^k_{2n} = \langle \nabla f(y^{i_{\max}}), u^{i_{\max}} - y^{i_{\max}} \rangle = \max_{i=1,2,\cdots,2n,} \langle \nabla f(y^i), u^i - y^i \rangle$$

Etape 6. *Si* $\theta^k_{2n} \leq 0$, *stop. Dans ce cas,* z^k *est une* $\varepsilon = \varepsilon_k$ *solution globale du problème* $(\mathcal{P})$, *où* $\epsilon_k = \alpha'_k - f(z^k)$ *et*

$$\alpha'_k = \max\{f(x) : x \in \mathcal{M}(z_k, \gamma_k, 2n)\},$$

$\gamma_k = 2(f(z_k) - t)$, *où* $t = \frac{1}{2} \min_{1,2,\cdots,l} \langle d, y^i \rangle$.

Etape 7. *Poser* $x^{k+1} = u^{i_{\max}}$, $k \leftarrow k + 1$ *retourner à Etape 2.*

A défaut d'avoir une bonne estimation de la borne supérieure par ce billet, nous avons entrepris de comparer RDCA à un algorithme classique de séparation-évaluation(Branch and Bound), dans lequel DCA est appelé pour améliorer la borne inférieure.

7.4 Algorithme de séparation-évaluation

On considère la décomposition en différence de fonctions concaves de la fonction objectif f suivante, $f(x) = g(x) - h(x)$, avec $g(x) = \frac{1}{2}x^T(Q - \rho I)x + q^T x$ et $h(x) = -\frac{1}{2}\rho \sum_{i=1}^{n} x_i^2$. où ρ est un réel tel que la matrice $Q - \rho I$ est définie négative.
La première étape de l'algorithme est la construction d'un rectangle $R_o \subset \mathbb{R}^n$ qui contient D. Pour cela on résout n programmes linéaires

$$\max\{x_i \ t.q \ x \in D\}, \ i = 1, 2, \cdots, n \tag{7.8}$$

pour obtenir les valeurs optimales $L^o_i, i = 1, 2, \cdots, n$ et

$$\min\{x_i \ t.q \ x \in D\}, \ i = 1, 2, \cdots, n \tag{7.9}$$

pour obtenir les valeurs optimales $l^o_i, i = 1, 2, \cdots, n$. Le rectangle peut alors s'écrire,

$$R^o = \{x : l^o_i \leq x_i \leq L^o_i, i = 1, 2, \cdots, n\}.$$

7.4.1 Borne supérieure

Soit $R = \{x : l_i \leq x_i \leq L_i, i = 1, 2, \cdots, n\}$ un rectangle dans R^n. Dans les méthodes de Branch and Bound, l'approche classique pour calculer la borne supérieure est d'utiliser un sur-estimateur concave de la fonction objectif. Quand elle est facile à calculer un sur-estimateur concave de choix pour une fonction est son enveloppe concave.

Définition 7.4.1 *L'enveloppe concave d'une fonction h sur un convexe D est une fonction h_{conc} telle que :*
(i) h_{conc} est concave sur D.
(ii) $h_{conc}(x) \geq h(x)\ \forall x \in D$.
(iii) Si $\tilde{h}$ est une fonction vérifiant (i) et (ii), alors $\tilde{h}(x) \geq h_{conc}(x)\ \forall x \in D$.

calculer l'enveloppe concave d'une fonction convexe sur un polyèdre. Puisque la fonction convexe $-h$ est séparable, son enveloppe concave sur le rectangle R est la somme de fonctions affines $h^{R_i}_{conc}(x_i)$ qui coïncident avec h_i aux extrémités du segment $[l_i, Li]$, c'est-à-dire la fonction

$$h^R_{conc}(x) = \sum_{i=1}^{n} h^{R_i}_{conc}(x_i)$$

où

$$h^{R_i}_{conc}(x_i) = -\frac{1}{2}\rho(l_i + L_i)x_i + \frac{1}{2}\rho l_i L_i.$$

Ainsi $g(x) + h^R_{conc}(x)$ est un sur-estimateur de f sur le domaine $\mathcal{K} = D \cap R$. La solution du problème concave

$$(UB) \quad \max\{g(x) + h^R_{conc}(x) : x \in \mathcal{K}\}$$

nous fournit un point x^R tel que,

$$g(x^R) + h^R_{conc}(x^R) \geq \max\{f(x) : x \in \mathcal{K}\} \geq f(x^R).$$

Donc $\beta(R) = g(x^R) + h^R_{conc}(x^R)$ est une borne supérieure pour f sur R et $f(x^R)$ est une borne inférieure de la valeur optimale $f_\star$.

7.4.2 Borne inférieure

La borne inférieure est calculée grâce à l'algorithme DCA en résolvant

$$(LB) \quad \max\{f(x) : x \in \mathcal{K}\}.$$

Algorithme 7.4.1 *Calcul de la borne inférieure(DCA2)*
Tant que $err > \varepsilon$ faire
résoudre

$$\max_{x \in \mathcal{K}} g(x) - \langle y^k + d, x\rangle$$

où $y^k = -\rho x^k$.
Fin Tant que

7.4.3 Algorithme de séparation-évaluation

Les lignes qui suivent décrivent la procédure séparation-évaluation pour résoudre $(\mathcal{P})$.

Algorithme 7.4.2 *Algorithme BBDCA*

Initialisation : On pose $R_0 = [l_0, L_0]$ *où* l_0 *et* L_0 *sont obtenus en résolvant les programmes linéaires* (4.2) *et* (4.1).
Calculer $h_{conc}^{R_0}$ *et résoudre le programme quadratique concave*

$$(R_0UB) \quad \max\{g(x) + h_{conc}^{R_0}(x) : x \in \mathcal{K}_0\}$$

pour obtenir la solution optimale x^{R_0} *et la valeur optimale correspondante* β_0. *Utiliser l'algorithme 7.4.1 pour résoudre le problème*

$$(R_0LB) \quad \max\{f(x) : x \in \mathcal{K}_0\}.$$

en prenant x_0^R *comme point initial. Soit* x_0 *la solution trouvée par DCA et* $\gamma_0 = f(x_0)$.
Si $\gamma - \beta_0 \leq \varepsilon$, *alors stop←vrai,* x_0 *est une* $\varepsilon-$*solution pour* $(\mathcal{P})$.
Sinon stop← faux finsi.
Poser $\mathcal{R} \leftarrow R_0$, $k \leftarrow 0$.
Tant que stop=faux faire,
- *sélectionner un rectangle* $R_k \in \mathcal{R}$ *tel que*

$$\beta_k = \beta(R_k) = \max\{\beta(R) : R \in \mathcal{R}\}.$$

- *Diviser* $R_k = \{x : l_i^k \leq x_i \leq L_i^k, i = 1, 2, \cdots, n\}$ *en deux sous- rectangles* R_{k1} *et* R_{k2} *en utilisant une subdivision rectangulaire normale choisie.*

- *Pour* R_{k1} *et* R_{k2} *calculer* $h_{conc}^{R_{ki}}$ *et résoudre*

$$(R_{ki}UB) \quad \max\{g(x) + h_{conc}^{R_{ki}}(x) : x \in \mathcal{K}_{ki}\}$$

  pour obtenir $x^{R_{ki}}$ *et* $\beta(R_{ki})$.
- *Pour* R_{k1} *et* R_{k2} *résoudre*

$$(R_{ki}LB) \quad \max\{f(x) : x \in \mathcal{K}_{ki}\}$$

  en utilisant l'algorithme 7.4.1 pour résoudre le problème en utilisant comme point initial $x^{R_{ki}}$ *soient* $x_{R_{ki}}^{DCA}$ *les solutions trouvées.*
- *Si l'on a* $f(x_{R_{ki}}^{DCA}) > \gamma_{k-1}$ *alors poser* $x_k = x_{R_{ki}}^{DCA}$ *et* $\gamma_k = f(x_k)$.
- *Poser* $\mathcal{R} \leftarrow (\mathcal{R} \backslash R_k) \cup \{R_{ki} : \beta(R_{ki}) > \gamma_k - \varepsilon, i = 1, 2\}$
- *Si* $\mathcal{R} = \varnothing$, *alors stop←vrai,* x_k *est une* $\varepsilon-$*solution, sinon* $k \leftarrow k + 1$ *finsi.*

Fin Tant que

Un point important de l'algorithme est la stratégie de subdivision des rectangles.

7.4.4 Stratégie de subdivision

Nous avons adopté la ω-subdivion. Pour un rectangle sélectionné R_k, $\beta(R_k) > f(x_k)$, donc, $h_{conc}(x^{R_k}) - h(x^{R_k}) > 0$. On détermine l'indice i_k tel que

$$i_k \in \arg\max_i \{h^i_{conc}(x_i^{R_k}) - h^i(x_i^{R_k})\}.$$

Puis on subdivise R_k en deux sous-rectangles,

$$R_{k1} = \{x \in R_k : x_{i_k} \leq x_{i_k}^{R_k}\},\ \ R_{k2} = \{x \in R_k : x_{i_k} \geq x_{i_k}^{R_k}\}.$$

7.5 Résultats numériques

Deux types de résultats numériques sont reportés. Dans le premier type(Type I) le but est exclusivement de voir combien de redémarrages sont nécessaires à DCA pour vérifier la condition de l'étape 6 de l'Algorithme 7.3.1. Dans le second type(Type II) la solution retournée par RDCA est comparée à la solution retournée par un Branch and Bound pour résoudre $(\mathcal{P})$.
Les procédures ont été écrites en C en double précision sous Windows XP avec processeur de 1.3 MHZ et une RAM de 512Mo. Les éléments propres ont été générés en utilisant MATLAB6.5. Les sous-problèmes linéaires et de minimisation quadratiques convexes sont résolus avec CPLEX8.0. Le temps d'exécution est reporté en secondes.

7.5.1 Résultats numériques pour le test de type I

Pour ce type, 70 problèmes générés aléatoirement ont été testés. Les éléments des matrices C, A et des vecteurs d, b sont générés avec leur signe de sorte que l'ensemble d'admissibilé est non vide. La matrice symétrique définie positive C est construite en suivant la méthodologie dans [15]. Plus précisément la matrice C est construite de manière suivante :
On pose $C = QDQ^T$ où Q est une matrice orthogonale et D une matrice diagonale. La matrice orthogonale Q est de la forme $Q^1Q^2Q^3$ où

$$Q^j = I - 2\frac{w^j w^{jT}}{\| w^j \|^2},\ j = 1, 2, 3.$$

Et les composantes du vecteur w^j sont choisies aléatoirement dans $(-1, 1)$, les éléments diagonaux de la matrice D sont choisies aléatoirement dans $(0, 5)$.
Le point initial de l'algorithme est pris comme étant,

$$x_i^\circ = 0,\ i = 1, 2, \cdots, n.$$

Le test d'arrêt pour DCA est $err \leq 10^{-6}$ où

$$err = \| x^{k+1} - x^k \| / (1 + \| x_k \|).$$

Dans la colonne nbDCA du Tableau 7.1, $p(k)$ signifie que sur p problèmes testés, DCA a redémarré k fois.
Deux enseignements sont à tirer de ces tests numériques :

n	m	max DCA iterations	nb DCA restarts	max CPU	min CPU	CPU average	average DCA	test global CPU average
10	5	5	4(1) 6(2)	1.14	0.12	0.83	0.1	0.73
20	20	3	9(1) 1(2)	1.38	1.36	1.367	0.1	1.267
50	30	3	9(1) 1(2)	12.90	6.89	8.21	0.0	8.20
100	100	3	10(1)	85.93	71.359	75.15	0.4	74.75
150	100	4	10(1)	229.47	207.06	213.05	0.9	230.15
200	200	4	10(1)	795.52	761.65	781.83	4.2	777.63
300	300	3	10(1)	3228.78	3049.37	3135.37	19.5	3115.87

TAB. 7.1 – Résultats numériques pour le test Type I avec RDCA2

– DCA redémarre au maximum 2 fois pour vérifier le test de l'Etape 7.
– Le temps pris par DCA est en moyenne largement inférieur au temps consacré à la construction de l'approximation de l'ensemble de niveau et à la vérification du test comme le montre la figure 7.2.

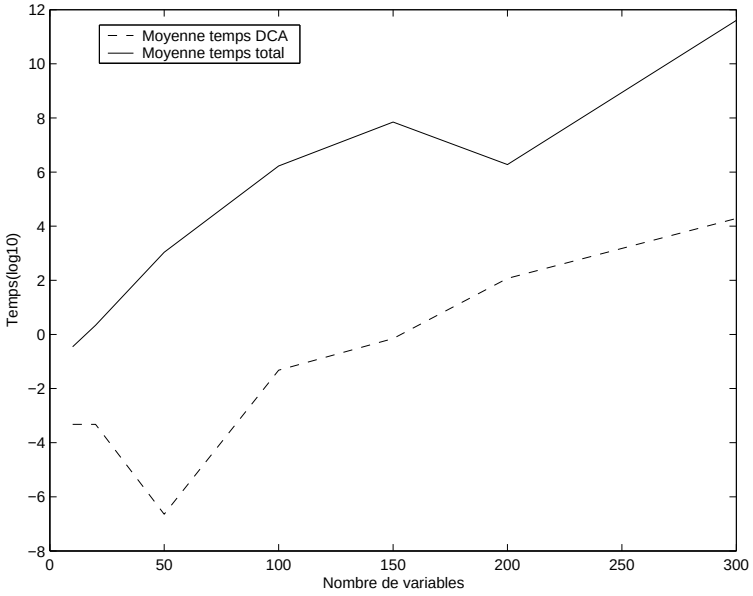


FIG. 7.2 – Temps moyen pris par DCA comparé au temps moyen général.

7.5.2 Résultats numériques pour le Type II

Pour ce second type de tests, les éléments du problème $(\mathcal{P})$ furent construits comme suit : On commence par générer aléatoirement une matrice $m \times n$ et on calcule sa décomposition en valeurs singulières pour obtenir les matrices U et V de dimensions respecives $n \times n$ et

$m \times m$. On pose

$$A = U \begin{pmatrix} \Sigma \\ \Omega \end{pmatrix} V^T,$$

où $\Sigma = diag(\sigma_i)$ avec les σ_i Choisis dans $[-1, 1]$ and $\Omega = diag(\omega_i)$ les ω_i choisis dans $[-1, 1]$. générer $x^\star$ tel que $x_i^\star \in [-10, 10]$.

$$b = -Ax^\star.$$

$$C = UDU^T$$

avec $D = diag(\delta_i)$ où $\delta_i \in [0, 10]$. On a pris λ_j dans $[0, 1]$ et

$$q = -Qx^\star + A\lambda.$$

Les tests ont été réalisés avec une précision de $10e^{-2}$ pour la procédure BBDCA. Les tests numériques ont été limités à 300 variables et 30 contraintes pour la procédure RDCA1 en raison du temps de calcul devenant très important et les limites de capacité mémoire.

Les résultats numériques sont reportés dans le Tableau 7.2. Dans ce tableau,
- $\geq$ signifie que le temps de calcul est supérieur à trois heures.
- $=$ signifie que le nombre d'itérations a été limité en raison du grand écart entre la borne inférieure et la borne supérieure.
- En gras RDCA1 et RDCA2 obtiennent une valeur de l'objectif supérieure où égale à la borne inférieure de BBDCA.

On observe au Tableau 7.2 que le temps de calcul moyen de RDCA1 est supérieur à celui de BBDCA, ceci est dû au nombre de problèmes linéaires qui doivent être résolus dans RDCA1. Pour 300 variables par exemple 2400 sous-problèmes de programmation linéaire doivent être résolus, ce nombre étant le nombre de points dans l'ensemble approximant l'ensemble de niveau. Par contre le temps de calcul de RDCA2 est bien inférieur au temps de calcul de RDCA1 et de BBDCA comme le montre la Figure 7.3.

Trois leçons sont à tirer du Tableau 7.2.

1. Tout d'abord ajouter des points à l'ensemble A_z^{2n} n'améliore pas la qualité de la solution retournée par RDCA1 comme l'attestent les résultats de l'algorithme RDCA2.
2. Les algorithmes RDCA1 et RDCA2 ne sont pas globaux, comme l'attestent leurs résultats comparés à ceux de BBDCA.
3. Le temps de calcul de RDCA2 est moindre que celui de BBDCA quand les dimensions du problème deviennent grandes, et le rapport temps d'exécution et qualité de la solution retournée est meilleur que celui de BBDCA dans ce cas.

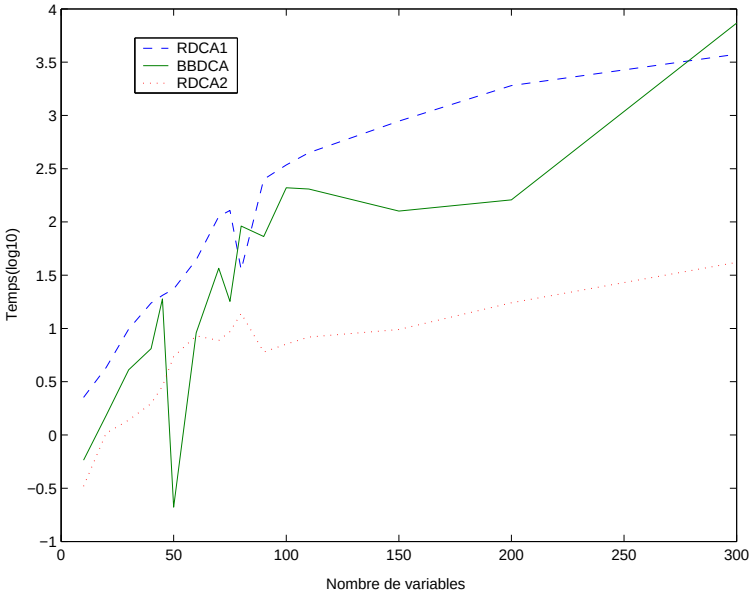


FIG. 7.3 – Temps de calcul en fonction de la dimension pour les algorithmes RDCA1, RDCA2 et BBDCA.

7.6 Combiné RDCA-BBDCA

Le dernier point mentionné ci-dessus nous a incité à développer un algorithme qui est une combinaison de RDCA2 et de BBDCA, dans lequel RDCA2 est appelé au tout début du processus d'optimisation pour fournir une bonne borne inférieure, et un bon point initial pour l'algorithme BBDCA.

Algorithme 7.6.1 *Combiné RDCA-BBDCA*

- *Lancer RDCA2 pour obtenir une borne inférieure LbInit de* $(\mathcal{P})$. *Soit* x_0 *le point trouvé par RDCA2.*
- *Lancer BBDCA en prennant* x_0 *comme premier meilleur point et poser* $\gamma_0 = f(x_0)$ *dans BBDCA.*

Pour ces tests numériques le nombre d'itérations à été limité à trois. Seuls les problèmes du Tableau 7.2 où RDCA2 trouvaient une valeur de l'objectif inférieure à la borne inférieure retournée par BBDCA ont été considérés. Les résultats sont reportés dans le Tableau 7.3. Les résultats numériques du Tableau 7.3 montrent que dès que la dimension du problème devient très grande la combinaison RDCA-BBDCA est meilleure en termes de rapport qualité de la solution et temps de calcul. Le temps de calcul de la combinaison RDCA-BBDCA devient moindre dès que la dimension du problème devient importante, comme le montre la Figure 7.4 Ceci confirme les observations de la fin du paragraphe précédent.

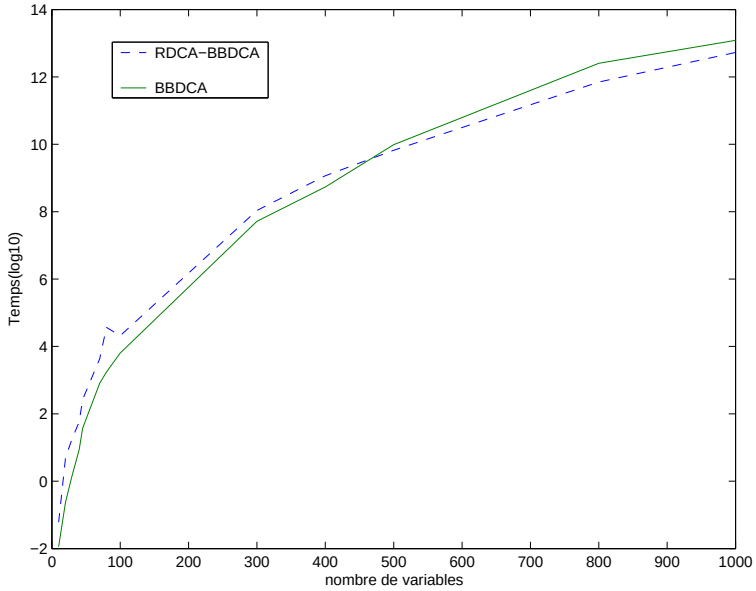


FIG. 7.4 – Temps de calcul en fonction de la dimension.

7.7 Conclusion

Une procédure de redémarrage de DCA a été présentée. Le but premier du travail était de développer une procédure de redémarrage de DCA. Le second but était de compléter le travail fait dans [53] sur l'approximation de l'ensemble de niveau, en proposant une procédure basée sur les techniques DC permettant non seulement de fournir de nouveaux points appartenant à l'ensemble de niveau mais aussi de fournir une idée sur la distance entre l'ensemble de niveau et l'ensemble qui l'approxime. La procédure de redémarrage semble efficace, mais la quête de solution globale par cette approche sans y apporter des modifications semble irréalisable en raison d'une part du nombre important de points qu'il faudrait calculer pour approximer au mieux l'ensemble de niveau et donc d'éventuels problèmes de capacité mémoire ; et d'autre part en raison du trop grand nombre de problèmes de programmation linéaire qu'il faudrait résoudre à chaque itération de l'algorithme. Une combinaison de la procédure de redémarrage de DCA et de l'algorithme de séparation-évaluation a été proposée. La combinaison des deux algorithmes donne de meilleures résultats en termes de rapport temps de calcul qualité de la solution que donne chacun des algorithmes pris individuellement. Ce travail est un premier pas vers des algorithmes basés sur les conditions d'optimalité globale et reste donc à ce titre un sujet de travail ouvert.

dim		RDCA1			RDCA2			BBDCA				
n	m	$NbRest$	$f(z)$	$temps$	$NbRest$	$f(z)$	$temps$	$iter$	$NbRest$	UB	LB	$temps$
10	1	2	$\mathbf{4.596e^5}$	5.015	2	$\mathbf{4.596e^5}$	0.430	2	1	$4.648e^5$	$\mathbf{4.596e^5}$	0.09
10	5	1	$3.142e^6$	2.254	1	$3.142e^6$	0.330	7	2	$3.404e^6$	$3.341e^6$	0.58
20	2	2	$\mathbf{6.777e^5}$	5.017	2	$\mathbf{6.777e^5}$	0.851	12	1	$6.870e^5$	$\mathbf{6.777e^5}$	0.77
20	10	2	$6.332e^5$	4.301	2	$6.332e^5$	1.041	17	1	$7.148e^5$	$7.071e^5$	1.52
30	3	2	$1.022e^6$	9.850	2	$1.022e^6$	1.382	22	1	$1.038e^6$	$1.026e^6$	4.09
40	4	2	$1.163e^6$	17.324	2	$1.163e^6$	1.973	23	1	$1.180e^6$	$1.168e^6$	6.46
45	22	1	$1.121e^6$	20.450	1	$1.121e^6$	2.874	27	1	$1.524e^6$	$1.475e^6$	18.95
50	5	1	$\mathbf{2.433e^6}$	23.423	1	$\mathbf{2.433e^6}$	5.458	1	1	$2.443e^6$	$\mathbf{2.433e^6}$	0.21
60	6	1	$\mathbf{2.598e^6}$	39.442	1	$\mathbf{2.598e^6}$	2.363	1	1	$2.611e^6$	$\mathbf{2.598e^6}$	0.28
60	30	1	$\mathbf{2.691e^6}$	44.143	1	$\mathbf{2.691e^6}$	8.542	30	1	$2.719e^6$	$\mathbf{2.691e^6}$	9.10
70	7	2	$2.566e^6$	112.525	2	$2.566e^6$	7.721	24	3	$2.668e^6$	$2.603e^6$	36.73
75	37	1	$\mathbf{3.555e^6}$	128.242	1	$\mathbf{3.555e^6}$	9.333	47	1	$3.591e^6$	$\mathbf{3.555e^6}$	17.97
80	8	1	$2.478e^6$	125.443	1	$2.478e^6$	4.947	38	1	$3.739e^6$	$2.787e^6$	96.28
80	40	2	$2.729e^6$	35.453	2	$2.729e^6$	13.870	42	2	$2.880e^6$	$2.833e^6$	91.36
90	9	2	$\mathbf{3.205e^6}$	251.060	2	$\mathbf{3.205e^6}$	6.000	50	1	$3.236e^6$	$\mathbf{3.205e^6}$	73.00
100	10	2	$3.148e^6$	343.125	2	$3.148e^6$	7.130	62	1	$3.214e^6$	$3.179e^6$	209.25
110	11	2	$\mathbf{3.471e^6}$	446.565	2	$\mathbf{3.471e^6}$	8.302	67	5	$3.590e^6$	$\mathbf{3.214e^6}$	203.96
150	15	1	$\mathbf{6.531e^6}$	884.505	1	$\mathbf{6.531e^6}$	9.774	73	1	$6.598e^6$	$\mathbf{6.531e^6}$	126.57
200	20	1	$\mathbf{8.540e^6}$	1907.453	1	$\mathbf{8.540e^6}$	17.425	9	2	$8.626e^6$	$\mathbf{8.540e^6}$	161.18
300	30	1	$8.803e^6$	3753.40	1	$8.803e^6$	41.730	151	4	$9.833e^6$	$9.732e^6$	7350.01
400	40	-	–	-	2	$1.285e^7$	126.632	131	8	$1.557e^7$	$1.311e^7$	$\geq$
500	50	-	–	-	2	$1.600e^7$	222.480	3	4	$2.492e^7$	$1.670e^6$	$\geq$
600	60	-	–	-	1	$\mathbf{2.197e^7}$	284.909	53	2	$2.825e^7$	$\mathbf{2.093e^7}$	$=$
700	70	-	–	-	1	$\mathbf{3.444e^7}$	489.533	8	1	$3.479e^7$	$\mathbf{3.444e^7}$	812.44
800	80	-	–	-	2	$2.619e^7$	994.179	6	6	$3.986e^7$	$2.669e^7$	$\geq$
900	90	-	–	-	1	$\mathbf{4.341e^7}$	1088.135	75	1	$4.385e^7$	$\mathbf{4.341e^7}$	11470.00
1000	100	-	–	-	2	$3.294e^7$	2007.427	3	4	$4.975e^7$	$3.403e^6$	$\geq$

TAB. 7.2 – Résultats numériques pour le test Type II.

dim		RDCA-BBDCA					BBDCA				
n	m	$NbRest$	$LbInit$	$LbFin$	Ub	$temps$	$NbRest$	$LbInit$	$LbFin$	Ub	$temps$
10	5	1	$3.086e^5$	$3.341e^5$	$3.798e^5$	0.43	1	$3.341e^5$	$3.341e^5$	$3.798e^5$	0.26
20	10	1	$6.332e^5$	$7.071e^5$	$9.393e^5$	1.62	1	$7.071e^5$	$7.071e^5$	$9.393e^5$	0.65
30	3	1	$1.022e^5$	$1.026e^5$	$1.412e^6$	2.42	1	$1.026e^5$	$1.026e^5$	$1.430e^6$	1.14
40	4	1	$1.163e^6$	$1.168e^6$	$1.566e^6$	3.39	1	$1.168e^6$	$1.168e^6$	$1.556e^6$	1.91
45	22	1	$1.421e^6$	$1.475e^6$	$2.201e^6$	5.36	2	$1.313e^6$	$1.475e^6$	$2.192e^6$	2.97
70	7	1	$2.502e^6$	$2.601e^6$	$3.433e^6$	12.42	2	$2.518e^6$	$2.535e^6$	$3.428e^6$	7.50
80	8	1	$2.478e^6$	$2.482e^6$	$3.821e^6$	13.97	2	$2.478e^6$	$2.833e^6$	$3.739e^6$	8.57
80	40	1	$2.760e^6$	$2.800e^6$	$3.948e^6$	23.65	2	$2.687e^6$	$2.833e^6$	$3.936e^6$	9.45
100	10	3	$3.154e^6$	$3.197e^6$	$4.921e^6$	19.95	2	$3.162e^6$	$3.179e^6$	$4.921e^6$	13.96
300	30	1	$8.858e^6$	$9.605e^6$	$1.491e^7$	262.40	2	$9.444e^6$	$9.628e^6$	$1.491e^7$	210.09
400	40	1	$1.285e^7$	$1.313e^7$	$1.990e^7$	536.48	2	$1.302e^7$	$1.310e^7$	$1.990e^7$	425.00
500	50	2	$1.600e^7$	$1.640e^7$	$2.488e^7$	906.48	3	$1.667e^7$	$1.674e^7$	$2.488e^7$	1015.59
800	80	2	$2.619e^7$	$2.659e^7$	$3.987e^7$	3682.18	4	$2.627e^7$	$2.669e^7$	$3.987e^7$	5419.80
1000	100	1	$3.294e^7$	$3.328e^7$	$4.985e^7$	6783.42	4	$3.308e^7$	$3.328e^7$	$4.985e^7$	8700.00

Tab. 7.3 – Comparaison de RDCA-BBDCA et de BBDCA.

Troisième partie

APPLICATIONS INDUSTRIELLES

Chapitre 8

Problème de mécanique de structure

Les systèmes mécaniques conçus que ce soit en construction mécanique, en génie civil ou en industrie chimique, sont généralement destinés à remplir leur fonction sans dommage notable durant le temps d'exploitation escompté, en étant soumis à un ensemble de sollicitations préalablement défini. Lors de sa conception, une structure doit remplir certains critères sur sa résistance et sa fiabilité vis-à-vis des charges qu'elle aura à supporter. Si l'ensemble des charges est connu avec exactitude, l'étude peut être abordée par la «méthode pas à pas» dans le cas contraire, l'étude est rendue possible par l'application de la «méthode directe». La méthode directe est composée de l'analyse limite et de l'adaptation ou *shakedown.*
La méthode directe conduit à la formulation d'un problème de programmation mathématique couplée à la méthode d'éléments finis et demande :

- la solution du problème de référence purement élastique,
- l'utilisation un processus d'optimisation, qui consiste à déterminer le facteur de charge contre la défaillance de la structure et permet de construire le champ de contraintes résiduelles.

Le problème traité est un problème non linéaire sous contraintes. Ce qui nécessite pour les applications industrielles un très grand nombre de variables d'optimisation et un temps de calcul élevé. Pour pouvoir résoudre ce problème d'optimisation à grande échelle avec un temps de calcul raisonnable, il est nécessaire d'utiliser un algorithme robuste. Les méthodes de points intérieurs ont montré leur efficacité pour le traitement des problèmes de grandes dimensions. Nous avons appliqué IPDCA présenté au Chapitre 3 au problème d'optimisation non linéaire en présence en utilisant les adaptations de l'Annexe A. Des résultats numériques encourangeants sont présentés à la fin de ce chapitre . Mais avant, nous présenterons les divers discrétisations qui conduisent au problème d'optimisation non linéaire en nous inspirant des présentations faites dans [3, 4, 5, 6].

8.1 Considérations générales et définitions

On considère le comportement d'une structure élastique parfaitement plastique $\mathcal{B}$ parfaitement élastique, de volume fini Ω à surface lisse $\partial\Omega$ composée de deux parties disjointes $\partial\Omega_p$ et $\partial\Omega_u$, où les conditions aux limites statiques et cinétiques sont prescrites ($\Omega = \partial\Omega_u \cup \partial\Omega_p$; $\partial\Omega_u \cap \partial\Omega_p = \varnothing$). La structure est sujette à des sollicitations externes quasi-statiques vari-

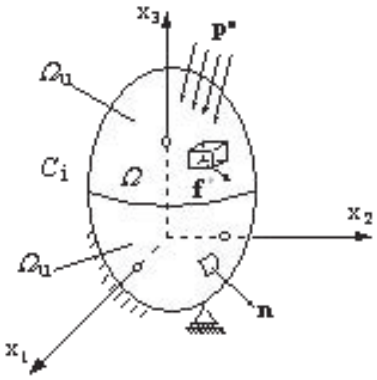


FIG. 8.1 – La structure $\mathcal{B}$.

ables $P(x,t) \in \mathcal{L}$ au temps t composées de forces de volume $f^\star(x,t)$ dans Ω, de tractions surfaciques $p^\star(x,t)$ sur $\partial\Omega_p$ et des déplacements $u^\star(x,t)$ sur $\partial\Omega_u$. $\mathcal{L}$ représente le domaine des charges.

8.2 Formulation continue de la méthode directe

Le théorie de la borne inférieure d'adaptation stipule qu'une structure $\mathcal{B}$ s'adapte sous sollicitations variables $P(x,t) \in \mathcal{L}$, s'il existe un facteur de sécurité $\alpha > 1$ et un champ de contraintes résiduelles indépendant du temps σ^r tel que sa superposition aux contraintes purement élastiques σ^E ne viole pas le domaine d'élasticité pour tout temps $t > 0$. Ceci peut se traduire par la contrainte

$$F(\alpha\sigma^E(x,t) + \sigma^r(x)) \quad \leq \quad 0, \quad \forall x \in \Omega. \tag{8.1}$$

Dans la littérature la contrainte (8.1) est convexe.
Le champ de contraintes purement élastiques pures satisfait le système d'équations

$$\begin{array}{rcccc} Div\ \sigma^E & = & -f & dans & \Omega \\ n.\sigma^E & = & p^\star & sur & \partial\Omega_p \\ u^E & = & u^\star & sur & \partial\Omega_u \end{array}$$

avec

$$\varepsilon^E = \frac{1}{2}(\nabla(u^E) + \nabla(u^E)^T). \tag{8.2}$$

Et le champ des contraintes résiduelles vérifie

$$\begin{array}{rcccc} Div\ \sigma^r & = & 0 & dans & \Omega \\ n.\sigma^r & = & 0 & sur & \partial\Omega_p \end{array}$$

où n est le vecteur normal à $\partial\Omega_p$.

8.3 Formulation discrète de la méthode directe

8.3.1 Discrétisation de contraintes élastiques

Pour calculer les contraintes purement élastiques σ^E, on utilise le principe du travail virtuel combiné à la discrétisation par éléments finis avec des fonctions de test pour les champs de déplacements. Alors les contraintes purement élastiques σ^E sont en équilibre avec les forces de volume $f^\star$ et les tractions surfaciques $p^\star$ si l'égalité suivante est vérifiée

$$\delta U_{int} = \delta U_{ext}$$

soit

$$\int_\Omega \{\delta\varepsilon^E(x)\}^T\{\delta\sigma^E(x)\}d\Omega = \int_{\partial\Omega p} \{\delta u^E\}\{p^\star\}dS + \int_\Omega \{\delta u^E\}^T\{f^\star\}d\Omega,$$

pour tout champ de déplacement virtuel δu^E et tout champ de déformation virtuelle $\delta\varepsilon^E$ vérifiant l'équation de compatibilité (8.2). Le champ de déplacement virtuel δu^E de chaque élément e est approximé grâce à la formule,

$$\{\delta u^E\} = \sum_{k=1}^{NKE} N_k u_k^e, \tag{8.3}$$

où N_k et u_k^e représente respectivement la k^{eme} matrice de forme et le k^{eme} vecteur de déplacement virtuel du k^{eme} nœud de l'élément e, respectivement. NKE représente le nombre total de nœuds de chaque élément. Le champ de déformation virtuelle $\delta\varepsilon^E(x)$ est obtenu par substitution de (8.3) dans (8.2) de sorte que l'on a,

$$\{\delta\varepsilon^E(x)\} = \sum_{k=1}^{NKE} B_k(x)\delta u_k^e, \tag{8.4}$$

où $[B]$ est la matrice de compatibilité dépendant des coordonnées. L'intégration de l'équation (8.4) doit être faite sur tous les points de Gauss NGE de l'élément considéré. Si i représente le i^{eme} point de Gauss, le vecteur de coordonnées correspondant se note x_i et,

$$\begin{aligned}
\int_\Omega \{\delta\varepsilon^E(x)\}^T\{\delta\sigma^E(x)\}d\Omega &= \{\delta u^E\}^T[\sum_{i=1}^{NGE} w_i|J|_i[B(x_i)]^T E[B(x_i)]]\{u^e\} &(8.5)\\
&= \{\delta u^E\}^T[K]\{u^e\} &(8.6)\\
&= \{\delta u^E\}^T F &(8.7)
\end{aligned}$$

où F est le vecteur de forces nodales, w_i le facteur de pondération, $|J|_i$ le déterminant de la matrice Jacobienne et $[K]$ la matrice de raideur.
Cette intégrale conduit pour le i^{eme} point de Gauss à,

$$\{\sigma^E(x_i)\} = [E][B(x_i)]\{u^e\}. \tag{8.8}$$

8.3.2 Discrétisation des champs de contraintes résiduelles

De façon analogue les champs de contraintes résiduelles peuvent être déterminé par l'équation

$$\int_{\Omega} \{\delta\varepsilon\}^T \{\delta\sigma^r\} d\Omega = 0. \tag{8.9}$$

En introduisant le vecteur du tenseur de déformation ε, la correspondante déformation virtuelle $\delta\varepsilon$ est donnée pour tout élément e par

$$\{\delta\varepsilon\} = \sum_{k=1}^{NKE} N_k \delta u_k^e, \tag{8.10}$$

En utilisant cette relation et en introduisant le vecteur inconnu de contraintes résiduelles σ_i^r à chaque point de Gauss i, on intègre numériquement la condition d'équilibre (8.9) en utilisant la technique de Gauss-Legendre. L'intégration doit être faite sur tous les NGE points de Gauss avec leur facteur de pondération w_i pour tout élément e, ce qui donne

$$\int_{\Omega} \{\delta\varepsilon\}^T \{\sigma^r\} d\Omega \;=\; \sum_{i=1}^{NGE} w_i |J|_i [\sum_{k=1}^{NKE} B_k \delta u_k^e] \sigma_i^r. \tag{8.11}$$

Une sommation sur toutes les contributions de tous les éléments et par la variation de tous les déplacements nodaux virtuels par rapport aux conditions aux limites, on obtient finalement le système linéaire

$$\int_{\Omega} \{\delta\varepsilon\}^T \{\sigma^r\} d\Omega \;=\; \sum_{i=1}^{NG} C_i \sigma_i^r = [C]\{\sigma^r\} = \{0\}, \tag{8.12}$$

où NG est le nombre de points total de Gauss de la structure, $[C]$ la matrice d'équilibre, définie à partir du systèmme discrétisé et des conditions aux limites et $\{\sigma^r\}$ le vecteur de contraintes résiduelles globales de la structure discrétisée.

8.3.3 Discrétisation par rapport au temps

Jusqu'ici aucune restriction n'a été faite sur le domaine des charges $\mathcal{L}$. On peut donc supposer $\mathcal{L}$ de forme quelconque. Cependant dans les cas pratiques, le nombre de charges indépendantes est limité, chacune variant entre des bornes bien définies. Si le nombre de telles charges est n, alors le domaine des charges peut être défini par un polyèdre de dimension n,

$$\mathcal{L} = \{P \mid P(x,t) = \sum_{j=1}^{n} \mu_j P_j(x),\ \mu_j \in [\mu_j^-, \mu_j^+]\} \tag{8.13}$$

où P est le vecteur de charges généralisé, μ_j des multiplicateurs scalaires ayant pour bornes inférieure et supérieure μ_j^- et μ_j^+ respectivement. P_j représente n charges fixes et indépendantes (forces de la structure, tractions surfaciques, changements de température,etc) correspondant

au sommet j. On note NV le nombre de sommets du polyèdre $\mathcal{L}$. De la convexité de la contrainte (8.1) et de la définition du champ des charges (8.13), on montre que la contrainte (8.1) est satisfaite à chaque instant t si,

$$F(\alpha\sigma_i^E(P_j)+\sigma_i^r) \;\leq\; 0,\;\; \forall j\in[1,NV] \;\; et \;\; \forall i\in[1,NG]. \tag{8.14}$$

Alors la formulation discrétisée du théorème d'adaptation pour déterminer le facteur de charge α est donnée par le problème d'optimisation

$$(\mathcal{M})\begin{cases} \max \alpha \\ tq \\ \sum\limits_{i=1}^{NG} C_i\sigma_i^r \;=\; \{0\} \\ F(\alpha\sigma_i^E(P_j)+\sigma_i^r) \;\leq\; 0, \\ j\in[1,NV],\;\; i\in[1,NG] \end{cases} \tag{8.15}$$

avec $NV=2^n$. Le nombre d'inconnues du problème d'optimisation (8.15) est $N=1+NG\times NS$ correspondant à α et $\{\sigma^r\}$. Le nombre de contraintes est $NV\times NG+NK$ où NS est la dimension du vecteur de contraintes en chaque point de Gauss et NK est le dégré de liberté des déplacements de la structure discrétisée.
Dans la section qui suit nous allons appliquer IPDCA au problème $(\mathcal{M})$.

8.4 Formulations équivalentes simples du programme convexe

8.4.1 Formulation équivalente primale

Après avoir précisé la fonction F dans le problème $(\mathcal{M})$, notre problème de mécanique de structure s'écrit :

$$(\mathcal{P})\begin{cases} \max\; \alpha \\ t.q \\ \sum\limits_{r=1}^{NG} C_rX_r=0 \\ (\sigma_{1r}-\sigma_{2r})^2+(\sigma_{2r}-\sigma_{3r})^2+(\sigma_{3r}-\sigma_{1r})^2+6\sigma_{4r}^2+6\sigma_{5r}^2+6\sigma_{6r}^2\leq 2a_r^2 \\ \sigma_r-\alpha\beta_r-X_r=0, \\ r=1,2,\cdots,NG. \end{cases}$$

Avec a, β_r donnés et $C_r\in\mathcal{M}(\mathbb{R})_{(3NK,6)}$, $\sigma_r\in\mathbb{R}^6$, $X_r=(X_{1r},\cdots,X_{6r})\in\mathbb{R}^6$ pour $r=1,2,\cdots,NG$.
Ce problème a $12NG+1$ variables et $3NK+7NG$ contraintes. Dans la pratique $NG>NK$ et NK est de l'ordre de plusieurs milliers. Ce qui conduit à un problème de grande dimension.

Le problème $(\mathcal{P})$ est un problème convexe et la reformulation que nous devons entreprendre doit conserver cette bonne propriété. De plus le problème reformulé doit avoir moins de contraintes et moins de variables que le problème d'origine donc plus facile à résoudre.
On définit l'application linéaire bijective

$$\begin{aligned} \mathcal{G} : \mathbb{R}^6 &\longrightarrow \mathbb{R}^6 \\ v &\longmapsto Tv \end{aligned}$$

Où $T \in \mathcal{M}(\mathbb{R})_{(6,6)}$ définie par :

$$T = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & & & \\ -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & & & \\ -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & & & \\ & & & \frac{1}{\sqrt{6}} & & \\ & & & & \frac{1}{\sqrt{6}} & \\ & & & & & \frac{1}{\sqrt{6}} \end{pmatrix}$$

La bijection inverse est définie par,

$$T^{-1} = \begin{pmatrix} 1 & -1 & & & & \\ & 1 & -1 & & & \\ 1 & & 1 & & & \\ & & & \sqrt{6} & & \\ & & & & \sqrt{6} & \\ & & & & & \sqrt{6} \end{pmatrix}$$

Si l'on pose $v = \mathcal{G}^{-1}(u)$ et donc $u = \mathcal{G}(v)$, en faisant jouer à u le rôle de σ_r et à v celui u_r dans $(\mathcal{P})$, la deuxième contrainte de $(\mathcal{P})$ devient,

$$u_{1r}^2 + u_{2r}^2 + (u_{1r} + u_{2r})^2 + u_{4r}^2 + u_{5r}^2 + u_{6r}^2 \leq 2a_r^2. \tag{8.16}$$

Puisque l'on a $\sigma_r = Tu_r$ la troisième contrainte de $(\mathcal{P})$ devient,

$$X_r - Tu_r + \alpha\beta_r = 0. \tag{8.17}$$

On a d'une part,

$$Tu_r = \sum_{i=1}^{6} T^i u_{ir} = \sum_{\substack{i=1 \\ i\neq 3}}^{6} T^i u_{ir} + T^3 u_{3r}, \tag{8.18}$$

où T^i désigne la i^{eme} colonne de la matrice T d'autre part (8.17) nous permet d'exprimer X_r en fonction de Tu_r, en remplaçant Tu_r par (8.18), on a,

$$\begin{aligned} C_r X_r &= C_r T u_r - \alpha C_r \beta_r \\ &= C_r(\sum_{\substack{i=1 \\ i\neq 3}}^{6} T^i u_{ir} + T^3 u_{3r}) - \alpha C_r \beta_r. \end{aligned}$$

On note

$$\bar{u}_r = \begin{pmatrix} \bar{u}_{1r} = u_{1r} \\ \bar{u}_{2r} = u_{2r} \\ \bar{u}_{3r} = u_{4r} \\ \bar{u}_{4r} = u_{5r} \\ \bar{u}_{5r} = u_{6r} \end{pmatrix}$$

et on a

$$C_r \sum_{\substack{i=1 \\ i\neq 3}}^{6} T^i u_{ir} = \sum_{\substack{i=1 \\ i\neq 3}}^{6} (C_r T^i) u_{ir} = \sum_{\substack{i=1 \\ i\neq 3}}^{6} (C_r T)^i u_{ir}.$$

On forme la matrice

$$A_r = \begin{pmatrix} (C_rT)^1 & (C_rT)^2 & (C_rT)^4 & (C_rT)^5 & (C_rT)^6 \end{pmatrix} \in \mathcal{M}(\mathbb{R})_{(3NK,5)}.$$

On peut donc écrire,

$$\sum_{\substack{i=1 \\ i\neq 3}}^{6} (C_rT)^i u_{ir} = A_r \bar{u}_r.$$

On introduit $x_r = u_{3r}$ pour $r = 1, \ldots, NG$ et on définit le vecteur $x \in \mathbb{R}^{NG}$ dont les composantes sont les x_r. On a ainsi

$$C_r T u_r = A_r \bar{u}_r + (C_rT)^3 x_r.$$

On définit la matrice

$$B = \begin{pmatrix} (C_1T)^3 & (C_2T)^3 & (C_3T)^3 & \ldots & (C_{NG}T)^3 \end{pmatrix} \in \mathcal{M}(\mathbb{R})_{(3NK,NG)}.$$

On pose

$$w = \sum_{r=1}^{NG} C_r \beta_r$$

A ce niveau le problème transformé est

$$\begin{cases} \max\ \alpha \\ \quad t.q \\ \quad \sum_{r=1}^{NG} A_r \bar{u}_r + Bx - \alpha w = 0 \\ \quad u_{1r}^2 + u_{2r}^2 + (u_{1r} + u_{2r})^2 + u_{4r}^2 + u_{5r}^2 + u_{6r}^2 \leq 2a_r^2. \\ \quad r = 1, 2, \cdots, NG. \end{cases}$$

L'étape suivante de la transformation est la diagonalisation des contraintes quadratiques. Pour plus de clarté dans la présentation, nous abandonnons momentanément l'indice r. Les contraintes quadratiques définissent une forme quadratique

$$f(u) = \frac{1}{2}\langle Q\bar{u}, \bar{u}\rangle = u_1^2 + u_2^2 + (u_1 + u_2)^2 + u_4^2 + u_5^2 + u_6^2$$

où $\bar{u}$ est redéfini comme suite $\bar{u} = (u_1, u_2, u_4, u_5, u_6)$ et la matrice Q est,

$$Q = \begin{pmatrix} 4 & 2 & & & \\ 2 & 4 & & & \\ & & 2 & & \\ & & & 2 & \\ & & & & 2 \end{pmatrix}$$

Soit LL^T la décomposition de Cholesky de la matrice $\frac{1}{2}Q$, avec

$$L = \begin{pmatrix} \sqrt{2} & & & & \\ \frac{1}{\sqrt{2}} & \frac{\sqrt{3}}{\sqrt{2}} & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{pmatrix}$$

On a

$$\begin{aligned} \langle \frac{1}{2} Q\bar{u}, \bar{u} \rangle &= \langle LL^T \bar{u}, \bar{u} \rangle \\ &= \langle L^T \bar{u}, L^T \bar{u} \rangle. \end{aligned}$$

Ainsi le problème modifié est,

$$(\mathcal{PM}) \begin{cases} \max\ \alpha \\ t.q \\ \sum_{r=1}^{NG} A_r \bar{u}_r + Bx - \alpha w = 0 \\ \| L^T \bar{u}_r \|^2 \leq 2a_r^2. \\ r = 1, 2, \cdots, NG. \end{cases}$$

En effectuant le changement de variable $u_r = L^T \bar{u}_r$ pour simplifier les notations on pose à nouveau $A_r = A_r L^{-T}$ et $u_r = \frac{u_r}{a_r\sqrt{2}}$. Le problème $(\mathcal{PM})$ devient,

$$(\mathcal{PM}) \begin{cases} \max\ \alpha \\ t.q \\ \sum_{r=1}^{NG} A_r u_r + Bx - \alpha w = 0 \\ \| u_r \|^2 \leq 1. \\ r = 1, 2, \cdots, NG. \end{cases}$$

Ce problème a $6NG + 1$ variables et $3NK + NG$ contraintes et reste un problème convexe.

Remarque 8.4.1 *La transformation effectuée n'est pas la seule qui nous permet d'obtenir un problème ayant moins de variables et moins de contraintes.*
En effet une transformation équivalente serait de tirer X_r du troisième bloc de contraintes du problème $(\mathcal{P})$ et de remplacer cette valeur dans les autres termes on obtient le problème,

$$\begin{cases} \max\ \alpha \\ t.q \\ \sum_{r=1}^{NG} C_r\sigma_r - \alpha\sum_{r=1}^{NG} C_r\beta_r = 0 \\ (\sigma_{1r}-\sigma_{2r})^2+(\sigma_{2r}-\sigma_{3r})^2+(\sigma_{3r}-\sigma_{1r})^2+6\sigma_{4r}^2+6\sigma_{5r}^2+6\sigma_{6r}^2 \leq 2a_r^2 \\ r=1,2,\cdots,NG. \end{cases}$$

Ce problème a également $6NG+1$ variables et $3NK+NG$ contraintes et reste un problème convexe. Cependant avec cette reformulation le Hessien du Lagrangien est potentiellement plus dense que dans la première reformulation en raison de la présence de contraintes quadratiques plus complexes. Cependant cette reformulation a l'avantage de ne pas trop transformer le problème d'origine, la structure de la matrice $[C]$ y est par exemple conservée.
Nous avons néanmoins opté pour la première reformulation en raison principalement de la structure plus creuse du Hessien du Lagrangien dans celle-ci. Ce paramètre est non négligeable dans notre cadre d'étude car nous aurons à faire à des problèmes de grandes dimensions.

□

8.4.2 Problème dual du problème reformulé

On définit la matrice

$$M = \begin{pmatrix} A_1 & A_2 & \dots & A_{NG} & B & -w \end{pmatrix} \in \mathcal{M}(\mathbb{R})_{(3NK,6NG+1)}.$$

De plus on a

$$-\alpha = \langle -e_{6NG+1}, (u,x,\alpha)^T\rangle$$

où $u=(u_1,u_2,\dots,u_{NG})$ et e_{6NG+1} est le vecteur dont toutes les composantes valent 0 sauf la 6NG+1-ème. Soit $\lambda \in \mathbb{R}^{3NK}$ le multiplicateur associé aux contraintes égalité, si on pénalise les contraintes égalité on obtient le problème $(\mathcal{P}_\lambda)$ suivant

$$g(\lambda) = \inf_{(x,\alpha,u_1,u_2,\cdots,u_{NG})\in \mathbb{R}^{NG}\times\mathbb{R}\times\mathcal{C}_1\times\mathcal{C}_2\times\cdots\times\mathcal{C}_{NG}} \{\langle -e_{6NG+1}, (u,x,\alpha)^T\rangle + \langle \lambda, M(u,x,\alpha)^T\rangle\}$$

où

$$\mathcal{C}_r = \{u_r \in \mathbb{R}^5 \ : \parallel u_r\parallel^2 \leq 1\}, \quad \text{pour i} = 1,2,\cdots,\text{NG}$$

et on a

$$u \in \mathcal{C} = \overset{NG}{\underset{r=1}{X}} C_r.$$

Le problème dual au problème $(\mathcal{PM})$ est,

$$(\mathcal{PD}) \quad \sup\{g(\lambda) \ : \ \lambda \in \mathbb{R}^{3NK}\}.$$

Calcul explicite de $g(\lambda)$

Les conditions d'optimalité s'écrivent,

$$(\mathcal{KKT}) \begin{cases} 0 \in -e_{6NG+1} + \lambda^T M + \partial\mathcal{X}_{\mathcal{C}\times\mathbb{R}^{NG}\times\mathbb{R}}(u,x,\alpha), \\ u_r \in \mathcal{C}_r, \\ \mu_r(\parallel u_r\parallel^2 - 1) = 0, \\ \mu_r \geq 0, \\ r = 1, 2, \cdots, NG. \end{cases}$$

Puisque $\partial\mathcal{X}_{\mathcal{C}\times\mathbb{R}^{NG}\times\mathbb{R}}(u,x,\alpha)$ est un cône, donc produits de cônes, on a

$$\begin{aligned} \partial\mathcal{X}_{\mathcal{C}\times\mathbb{R}^{NG}\times\mathbb{R}}(u,x,\alpha) &= \partial\mathcal{X}_{\mathcal{C}}(u) \times \partial\mathcal{X}_{\mathbb{R}^{NG}}(x) \times \partial\mathcal{X}_{\mathbb{R}}(\alpha) \\ &= \partial\mathcal{X}_{\mathcal{C}_1}(u_1) \times \partial\mathcal{X}_{\mathcal{C}_2}(u_2) \times \ldots \times \partial\mathcal{X}_{\mathcal{C}_{NG}}(u_{NG}) \times \partial\mathcal{X}_{\mathbb{R}^{NG}}(x) \times \partial\mathcal{X}_{\mathbb{R}}(\alpha) \\ &= \partial\mathcal{X}_{\mathcal{C}_1}(u_1) \times \partial\mathcal{X}_{\mathcal{C}_2}(u_2) \times \ldots \times \partial\mathcal{X}_{\mathcal{C}_{NG}}(u_{NG}) \times 0_{\mathbb{R}^{NG}} \times 0_{\mathbb{R}} \end{aligned}$$

de plus on a

$$\partial\mathcal{X}_{\mathcal{C}r} = \begin{cases} 0 & si \ \parallel u_r \parallel^2 < 1 \\ \mathbb{R}_+ u_r & si \ \parallel u_r \parallel^2 = 1. \end{cases}$$

Le système $(\mathcal{KKT})$ devient

$$(\mathcal{KKT}) \begin{cases} 0 = -e_{6NG+1} + \lambda^T M + ((\mu_1 u_1, \mu_2 u_2, \ldots, \mu_{NG} u_{NG}), 0_{\mathbb{R}^{NG}}, 0_{\mathbb{R}}), \\ u_r \in \mathcal{C}_r, \\ \mu_r(\parallel u_r\parallel^2 - 1) = 0, \\ \mu_r \geq 0, \\ r = 1, 2, \cdots, NG. \end{cases}$$

Notons

$$\bar{g}(\lambda) = \langle -e_{6NG+1}, (u,x,\alpha)^T\rangle + \langle \lambda, M(u,x,\alpha)^T\rangle. \tag{8.19}$$

Puisque

$$\langle M^T\lambda, (u,x,\alpha)\rangle = \sum_{r=1}^{NG}\langle A_r^T\lambda, u_r\rangle + \langle B^T\lambda, x\rangle - \alpha w^T\lambda, \tag{8.20}$$

on a

$$\bar{g}(\lambda) = -\alpha(1 + w^T\lambda) + \sum_{r=1}^{NG}\langle A_r^T\lambda, u_r\rangle + \langle B^T\lambda, x\rangle. \tag{8.21}$$

Puisque α et x sont non bornés on peut avoir $\tilde{g}(\lambda) \longrightarrow \infty$. Pour éviter cette situation on annule les termes où α et x interviennent.
Le problème dual est donc,

$$(\mathcal{PD}) \quad \sup\{\bar{g}(\lambda) \ : \ B^T\lambda = 0, 1 + w^T\lambda = 0\}, \tag{8.22}$$

soit

$$(\mathcal{PD}) \quad \sup\{-\sum_{r=1}^{NG} \| A_r^T\lambda \| \ : \ B^T\lambda = 0, 1 + w^T\lambda = 0\}, \tag{8.23}$$

car le problème dual est séparable et

$$\inf\{\langle A_r^T\lambda, u_r\rangle \ : \ u_r \in \mathcal{C}_r\}$$

a pour solution

$$u_r^\star = -\frac{A_r^T\lambda}{\| A_r^T\lambda \|}$$

si $\lambda \neq 0$ et tout $u_r \in C_r$ est solution sinon.

Le problème dual $(\mathcal{PD})$ a $3NK$ variables et $NG + 1$ contraintes, il est convexe mais non différentiable.

8.5 IPDCA pour résoudre le problème convexe $(\mathcal{PM})$

Le problème $(\mathcal{PM})$ est du type de problèmes $(\mathcal{P}_{\mathcal{EI}})$ dont les calculs pour le résoudre avec IPDCA sont présentés en Annexe A. Le problème $(\mathcal{PM})$ étant un problème convexe, est un faux problème DC Pour lui appliquer IPDCA, on le reformule sous la forme DC

$$(\mathcal{P}_{dc}) \begin{cases} \max \ f(X) = g(X) - h(X) \\ t.q \\ \quad H(X) = 0 \\ \quad G(X) \leq 0. \end{cases}$$

où X est un vecteur de $\mathbb{R}^{6NG+1}$ représentant le vecteur des variables du problème $(\mathcal{PM})$, $H \in \mathbb{R}^{3NK}$, $G \in \mathbb{R}^{NG}$ représentant respectivement les contraintes linéaires et les contraintes non linéaires du problème $(\mathcal{PM})$, g est une fonction fortement convexe(par exemple $g(X) = f(X) + \frac{1}{2}\rho \| X \|^2$) et $h(X) = \frac{1}{2}\rho \| X \|^2$, avec ρ un nombre positif.

Avant de pouvoir appliquer IPDCA au problème $(\mathcal{PM})$ nous devons nous assurer que celui-ci vérifie les hypothèses qui garantissent la convergence de IPDCA. Tout d'abord les contraintes sont qualifiées car H est linéaire et dans la pratique surjective de plus G est convexe et il existe X telque $G(X) < 0$. La qualification des contraintes implique que le problème dual admet une solution optimale $\lambda^\star$ et on a toujours

$$\alpha \leq \sum_{r=1}^{NG} \| A_r^T\lambda_r^\star \|, \ \ \forall\alpha$$

donc α est borné supérieurement, de plus $\alpha = 0$, $X_r = 0$, $u_r = 0$ est un point admissible du problème, donc $\alpha \geq 0$ ainsi α est borné ; de plus u_r est borné grâce aux contraintes

$\| u_r \| \leq a_r$ et grâce aux équations $X_r + \alpha\beta_r - Tu_r = 0$, X_r est aussi borné. Les conditions de convergence de l'algorithme sont remplies.

Pour résoudre le problème convexe $(\mathcal{PM})$ avec IPDCA, on applique IPDCA au faux problème DC $(\mathcal{P}_{dc})$. Les principales étapes de ce processus sont présentées ci-dessous.
Tout d'abord le problème $(\mathcal{P}_{dc})$ est réécrit en ajoutant des variables d'écart $W \in \mathbb{R}_+^{NG}$ et le terme logarithmique barrière à la fonction objectif. On obtient le problème

$$(\mathcal{P}_{dc}^{\mu}) \begin{cases} \max\ f_\mu(X,W) = f(X) - \mu \sum_{i=1}^{NG} log(W_i) \\ t.q \\ \quad H(X) = 0 \\ \quad G(X) + W = 0. \end{cases}$$

Dans l'approche classique des méthodes de points intérieurs, pour résoudre le problème $(\mathcal{P}_{dc})$, un pas de Newton est effectué au système KKT du problème $(\mathcal{P}_{dc}^{\mu})$, une longueur de pas primal et dual est par la suite calculée à l'aide d'une fonction de mérite, puis suit la mise à jour du paramètre barrière μ. R. Vanderbei a développé une factorisation du type factorisation de Cholesky pour résoudre les systèmes linéaires ayant des matrices quasi-définies. Pour que cette technique puisse être utilisée dans la méthode de Newton pour résoudre le système de KKT, IPDCA applique une méthode de points intérieurs pas directement au problème $(\mathcal{P}_{dc}^{\mu})$ mais au problème

$$(\mathcal{P}_{dc}^{\mu})_k \begin{cases} \max\ f_{\mu,k}(X,W) = g(X) - \langle X, Y_k \rangle - \mu \sum_{i=1}^{NG} log(W_i) \\ t.q \\ \quad H(X) = 0, \\ \quad G(X) + W = 0, \end{cases}$$

où $Y_k = \partial H(X_k) = \rho X_k$.
En d'autres termes IPDCA effectue un pas de Newton au problème convexe $(\mathcal{P}_{dc}^{\mu})_k$ au lieu de le résoudre complètement pour déterminer l'itéré suivant X^{k+1} comme le fait DCA. En contre partie IPDCA doit calculer à chaque itération la longueur de pas primale et la longueur de pas duale grâce à une fonction de mérite et mettre à jour le paramètre barrière μ. En plus pour obtenir une matrice quasi-définie lors de la résolution avec la méthode de Newton du système KKT nous avons adopté lors de notre implémentation la régularisation par ajout de terme constant présentée à la Section A.2.2. Ce choix fut motivé par le fait que nous avons à faire à des problèmes de grandes dimensions ; utiliser la régularisation par ajout de variables et de contraintes comme le fait Vanderbei et comme présenté à la Section A.2.1 demanderait plus d'espace mémoire pour un résultat équivalent.

Nous avons choisit de comparer IPDCA à LANCELOT sur ce problème car LOQO s'est avéré incapable de résoudre les problèmes au-delà d'une certaine dimension ; cette observation fut déjà faite à la Section 3.8.
Nous donnons dans la section qui suit une brève description de LANCELOT.

8.6 LANCELOT

LANCELOT [186, 187] est un logiciel d'optimisation qui emploie une approche Lagrangien augmenté pour traiter toutes les contraintes autres que les bornes sur les variables. Les bornes sont traitées explicitement au niveau supérieur dans un sous-problème. Ce sous-problème est un problème d'optimisation non linéaire n'ayant que des bornes sur les variables, est approximativement résolu à chaque itération.

Pour résoudre le problème en présence, LANCELOT combine une approche de région de confiance adaptée pour manipuler les bornes sur les variables, les techniques de gradient projeté, et les structures de données spéciales(groupe partiellement séparable) pour exploiter la structure du problème sous jacent.

Le logiciel fournit en plus des algorithmes itératifs et directs pour résoudre les systèmes linéaires(pour les équations de Newton), une variété d'algorithmes de préconditionnement, des méthodes quasi-Newtoniennes et Newtoniennes, dispose de routines pour le calcul des gradients analytiques et par différence-finie, et un décodeur automatique capable de lire les problèmes exprimés en format standard d'entrée SIF[1] [134].

LANCELOT est écrit en Fortran 77 de norme ANSI, une interface avec AMPL est également disponible. Les versions simple et à double précision sont disponibles. Nous avons utilisé la version à double précision pour nos tests numériques.

8.7 Résultats numériques

Le code est écrit en C et testé sur un nœud du cluster IBM 1600 installé au CRIHAN[2], le travail fut à chaque fois soumis en *batch*(traitement par lots). La compilation fut faite grâce à la commande *xlc,* l'option *-q64* fut utilisée ; celle-ci provoque la création par la compilation de fichiers objet en mode d'adressage 64 bits. Cette option a une incidence sur les quantités mémoire que peuvent occuper la stack et le heap. En effet en mode 64 bits, ils peuvent occuper plusieurs millions de Go de mémoire.

La précision fut fixée à $10e^{-4}$ pour LANCELOT, NIPA et pour IPDCA, cette valeur nous fut suggérée par nos interlocuteurs de l'IAM. Le temps de résolution est reporté en secondes

[1] SIF pour standard input format
[2] le CRIHAN est le Centre de ressources d'Informatique de Haute-Normandie

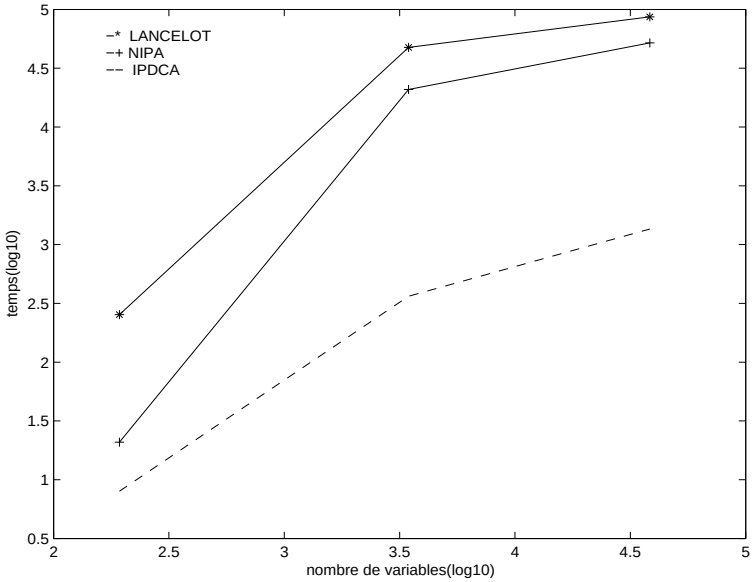


FIG. 8.2 – Comparaison des temps de calcul pour LANCELOT, NIPA et IPDCA.

ou en nombre de jours, heures et minutes pour le dernier exemple.

Les dérivées premières et secondes pour IPDCA et NIPA ont été fournies aux algorithmes.

Dans le Tableau 8.1,
n désigne le nombre de variables du problème,
m désigne le nombre de contraintes du problème,
iter le nombre d'itérations,
temps le temps d'exécution,
– l'algorithme n'a pas pu résoudre le problème.

On remarque la supériorité d'IPDCA sur les deux autres algorithmes ; on remarque en particulier la grande supériorité d'IPDCA sur LANCELOT en termes de temps de calcul ; la Figure 8.2 l'illustre clairement. On relève également que la solution retournée par IPDCA est meilleure que celle des deux autres algorithmes pour la précision choisie. Une précision plus petite conduirait à une solution retournée par LANCELOT au moins aussi bonne que celle retournée par IPDCA, le temps de calcul étant bien entend beaucoup plus important dans ce cas. Pour les problèmes P4 et P5, il n'a pas été possible de générer les fichiers SIF nécessaires à LANCELOT, en raison de la grande taille des problèmes. Pour le problème P4, NIPA fut arrêté au bout de trois jours en raison de sa trop lente progression vers la solution et le problème P5 en raison de sa dimension ne fut pas testé par cet algorithme.

Notons que pour ces tests numériques nous avons utilisé la fonction **dgesv** de CLAPACK

noms	*dim*		*iter*			*obj*			*temps*		
	n	m	LANC	NIPA	IPDCA	LANC	NIPA	IPDCA	LANC	NIPA	IPDCA
P1	193	148	81140	10	138	1.3	1.5	1.5	253.35	20.86	7.98
P2	3457	2268	12350	15	209	47.4	50.4	50.9	47565.45	20830.12	363.67
P3	38401	25046	17500	17	172	2.4	2.4	2.4	86400.00	51960.25	1356.63
P4	57601	36552	-	-	1334	-	-	5.1	-	-	1828.66
P5	192001	125200	-	-	1438	-	-	5.5	-	-	2j0h33min

TAB. 8.1 – Comparaison de IPDCA, NIPA et LANCELOT sur le problème de mécanique.

pour résoudre le système linéaire (2.8) de l'algorithme NIPA. Il s'agit d'une stratégie différente de celle présentée au Chapitre 2, nous n'avons cependant observé aucune instabilité de l'algorithme avec cette nouvelle approche, mais la meilleure stratégie pour résoudre le système linéaire (2.8) reste à l'étude.

L'utilisation de la fonction de pénalité L_1 peut provoquer un ralentissement de l'algorithme et parfois une impossibilité d'atteindre une tolérance d'optimalité prédéfinie. C'est l'effet Maratos[3] ; cet effet pourra être annihilé par l'incorporation dans l'algorithme d'une correction du second ordre en adaptant la stratégie présentée dans [195] par exemple, ou par l'intégration d'une procédure de recherche linéaire non-monotone [29, 30]. Nous n'avons pas été confronté à cet effet lors des tests numériques effectués, mais l'intégration de ces procédures fera parti de nos développements futurs pour accroître la robustesse de notre algorithme.

[3]la fonction de mérite L_1 peut parfois rejeter des pas qui pourraient permettre un bon progrès vers la solution.

Chapitre 9

Combiné IPDCA-DCA dans un processus d'apprentissage avec noyaux non-positifs

Depuis une dizaine d'années, une révolution partielle est en train de refaçonner le domaine de la reconnaissance de forme à travers des algorithmes tels que les Séparateurs à Vaste Marge (SVM) et les méthodes à noyau [168, 169]. Ces méthodes, qui connaissent encore d'importants développements dans des domaines aussi divers que la détection, la classification, la régression ou encore l'estimation [170, 171, 172, 176], présentent les avantages des réseaux de neurones de type perceptron multicouche sans en avoir les inconvénients. En effet, ils ont le caractère «universel»(ils peuvent tout apprendre) des réseaux de neurones mais, outre la relative facilité de leur mise en oeuvre, ils sont étayés par des fondements théoriques solides qui assurent notamment l'existence et l'unicité de la solution du problème d'apprentissage. Et pratiquement, ils présentent d'excellentes performances. Tout cela est lié à la nature des «noyaux». Les noyaux sont des mesures de similarité entre des observations qui définissent implicitement une projection dans un espace de grande dimension [175, 173, 174]. Le noyau est alors vu comme le produit scalaire de la projection des données sur un espace de Hilbert à noyau reproduisant, défini comme l'espace des caractéristiques. En remarquant que les fonctions linéaires peuvent être estimées et évaluées complètement à partir de certains produits scalaires, il est possible en utilisant les noyaux d'apprendre des fonctions qui vont être linéaires dans l'espace des caractéristiques de dimension infinie, mais non linéaires dans l'espace des observations. Ce principe permet donc d'étendre une large classe d'algorithmes linéaires au non linéaires sans qu'il soit nécessaire de recourir à d'autres développements théoriques. Dans [167] Mangasarian et *al.* présentent un certain nombre de contributions de l'optimisation pour résoudre les problèmes de fouille de données(Data Mining) après avoir présenté les domaines de leurs applications ; ils y listent également un certain nombre de défis qui restent à être relevés par les spécialistes de l'optimisation. Très récemment de nouvelles approches pour la classification et la selection ont été proposées dans le cadre générale de l'optimisation, incluant les SVMs linéaires et non linéaires [194] et en utilisant des techniques de l'optimisation DC et DCA.

Dans ce chapitre nous allons appliquer IPDCA-DCA à un problème de Séparateurs à Vaste Marge. Le problème est posé comme problème d'interpolation et reformulé comme programme quadratique non convexe en suivant la méthodologie présentée dans [145].

9.1 Séparateurs à Vaste Marge

9.1.1 Espace de Krein à noyau reproduisant

Nous souhaitons trouver une fonction f interpolante, cette interpolation est faite à partir de noyaux. Un *noyau* est une fonction k de deux variables symétrique et positive à valeurs dans $\mathbb{R}$.
A partir du noyau on construit l'ensemble de combinaisons linéaires finies

$$\mathcal{H}_o = \{f \ : \mathbb{R}^n \to \mathbb{R} \ : \ \exists p \in \mathbb{N}, \alpha \in \mathbb{R}^n, \ \{x_i\}_{i=1,2,\cdots,p}, \ x_i \in \mathbb{R}^n, \ f(x) = \sum_{i=1}^{n} \alpha_i k(x, x_i)\}.$$

On définit la forme bilinéaire

$$\langle ., . \rangle_{H_o} \ : \ \mathcal{H}_o \times \mathcal{H}_o \to \mathbb{R}, \ \ (f, g) \longmapsto \langle f, g \rangle = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \beta_j k(x_i, x_j) \tag{9.1}$$

avec $f(x) = \sum_{i=1}^{n} \alpha_i k(x, x_i)$ et $g(x) = \sum_{i=1}^{n} \beta_i k(x, x_i)$, $\langle ., . \rangle_{\mathcal{H}_o}$ définit un produit scalaire, on en déduit la norme,

$$||f||_{\mathcal{H}_o} = \sqrt{\langle f, f \rangle_{\mathcal{H}_o}}.$$

Muni de ce produit scalaire $\mathcal{H}_o$ est un espace préhilbertien dans lequel la *propriété de reproduction*

$$\langle f(.), f(x, .) \rangle_{\mathcal{H}_o} = f(x)$$

est vérifiée. Pour obtenir un espace de Hilbert on pose

$$\mathcal{H} = \bar{\mathcal{H}_o}.$$

Ainsi défini $\mathcal{H}$ est un espace de Hilbert dit à noyau reproduisant.

Dans le cas où la fonction k n'est pas positive, on se donne deux noyaux positifs k^+ et k^-. Et on fait l'hypothèse de l'existence de la décomposition

$$k = k^+ - k^-.$$

L'effet imédiat de cette hypothèse est qu'on peut écrire,

$$\langle f, f \rangle_{\mathcal{H}_o} = \langle f^+, f^+ \rangle_{\mathcal{H}_o^+} - \langle f^-, f^- \rangle_{\mathcal{H}_o^-},$$

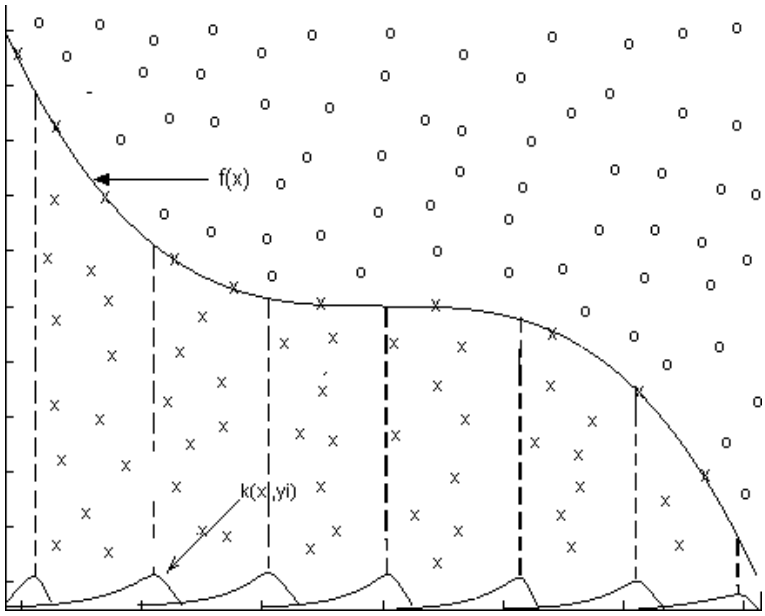


FIG. 9.1 – Fonction f obtenue par interpolation.

on a deux structures Hilbertiennes. En posant

$$\mathcal{H} = \bar{\mathcal{H}}_o^+ \ominus \bar{\mathcal{H}}_o^-,$$

$\mathcal{H}$ ainsi défini est un espace de Krein à noyau reproduisant.

9.1.2 Problème d'interpolation

Considérons l'échantillon (x_i, y_i), $i \in \{1, 2, \cdots, n\}$ avec le codage $y_i \in \{-1, +1\}$. Le problème d'interpolation se formule comme suit :

$$\text{trouver } \; f \; \text{ dans } \; \mathcal{H} \; \text{ tel que } \; f(x_i) = y_i \;\; i = 1, 2, \cdots, n.$$

Pour cela on définit l'opérateur $\mathcal{T}$ de manière suivante

$$\mathcal{T} \;\; : \;\; \mathcal{H} \to \mathbb{R}^n, \;\; f \longmapsto \mathcal{T}f = \begin{pmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{pmatrix}. \tag{9.2}$$

L'opérateur $\mathcal{T}$ s'appelle *opérateur d'évaluation.* Le problème d'interpolation peut donc s'écrire sous la forme variationnelle,

$$\text{trouver } \; f \; \text{ dans } \; \mathcal{H} \; \text{ tel que } \; \mathcal{T}f = y. \tag{9.3}$$

Mais (9.3) équivaut à,

$$\forall \alpha \in \mathbb{R}^n, \;\; \alpha^T \mathcal{T} f = \alpha^T y.$$

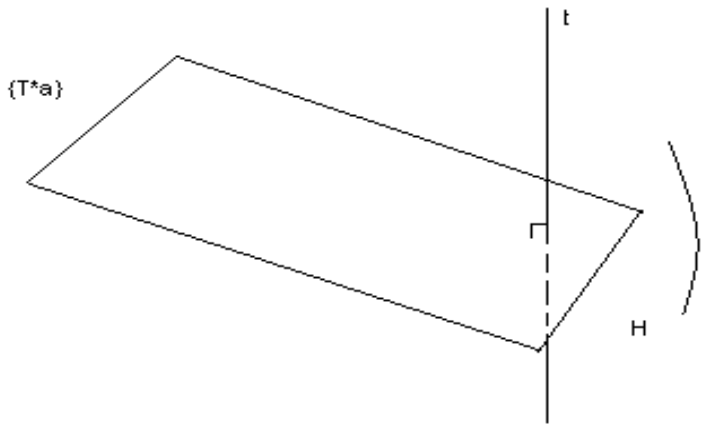


FIG. 9.2 – Orthogonal à l'espace de Krein $\mathcal{H}$.

On pose $\mathcal{I} = \{f \in \mathcal{H} \ : \ \mathcal{T}f = y\}$ l'ensemble des fonctions interpolantes.
On a la chaîne d'équivalences,

$$\forall t \in \mathcal{I}, \ \alpha^T \mathcal{T} f = \alpha^T \mathcal{T} t \ \ \forall \alpha \in \mathbb{R}^n \iff \alpha^T \mathcal{T}(f-t) = 0 \iff \langle \mathcal{T}^\star \alpha, f-t \rangle_{\mathcal{H}} = 0 \ \ \forall \alpha \in \mathbb{R}^n.$$

Cette chaîne d'équivalences nous fournit une caractérisation la solution du problème par orthogonalité.
Dans le cas Hilbertien la solution de (9.3) serait la solution de norme minimale.

On vient de voir que $\mathcal{T}^\star \alpha = f$ et donc $\mathcal{T}\mathcal{T}^\star \alpha = y$ car $\mathcal{T}f = y$.
Mais on sait également que $f(x) = \sum_{i=1}^{n} \alpha_i k(x, x_i)$ et donc,

$$\mathcal{T}^\star \alpha = \sum_{i=1}^{n} \alpha_i k(x, x_i).$$

On en déduit que

$$\mathcal{T}^\star \mathcal{T} \alpha = \begin{pmatrix} \sum_{i=1}^{n} \alpha_i k(x_1, x_i) \\ \vdots \\ \sum_{i=1}^{n} \alpha_i k(x_n, x_i) \end{pmatrix} = \mathcal{K}\alpha$$

où $\mathcal{K}_{ij} = k(x_i, x_j)$. On a le problème d'interpolation de norme minimale,

$$\left\{\begin{array}{l} \underset{f\in\mathcal{H}}{\text{stabiliser}} \ \frac{1}{2}\langle f, f\rangle_{\mathcal{H}} \\ t.q. \\ \quad Tf = y \end{array}\right. \iff \left\{\begin{array}{rcl} \mathcal{K}\alpha & = & y \\ f & = & \mathcal{T}^\star\alpha. \end{array}\right.$$

9.1.3 Problème de programmation quadratique non convexe

Dans le cadre d'espace de Krein $\mathcal{H}$, le problème de séparateurs à vaste marge du problème de discrimination à deux classes est le problème d'optimisation,

$$\left\{\begin{array}{l} \underset{f\in\mathcal{H},\xi,b}{\text{stabiliser}} \ \frac{1}{2}\langle f, f\rangle_{\mathcal{H}} + C(\sum_{i=1}^{n}\xi_i)^p \\ t.q. \\ \quad y_i(f(x_i)+b) \geq 1-\xi_i, \ \ i = 1, 2, \cdots, n \\ \quad \xi_i \geq 0, \ \ i = 1, 2, \cdots, n. \end{array}\right. \tag{9.4}$$

où C et p sont des paramètres qui doivent être déterminés à l'avance et définissent le coût de la violation des contraintes mesurée par les variables d'écart $\{\xi_i\}_{i=1}^n$. Dans ce problème la marge est maximisée tout en payant une pénalité proportionnelle à la violation des contraintes.

Pour résoudre le problème (9.4), on définit le Lagrangien

$$\mathcal{L}(f, b, \xi, \alpha, \gamma) = \frac{1}{2}\langle f, f\rangle_{\mathcal{H}} + C(\sum_{i=1}^{n}\xi_i)^p - \sum_{i=1}^{n}\alpha_i(y_i(f(x_i)+b) - 1 + \xi_i) - \sum_{i=1}^{n}\gamma_i\xi_i.$$

La solution du problème (9.4) est alors point selle de $\mathcal{L}(f, b, \xi, \alpha, \gamma)$, qui doit être minimizer par rapport à f, ξ, et b, et maximiser par rapport à $\alpha \geq 0$ et $\gamma \geq 0$. En différentiant et en posant le résultat égale à zéro on obtient,

$$\begin{array}{rcl} \frac{\partial\mathcal{L}(f,b,\xi,\alpha,\gamma)}{\partial f} & = & f - \sum_{i=1}^{n}\alpha_i y_i k(x_i, .) \qquad\qquad = \ 0, \\ \frac{\partial\mathcal{L}(f,b,\xi,\alpha,\gamma)}{\partial b} & = & \sum_{i=1}^{n}\alpha_i y_i \qquad\qquad = \ 0, \\ \frac{\partial\mathcal{L}(f,b,\xi,\alpha,\gamma)}{\partial \xi} & = & \left\{\begin{array}{rcl} pC(\sum_{i=1}^{n}\xi_i)^{p-1} - \alpha_i - \gamma_i & = & 0 \ \text{ si } \ p > 1 \\ C - \alpha_i - \gamma_i & = & 0 \ \text{ si } \ p = 1. \end{array}\right. \end{array} \tag{9.5}$$

Pour $p > 1$ en posant $\sum_{i=1}^{n}\xi_i = (\frac{\delta}{pC})^{\frac{1}{p-1}}$, de la première équation on obtient,

$$f = \sum_{i=1}^{n}\alpha_i y_i k(x_i, .).$$

En remplassant dans le Lagrangien on obtient

$$F(\alpha,\gamma)=\sum_{i=1}^{n}\alpha_i-\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j k(x_i,x_j)-\frac{\delta^{\frac{1}{p-1}}}{(pC)^{\frac{1}{p-1}}}(1-\frac{1}{p}).$$

Alors pour obtenir le séparateur à vaste marge on résout,

$$\begin{cases}\underset{\alpha,\delta}{\text{stabiliser}}\ \ \langle e,\alpha\rangle-\frac{1}{2}\alpha^T\mathcal{K}\alpha-\frac{\delta^{\frac{1}{p-1}}}{(pC)^{\frac{1}{p-1}}}(1-\frac{1}{p})\\ t.q.\\ \quad\langle y,\alpha\rangle=0\\ \quad\alpha\leq C\\ \quad\alpha\geq 0.\end{cases}\tag{9.6}$$

Quand $k=1$, c'est-à-dire qu'on pénalise linéairement la violation des contraintes et le problème (9.6) devient

$$\begin{cases}\underset{\alpha}{\text{stabiliser}}\ \ \langle e,\alpha\rangle-\frac{1}{2}\alpha^T\mathcal{K}\alpha\\ t.q.\\ \quad\langle y,\alpha\rangle=0\\ \quad\alpha\leq C\\ \quad\alpha\geq 0.\end{cases}\tag{9.7}$$

La solution du problème de séparateurs à vaste marge sera alors donné par la résolution d'un problème d'optimisation quadratique en dimension n sous une contrainte linéaire et des contraintes de boîte. D'autre part il est montré dans [181] que le problème de séparateurs à vaste marge dans le cas des noyaux non positifs peut avoir la même interprétation que dans le cas des noyaux positifs. Dans les sections qui suivent le problème de stabilisation sera vu comme problème de maximisation.

9.2 IPDCA-DCA pour l'apprentissage

Nous venons de voir à la section précédente que les problèmes de séparateurs à vaste marge, peuvent se poser comme programme quadratique non convexe,

$$(SVM)\begin{cases}\max\ \ \langle e,x\rangle-\frac{1}{2}\langle Qx,x\rangle\\ \ \langle y,x\rangle=0\\ \ 0\leq x\leq C.\end{cases}$$

Avec $Q = Q^T \in \mathbb{R}^{n\times n}$, $C \in \mathbb{R}^n$ un vecteur dont toutes les composantes sont des constantes positives toutes égales, e le vecteur de $\mathbb{R}^n$ dont les composantes sont toutes égales à 1, $y \in \{-1, 1\}^n$.
On considère le problème DC suivant qui découle du problème (SVM),

$$\min_{x\in\mathbb{R}^n}\{\mathcal{X}_{\mathcal{K}}(x) + (\frac{1}{2}x^T(Q+\lambda I)x - e^T x) - \frac{1}{2}\lambda x^T x\}$$

où λ est déterminé pour que la matrice $Q + \lambda I$ soit définie positive et
$\mathcal{K} = \{x \in \mathbb{R}^n \ : \ \langle y, x\rangle = 0,\ 0 \le x \le C\}$.

L'algorithme DCA qui en découle est le suivant,

Algorithme 9.2.1 *DCA pour résoudre le problème* (SVM)

0. Soient $\bar{x}^\circ$ un point initial et ε une précision choisie, poser $k = 0$.
1. A l'étape k résoudre le problème de programmation quadratique convexe

$$\min_{x\in\mathcal{K}}\frac{1}{2}x^T(Q+\lambda I)x - e^T x - \frac{1}{2}\lambda x_k^T x$$

on pose x_{k+1} la solution de ce problème.
2. si $\| x_{k+1} - x_k \| /(1+ \| x_k \|_k) \le \varepsilon$ alors fin, sinon $k \leftarrow k+1$ retourner en 1.

La particularité de l'algorithme IPDCA-DCA ci-dessous est l'introduction d'une étape suplémentaire par rapport à l'algorithme IPDCA-DCA présenté au Chapitre 3. Cette étape suplémentaire a pour objectif de déterminer les composantes qui seront éventuellement nulles à la solution. En effet les études empiriques ont montré qu'à la solution une grande partie des composantes de la solution est nulle.

Algorithme 9.2.2 *IPDCA-DCA pour l'apprentissage*

1. ***Lancer*** *IPDCA pour déterminer un x°*
2. ***Déterminer*** *le point $\bar{x}^\circ$ de la manière suivante :*
Pour $i = 1, 2, \cdots, n$
si $x_i^\circ < \nu$ ***alors*** *fixer $\bar{x}_i^\circ = 0$*
sinon *poser $\bar{x}_i^\circ = x_i^\circ$.*
3. ***Lancer*** *DCA à partir du point $\bar{x}^\circ$.*

9.3 Résultats numériques

Pour nos tests numériques le système d'exploitation est Windows XP, la machine un PC de 1300 Mhz de vistesse processeur avec 512K de RAM, les programmes on été écrit en C, la précision choisie pour DCA, IPDCA est $\varepsilon = 10e^{-6}$. Le calcul de la plus petite valeur propre

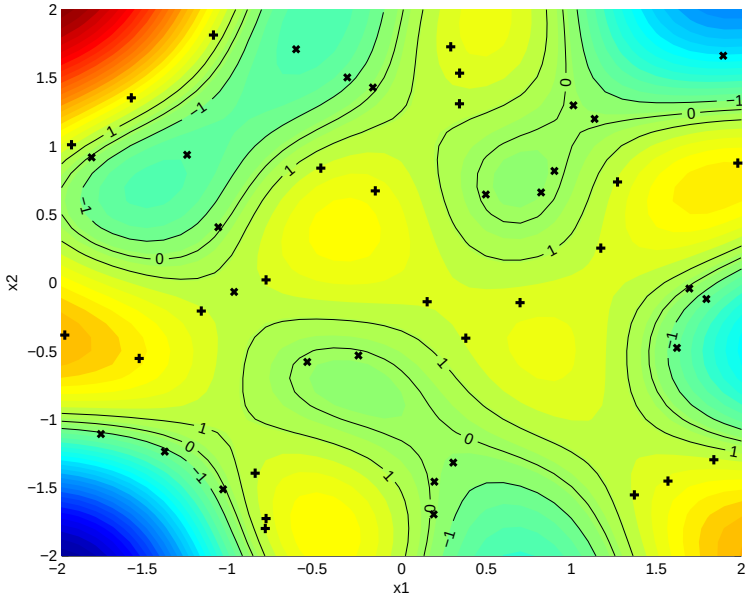


FIG. 9.3 – Forme des données utilisées pour tester IPDCA-DCA.

a été effectué par MATLAB.

Pour ces tests préliminaires, les données sont générées en tirant aléatoirement des points en deux dimensions sur un damier (deux classes). Les étiquettes sont attribuées en fonction de leurs coordonnées. Ensuite le noyau utilisé est celui d'Epanechnikov

$$k(x_1, x_2) = \max\left(0, \left(1 - \frac{(x_1 - x_2)^2}{2\sigma^2}\right)\right).$$

On définit la matrice $\mathcal{K} = \{k(x_i, x_j)\}_{ij}$ de dimension $n \times n$ cette matrice est pré et post multipliée par les étiquettes y. On pose $Q = \{q_{ij}\}_{ij}$ la matrice obtenue. Cette transformation fait de la matrice Q une matrice symétrique indéfinie.

On prend $C = (500, 500, \cdots, 500)^T$. La dimension maximale testée est 4500 variables, la particularité de ce problème est la forte densité du Hessien de la fonction objectif.

La Figure 9.4 présente la complexité d'IPDCA-DCA pour les données dont la génération fut présentée plus haut.
La complexité de notre algorithme peut être améliorée en tenant compte de la structure spéciale des contraintes de ce problème (contrainte boite plus une seule contrainte linéaire) dans l'adaptation d'IPDCA. D'autre part il est important d'appliquer IPDCA à des problèmes quadratiques réellement non convexes dans la mesure où d'autres méthodes de points intérieurs ou de contraintes actives sont performantes, par exemple simpleSVM [146] qui est une méthode itérative utilisant le principe des contraintes actives ou [188], où l'on développe

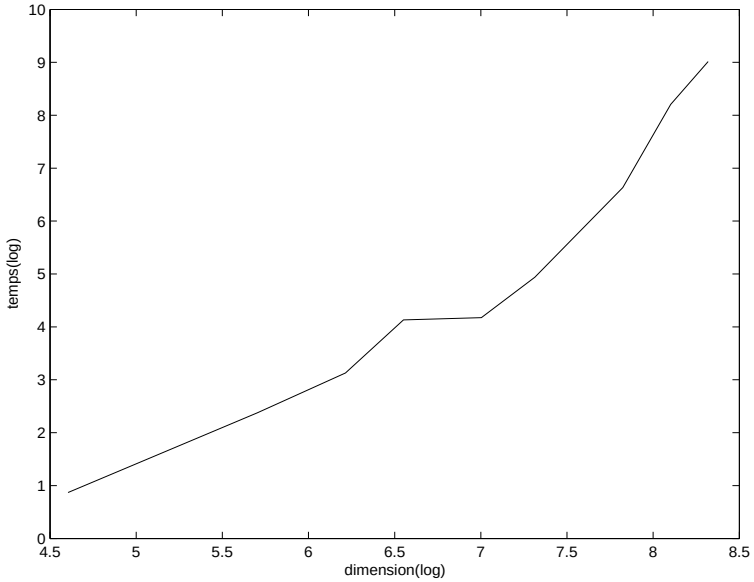


FIG. 9.4 – Complexité d'IPDCA-DCA sur le problème d'apprentissage.

trois familles de méthodes (Lemke, contraintes actives et points intérieurs) pour traiter numériquement les programmes quadratiques convexes issus de SVM à noyau positif. Signalons enfin que le problème non linéaire SVM étudié ici est en réalité équivalent à un problème quadratique convexe.

Conclusion et perspectives

La contribution première de ce travail est la combinaison de deux approches d'optimisation : les méthodes de points intérieurs qui ont montré leur efficacité sur le plan numérique en traitant des problèmes de grandes tailles, et les techniques d'optimisation DC qui donnent un cadre théorique élégant et robuste pour gérer la non-convexité à travers le schéma algorithmique DCA. IPDCA présenté au Chapitre 3, est le premier fruit d'une telle association. Les tests numériques sur des problèmes académiques et sur un problème industriel de grandes tailles, montrent la supériorité de cette approche comparée au schéma classique de points intérieurs. Comparée à DCA, notre algorithme n'a besoin de résoudre qu'un seul système linéaire à chaque itération ce qui le rend globalement plus rapide que DCA ; même si DCA trouve directement la solution globale pour une certaine classe de problèmes. IPDCA hérite de DCA la flexibilité dans la décomposition DC de l'objectif, mais supporte moins de décompositions DC que DCA car limité à la classe de fonctions C^2. Par rapport au schéma classique de points intérieurs, la non-convexité est gérée d'emblée au tout début de l'optimisation grâce à la décomposition DC de la fonction objectif et non par des heuristiques incorporées dans la procédure de factorisation de la matrice intervenant dans le système de Newton.
Nous avons également introduit une procédure à deux phases associant IPDCA pour calculer un bon point initial qui servira à DCA de complèter l'optimisation. Des résultats numériques encourageants sont reportés. Mais une telle procédure doit être utilisée que si on a des doutes sur la qualité du point initial que l'on donne à DCA, car elle peut s'avérer coûteuse. Une autre mise en garde à observer est que, si IPDCA trouve un point qui est un minimum local et non un simple point de KKT, les résultats numériques montrent que DCA n'améliore pas cette solution. Ce qui est normal car DCA est construit à partir des conditions d'optimalité locales.

Des tests numériques plus importants et sur une classe de problèmes plus large seront nécessaires pour faire de IPDCA un logiciel au même titre que ceux qui représentent l'état de l'art actuel dans le domaine des points intérieurs.

Nous avons également introduit un nouvel algorithme(NIPA) basé sur une reformulation des conditions d'optimalité. Si cette approche n'est pas nouvelle dans le domaine des problèmes de complémentarité, son association à la technique du filtre l'est dans le domaine de l'optimisation non linéaire. L'apport essentiel de cette approche est qu'elle nous affranchit de deux obstacles :
• le maintien de la positivité de certaines variables d'ecart à chaque itération ;

- la gestion du paramètre barrière qui est désormais considéré comme variable à part entière, et à ce titre géré automatiquement par l'algorithme.

Ce travail préliminaire nécessite d'autres élaborations sur le plan théorique, et sur le plan numérique, malgré des tests numériques encourageants sur des problèmes de petites dimensions et sur le problème de mécanique de structures de grandes dimensions.

Nous avons exploré la combinaison IPDCA-DCA dans un schéma séparation-évaluation d'abord pour résoudre les les problèmes quadratiques au Chapitre 4, puis au Chapitre 5 pour les problèmes de coupe maximale. Cette étude nous a montré dans le premier cas que cette combinaison entraînait moins de redémarrages pour DCA et donc un gain certain en terme de temps d'exécution comparé à l'approche classique dans laquelle DCA seul est utilisé pour améliorer la borne supérieure, tout en obtenant des résultats comparables. La seconde étude, celle portant sur les problèmes à coupe maximale ne confirma pas cette observation ; dans ce cadre, le temps d'exécution de l'algorithme s'avèra beaucoup plus long et les solutions pas toujours dans $\{0,1\}^n$ ce qui fut le cas dans l'approche classique.

Au Chapitre 7 nous avons proposé une méthode de redémarrage de DCA basée sur les conditions d'optimalité globales. De cette procédure nous pouvons tirer deux enseignements essentiels :

- les algorithmes basés sur les conditions globales comme ceux présentés dans [52, 54, 53, 47, 48, 49, 50] ne donnent pas la solution globale contrairement à ce qui semble y être affirmé ;
- l'intégration de DCA dans ces algorithmes, conduit à une procédure de redémarrage de DCA qui coûte moins chère en terme de temps et de mémoire utilisés, qu'un algorithme de séparation-évaluation classique pour des problèmes quadratiques de tailles moyennes, tout en trouvant des solutions comparables ou à peine moins bonnes que les ε-solutions de l'algorithme de séparation-évaluation.

L'optimisation monotone due à Hoang Tuy et dont nous avons rappelé les principales propriétés au Chapitre 6 associé à DCA dans un schéma séparation-évaluation conduit à des modèles élégants qui présentent néanmoins numériquement des limites qui rendent leur application effective inadaptée aux problèmes de grandes dimensions.

Les algorithmes développés au Chapitre 2 et Chapitre 3 ont été appliqués à un problème de mécanique de structure de grande dimension, IPDCA s'est avéré le plus performant des trois algorithmes. En particulier IPDCA a montré sa supériorité sur le code LANCELOT en résolvant un problème que LANCELOT résout en près d'une semaine en à peine trente minutes. Malgré cette supériorité constatée, nous sommes convaincus que pour des dimensions plus importantes, ce qui est courant en mécanique, la parallélisation du code est une option inévitable.
Nous avons également appliqué IPDCA au problème de séparation à vaste marge avec noyau non positif et nous avons obtenu des résultats prometteurs. Du travail reste néanmoins à faire dans ce domaine notamment pour identifier les variables qui seront nulles à la solution lors

des premières itérations, ceci pourrait réduire considérablement la complexité des problèmes car leur dimension sera plus petite. Des travaux [184, 185, 190] ont été entrepris dans cette direction mais restent encore restreints au cadre expérimental, leur exploitation dans notre contexte nécessite encore des élaborations malgré des résultats numériques encourangeants [183].

Dans cette thèse nous avons exploré les deux aspects de l'optimisation ; l'aspect local par l'introduction de deux nouveaux algorithmes, et l'aspect global pour lequel plusieurs pistes furent explorées. Des tests numériques comparatifs furent à chaque fois effectués et ceux-ci ont mis en exergue les points forts et les éventuelles faiblesses des approches proposées. Les méthodes et approches proposées seront le socle sur lequel reposeront nos recherches futures.

Nos futurs projets de recherche porteront par exemple sur les trois points suivants :
- Continuer le développement du code IPDCA pour en faire un des algorithmes représentant l'état de l'art dans le domaine.
- La parallélisation partielle ou complète du code de points intérieurs IPDCA pour pouvoir résoudre des problèmes d'optimisation industrielles de taille réelle en des temps raisonnables.
- Continuer le développement du code NIPA pour en faire une alternative fiable aux méthodes de points intérieurs classiques.

Annexe A

IPDCA pour le problème $(\mathcal{P}_{\mathcal{EI}})$

Le but de ce chapitre est de préciser deux points essentiels de l'algorithme IPDCA présenté au Chapitre 3 pour le problème $(\mathcal{P}_{\mathcal{EI}})$. Ces points sont :

– l'intégration de la régularisation DC dans une méthode de points intérieurs pour le problème $(\mathcal{P}_{\mathcal{EI}})$.
– La transformation du système linéaire issu de la linéarisation des conditions nécessaires d'optimalité du premier ordre en un système quasi-défini dont la matrice est fortement factorisable.

Nous rappelons que notre but est de déterminer un point de karush-Khun-Tucker du problème de programmation non linéaire suivant

$$(\mathcal{P}_{\mathcal{EI}}) \begin{cases} \min f(x) & \\ Ax - b & = \ 0, \\ c(x) & \geq \ 0, \\ x \in \mathbb{R}^n, & \end{cases}$$

où $f : \mathbb{R}^n \to \mathbb{R}$ et $c : \mathbb{R}^n \to \mathbb{R}^{m_1}$ sont au moins de classe C^2, et c supposé *concave.* $A \in \mathbb{R}^{m_2 \times n}$ une matrice *surjective*, et $b \in \mathbb{R}^{m_2}$ un vecteur.

La section qui suit sera consacrée à l'intégration de la régularisation DC dans une méthode de points intérieurs, puis dans la dernière partie du chapitre nous présenterons deux types de régularisations pour transformer le système linéaire issu de la linéarisation des conditions nécessaires d'optimalité du premier ordre en un système quasi-défini.

A.1 Méthode de points intérieurs primal-dual

La première étape dans l'approche de méthode de points intérieurs considérée dans ce chapitre consiste à introduire les variables d'écarts pour transformer d'une part les contraintes d'inégalités en contraintes d'égalités et d'autre part prendre en compte le fait que les variables primales x sont libres, c'est-à-dire sans bornes. Pour les gérer nous avons suivi l'approche décrite dans [31] et rappelée au Chapitre 2. Le problème transformé est,

$$\begin{cases} \min f(x) & & \\ Ax-b & = & 0, \\ c(x)-w & = & 0, \\ x-y+z & = & 0, \\ w \geq 0, & y \geq 0 & , z \geq 0. \end{cases}$$

avec $w \in \mathbb{R}_I^m$ et $y, z \in \mathbb{R}^n$. Le problème est par la suite transformé en une suite de problèmes en ajoutant des termes barrière à la fonction f. Le problème transformé s'écrit,

$$(P_\mu) \begin{cases} \min f_\mu(w,x,y,z) & & \\ Ax-b & = & 0, \\ c(x)-w & = & 0, \\ x-y+z & = & 0, \end{cases}$$

où $f_\mu(w,x,y,z) = f(x) - \mu\sum_{i=1}^{m_I} log(w_i) - \mu\sum_{j=1}^{n} log(y_j) - \mu\sum_{j=1}^{n} log(z_j)$.
Le terme logarithmique barrière est utilisé pour assurer implicitement les inégalités $w \geq 0$, $y \geq 0$, et $z \geq 0$.
Nous supposons pour toute la suite disposer d'une décomposition DC de la fonction objectif $f = g - h$, avec g et h convexes.
Posons

$$\begin{aligned} g_\mu(w,x,y,z) &= g(x) - \mu\sum_{i=1}^{m_I} log(w_i) - \mu\sum_{j=1}^{n} log(y_j) - \mu\sum_{j=1}^{n} log(z_j), \\ h_\mu(w,x,y,z) &= h(x). \end{aligned}$$

on suppose à l'itération k de disposer de x_k entre autres. Comme au Chapitre 3 on linéarise la partie concave de la décomposition de l'objectif f au point x_k. Puis on résoudra approximativement à chaque itération le problème,

$$(\mathcal{DC}_\mu) \begin{cases} \min g_\mu(w,x,y,z) & - & \nabla^T h(x_k)x \\ Ax-b & = & 0, \\ c(x)-w & = & 0, \\ x-y+z & = & 0. \end{cases}$$

Soit

$$L_\mu = g_\mu(w,x,y,z) - \nabla^T h(x_k)x - \lambda_E^T(Ax-b) - \lambda_I^T(c(x)-w) - s^T(x-y+z) \qquad (A.1)$$

le Lagrangien associé au problème ($\mathcal{DC}_\mu$). Les conditions d'optimalité du premier ordre s'écrivent,

$$(KKT_\mu) \begin{cases} \nabla_x L_\mu & = & \nabla_x g_\mu(w,x,y,z) - \nabla h(x_k) - A^T\lambda_E - B(x)^T\lambda_I - s & = & 0, \\ \nabla_w L_\mu & = & -\mu W^{-1}e + \lambda_I & = & 0, \\ \nabla_y L_\mu & = & -\mu Y^{-1}e + s & = & 0, \\ \nabla_z L_\mu & = & -\mu Z^{-1}e - s & = & 0, \\ \nabla_{\lambda_E} L_\mu & = & -Ax + b & = & 0, \\ \nabla_{\lambda_I} L_\mu & = & -c(x) + w & = & 0, \\ \nabla_s L_\mu & = & -x + y - z & = & 0, \end{cases}$$

avec $B(x) = \nabla c(x)$, $W = diag(w_i : i = 1, 2, \cdots, m_I)$., $Z = diag(z_i : i = 1, 2, \cdots, n)$, $Y = diag(y_i : i = 1, 2, \cdots, n)$.
En posant $v = \mu Z^{-1} e$ et en remarquant que $\nabla_x g_\mu(w, x, y, z) = \nabla g(x)$, après avoir prémultiplié la seconde et la troisième équation de (KKT_μ) par W et Y respectivement.
En posant

$$F_\mu(w, x, y, z; \lambda_E, \lambda_I, v, s) = \begin{pmatrix} \nabla g(x) - \nabla h(x_k) - A^T \lambda_E - B(x)^T \lambda_I - s \\ -\mu e + W \lambda_I \\ -\mu e + Y s \\ -\mu e + Z v \\ -Ax + b \\ -c(x) + w \\ -v - s \\ -x + y - z \end{pmatrix}$$

on doit résoudre le système d'équations non-linéaires,

$$F_\mu(w, x, y, z; \lambda_E, \lambda_I, v, s) = 0. \tag{A.2}$$

Remarquons qu'au point $\Pi_k = (x_k, w_k, y_k, z_k; \lambda_{E,k}, \lambda_{I,k}, s_k, v_k,)$ ce système traduit les conditions d'optimalité du problème (P_E) en Π_k, car on note que $\nabla g(x_k) - \nabla h(x_k) = \nabla f(x_k)$.
On utilise la méthode de Newton pour résoudre le système non linéaire (A.2). A la k-ème itération et pour μ fixé le système linéaire à résoudre est,

$$J(\Pi_k) \triangle \Pi = -F_\mu(\Pi_k), \tag{A.3}$$

où $J(\Pi_k)$ est la matrice Jacobienne de $F_\mu(\Pi_k)$ et $\triangle \Pi = (\triangle x, \triangle w, \triangle y,$
$\triangle z; \triangle \lambda_E, \triangle \lambda_I, \triangle s, \triangle v)$. Le calcul explicite de la matrice $J(\Pi_k)$ donne,

$$J(\Pi_k) = \begin{pmatrix} G_k & 0 & 0 & 0 & -A^T & -B_k^T & -I_n & 0 \\ 0 & \Lambda_{I,k} & 0 & 0 & 0 & W_k & 0 & 0 \\ 0 & 0 & S_k & 0 & 0 & 0 & Y_k & 0 \\ 0 & 0 & 0 & V_k & 0 & 0 & 0 & Z_k \\ -A & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -B_k & I_{m_I} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -I_n & -I_n \\ -I_n & 0 & I_n & -I_n & 0 & 0 & 0 & 0 \end{pmatrix} \tag{A.4}$$

avec $G_k = G(x_k)$, le hessien du Lagrangien au point x_k, $B_k = B(x_k)$, $S_k = diag(s_k^i : i = 1, 2, \cdots, n)$, $\Lambda_{I,k} = diag(\lambda_{I,k}^i : i = 1, 2, \cdots, n)$, $V_k = diag(v_k^i : i = 1, 2, \cdots, n)$ et I_n, I_{m_I} et par la suite I_{m_E} les matrices identité d'ordre n, m_I et m_E respectivement.
Par souci de clarté dans notre exposé nous allons momentanément abandonner l'indexation par k.

On pose,

$$\begin{array}{rcl}
\sigma & = & \nabla f(x) - A^T\lambda_E - B(x)^T\lambda_I - s, \\
\gamma_1 & = & \mu e - W\lambda_I, \\
\gamma_2 & = & \mu e - Ys, \\
\gamma_3 & = & \mu e - Zv, \\
\rho_1 & = & Ax - b, \\
\rho_2 & = & c(x) - w, \\
\beta_1 & = & v + s, \\
\beta_2 & = & x - y + z,
\end{array}$$

alors le système linéaire s'écrit,

$$\begin{pmatrix}
G(x) & 0 & 0 & 0 & -A^T & -B(x)^T & -I_n & 0 \\
0 & \Lambda_I & 0 & 0 & 0 & W & 0 & 0 \\
0 & 0 & S & 0 & 0 & 0 & Y & 0 \\
0 & 0 & 0 & V & 0 & 0 & 0 & Z \\
-A & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-B(x) & I_{m_I} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -I_n & -I_n \\
-I_n & 0 & I_n & -I_n & 0 & 0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
\triangle x \\ \triangle w \\ \triangle y \\ \triangle z \\ \triangle\lambda_E \\ \triangle\lambda_I \\ \triangle s \\ \triangle v
\end{pmatrix}
=
\begin{pmatrix}
-\sigma \\ \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \rho_1 \\ \rho_2 \\ \beta_1 \\ \beta_2
\end{pmatrix}.$$

Par élimination successive on a de la septième équation,

$$\triangle v = -\triangle s - \beta_1$$

et dans la quatrième équation du système on remplace $\triangle v$ par cette valeur on obtient,

$$\triangle z = V^{-1}Z\triangle s + \beta_1 + V^{-1}Z\beta_1 + V^{-1}\gamma_3$$

d'autre part d'après la dernière équation du système,

$$\triangle y = \triangle x + \triangle z + \beta_2$$

donc

$$\triangle y = \triangle x + V^{-1}Z\triangle s + \beta_1 + V^{-1}Z\beta_1 + V^{-1}\gamma_3 + \beta_2.$$

En posant $E = (YS^{-1} + ZV^{-1})^{-1}$, on a,

$$\triangle s = -E\triangle x + E[V^{-1}(Z\beta_1 + \gamma_3) + \beta_1 + S^{-1}\gamma_2].$$

En notant également que la sixième équation du système nous donne,

$$\triangle w = -D\triangle\lambda_I + \Lambda_I^{-1}\gamma_1,$$

où $D = \Lambda_I^{-1}W$. On aboutit après prémultiplication des équations par -1 le système réduit,

$$\begin{pmatrix} -(G(x)+E) & A^T & B(x)^T \\ A & 0 & 0 \\ B(x) & 0 & D \end{pmatrix} \begin{pmatrix} \triangle x \\ \triangle \lambda_E \\ \triangle \lambda_I \end{pmatrix} = \begin{pmatrix} \Sigma \\ -\rho_1 \\ -\rho_2 + \Lambda_I^{-1}\gamma_1 \end{pmatrix}. \tag{A.5}$$

avec $\Sigma = \sigma - E[V^{-1}(Z\beta_1 + \gamma_3) + \beta_1 + S^{-1}\gamma_2]$.
Notons H la matrice intervenant dans ce système. A ce niveau nous avons appliqué la régularisation primale pour gérer la non-convexité et pour gérer les variables libres x. Comme nous pouvons le voir cette régularisation ne suffit pas à rendre le système linéaire réduit (A.5) quasi-défini, car le bloc 2×2 de la matrice H est nul.
Pour rendre la matrice H quasi-définie nous examinerons deux types de régularisations dans la Section A.2.

Pour le moment nous supposons la direction de Newton $\triangle\Pi_k$ calculée, nous pouvons déterminer l'itéré suivant,

$$\Pi_{k+1} = \Pi_k + \Upsilon_k \triangle \Pi_k,$$

où $\Upsilon_k = diag\{\alpha_{x_k} I_n, \alpha_{w_k} I_{m_I}, \alpha_{y_k} I_n, \alpha_{z_k} I_n, \alpha_{\lambda_{E,k}} I_{m_E}, \alpha_{\lambda_{I,k}} I_{m_I}, \alpha_{s_k} I_n, \alpha_{v_k} I_n, \}$. Les longueurs de pas $\alpha_{x_k}, \alpha_{w_k}, \alpha_{y_k}, \alpha_{z_k}, \alpha_{\lambda_{E,k}}, \alpha_{\lambda_{I,k}}, \alpha_{s_k}, \alpha_{v_k}$ sont déterminées dans $(0, 1]$ et pourraient éventuellement être égales ou différentes les unes des autres. De plus puisque les variables d'écart, w_k, y_k, z_k et les variables duales $\lambda_{I,k}$, v_k, s_k, sont toutes positives à la solution du problème barrière $(\mathcal{DC}_\mu)$, cette propriété est maintenue pour toutes les itérations. Pour cela on définit,

$$\begin{aligned}
\alpha_{w_k}^{\max} &= \gamma \max_{1 \le j \le m_I} \{-\tfrac{w_k^{(j)}}{\triangle w_k^{(j)}} : \triangle w_k^{(j)} < 0\}, \\
\alpha_{y_k}^{\max} &= \gamma \max_{1 \le j \le n} \{-\tfrac{y_k^{(j)}}{\triangle y_k^{(j)}} : \triangle y_k^{(j)} < 0\}, \\
\alpha_{z_k}^{\max} &= \gamma \max_{1 \le j \le n} \{-\tfrac{z_k^{(j)}}{\triangle z_k^{(j)}} : \triangle z_k^{(j)} < 0\}, \\
\alpha_{\lambda_{I,k}}^{\max} &= \gamma \max_{1 \le j \le m_I} \{-\tfrac{\lambda_{I,k}^{(j)}}{\triangle \lambda_{I,k}^{(j)}} : \triangle \lambda_{I,k}^{(j)} < 0\}, \\
\alpha_{s_k}^{\max} &= \gamma \max_{1 \le j \le n} \{-\tfrac{s_k^{(j)}}{\triangle s_k^{(j)}} : \triangle s_k^{(j)} < 0\}, \\
\alpha_{v_k}^{\max} &= \gamma \max_{1 \le j \le n} \{-\tfrac{v_k^{(j)}}{\triangle v_k^{(j)}} : \triangle v_k^{(j)} < 0\},
\end{aligned}$$

qui représentent les longueurs de pas maximales autorisées et qui garantiraient la positivité des variables respectives, avec $\gamma \in (0, 1]$.
La distance du point Π_k au chemin central est mesurée en utilisant la norme euclidienne des conditions d'optimalité perturbées, soit $\| F_\mu(\Pi_k) \|$ une fois que celle-ci est plus petite qu'un ε_μ préfixé ; on fait par la suite décroître le paramètre μ et le processus est répété jusqu'à ce que μ devienne zéro.

A.2 Du système indéfini au système quasi-défini

A.2.1 Régularisation par ajout de variables d'écart

La matrice intervenant dans le système (A.5) n'est pas quasi-définie. Pour la rendre quasi-définie, Vanderbei [37, 34] ajoute des variables d'écart p aux contraintes d'égalité du problème $(\mathcal{P})$ et de nouvelles contraintes $p + t = 0$, de sorte que le problème $(\mathcal{DC}_\mu)$ serait,

$$(P_{Vb}) \begin{cases} \min \hat{g}_\mu(w,x,y,z,p,q) & - & \nabla^T h(x_k)x \\ Ax - b - p & = & 0, \\ p + q & = & 0, \\ c(x) - w & = & 0, \\ x - y + z & = & 0, \end{cases} \tag{A.6}$$

où $\hat{g}_\mu(w,x,y,z,p,q) = g_\mu(w,x,y,z) - \mu\sum_{j=1}^{m_E} log(p_j) - \mu\sum_{j=1}^{m_E} log(q_j)$.

Soit

$$L_\mu = \hat{g}_\mu - \nabla^T h(x_k)x - \lambda_E^T(Ax - b - p) - \lambda_I^T(c(x) - w) - s^T(x - y + z) - u^T(p + q)$$

le Lagrangien associé au problème (P_{Vb}) avec $\hat{g}_\mu = \hat{g}_\mu(w,x,y,z,p,q)$.
Les conditions nécessaires d'optimalité du premier ordre s'écrivent,

$$\begin{cases} \nabla_x L_\mu & = & \nabla_x g_\mu(w,x,y,z) - \nabla h(x_k) - A^T\lambda_E - B(x)^T\lambda_I - s & = & 0, \\ \nabla_w L_\mu & = & -\mu W^{-1}e + \lambda_I & = & 0, \\ \nabla_y L_\mu & = & -\mu Y^{-1}e + s & = & 0, \\ \nabla_z L_\mu & = & -\mu Z^{-1}e - s & = & 0, \\ \nabla_p L_\mu & = & \lambda_E - u - \mu P^{-1}e & = & 0, \\ \nabla_q L_\mu & = & -\mu Q^{-1}e - u & = & 0, \\ \nabla_{\lambda_E} L_\mu & = & -Ax + b + p & = & 0, \\ \nabla_{\lambda_I} L_\mu & = & -c(x) + w & = & 0, \\ \nabla_s L_\mu & = & -x + y - z & = & 0, \\ \nabla_u L_\mu & = & -p - q & = & 0, \end{cases}$$

avec $P = diag(p_i : i = 1, 2, \cdots, m_E)$ et $Q = diag(q_i : i = 1, 2, \cdots, m_E)$.
On pose
$\Pi = (x, w, y, z, p, q, \lambda_E, \lambda_I, s, v, u, l, t)$,
$\triangle\Pi = (\triangle x, \triangle w, \triangle y, \triangle z, \triangle p, \triangle q, \triangle\lambda_E, \triangle\lambda_I, \triangle s, \triangle v, \triangle u, \triangle l, \triangle t)$,
$l = \mu Q^{-1}e$ et $t = \mu P^{-1}e$,
et en adoptant les notations du paragraphe précédent on a

$$\mathcal{F}_\mu(\Pi) = \begin{pmatrix} \nabla_x g_\mu(w,x,y,z,p,q) - \nabla h(x_k) - A^T\lambda_E - B(x)^T\lambda_I - s \\ -\mu e + W\lambda_I \\ -\mu e + Ys \\ -\mu e + Zv \\ -\mu e + Pt \\ -\mu e + Ql \\ \lambda_E - u - t \\ -l - u \\ -p - q \\ -Ax + b + p \\ -c(x) + w \\ -s - v \\ -x + y - z \end{pmatrix}$$

Résoudre $\mathcal{F}_\mu(\Pi) = 0$ en utilisant la méthode de Newton conduit à résoudre une suite de systèmes linéaires,

$$\nabla\mathcal{F}_\mu(\Pi_k)\triangle\Pi = -\mathcal{F}_\mu(\Pi_k), \tag{A.7}$$

Le calcul explicite de la matrice $\nabla\mathcal{F}_\mu(\Pi_k)$ est donné au Tableau A.1. Dans ce tableau $L = diag(l_i : i = 1, 2, \cdots, m_E)$ et $T = diag(t_i : i = 1, 2, \cdots, m_E)$.
En adoptant un processus d'élimination similaire à celui de la section précédente on obtient le système linéaire réduit,

$$\begin{pmatrix} -(G(x)+E) & A^T & B(x)^T \\ A & F & 0 \\ B(x) & 0 & D \end{pmatrix} \begin{pmatrix} \triangle x \\ \triangle\lambda_E \\ \triangle\lambda_I \end{pmatrix} = \begin{pmatrix} \Sigma \\ -\Omega \\ -\rho_2 + \Lambda_I^{-1}\gamma_1 \end{pmatrix}. \tag{A.8}$$

avec $F = diag(\frac{q_i p_i}{p_i t_i + q_i l_i} : i = 1, 2, \cdots, m_E)$, matrice diagonale définie positive et

$$\Omega = F(\rho_1 + (\gamma_6 - \gamma_5 + P^{-1}\gamma_4) - Q^{-1}\gamma_7 - (Q^{-1}T)\rho_3),$$

où

$$\begin{aligned} \gamma_4 &= \mu e - Pt, \\ \gamma_5 &= u + l, \\ \gamma_6 &= -\lambda_E + u + t, \\ \gamma_7 &= \mu e - Ql, \\ \rho_3 &= p + q. \end{aligned}$$

La matrice intervenant dans le système (A.8) est quasi-définie.

A.2.2 Régularisation par ajout de termes constants ou variables

Gill et *al.*[132] Saunders et *al.* [79, 80] ont utilisé une approche différente de la précédente pour rendre le système (A.5) quasi-défini. Dans leur approche une perturbation $-\delta^2 I_E$ est

ajoutée à la matrice $J(\Pi)$ comme suit :

$$\begin{pmatrix} G(x) & 0 & 0 & 0 & -A^T & -B(x)^T & -I_n & 0 \\ 0 & \Lambda_I & 0 & 0 & 0 & W & 0 & 0 \\ 0 & 0 & S & 0 & 0 & 0 & Y & 0 \\ 0 & 0 & 0 & V & 0 & 0 & 0 & Z \\ -A & 0 & 0 & 0 & -\delta^2 I_E & 0 & 0 & 0 \\ -B(x) & I_{m_I} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -I_n & -I_n \\ -I_n & 0 & I_n & -I_n & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \triangle x \\ \triangle w \\ \triangle y \\ \triangle z \\ \triangle \lambda_E \\ \triangle \lambda_I \\ \triangle s \\ \triangle v \end{pmatrix} = \begin{pmatrix} -\sigma \\ \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ -\Omega \\ \rho_2 \\ \beta_1 \\ \beta_2 \end{pmatrix}.$$

avec $\Omega = -Ax + b$.
Toujours en adoptant une réduction similaire à celle de la section précédente on obtient le système réduit

$$\begin{pmatrix} -(G(x)+E) & A^T & B(x)^T \\ A & \delta^2 I_E & 0 \\ B(x) & 0 & D \end{pmatrix} \begin{pmatrix} \triangle x \\ \triangle \lambda_E \\ \triangle \lambda_I \end{pmatrix} = \begin{pmatrix} \Sigma \\ \Omega \\ -\rho_2 + \Lambda_I^{-1}\gamma_1 \end{pmatrix}. \tag{A.9}$$

Algorithmiquement l'ajout de variables d'écart avec zéro comme bornes supérieures et inférieures à toutes les lignes de la matrice A comme le fait Vanderbei produit en grande partie l'effet escompté par la régularisation $\delta^2 I_E$, sans pour autant perturber le problème. Cependant cet effet diminue lorsqu'on approche de la solution.
Une analyse de l'erreur relative faite sur la solution en utilisant une telle régularisation a été faite dans [79].
Dans [101] une régularisation semblable est utilisée. Ici la matrice $(G(x)+E)$ est remplacée par la matrice $(G(x)+E)+R_p$ et la matrice $\delta^2 I_E$ par une matrice R_d^1 de sorte que la matrice du système (A.5) devient,

$$M = \begin{pmatrix} -(G(x)+E)+R_p) & A^T & B(x)^T \\ A & R_d^1 & 0 \\ B(x) & 0 & D \end{pmatrix}$$

avec R_p et R_d^1 des matrices diagonales choisies dynamiquement pendant la factorisation de Cholesky de la matrice M. Ici la seule modification apportée au schéma classique de la factorisation de Cholesky est l'ajout d'une analyse des éléments de la matrice candidats à être pris comme pivot. L'idée est la suivante, une fois qu'un pivot m_{ii} est choisi, son signe est déterminé et il est vérifié si la valeur absolue dudit pivot est trop petite. Si tel est le cas, le pivot est remplacé par $m_{ii} - r_p^i$ si le pivot est négatif, ou par $m_{ii} + r_d^{1,i}$ s'il est positif. Les candidats m_{ii} à être pris comme pivot seront considérés comme étant trop petits si $|m_{ii}| < \epsilon h_{\max}$, où $h_{\max}$ est le plus grand élément diagonal de la matrice H et ϵ la précision relative de la machine.
Dans [101] des résultats comparatifs avec LOQO [36] en utilisant cette approche sont reportés et montrent la supériorité de cette approche sur des problèmes quadratiques convexes. L'attrait majeur de cette approche est d'utiliser une très petite régularisation lorsqu'un pivot est

accepté et une plus grande lorsqu'un pivot très petit est rencontré. Ceci permet de perturber de la moindre manière possible le problème car la régularisation ne se concentre que sur les pivots potentiellement instables.

$$\nabla \mathcal{F}_\mu(\Pi_k) = \begin{pmatrix}
G(x) & 0 & 0 & 0 & 0 & 0 & -A^T & -B(x)^T & -I_n & 0 & 0 & 0 & 0 \\
0 & \Lambda_I & 0 & 0 & 0 & 0 & 0 & W & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & S & 0 & 0 & 0 & 0 & 0 & Y & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & V & 0 & 0 & 0 & 0 & 0 & Z & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & P & 0 & 0 & 0 & 0 & 0 & 0 & 0 & T \\
0 & 0 & 0 & 0 & 0 & Q & 0 & 0 & 0 & 0 & 0 & L & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & I_{m_E} & 0 & 0 & 0 & -I_{m_E} & 0 & -I_{m_E} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -I_{m_E} & -I_{m_E} & 0 \\
0 & 0 & 0 & 0 & -I_{m_E} & -I_{m_E} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-A & 0 & 0 & 0 & I_{m_E} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-B(x) & I_{m_I} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -I_n & -I_n & 0 & 0 & 0 \\
-I_n & 0 & I_n & -I_n & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

Tab. A.1 – Calcul explicite de $\nabla \mathcal{F}_\mu(\Pi_k)$

Annexe B

Large-scale nonlinear programming and lower bound direct method in engineering applications : A combined interior point and DC programming approach

The first part of this chapter that treats the formulation of the problem as a nonlinear optimization problem is from [9, 10].

B.1 Introduction

To determine the ultimate loads for mechanical structures beyond the range of elasticity is one of the most important design tasks in civil and mechanical engineering. The two major reasons for this are : the industrial need for lightweight, raw material saving structures and structural elements ; the assessment of the critical and post-critical behaviour of structures, even if their service conditions are confined to the elastic domain of material behaviour. This task is particularly difficult in case when the considered structure is subjected to variable loading and where the evolution of the loads as function of time is not precisely known. From mathematical point of view this leads to ill-posed problems whose handling by conventional step-by-step calculation methods is not immediate, if possible at all. Nevertheless, for most of real-life problems, the assumption of precisely known loading histories is simply unrealistic. Direct method fills this gap [1, 2]. For its application, only the form of the domain of generalised loads has to be known and no information is required about the evolution of loads as function of time within this domain. The direct method, namely limit and shakedown analysis, leads to a problem of mathematical programming in conjunction with finite element methods which requires a large amount of computer memory if other than one or two-dimensional structures are studied [3, 4]. Furthermore, for nonlinear yield conditions

(e.g. von Mises criteria), the solution of the respective nonlinear optimisation problem often requires highly iterative procedures and is therefore very time-consuming. This results from the fact, that the nonlinear programming approaches imply adopting solution schemes based more on purely mathematical considerations than on the physics problem, thus often becoming unnecessarily expensive. In this paper, we show how to overcome this problem by using a software package for solving large-scale nonlinear optimisation problems. The methodology will be shown by a simple example.

B.2 General considerations and definitions

We consider the behaviour of a three-dimensional elastic-perfectly plastic structure or body $\mathcal{B}$ of finite volume Ω with a sufficiently smooth surface $\partial\Omega$ consisting of the disjoint parts $\partial\Omega_p$ and $\partial\Omega_u$, where statical and kinematical boundary conditions, respectively, are prescribed $\Omega = \partial\Omega_p \cup \partial\Omega_u$; $\emptyset = \partial\Omega_p \cap \partial\Omega_u$. The body is subjected to the quasi-statically varying external agencies $P(x,t) \in \mathcal{L}$ at time t consisting of body forces $f^*(x,t)$ in Ω, surface tractions $p^*(x,t)$ on $\partial\Omega_p$ and given displacements $u^*(x,t)$ on $\partial\Omega_u$.
According to the restriction to geometrically linear theory, the total strains rates can be split into purely elastic and purely plastic parts, respectively

$$\dot{\epsilon} = \dot{\epsilon}^e + \dot{\epsilon}^p \tag{B.1}$$

We assume the validity of the normality rule for plastic flow, such that

$$\dot{\epsilon}^p \in \partial\phi(\sigma) \tag{B.2}$$

where $\partial\phi(\sigma)$ denotes the sub-gradients of the plastic potential $\phi(\sigma)$ is the indicator function of a convex elastic domain $\mathcal{C}$ of all plastically admissible stress states

$$\sigma \in \mathcal{C} \text{ such that } \mathcal{C} = \{\sigma / F(\sigma) = \left(3/2(\sigma^D : \sigma^D)\right)^{1/2} - \sigma_Y \leq 0\} \tag{B.3}$$

Here, it is assumed that the yield function $F(\sigma)$ is of von Mises type, where σ^D denotes the deviatoric part of stress tensor ($\sigma^D = \sigma - \sigma_H I$) and σ_Y the uniaxial yield stress.

B.3 Formulation of the lower bound direct method

The lower bound theorem of shakedown can be expressed as follows [1] :
The body $\mathcal{B}$ shakes down with respect to the given loading $P(x,t) \in \mathcal{L}$ if there exists a safety factor $\alpha > 1$ and a time-independent field of residual stresses $\sigma^r(x)$ such that its superposition with the purely elastic stresses σ^E does not exceed the yield condition for any time $\alpha > 0$

$$F(\alpha\sigma^E(\mathbf{x},t) + \sigma^r(\mathbf{x})) \leq 0, \quad \forall \mathbf{x} \in \Omega \tag{B.4}$$

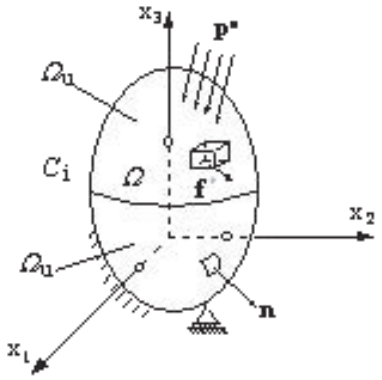


FIG. B.1 – Structure or body $\mathcal{B}$.

The field of purely elastic stresses satisfies the following system of equations

$$\text{Div}\,\sigma^E = -f^* \qquad \text{in } \Omega \tag{B.5}$$

$$n \cdot \sigma^E = p^* \qquad \text{on } \partial\Omega_p \tag{B.6}$$

$$u^E = u^* \qquad \text{on } \partial\Omega_u \tag{B.7}$$

with

$$\epsilon^E = \frac{1}{2}\big(\nabla(u^E) + \nabla(u^E)^T\big) \tag{B.8}$$

$$\epsilon^E = E^{-1} : \sigma^E \tag{B.9}$$

and the field of residual stresses satisfies

$$\text{Div}\,\sigma^r = 0 \qquad \text{in } \Omega \tag{B.10}$$

$$n \cdot \sigma^r = 0 \qquad \text{on } \partial\Omega_p \tag{B.11}$$

where n is the outward normal vector to $\partial\Omega_p$.

B.4 Discrete formulation of lower bound direct method

Any discrete version of the lower bound direct method presented above preserves the relevant bounding properties if the following conditions are satisfied simultaneously :
(i) the solution of the purely elastic stresses (B.5-B.6) is exact ;
(ii) the residual stress field satisfies point-wise the homogeneous equilibrium equations (B.10-B.11)
(iii) the yield condition (B.4) is satisfied everywhere in Ω
In order to make the numerical approach as general as possible, we use here the displacement method. In this case the well-known displacement element formulations involving e.g. isoparametric elements can be applied. For that purpose it is necessary to transform the statical equations (B.5-B.11) from their local form into the equivalent global form (B.5-B.6).

B.4.1 Discretisation of the purely elastic stresses

To calculate the purely elastic stresses σ^E, we use the virtual work principle combined with the finite element discretisation with test functions for the displacement fields. Then, the purely elastic stresses σ^E are in equilibrium with body forces f^* and surface tractions p^* if the following equality holds

$$\delta U_{int} = \delta U_{ext} \tag{B.12}$$

or

$$\int_\Omega \{\delta\epsilon^E(x)\}^T\{\sigma^E(x)\}d\Omega = \int_{\partial\Omega p} \{\delta u^E\}^T\{p^*\}dS + \int_\Omega \{\delta u^E\}^T\{f^*\}d\Omega \tag{B.13}$$

for any virtual displacement field δu^E and any virtual strain field $\delta\epsilon^E$ satisfying the compatibility condition (B.8). The virtual displacement field δu^E of each element e is approximated according to

$$\{\delta u^E\} = \sum_{k=1}^{NKE} N_k u_k^e \tag{B.14}$$

where N_k and δu_k^e denote the k-th shape function matrix and the vector of virtual displacements of the k-th node of the element e, respectively. NKE denotes the total number of nodes of each element. The virtual strain field $\delta\epsilon^E(x)$ is derived by substitution of eqn. (B.14) into (B.8), such that

$$\{\delta\epsilon^E(x)\} = \sum_{k=1}^{NKE} B_k(x)\delta u_k^e \tag{B.15}$$

where $[B]$ is the compatibility matrix depending on the coordinates. The integration of (B15) has to be carried out over all Gaussian points NGE in the considered element, where the index i refers to the i-th Gaussian point. The corresponding coordinate vector shall be denoted by x_i, i.e.

$$\begin{aligned}\int_\Omega \{\delta\epsilon^E(x)\}^T\{\sigma^E(x)\}d\Omega &= \{\delta u^E\}^T\Big\{\sum_{i=1}^{NGE} w_i|J|_i[B(x_i)]^T E[B(x_i)]\Big\}\{u^e\} \\ &= \{\delta u^E\}^T[K]\{u^e\} \\ &= \{\delta u^E\}^T F\end{aligned} \tag{B.16}$$

where $\{F\}$ denotes the vector of nodal forces, w_i the weighting factors, $|J|_i$ the determinant of the Jacobian matrix and $[K]$ the stiffness matrix.
This integral leads for the i-th Gaussian point to

$$\{\sigma^E(x_i)\} = [E][B(x_i)]\{u^e\} \tag{B.17}$$

B.4.2 Discretisation of the residual stress field

Analogously, the field of residual stresses can be determined by

$$\int_{\Omega} \{\delta\epsilon\}^T \{\sigma^r\} d\Omega = 0. \tag{B.18}$$

By introducing a vector form for the strain tensor ϵ, the corresponding virtual strains $\delta\epsilon$ are given in each element "e" by

$$\{\delta\epsilon\} = \sum_{k=1}^{NKE} B_k \delta u_k^e \tag{B.19}$$

The shape functions of the considered element are the same as for the determination of the purely elastic stresses. Using this relation and introducing the unknown residual stress vector σ_i^r at each Gaussian point i, the equilibrium condition (B.18) is integrated numerically by using the well-known Gauss-Legendre technique. The integration has to be carried out over all Gaussian points NGE with their weighting factors w_i in the considered element "e"

$$\int_{\Omega} \{\delta\epsilon\}^T \{\sigma^r\} d\Omega = \sum_{i=1}^{NGE} w_i |J|_i \Big[\sum_{k=1}^{NKE} B_k \delta u_k^e \Big] \sigma_i^r. \tag{B.20}$$

By summation of the contributions of all elements and by variation of the virtual node-displacements with regard to the boundary conditions, one finally gets the linear system of equations [5, 6]

$$\sum_{i=1}^{NG} C_i \sigma_i^r = [C]\{\sigma^r\} = \{0\} \tag{B.21}$$

where NG denotes the total number of Gaussian points of the reference body, $[C]$ is a constant equilibrium matrix, uniquely defined by the discretised system and the boundary conditions and $\{\sigma^r\}$ is the global residual stress vector of the discretised reference body.

B.4.3 Discretisation of the time variable

Up to now, no restrictions have been made to the load domain $\mathcal{L}$. Thus $\mathcal{L}$ can be of arbitrary form. However in many practical cases the number of independent loads is restricted, each varying between some given bounds. If the number of such independent loads is n, then the load domain is defined by an n-dimensional polyhedron

$$\mathcal{L} = \Big\{ P/P(x,t) = \sum_{j=1}^{n} \mu_j(t) P_j(x),\ \mu_j \in [\mu_j^-, \mu_j^+] \Big\} \tag{B.22}$$

where P is the vector of generalised loads, j are scalar multipliers with upper and lower bounds μ_j^+ and μ_j^-, respectively. P_j represents n fixed and independent generalised loads (e.g. body forces, surface tractions, prescribed boundary displacements, temperature changes or

combinations of them). For subsequent considerations the corners of the polyhedron (load domain $\mathcal{L}$) are numbered by the index j, such that $j = 1, \ldots, NV$, where NV denotes the total number of corners. The loads, which correspond to each corner of $\mathcal{L}$ are characterised symbolically by P_j. In view of the convexity of the yield function F (B.4) and due to the above assumption on the load domain $\mathcal{L}$ it can be shown that [7]

$$F(\alpha\sigma^E(x,t) + \sigma^r(x)) \leq 0 \tag{B.23}$$

is fulfilled at any time t, if

$$F(\alpha\sigma_i^E(P_j) + \sigma_i^r) \leq 0 \tag{B.24}$$

holds for all $j \in [1, NV]$ and for all $i \in [1, NG]$. Then the discretised formulation of the static shakedown theorem for the determination of the shakedown loading factor is given by

$$\alpha_{SD} = \max_{\sigma_i^r} \alpha \tag{B.25}$$

with the subsidiary conditions

$$\sum_{i=1}^{NG} C_i\sigma_i^r = \{0\} \tag{B.26}$$

$$F(\alpha\sigma_i^E(P_j) + \sigma_i^r) \leq 0 \tag{B.27}$$

$$i \in [1, NG] \text{ and } j \in [1, NV]$$

The yield criterion has to be fulfilled at Gaussian points $i \in [1, NG]$ and in each load corner $j \in [1, NV]$, where $NV = 2^n$. The number of unknowns of the optimisation problem (B.25)-(B.27) is $N = 1 + NG \times NS$ corresponding to α and $\{\sigma^r\}$. The number of constraints is $NV \times NG + NK$, where NS is the dimension of the stress vector at each Gaussian point and NK denotes the degrees of freedom of displacements of the discretised body. This problem can not be solved efficiently by classical algorithms of optimisation because for engineering problems the number of unknowns is in general very high. To overcome the time-consuming is to use a software package for solving large-scale non linear optimisation problems.

B.5 Solving large-scale nonlinear optimisation by IPDCA

B.5.1 New equivalent simpler convex program

We will present in the last section some computational results performed by a primal-dual the interior point with DC regularization algorithm (IPDCA).

To increase the computational performance of the algorithm, we will first reformulate the convex program above into equivalent simpler form. For this end we rewrite this problem as follows,

$$(\mathcal{P})\begin{cases}\max\ \alpha\\ s.t\\ \sum\limits_{r=1}^{NG} C_r X_r = 0\\ (\sigma_{1r}-\sigma_{2r})^2+(\sigma_{2r}-\sigma_{3r})^2+(\sigma_{3r}-\sigma_{1r})^2+6\sigma_{4r}^2+6\sigma_{5r}^2+6\sigma_{6r}^2\le 2a_r^2\\ \sigma_r-\alpha\beta_r-X_r=0,\\ r=1,...,NG.\end{cases}$$

where β_r and $C_r \in \mathcal{M}_{(3NK,6)}(\mathbb{R})$ (the vector space of $(3NK)\times 6$ real matrices) are given, $\sigma_r \in \mathbb{R}^6$, $a_r \in \mathbb{R}$, $X_r := (X_{1r},...,X_{6r}) \in \mathbb{R}^6$ for $r = 1,...,NG$.
The problem has $12NG+1$ variables and $3NK+7NG$ constraints. In practice $NG > NK$ and the values of NK are about several thousands. It then leads to large-scale convex programs. We shall present below an equivalent convex program with less variables and constraints.
Let $T \in \mathcal{M}_{(6,6)}(\mathbb{R})$ be the following nonsingular matrix whose inverse T^{-1} is easily computed by

$$T = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & & & \\ -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & & & \\ -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & & & \\ & & & \frac{1}{\sqrt{6}} & & \\ & & & & \frac{1}{\sqrt{6}} & \\ & & & & & \frac{1}{\sqrt{6}} \end{pmatrix}$$

and

$$T^{-1} = \begin{pmatrix} 1 & -1 & & & & \\ & 1 & -1 & & & \\ 1 & & 1 & & & \\ & & & \sqrt{6} & & \\ & & & & \sqrt{6} & \\ & & & & & \sqrt{6} \end{pmatrix}$$

Set $v = T^{-1}u$ and then $u = Tv$. Making u (resp. v) play the role of σ_r (resp. u_r in $(\mathcal{P})$), the second constraints in $(\mathcal{P})$ becomes

$$u_{1r}^2+u_{2r}^2+(u_{1r}+u_{2r})^2+u_{4r}^2+u_{5r}^2+u_{6r}^2 \le 2a_r^2$$

Since $\sigma_r = Tu_r$, the third constraint in $(\mathcal{P})$ takes the form

$$X_r - Tu_r + \alpha\beta_r = 0. \tag{B.28}$$

We have

$$Tu_r = \sum_{i=1}^{6} T^i u_{ir} = \sum_{\substack{i=1\\ i\neq 3}}^{6} T^i u_{ir} + T^3 u_{3r}, \tag{B.29}$$

where T^i denotes the i^{th} column of the matrix T. On the other hand, it follows from (28) and (29) that :

$$\begin{aligned} C_r X_r &= C_r T u_r - \alpha C_r \beta_r \\ &= C_r(\sum_{\substack{i=1 \\ i\neq 3}}^{6} T^i u_{ir} + T^3 u_{3r}) - \alpha C_r \beta_r. \end{aligned}$$

Consider now the new variables $\overline{u}_r$ in $\mathbb{R}^5$.

$$\bar{u}_r = (\bar{u}_{1r} := u_{1r},\ \bar{u}_{2r} := u_{2r},\ \bar{u}_{3r} := u_{4r},\ \bar{u}_{4r} := u_{5r},\ \bar{u}_{5r} := u_{6r})$$

and the matrix

$$A_r = \left(\begin{array}{ccccc} (C_r T)^1 & (C_r T)^2 & (C_r T)^4 & (C_r T)^5 & (C_r T)^6 \end{array} \right) \in \mathcal{M}_{(3NK,5)}(\mathbb{R}).$$

We can write

$$\sum_{\substack{i=1 \\ i\neq 3}}^{6} (C_r T)^i u_{ir} = A_r \bar{u}_r.$$

Let x be the vector in $\mathbb{R}^{NG}$ whose components x_r are defined by

$$x_r := u_{3r} \text{ for } r = 1, \ldots, NG$$

It follows that

$$C_r T u_r = A_r \bar{u}_r + (C_r T)^3 x_r.$$

In the sequel B is the matrix

$$B = \left(\begin{array}{cccc} (C_1 T)^3 & (C_2 T)^3 & (C_3 T)^3 & \ldots \quad (C_{NG} T)^3 \end{array} \right) \in \mathcal{M}_{(3NK,NG)(\mathbb{R})}.$$

and we set

$$w := \sum_{r=1}^{NG} C_r \beta_r$$

At this point, the problem $(\mathcal{P})$ is equivalently transformed into the following

$$\begin{cases} \max\ \alpha \\ \ s.t \\ \ \sum_{r=1}^{NG} A_r \bar{u}_r + Bx - \alpha w = 0 \\ \ u_{1r}^2 + u_{2r}^2 + (u_{1r} + u_{2r})^2 + u_{4r}^2 + u_{5r}^2 + u_{6r}^2 \leq 2a_r^2. \\ \ r = 1, ..., NG. \end{cases}$$

In the next we shall transform the above convex quadratic constraints into Euclidean ball constraints. For this we first consider the convex quadratic form

$$f(u) = \frac{1}{2}\langle Q\bar{u}, \bar{u}\rangle = u_1^2 + u_2^2 + (u_1 + u_2)^2 + u_4^2 + u_5^2 + u_6^2$$

where $\bar{u} := (u_1, u_2, u_4, u_5, u_6)$ and the matrix Q is given by

$$Q = \begin{pmatrix} 4 & 2 & & & \\ 2 & 4 & & & \\ & & 2 & & \\ & & & 2 & \\ & & & & 2 \end{pmatrix}$$

The Cholesky factorization LL^T of the matrix $\frac{1}{2}Q$ is easily computed by

$$L = \begin{pmatrix} \sqrt{2} & & & & \\ \frac{1}{\sqrt{2}} & \frac{\sqrt{3}}{\sqrt{2}} & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{pmatrix}$$

Hence

$$\begin{aligned} \langle \frac{1}{2} Q\bar{u}, \bar{u} \rangle &= \langle LL^T \bar{u}, \bar{u} \rangle \\ &= \langle L^T \bar{u}, L^T \bar{u} \rangle. \end{aligned}$$

The transformed problem takes then the form

$$(\mathcal{PM}) \begin{cases} \max\ \alpha \\ s.t \\ \sum_{r=1}^{NG} A_r \bar{u}_r + Bx - \alpha w = 0 \\ \| L^T \bar{u}_r \|^2 \leq 2a_r^2. \\ r = 1, ..., NG. \end{cases}$$

Finally , by using the new change of variables $u_r = L^T \bar{u}_r$ and, keeping the same notations A_r and u_r for $A_r L^{-T}$ and $\frac{u_r}{a_r\sqrt{2}}$ respectively (i.e., $A_r := A_r L^{-T}$ and $u_r := \frac{u_r}{a_r\sqrt{2}}$) for notation simplicity, the problem $(\mathcal{PM})$ gets the form

$$(\mathcal{PM}) \begin{cases} \max\ \alpha \\ s.t \\ \sum_{r=1}^{NG} A_r u_r + Bx - \alpha w = 0 \\ \| u_r \|^2 \leq 1. \\ r = 1, ..., NG. \end{cases}$$

This convex program has $6NG + 1$ variables and $3NK + NG$ constraints. In the following section we will describe IPDCA to solve the convex problem $(\mathcal{PM})$

B.5.2 The IPDCA algorithm

IPDCA find a KKT point of the following nonlinear programming problem

$$(\mathcal{P}_{\mathcal{EI}}) \begin{cases} \min f(x) & \\ Ax - b & = & 0, \\ c(x) & \geq & 0, \\ x \in \mathbb{R}^n, & \end{cases}$$

where f and $c : \mathbb{R}^n \to \mathbb{R}^{m_I}$ are two twice continuous differentiable functions and c is supposed to be *concave.* $A \in \mathbb{R}^{m_E \times n}$ a *surjective* matrix, and $b \in \mathbb{R}^{m_E}$ a vector.

We will assume that we have constraints qualification in the sense presented in Section 1.1.4.

The first step in the interior-point approach considered is to add slack variables w to each of the inequality constraints in $(\mathcal{P}_{\mathcal{EI}})$ and to add a linear constraints in order to handle free variable x. The problem $(\mathcal{P}_{\mathcal{EI}})$ is then reformulated to,

$$(\mathcal{P}) \begin{cases} \min f(x) & & \\ Ax - b & = & 0, \\ c(x) - w & = & 0, \\ x - y + z & = & 0, \\ w \geq 0, & y \geq 0 & , z \geq 0. \end{cases}$$

The second step is to consider the problem with the barrier objective function

$$(\mathcal{P}_\mu) \begin{cases} \min \bar{f}_\mu(x, w, y, z) & \\ Ax - b & = & 0, \\ c(x) - w & = & 0, \\ x - y + z & = & 0, \\ w > 0, y > 0, z > 0, & \end{cases}$$

with

$$\bar{f}_\mu(x, w, y, z) = f(x) - \mu \sum_{i=1}^{m_I} log(w_i) - \mu \sum_{j=1}^{n} log(y_j) - \mu \sum_{j=1}^{n} log(z_j).$$

IPDCA requires a DC decomposition of $f = g - h$, where g and h are two convex functions. By linearizing the concave component of the objective function, IPDCA solves approximately the problem,

$$(\mathcal{DC}_k) \begin{cases} \min \bar{g}_k(x) & & \\ Ax - b & = & 0, \\ c(x) - w & = & 0, \\ x - y + z & = & 0, \\ w \geq 0, & y \geq 0 & , z \geq 0. \end{cases}$$

where $\bar{g}_k(x) = g(x) - \nabla^T h(x_k)x$.
The problem is then transformed to a sequence of problems with the logarithmic barrier function :

$$(\mathcal{DC}_\mu)\begin{cases} \min g_\mu(w,x,y,z) & - & \nabla^T h(x_k)x \\ Ax - b & = & 0, \\ c(x) - w & = & 0, \\ x - y + z & = & 0, \end{cases}$$

with

$$\begin{aligned} g_\mu(w,x,y,z) &= g(x) - \mu\sum_{i=1}^{m_I} log(w_i) - \mu\sum_{j=1}^{n} log(y_j) - \mu\sum_{j=1}^{n} log(z_j), \\ h_\mu(w,x,y,z) &= h(x). \end{aligned}$$

B.5.2.1 Step computation

The Lagrangian of $(\mathcal{DC}_\mu)$ is

$$L_\mu = g_\mu(w,x,y,z) - \nabla^T h(x_k)x - \lambda_E^T(Ax - b) - \lambda_I^T(c(x) - w) - s^T(x - y + z) \quad \text{(B.30)}$$

Expressing the first order necessary conditions for $(\mathcal{DC}_\mu)$ yields

$$(KKT_\mu)\begin{cases} \nabla_x L_\mu & = & \nabla_x g_\mu(w,x,y,z) - \nabla h(x_k) - A^T\lambda_E - B(x)^T\lambda_I - s & = & 0, \\ \nabla_w L_\mu & = & -\mu W^{-1}e + \lambda_I & = & 0, \\ \nabla_y L_\mu & = & -\mu Y^{-1}e + s & = & 0, \\ \nabla_z L_\mu & = & -\mu Z^{-1}e - s & = & 0, \\ \nabla_{\lambda_E} L_\mu & = & -Ax + b & = & 0, \\ \nabla_{\lambda_I} L_\mu & = & -c(x) + w & = & 0, \\ \nabla_s L_\mu & = & -x + y - z & = & 0, \end{cases}$$

with $B(x) = \nabla c(x)$, $W = diag(w_i : i = 1, 2, \cdots, m_I)$., $Z = diag(z_i : i = 1, 2, \cdots, n)$, $Y = diag(y_i : i = 1, 2, \cdots, n)$.
Let us set $v = \mu Z^{-1}e$ and since $\nabla_x g_\mu(w,x,y,z) = \nabla g(x)$, after premultiplying the second and the third equation of (KKT_μ) by W and Y respectively.
Let us set

$$F_\mu(w,x,y,z;\lambda_E,\lambda_I,v,s) = \begin{pmatrix} \nabla g(x) - \nabla h(x_k) - A^T\lambda_E - B(x)^T\lambda_I - s \\ -\mu e + W\lambda_I \\ -\mu e + Ys \\ -\mu e + Zv \\ -Ax + b \\ -c(x) + w \\ -v - s \\ -x + y - z \end{pmatrix}$$

We should solve the nonlinear system of equations,

$$F_\mu(w, x, y, z; \lambda_E, \lambda_I, v, s) = 0. \tag{B.31}$$

Note that at the point $\Pi_k = (x_k, w_k, y_k, z_k; \lambda_{E,k}, \lambda_{I,k}, s_k, v_k,)$ this system is the KKT conditions of the problem $(\mathcal{P})$ at the point Π_k, because $\nabla g(x_k) - \nabla h(x_k) = \nabla f(x_k)$.
Using the Newton method to solve the nonlinear system (B.31). At the k-th iteration and for μ fixed, the linear system to solve is,

$$J(\Pi_k)\triangle\Pi = -F_\mu(\Pi_k), \tag{B.32}$$

where $J(\Pi_k)$ is the Jacobian matrix of $F_\mu(\Pi_k)$ and $\triangle\Pi = (\triangle x, \triangle w, \triangle y,$
$\triangle z; \triangle\lambda_E, \triangle\lambda_I, \triangle s, \triangle v)$. With

$$J(\Pi_k) = \begin{pmatrix} G_k & 0 & 0 & 0 & -A^T & -B_k^T & -I_n & 0 \\ 0 & \Lambda_{I,k} & 0 & 0 & 0 & W_k & 0 & 0 \\ 0 & 0 & S_k & 0 & 0 & 0 & Y_k & 0 \\ 0 & 0 & 0 & V_k & 0 & 0 & 0 & Z_k \\ -A & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -B_k & I_{m_I} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -I_n & -I_n \\ -I_n & 0 & I_n & -I_n & 0 & 0 & 0 & 0 \end{pmatrix} \tag{B.33}$$

with $G_k = G(x_k)$, the Hessian of the Lagrangian at x_k, $B_k = B(x_k)$, $S_k = diag(s_k^i : i = 1, 2, \cdots, n)$, $\Lambda_{I,k} = diag(\lambda_{I,k}^i : i = 1, 2, \cdots, n)$, $V_k = diag(v_k^i : i = 1, 2, \cdots, n)$ and I_n, I_{m_I}, I_{m_E} the identity matrices.
For sick of clarity in our presentation we will leave for the moment the indexation with k.
Let us set,

$$\begin{aligned} \sigma &= \nabla f(x) - A^T\lambda_E - B(x)^T\lambda_I - s, \\ \gamma_1 &= \mu e - W\lambda_I, \\ \gamma_2 &= \mu e - Ys, \\ \gamma_3 &= \mu e - Zv, \\ \rho_1 &= Ax - b, \\ \rho_2 &= c(x) - w, \\ \beta_1 &= v + s, \\ \beta_2 &= x - y + z, \end{aligned}$$

then the linear system can be written,

$$\begin{pmatrix} G(x) & 0 & 0 & 0 & -A^T & -B(x)^T & -I_n & 0 \\ 0 & \Lambda_I & 0 & 0 & 0 & W & 0 & 0 \\ 0 & 0 & S & 0 & 0 & 0 & Y & 0 \\ 0 & 0 & 0 & V & 0 & 0 & 0 & Z \\ -A & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -B(x) & I_{m_I} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -I_n & -I_n \\ -I_n & 0 & I_n & -I_n & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \triangle x \\ \triangle w \\ \triangle y \\ \triangle z \\ \triangle\lambda_E \\ \triangle\lambda_I \\ \triangle s \\ \triangle v \end{pmatrix} = \begin{pmatrix} -\sigma \\ \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \rho_1 \\ \rho_2 \\ \beta_1 \\ \beta_2 \end{pmatrix}.$$

By a successive alimination process we have from the seventh equation we have

$$\triangle v = -\triangle s - \beta_1 \tag{B.34}$$

and by replacing $\triangle v$ in the fourth equation we get,

$$\triangle z = V^{-1}Z\triangle s + \beta_1 + V^{-1}Z\beta_1 + V^{-1}\gamma_3 \tag{B.35}$$

the last equation of the system gives,

$$\triangle y = \triangle x + \triangle z + \beta_2 \tag{B.36}$$

so

$$\triangle y = \triangle x + V^{-1}Z\triangle s + \beta_1 + V^{-1}Z\beta_1 + V^{-1}\gamma_3 + \beta_2. \tag{B.37}$$

If we set $E = (YS^{-1} + ZV^{-1})^{-1}$, we have,

$$\triangle s = -E\triangle x + E[V^{-1}(Z\beta_1 + \gamma_3) + \beta_1 + S^{-1}\gamma_2]. \tag{B.38}$$

The sixth equation of the system gives,

$$\triangle w = -D\triangle\lambda_I + \Lambda_I^{-1}\gamma_1, \tag{B.39}$$

where $D = \Lambda_I^{-1}W$. We get after premultiplying the equations by -1 the reduced linear system,

$$\begin{pmatrix} -(G(x)+E) & A^T & B(x)^T \\ A & 0 & 0 \\ B(x) & 0 & D \end{pmatrix} \begin{pmatrix} \triangle x \\ \triangle\lambda_E \\ \triangle\lambda_I \end{pmatrix} = \begin{pmatrix} \Sigma \\ -\rho_1 \\ -\rho_2 + \Lambda_I^{-1}\gamma_1 \end{pmatrix}. \tag{B.40}$$

with $\Sigma = \sigma - E[V^{-1}(Z\beta_1 + \gamma_3) + \beta_1 + S^{-1}\gamma_2]$.
Note that the linear system (B.40) is not quasidefinite because the 2×2 matrix block is zero. To make it quasidefinite we replace this 2×2 matrix block by the perturbation $\delta^2 I_{m_E}$ to get the perturbed linear system,

$$\begin{pmatrix} -(G(x)+E) & A^T & B(x)^T \\ A & \delta^2 I_{m_E} & 0 \\ B(x) & 0 & D \end{pmatrix} \begin{pmatrix} \triangle x \\ \triangle\lambda_E \\ \triangle\lambda_I \end{pmatrix} = \begin{pmatrix} \Sigma \\ -\bar{\rho}_1 \\ -\rho_2 + \Lambda_I^{-1}\gamma_1 \end{pmatrix}, \tag{B.41}$$

with $\bar{\rho}_1 = Ax - b + \delta^2(\lambda_E - \lambda_{E,k})$.
We note that the right hand side of this equation is the KKT conditions of the *dual regularized problem*,

$$(\mathcal{DR}_k)\begin{cases} \max f_\mu(x) + \lambda_E^T b - \frac{\delta}{2}(\lambda_E - \lambda_{E,k})^T(\lambda_E - \lambda_{E,k}) - \lambda_I^T c(x) + [\lambda_I^T B(x)]^T x + \nabla f(x)^T x + s^T z \\ \nabla f(x) - A^T\lambda_E - B(x)^T\lambda_I - s = 0, \\ \lambda_E, \lambda_I > 0, s > 0, \end{cases}$$

with

$$f_\mu(x) = f(x) - \mu\sum_{i=1}^{m_I} log(\lambda_{I,i}) - \mu\sum_{j=1}^{n} log(s_j).$$

The new quadratic term present in the objective penalyzes directions that move the current iterate far away from the iterate $\lambda_{E,k}$.
The problem $(\mathcal{DR}_k)$ is the regularized of the problem

$$(\mathcal{D}) \left\{ \begin{array}{l} \max f_\mu(x) + \lambda_E^T b - \lambda_I^T c(x) + [\lambda_I^T B(x)]^T x + \nabla f(x)^T x + s^T z \\ \nabla f(x) - A^T\lambda_E - B(x)^T\lambda_I - s = 0, \\ \lambda_E, \lambda_I > 0, s > 0, \end{array} \right.$$

that is the dual of the problem $(\mathcal{P})$.
Note that by the choice $(\lambda_{E,k}, \lambda_{I,k}, s^k)$ as the current dual triplet at the iteration k, we ensure that the right hand side of the reduce system is not altered, that is,

$$\bar{\rho}_1 = \rho_1 = Ax - b,$$

while we benefit with the regularization $\delta^2 I_{m_E}$; and the linear system that we solve in IPDCA at each iteration is,

$$\left(\begin{array}{ccc} -(G(x)+E) & A^T & B(x)^T \\ A & \delta^2 I_{m_E} & 0 \\ B(x) & 0 & D \end{array} \right) \left(\begin{array}{c} \triangle x \\ \triangle\lambda_E \\ \triangle\lambda_I \end{array} \right) = \left(\begin{array}{c} \Sigma \\ -\rho_1 \\ -\rho_2 + \Lambda_I^{-1}\gamma_1 \end{array} \right). \tag{B.42}$$

The approach that we have adopted here is close to the primal-dual regularization approach presented in [79, 80, 101], the diffence resides in the choice of the primal regularization. In our approach the primal régularization is the DC regularization.

To avoid using definite matrix perturbations, another alternative suggested by R. Vanderbei to treat the equality affine constraints consists in replacing these constraints by,

$$\begin{array}{rcl} Ax + b - t & = & 0 \\ t + v & = & 0, \\ t \geq 0,\ v \geq 0. & & \end{array}$$

With this approach we add linear constraints and two variables, compared to our approach, this approach consumes more memory for a comparable result.

The matrix involved in (B.42) is now quasi-definite and is known to be strongly factorizable to a form of Cholesky-like factorization [37].

To compute the next iterate,

$$\Pi_{k+1} = \Pi_k + \Upsilon_k \triangle\Pi_k,$$

with $\Upsilon_k = diag\{\alpha_{x_k} I_n, \alpha_{w_k} I_{m_I}, \alpha_{y_k} I_n, \alpha_{z_k} I_n, \alpha_{\lambda_{E,k}} I_{m_E}, \alpha_{\lambda_{I,k}} I_{m_I}, \alpha_{s_k} I_n, \alpha_{v_k} I_n\}$. The steps sizes α_{x_k}, α_{w_k}, α_{y_k}, α_{z_k}, $\alpha_{\lambda_{E,k}}$, $\alpha_{\lambda_{I,k}}$, α_{s_k}, α_{v_k}, are determined in $(0, 1]$ and could be equal each

other. Moreover the slack variables, w_k, y_k, z_k and dual variables $\lambda_{I,k}$, s_k, are all positive at the solution this propriety is maintained during the iteration process. We define,

$$\begin{aligned}
\alpha_{w_k}^{\max} &= \gamma \max_{1\leq j\leq m_I} \{-\tfrac{w_k^{(j)}}{\triangle w_k^{(j)}} : \triangle w_k^{(j)} < 0\},\\
\alpha_{y_k}^{\max} &= \gamma \max_{1\leq j\leq n} \{-\tfrac{y_k^{(j)}}{\triangle y_k^{(j)}} : \triangle y_k^{(j)} < 0\},\\
\alpha_{z_k}^{\max} &= \gamma \max_{1\leq j\leq n} \{-\tfrac{z_k^{(j)}}{\triangle z_k^{(j)}} : \triangle z_k^{(j)} < 0\},\\
\alpha_{\lambda_{I,k}}^{\max} &= \gamma \max_{1\leq j\leq m_I} \{-\tfrac{\lambda_{I,k}^{(j)}}{\triangle \lambda_{I,k}^{(j)}} : \triangle \lambda_{I,k}^{(j)} < 0\},\\
\alpha_{s_k}^{\max} &= \gamma \max_{1\leq j\leq n} \{-\tfrac{s_k^{(j)}}{\triangle s_k^{(j)}} : \triangle s_k^{(j)} < 0\},\\
\alpha_{v_k}^{\max} &= \gamma \max_{1\leq j\leq n} \{-\tfrac{v_k^{(j)}}{\triangle v_k^{(j)}} : \triangle v_k^{(j)} < 0\},
\end{aligned}$$

that represent the maximal steps sizes autorized to keep the positivity of the respective variables, with $\gamma \in (0, 1]$.

Once the maximal steps sizes are computed, we set

$$\Upsilon^{\max} = \min\{\alpha_{w_k}^{\max}, \alpha_{y_k}^{\max}, \alpha_{z_k}^{\max}, \alpha_{\lambda_{I,k}}^{\max}, \alpha_{s_k}^{\max}, \alpha_{v_k}^{\max}\}$$

a backtracking line search is performed in order to decrease the L_1 merit function.

$$\mathcal{M}_{\sigma\mu}(x, w, y, z) = \bar{f}_\mu(x, w, y, z) + \sigma \| \; Ax - b \; \| + \sigma \| \; c(x) - w \; \| + \sigma \| \; x - y + z \; \|, \quad \text{(B.43)}$$

where σ is a positive penalty parameter. This parameter is updated at each iteration. Clearly the step size Υ_k is computed in $(0, \Upsilon^{\max}]$ and verifies the Armijo's condition,

$$\mathcal{M}_{\sigma\mu}(\Pi_k + \Upsilon_k \triangle\Pi) \leq \mathcal{M}_{\sigma\mu}(\Pi_k) + \beta\alpha_k \mathcal{M}'_{\sigma\mu}(\Pi_k; \triangle\Pi), \quad \text{(B.44)}$$

with β a constant in $(0, \frac{1}{2})$. If the condition (B.44) is not satified, then a new step size is taken in $[\beta_1\Upsilon_k, \beta_2\Upsilon_k]$, with $\beta_1, \beta_2 \in (0, 1)$ and $\beta_1 < \beta_2$. The procedure is repeated until (B.44) is satisfied. Then the new primal-dual iterate is,

$$\Pi_{k+1} = \Pi_k + \Upsilon_k \triangle\Pi.$$

The distance of the point Π_k to central path is mesured using the euclidian norm of the perturbed optimality conditions, that is $\| \; F_\mu(\Pi_k) \; \|$ less than a given precision ε ; the parameter μ is decrised and the process is repeated until μ becomes zero.

B.5.2.2 The IPDCA algorithm for the problem $(\mathcal{P}_{\mathcal{EI}})$

We will now describe the main lines of IPDCA to solve the problem $(\mathcal{P}_{\mathcal{EI}})$.

Algorithme B.5.1 *A primal-dual interior point algorithm(IPDCA)*

1. *Chose* Π_0, $\beta \in (0, \frac{1}{2})$, $\theta \in (0,1)$ *and* σ_0 *and a precision* ε. *Set* $k = 0$.
2. *Compute the step* $\triangle\Pi_k$ *using* (B.31)-(B.42).
3. *Compute the step size* Υ_k *using the condition* (B.44), *if there is no step size satisfying* (B.44) *then stop, else set*
$$\Pi_{k+1} = \Pi_k + \Upsilon_k \triangle\Pi.$$
4. *If* $\| F_\mu(\Pi_k) \| \leq \varepsilon$ *then stop, else update* $\mu_{k+1} = \theta\mu_k$ *and update* σ_k *set* $k = k+1$ *goto step 2.*

An important point in the algorithm just presented is the rule to update the penalty parameter σ_k. In the present version of IPDCA we have adopted the following rule,

if $\sigma_{k-1} \geq \| \tilde{\lambda}_k \| + \bar{\sigma}$ **then**
$\sigma_k = \sigma_{k-1}$
else
$\sigma_k = \max\{\zeta\sigma_{k-1}, \| \tilde{\lambda}_k \| + \bar{\sigma}\}$
end

with $\zeta > 1$, $\bar{\sigma}$ a given positive number, and

$$\tilde{\lambda}_k = \| \lambda_{I,k} + \triangle\lambda_I \| + \| \lambda_{E,k} + \triangle\lambda_E \| + \| s_k + \triangle s \| .$$

B.5.3 Computational results

We first reformulate the convex program $(\mathcal{PM})$ as a false DC Program of the form

$$(Pdc) \min\{f(x) = g(x) - h(x) : x \in \mathbb{R}^n, H(x) = 0, G(x) \leq 0\}$$

where g is a function strongly convex (for example $g(x) = f(x) + \frac{\rho}{2} \| x \|^2$ and $h(x) = \frac{\rho}{2} \| x \|^2$, with ρ being a positive number, we take $\rho = 10e^{-3}$ for our tests). The optimality tolerance is $10e^{-4}$ as suggested by our interlocutor of IAM.

The IPDCA has been implemented in standard C, on one node of a cluster IBM 1600.
In Table B.1,
- n : number of variables,
- m : number of constraints,
- *iter* : number of iterations,
- *time* : time taken to solve the problem reported in seconds or days, hours minutes for the last problem,
- " $-$ " : the algorithm failed to solve the problem.

The computational results show that the running-time of IPDCA is very small in comparison with that of the standard LANCELOT code(100 times faster).

noms	*dim*		*iter*			*obj*			*time*		
	n	m	LANC	NIPA	IPDCA	LANC	NIPA	IPDCA	LANC	NIPA	IPDCA
P1	193	148	81140	10	138	1.3	1.5	1.5	253.35	20.86	7.98
P2	3457	2268	12350	15	209	47.4	50.4	50.9	47565.45	20830.12	363.67
P3	38401	25046	17500	17	172	2.4	2.4	2.4	86400.00	51960.25	1356.63
P4	57601	36552	-	-	1334	-	-	5.1	-	-	1828.66
P5	192001	125200	-	-	1438	-	-	5.5	-	-	2d0h33min

TAB. B.1 – Comparison of IPDCA, NIPA and LANCELOT on the problem.

Annexe C

IPDCA and IPDCMECA user's guide

C.1 IPDCA user's guide

In this section we will describe,
(1) how to install IPDCA on your machine,
(2) how to use the subroutine library to formulate and solve your optimization problems.

C.1.1 Installation

The IPDCA directory contains the following files
`ipdca.lib` Archive file containing the IPDCA functions library.
`ipdca.c` A file containing the main program for IPDCA.
`exple.c` A file containing an example on how IPDCA functions can be used.
`ipdca.h` A header file containing the function prototypes for some functions in the IPDCA library. This file will be included in any program file in which IPDCA functions library are called
You will need to have ADOL-C installed on your machine. ADOL-C is used to compute first and second derivatives in IPDCA.

C.1.2 Coding of the function subroutines

We believe that the easiest way to explain how to use the function library is by considering an example ; so we present here the contents of `exple.c` file.

```
#include<ipdca.h>
int main()
{


  IPDCA *ipdc;
```

```
  /*Initialize the problem*/

  ipdc = initpb();
  if (ipdc == NULL) {printf(" Initialization failure!\n"); exit(1);}

  /*Call the solver*/
  solvepb(ipdc);

  return 0;
}


/*
   Specify IPDCA setup
*/
void IPDCAsetup(*IPDCA*ipdc)
{

   /*The problem name*/
    strcpy(name,"mypbname");

   /*The problem dimension*/

    ipdc->n   = 2; /*number of variables*/
    ipdc->mle = 0; /*number of linear equality constraints*/
    ipdc->mli = 1; /*number of nonlinear inequality constraints*/


   /*Optimization tolerances*/

    ipdc->feastol = 1e-6; /*feasibility tolerance*/
    ipdc->kttol   = 1e-4; /*tolerance for the other KT conditions*/

}

/*
Objective function
*/

void ffunc(double*x,double*f)
{
```

```
   *f = x[0]*x[0]-x[1]*x[1];
}

/*
The DC decomposition of the objective function must be provided by the user.
That is the user must provide the two convex functions g and h with f=g-h.
*/

/*
The convex function g
*/

void gfunc(double*x,double*g)
{
  *g = x[0]*x[0];
}
/*
The convex function h
*/
void hfunc(double*x,double*h)
{
 *h = x[1]*x[1];
}
/*
The constraint function. Bounds on variables are considered as constraints.
*/
void cfunc(double*x,double*Cl,double*Cnl)
{
   Cnl[0] = -x[0]*x[0]-x[1]*x[1]+1;
}
```

C.2 IPDCMECA user's guide

IPDCMECA is the IPDCA algorithm designed for the problem presented in Chapter 8. In the following lines we will describe the contents of the directory IPDCMECA released to our partners of IAM and how to use IPDCMECA to solve the problem.

The problem is solved in two steps : first the problem is transformed in the sudirectory PBTRF ; the transformed problem is exported in the subdirectory data, then solved by calling ipdca in the main directory.

```
In the IPDCMECA folder:
  Some .h files, a main.c file a Makefile  and directories
```

```
 data :
       This directory must contain the following files cont-
       aining the data of the transformed mechanic problem
        - meca_num.txt
        - sparsA.txt
        - sparsB.txt
        - sparsW.txt
        - CONST-A(provided by the user)

four parameters can be set by the user in the main.c file,

feastol   the feasible tolerance its default value is 10e-6
kkttol    the tolerance for the other KKT equations its
          default value is 10e-4.
itlim     the iterations limit its default value is 500
printlev  the print level. 1 to have the iterations printed
          on the screen 0 otherwise.

If the tolerance defined by kkttol cannot be reached,
the optimization process will end with a feasible solution.

Note that in the present version to test IPDCMECA you should renaim
sparsA_big.txt for example into sparsA.txt and so for the three other
corresponding files sparsB_big.txt, sparsW_big.txt, meca_num_big.txt
in the directory in order to test IPDCMECA.

 data_pb_orig : The data of the original problem that should be
                transformed before passing it to IPDCA.
               Included two files
                 - CMAT
                 - SIGMA
               or ninetheen files(eighteen Cmat-mn files and SIGMA)
               to test.

 NOTES : 1) - The original data should be provided in the form
              in CMAT and SIGMA files.
         2) - The original data should be provided in the form
              in CMAT-mp and SIGMA files if the C matrix is provided
              in eighteen differents files.

 doc:
       documentation.
```

```
  lib :
      Library file that contains ipdca.a compiled with
      gcc 3.22 on RedHat 9.0

  results :
       The primal solution is written in this directory.

  PBTRF(Tranform the problem) :
        Contains files
           - transfpb.h
           - transfpb.c


        In transfpb.h you should set the parameter workpar
        to a large unsigned integer depending to your problem size.
        This parameter determines the working space to reformulate
        the problem. It represents the number of nonzero elements of
        the linear constraint matrix of the transformed problem.

        In this directory before running make you should set
        the correct paths in the file transfpb.c and the makefile
        if needed.

 Additional Sofware : none

 The compiler:

        compiler  gcc 3.22

  Supported Platform : Linux Redhat9.0

To execute the program just launch gcc transfpb.c -lm or make then launch
a.out to transform  the problem.
```

Bibliographie

[1] E. Melan, *Theorie Statisch Unbestimmter Systeme aus Ideal-Plastischem Baustoff,* Sitber. Akad. Wiss., Wien, Abt. IIA 145, 195-218, 1936.

[2] W.T.Koiter, *General theorems for elastic-plastic solids,* In : Progress in solid mechanics. Ed. : I.N. Sneddon ; R. Hill, North-Holland, Amsterdam, pp. 165-221, 1960.

[3] A. Hachemi, D. Weichert, *Numerical shakedown analysis of damaged structures,* Comput. Methods Appl. Mech. Engrg., 160, 57-70, 1998.

[4] D. Weichert, A. Hachemi, *Influence of geometrical nonlinearities on the shakedown analysis of damaged structures,* Int. J. Plasticity, 14, 891-907, 1998.

[5] D. Weichert, A. Hachemi, F. Schwabe, *Shakedown analysis of composites,* Mech. Res. Comm., 26, 309-318, 1999.

[6] D. Weichert, A. Hachemi, F. Schwabe, *Failure investigation of fiber-reinforced composite materials by shakedown analysis,* In : Inelastic Analysis of Structures under Variable LoadsTheory and Engineering Applications. Ed : D. Weichert ; G. Maier, Kluwer Academic Publishers, pp. 107-119, 2000.

[7] J.A. König, M. Kleiber, *On a new method of shakedown analysis,* Bull. Acad. Polon. Sci., Sér. Sci., **26**, 165-171, 1978.

[8] F. Akoa, Le Thi Hoai An, Pham Dinh Tao, *An Interior Point Algorithm with DC Regularization for Nonconvex Quadratic Programming,* Modelling, Computation and Optimization in Information Systems and Management Sciences. pp 87-96. Hermes Science Publishers, edited by Le Thi Hoai An and Pham Dinh Tao.

[9] F. Akoa, A. Hachemi, Le Thi Hoai An, S. Mouhtamid, Pham Dinh Tao, *Large-scale nonlinear programming and lower bound direct method in engineering applications : A combined interior point and DC programming approach* Submitted to Special Issue of Journal of Global Optimization : "Modelling, Computation and Optimization in Systems Engineering."

[10] A. Hachemi, S. Mouhtamid, Le Thi Hoai An, Pham Dinh Tao, *Large-scale nonlinear programming and lower bound direct method in engineering applications* Modelling, Computation and Optimization in Information Systems and Management Sciences. pp 299-310. Hermes Science Publishers, edited by Le Thi Hoai An and Pham Dinh Tao.

[11] R.B. Schnabel, E. Eskow, *A Revised Modified Cholesky Factorization Algorithm,* SIAM Journal on optimization. Volume 9, Number 4. pp 1135-1148.

[12] M. Fukushima, *Merit function for variational inequality and complementarity problems,* In G. Di Pillo and F. Gianessi(eds.), *Nonlinear optimization and Applications.* Plenum Press New York, NY, 1996, 155-170.

[13] A. Fisher, *A special Newton type Optimization methods for positive semidefinite linear complementarity problems,* Technical repport, Institut of Numerical Mathematics, Dresden University of Technology. Dresden Germany, 1992.

[14] A. Fisher, *A special Newton type Optimization methods,* Optimization, 24, 269-284, 1992.

[15] J.J. Moré, D.G. Sorensen, *Computing a trust region step.* SIAM J. Sci. Stat.Comput. 4, 553-572.

[16] R. Fletcher and S.Leyffer, *Nonlinear programming without a penalty function,* Technical Repport NA/171, University of Dundee, Dept. of Mathematics, Dundee Scotland, 1997.

[17] H. Jiang, *Smoothed Fisher-Burmeister Equations methods for the complementarity problem,* CSIRO Mathematical and Information Sciences.

[18] A. Fisher, *On the local superlinear convergence of a Newton-type method for LCP under weak conditions.* Technical repport MATH-NM-07-1993, Institut of Numerical Mathematics, Dresden University of Technology. Dresden Germany, 1993.

[19] A. Fisher, *Defecktfunctionale bei Ungleichungen und gedämpfte gestörte Newton-Verfahren.* Preprint 07-11-85, Department of Mathematics, TU Dresden, 1985.

[20] C. Kanzow, *Nonlinear complementarity as unconstrained optimization problem.* Preprint 67, Institut of Applied Mathematics University of Hamburg. Hamburg, Germany, 1993.

[21] S. Engelke and C. Kanzow, *Predictor-corrector smoothing methods for the solution of linear programming.* Preprint, Institut of Applied Mathematics University of Hamburg. Hamburg, Germany, 2000.

[22] S. Engelke and C. Kanzow, *Predictor-Corrector Smoothing Methods for Linear Programs with More Flexible Update of the Smoothing Parameter.* Computational Optimization and Applications,Kluwer Academic publishers, 23, 299-320, 2002.

[23] S. Engelke and C. Kanzow, *Improved smoothing-type methods for the solution of linear programming.* Preprint, Institut of Applied Mathematics University of Hamburg. Hamburg, Germany, 2000.

[24] C. Kanzow, *Some noninterior continuation methods for linear complementarity problems.* SIAM J. Matrix Anal. Appl., 17(1996) p.851-868.

[25] C. Kanzow, *Global convergence properties of some iterative methods for linear complementarity problems.* Preprint 72, Institut of Applied Mathematics University of Hamburg. Hamburg, Germany, 1993, revised 1994.

[26] C. Kanzow, *Some equation-based methods for the nonlinear complementarity problems.* Preprint 63, Institut of Applied Mathematics University of Hamburg. Hamburg, Germany, 1993.

[27] L. QI, and H. Jiang, *karush-Kuhn-Tucker equations and convergence analysis for Newton and Quasi-newton methods for solving these equations.* Technical Repport, AMB 94/5, School of Mathematics, University of New South Wales, Australia, 1994.

[28] P. Tseng, *Growth behavior of a class of merit functions for the nonlinear complementarity problem,* Manuscript, 1994.

[29] J.F. Bonnans, E.R. Panier, A.L. Tits, J.L. Zhou, *Avoiding the Maratos effect by means of a nonmonotone line search II. Inequality constrained problems-feasible iterates,* SIAM Journal on Numerical Analysis, v.29 n.4, p.1187-1202, Aug. 1992.

[30] Y.H. Dai, *On the Nonmonotone Line Search* Journal of Optimization Theory and Applications 112 (2) : 315-330, February 2002.

[31] R.J. Vanderbei and D. Shanno, *An interior-point algorithm for nonconvex nonlinear programming* Statistics and Operations Research. Princeton SOR-97-21.

[32] R.J. Vanderbei, *Linear Programming. Foundations and Extensions.* Second Edition - International Series in Operations Research and Management Science, Volume 37.

[33] R.J. Vanderbei, *Large-scale nonlinear AMPL models* http ://www.sor.princeton.edu/ rvdb/ampl/nlmodels/index.html

[34] R.J. Vanderbei, *An interior-point algorithm for quadratic programming* Statistics and Operations Research. Princeton SOR-94-15.

[35] R.J. Vanderbei, *Symmetric quasi-definite matrices,* SIAM Journal on Optimization, 5(1) :100-113, 1995.

[36] R.J. Vanderbei, *LOQO User's Manual version 4.05.*
Operations Research and Financial Engeneering. Technical Report No. ORFE-99-, october, 2000.

[37] R.J. Vanderbei and D. Shanno, *An interior-point algorithm for nonconvex nonlinear programming : Filter methods and merit functions.* Statistics and Operations Research. Princeton ORFE-00-06.

[38] W. Hock, K. Schittkowski, *Test examples for nonlinear programming codes,* Lecture notes in econom. and Math. Systems 187, Springer, New York, 1981.

[39] A. Wächter and L.T. Biegler, *Failure of global convergence for a class of interior point methods for nonlinear programming ,* Math. Program. ; 88(2000), 565-574.

[40] L. Qi and G. Zhou, *A smoothing Newton method for ball constrained variational inequalities with applications.* Statistics and Operations Research. Princeton SOR-97-21.

[41] S. Smale, *Algorithms for solving equations* Proceeding of international congress of mathematicians, Edited by Gleason, A. M. American Mathematics Society. Providence Rhode Island, 1987, pp.172-195.

[42] B. Chen, *A non interior continuation method for quadratic and linear programming* SIAM J. Optim., 3(1993), 503-515.

[43] B. Chen, and P.T. Harker, *Smooth approximations to nonlinear complementarity problems,* SIAM J. Optim., 7(1997), 403-420.

[44] D. Sun, *A regularization Newton method for solving nonlinear complementarity problems,* Appl. Math. Optim., 36(1999), 315-339.

[45] J. Burke, and S. Xu, *The global linear convergence of a non-interior path-following algorithm for linear complementarity problems* Math. Oper. Res., 23(1998), 719-734.

[46] J. Burke, and S. Xu, *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, 1999, pp. 45-64.

[47] A.S. Strekalovsky, *On the Global Extremum Problem,* Soviet Math. Doklady, 292(5), pp.1062-1066, 1987.

[48] A.S. Strekalovsky, *Global Optimality Conditions for Nonconvex Optimization* Journal of Global Optimization, 12, pp.415-434, 1998.

[49] A.S. Strekalovsky, R. Enkhbat, *Global Maximum of Convex Functions on an arbitrary Set.* Dep. in VINITI, Irkutsk, No. 1063, pp. 1-27, 1990.

[50] A.S. Strekalovsky, *Global Optimality Conditions for Nonconvex Optimization* Journal of Global Optimization, 12, pp.415-434, 1998.

[51] A.S. Strekalovsky, *DC Programming : Theory and Algorithms* Institute of System Dynamics and Control Thory SB of RAS, May 1999.

[52] R. Enkhbat, T. Ibaraki, *On Extrema of a Strongly Convex Function* Tech. Rep. University of Kyoto.

[53] R. Enkhbat, T. Ibaraki, *On the Maximization and Minimization of a Quasiconvex Function* Tech. Rep. University of Kyoto, 2001.

[54] R. Enkhbat, T. Ibaraki, *On the Maximization and Minimization of a Quasiconvex Function* ech. Rep. University of Kyoto, 2001.

[55] Ioannis Akrotirianakis, Berç Rustem, *A primal-dual interior point algorithm with an exact différentiable merit function for general nonlinear programming problems* Tech. Rep. 98-09. Department of computing, Imperial College of Science, Technology and Medecine, London , 1999.

[56] H. Yamashita, *A globally convergent primal-dual interior point method for constrained optimization.* Tech. Rep. Mathematical Systems Institute Inc, 2-5-3 Shinjuku, Shinjuku-ku, Tokyo, Japan, 1995.

[57] J.F. Bonnans, J.-C. Gilbert, C. Lemaréchal, C. Sagastizábal, *Numerical Optimization - Theoretical and Practical Aspects.* Springer Verlag, Berlin, 2001.

[58] A.V. Fiacco, G.P. McCormick, *Nonlinear Programming : Sequencial Unconstrained Minimization Techniques.* Research Analysis Corperation, McLean Virginia, 1968. Republished in 1990 by SIAM, Philadelphia.

[59] *Deliverable D1 : Algorithms for Solving Nonlinear Constrained and Optimization Problems : The State of The Art,* Bliek, C.(ed),The Coconut Consortium, jun 2001.

[60] M.H. Wright, *The interior point revolution in constrained optimization.* R. In : Deleone A. Murli, P.M. Pardalosand G. Toraldo(eds), High-Performance Algorithms and Software in Nonlinear Optimization, Kluwer Academic Publishers, Dordrecht, pp. 359-381, 1998.

[61] D.M. Gay, M.L. Overton, M.H. Wright, *Primal-dual interior point methods for nonconvex nonlinear programming.* In Y. Yuan, editor, advanced in nonlinear Programming, pp. 341-407. Kluwer Academic Publishers, Dordtrecht, 1998.

[62] P.M. Pardalos, *Global optimization algorithms for linearly constrained indefinite quadratic problems,* Comput., 21, 87-97, 1991.

[63] P.M. Pardalos and G. Scnitger, *Checking local optimality in constrained quadratic programming is NP-hard.* Oper. Res. Let. 7, 33-35, 1988.

[64] N.I.M. Gould and P.L. Toint, *A quadratic programming bibliography.* Numerical Analysis Group Internal Repport 2000-1, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 2001.

[65] P.M. Pardalos and S.A. Vavasis, *Quadratic programming with one negative eigenvalue is NP-hard,* J. Global Optimization, 21, 843-855, 2002.

[66] S.A. Vavasis, *Quadratic programming is NP,* Info. Proc. Lett., 36, 73-77, 1990.

[67] S.A. Vavasis, *Nonlinear optimization : Complexity issues,* Oxford University press, New York, 1991.

[68] C.A. Floudas, V. Visweswarn, *Quadratic optimization.,* In : R. Horst and P.M. Pardalos(eds), Handbook of Global Optimization, Kluwer Academic Publishers, Dordrecht/Boston/London, pp. 217-270.

[69] S. Sahani, *Computationally related problems* SIAM J. Comp., 3, 262-279.

[70] R.B. Lehoucq, D.C. Sorensen, C. Yang, *ARPACK User's Guide : Solution of Large Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods* Oct, 1997.

[71] *CPLEX Optimization ILOG, Inc*Using the CPLEX^R Callable Library and CPLEX Barrier and Mixed Integer Solver Options(version 5. 0), Incline Village, 1997.

[72] M.H. Wright, *Why a pure primal Newton barrier step may be infeasible.* SIAM Journal on Optimization, 5(1) :1-12, 1995.

[73] B.D. Burow, *Mixed Langage Programming.*
Universitat Hamburg, II. Institut fur Experimentalphysik.

[74] Y.-B. Zhao and D. Li *A globally and Locally Superlinearly Convergent Non-Interior-Point Algorithm for $P_\circ$ LCPs'* SIAM J. Optim., Vol. 13, No. 4, pp. 1195-1221.

[75] M. Kojima, S. Mizuno, A.Yoshise, *A polynomial time algorithm for a class of linear complementary problems.* Math. Programming, 44 :27-41, 1989.

[76] J.E. Falk, K.R. Hoffman, *A successive underestimation method for concave minimization,* Mathematics of Operations Research 1, 251-259, 1976.

[77] N.K. Karmakar, *A new polynomial time algorithm for linear programming.* combinatorica, 4 :373-395, 1984.

[78] K.A. McShane, C.L. Monna, D.F. Shanno, *An implementation of primal-dual interior point method for linear programming.* ORSA Journal o Computing, 1(2) :70-83, 1989.

[79] M. Saunders, *Cholesky-based methods for sparse least squares : the benefits of regularization.* In Linear and Nonlinear Conjugate Gradient-Related Methods, L. Adams and L. Nazareth, eds., SIAM Philadelphia, USA, 1996, pp. 92-100.

[80] M. Saunders, J.A. Tomlin, *Solving regularized linear programs using barrier methods and KKT systems.* Technical Repport SOL 96-4, Systems Optimization LMaboratory, Dept. of Operations Researcf, Stanford University, Stanford, CA 94305, USA, December 1996.

[81] P. Gill, M. Saunders, J.R. Shinnerl, *On the stability of cholesky factorization for symmetric quasidefinite systems,.* SIAM J. Matrix Anal. Appl. 17(1) :35-46, 1996.

[82] I.J. Lustig, R.E. Marsten, D.F. Shanno, *Comutational experience with primal-dual interior point method for linear programming.* Linear Algebra and Applications, 152 :191-222, 1991.

[83] Y. Zang, R.A. Tapia, *A superlinear convergent polinomial primal dual interior point method for linear programming.* SIAM J. Optimization, 3(1) :118-133, February 1993.

[84] F.A. Potra, *An infeasible interior-point predictor-corrector algorithm linear programming.* Technical Report 26, Departement of Mathematics, University of Iowa, Iowa City, IA, 1992.

[85] R. Monteiro, I. Adler, *Interior path following algorithm primal-dual algorithms-part 2 : Convex quadratic programming..* Mathematical Programmming, 44 :43-66, 1989.

[86] S.J. Wright, *An interior point algorithm for linearly constrained optimization.* SIAM J. Optimization, 2 :450-473, 1992.

[87] J.P. Vial, *Computational experience with a primal-dual interior point method for smooth convex programming.* Optimization methods and software, 3 :285-310, 1994.

[88] K.M. Anstreicher, J.P. Vial, *On the convergence of an infeasible primal-dual interior point method for convex programming.* Optimization methods and software, 3 :273-283, 1994.

[89] G.P. McCormick, J.E. Falck, *The superlinear convergence of a nonlinear primal-dual algorithm.* Technical Report GWU/OR/Serial T-550/91, Departement of Operation Research, The George Washinton University, Washington DC, 1991.

[90] A.S. El Bakry, R.A. Tapia, T. Tsuchiya, Y. Zang, *On the formulation and theory of the newton interior point method for nonlinear programming.* JOTA, 89(3) :507-541, 1996.

[91] A.L. Tits, A. Wächter, S. Bakhtiari, T.J. Urban, and C.T. Lawrence *A Primal-Dual Interior-Point Method for Nonlinear Programming with Strong Global and Local Convergence Properties.* SIAM Journal on Optimization 14(1), pp. 173-199, 2003.

[92] R. Horst, N.V. Thoai, *A New algorithm for Solving the General Quadratic Programming Problem,* Forshchungsberich NO. 3-1993, Universtat Trier.

[93] Thai Quynh Phong, Le Thi Hoai An, Pham Dinh Tao, *On the global solution of linearly constrained indefinite quadratic minimization problems by decomposition branch and bound method* RAIRO, Recherche Opérationnelle 30(1), 31-49.

[94] Thai Quynh Phong, *Analyse Numérique des Méthodes d'Optimissation Globale codes et simulations numériques. Applications.* Thèse de doctorat, Université de Rouen, Décembre 1994.

[95] M. Le Dung, T.Q. Phong and P. Dinh Tao, *Decomposition methods for solving a class of nonconvex programming problems dealing with bilinear and quadratic function.* Compuational Optimization and Applications 4(1995) 203-216.

[96] A. Wächter, *An Interior Point Algorithm for Large-Scale Nonlinear Optimization with Applications in Process Engineering.* Phd Thesis, Carnegie Mellon University, 2002.

[97] J.-C. Gilbert, R.H. Byrd, J. Nocedal *A trust region method based on interior point techniques for nonlinear programming.* Mathematical Programming, 89 (2000) 149-185.

[98] R.H. Byrd, M.E. Hribar, J. Nocedal *An interior point algorithm for large scale nonlinear programming.* SIAM Journal on Optimization, 9(4) :877-900, 2000.

[99] L. Chauvier, A. Fuduli, J.-C. Gilbert, *A truncated SQP algorithm for solving nonconvex equality constrained optimization problems* INRIA Rapport de Recherche n° 4346, 26 décembre 2001.

[100] B. Kalantari, J.B. Rosen, *Algorithm for global minimization of linearly constrained concave quadratic functions.* Mathematic of Operations Research, 12 :544-561, 1987.

[101] A. Altman, J. Gondzio *Regularized symmetric Indefinite Systems in Interior Point Methods for Linear and Quadratic Optimization.* Logilab Technical Report 1998.6.

[102] Le Thi Hoai An, Pham Dinh Tao, F. Akoa, *An Interior Point Algorithm with DC Regularization (IPDCA) for Nonlinear programming,* Tech. Rep. Laboratory of Mathematics, Insa - Rouen, 2003.

[103] Le Thi Hoai An, *Analyse numérique des algorithmes de l'optimization D.C : Approches locale et globale,* Thèse de doctorat, Université de Rouen, Décembre 1994.

[104] Le Thi Hoai An, Pham Dinh Tao, *DC Programming : Theory, Algorithms and Applications. The State of the Art*(28 pages), Proceedings (containing the refereed contributed papers) of The First International Workshop on Global Constrained Optimization and Constraint Satisfaction (Cocos' 02), Valbonne-Sophia Antipolis, France, October 2-4, 2002.

[105] Le Thi Hoai An, *Contribution à l'optimisation non convexe et l'optimisation globale : Théorie, Algorithmes et Applications*, Habilitation à Diriger des Recherches (HDR), Université de Rouen, Juin 1997

[106] Pham Dinh Tao and Le Thi Hoai An, *Convex analysis approach to DC programming : Theory, Algorithms and Applications,* Acta Mathematica Vietnamica, dedicated to Professor Hoang Tuy on the occasion of his 70th birthday, Vol. **22**, Number 1 (1997).

[107] Pham Dinh Tao, Le Thi Hoai An, *DC Optimization Algorithm for Solving The Trust Region Problem*SIAM Journal on Optimization, Vol. **8**, Number 2, (1998), pp. 476-505

[108] Le Thi Hoai An, Pham Dinh Tao, *Large Scale Molecular Optimization From Distance Matrices by a DC Optimization Approach.*SIAM Journal on Optimization, Vol. **14**, No 1, pp. 77-116, 2003.

[109] Le Thi Hoai An, Pham Dinh Tao, *The DC (difference of convex functions) Programming and DCA revisited with DC models of real world nonconvex optimization problems.* Annals of Operations Research 133, 23-46, 2005.

[110] Le Thi Hoai An, Pham Dinh Tao, Pham Tien Son, *DCA for Computing Global Minima of the Lennard Jones Potential Energy Function.* Submitted

[111] Le Thi Hoai An, Pham Dinh Tao, *A continuous approach for globally solving linearly constrained quadratic zero-one programming problems* Optimization 50 (1-2) : 93-120 2001

[112] Le Thi Hoai An, Pham Dinh Tao, *A continuous approach for Globally Solving Mixed Integer Programming.* Sixth SIAM Conference on Optimization, Georgia, USA, Atlanta, may 10-12, 1999.

[113] Le Thi Hoai An, Tao Pham Dinh, *A Branch-and-Bound method via D.C. Optimization Algorithm and Ellipsoidal techniques for Box Constrained Nonconvex Quadratic Programming Problems,* Journal of Global Optimization 13, pp. 171-206 (1998).

[114] Le Thi Hoai An, Tao Pham Dinh, and M. Le Dung, *A combined D.C. Optimization - Ellipsoidal Branchand- Bound Algorithm for Solving Nonconvex Quadratic Programming Problems,* Journal of Combinatorial Optimization, Vol. 2, No. 1, pp. 9-28.

[115] Le Thi Hoai An, Tao Pham Dinh, *Solving a class of linearly constrained indefinite quadratic problems by d.c. algorithms,* Journal of Global Optimization, 11(1997), pp. 253-285.

[116] Le Thi Hoai An, Tao Pham Dinh, *DC Programming Approachs for globally Solving Linear Complementarity Problems,* Technical Report, Laboratory of Mathematics, Insa-Rouen 2003, Submitted.

[117] Le Thi Hoai An, T. P. Phuc, Pham Dinh Tao, *A DC programming approach to the strategic supply chain problem from qualified partner set,* Modelling, Computation and Optimization in Information Systems and Management Sciences. pp 327-335. Hermes Science Publishers, edited by Le Thi Hoai An and Pham Dinh Tao.

[118] Le Thi Hoai An, Tao Pham Dinh, T. Huynh Van Ngai, *Exact penalty techniques in DC Programming*, Technical Report, Laboratory of Mathematics, Insa-Rouen, 2004. Submitted.

[119] P.M. Pardalos, J.B. Rosen, *Constrained Global Optimization : Algorithms and Application,* Springer-Verlag, Berlin, 1987.

[120] P. Spellucci, *Numerical experiments with modern methods for large scale QP-problems,* Technical University at Darmstadt, Department of Mathematics, 1996.

[121] C.A. Floudas, P.M. Pardalos, *Handbook of Test Problems in Local and Global Optimization* Kluwer Academic Publishers 1999

[122] A.T. Phillips, J.B. Rosen, *A parallel algorithm for constrained concave quadratic quadratic global minimization.* Mathematic programming, 42 :421-448, 1988.

[123] R. Horst, H. Tuy, *Global Optimization : Deterministic Approaches,* 2cd revised edition, Springer-Verlag, Berlin, 1993

[124] J.B. Hiriart-Urruty, *From convex optimization to nonconvex optimization. Part I : Necessary and sufficient conditions for global optimality,* Nonsmooth Optimization and Related Topics, Ettore Majorana International Sciences, Series 43, Plenum Press.

[125] J.E. Falk, R.M. Soland, *An algorithm for separable nonconvex programming problems* Management Science 15, 550-569, 1969.

[126] J.B. Hiriart-Urruty, *Generalized differentiability, dualityand optimiation for problems dealing with differences of convex functions,* Lecture Notes in Economics en MAth. Systems, 256(1985), pp 37-70.

[127] B. Chen, X. Chen, *A Global Linear and Local Quadratic Continuation Smoothing Method for Variational Inequalities with Box Constraints.* AMR 97/6, Applied Mathematics Report, University of New South Wales, Sydeney, March 1997

[128] A. Griewank and D. Juedes and J. Srinivasan, *ADOL-C, a Package for the Automatic Differentiation of Algorithms Written in C/C"*, Mathematics and Computer Science Div., Argonne National Laboratory. August 22, - 14 -, 1990.

[129] S.A. Gabriel, J.J. Moré, *Smoothing of mixed complementarity problems,* MCS-P541-0995, Tech. Rep., Mathematics and Computer Science Division, Argone National Laboratory, Argonne, Illinois, 1995.

[130] C. Chen, O.L. Mangasarian, *A class of smoothing functions for nonlinear and mixed complementarity problems,* Comp. Optim. and Appl., 5(1996), 97-138.

[131] P.E. Gill, W. Murray, M.H. Wright, *Practical optimization,* Academic Press, London, 1981.

[132] P.E. Gill, W. Murray, D.B. Poncélon, M.A. Saunders, *Solving reduced KKT system in barrier methods for linear programming,* in G. A. Watson and D. Griffiths(eds), Numerical Analysis 1993, Pitman Research Notes in Mathematics 303, Longmans Press, pp. 89-104, 1994.

[133] S.H. Cheng, N.J. Higham, *A Modified Cholesky Algorithm Based on Symmetric Indefinite Factorization,* SIAM J. Matrix Anal. Appl., 19(1998) pp. 1097-1110.

[134] A.R. Conn, N.I.M. Gould and P.L. Toint, *The SIF Reference Document* http ://www.numerical.rl.ac.uk/lancelot/sif/sifhtml.html

[135] M. Gertz, J. Nocedal, A. Sartenaer, *A starting-point strategy for nonlinear-interior methods* Report OTC 2003/4, Optimization Technology Center, March 2003, submitted for publication in Applied Mathematics Letters

[136] R. Fletcher, S. Leyffer and P.L. Toint, *On the Global Convergence of a Filter-SQP Algorithm,* SIAM Journal on Optimization, vol.13(1), pp. 44-59, 2002.

[137] R. Fletcher, N. Gould, S. Leyffer, P.L. Toint and A. Wächter, *On the Global Convergence of Trust-Region SQP-Filter Algorithms for General Nonlinear Programming,* SIAM Journal on Optimization, vol. 13(3), pp. 635-659, 2002.

[138] N.I.M. Gould and P.L. Toint, *Global Convergence of a Hybrid Trust-Region SQP-Filter Algorithm for General Nonlinear Programming,* in System Modeling and Optimization XX, E. Sachs and R. Tichatschke (eds.), Kluwer, Dordrecht, pp. 23–54, 2003.

[139] S. Ulbrich, *On the Superlinear Local Convergence of a Filter-SQP Method,* Mathematical Programming Ser. B, Vol. 100 (2004), pp. 217-245 (Special Issue for Roger Fletcher).

[140] M. Ulbrich, S. Ulbrich, and L.N. Vicente, *Globally Convergent Primal-Dual Interior-Point Filter Method for Nonlinear Programming,* Mathematical Programming Ser. A (2003), Vol. 100, pp. 379-410.

[141] M. Ulbrich and S. Ulbrich, *Non-monotone Trust Region Methods for Nonlinear Equality Constrained Optimization without a Penalty Function,* Math. Program. Ser. B Vol. 95 (2003), pp. 103–135.

[142] L. Ekeland, R. Temam, *Analyse Convexe et Problèmes Variationnels,* Dunod, Paris, 1974.

[143] H. Tuy, *Monotonic Optimization : Problems and Solution Approaches* SIAM Journal on Optimization 11 :2(2000), 464-494.

[144] Ng.T. Hoai Phuong, H. Tuy, *A Monotonicity Based Approach to Non Convex Quadratic Minimization* Tech. Rep. Vietnamese Academy of Science and Technology Institute of Mathematics, 2002, to appear in Vietnam J. Math.

[145] C.S. Ong, X. Mary, S. Canu, A.J. Smola, *Learning with Non-Positive Kernels*, In Proceedings of the 21st International Conference on Machine Learning, 2004. pp. 639-646.

[146] G. Loosli, S. Canu, S.V.N. Vishwanathan, A.J. Smola and M. Chattopadhyay, *Une boîte à outils rapide et simple pour les SVM,* CAP 2004 - Conférence d'Apprentissage, 2004, Michel Liquière and Marc Sebban editors, 113-128, Presses Universitaires de Grenoble, 9-782706-112249.

[147] R. Fletcher, S. Leyffer and P.L. Toint, *On the Global Convergence of and SLP-Filter Algorithm,* Report 98/13, Dept of Mathematics, FUNDP, Namur (B), 1998.

[148] C. Sainvitu, *Les méthodes de filtre en optimisation non linéaire : une comparaison des variantes avec recherche linéaire et région de confiance* Mémoire, Dept of Mathematics, FUNDP, Namur (B), 2002.

[149] N. Gould, C. Sainvitu, P.L. Toint, *A filter-trust-region method for unconstrained optimization* Report 04/03, Dept of Mathematics, FUNDP, Namur (B), 2004.

[150] A. Wächter and L.T. Biegler, *On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming* Research Report RC 23149, IBM T. J. Watson Research Center, Yorktown, USA, March 2004.

[151] A. Wächter and L.T. Biegler, *Line Search Filter Methods for Nonlinear Programming : Motivation and Global Convergence* Research Report RC 23036, IBM T. J. Watson Research Center, Yorktown, USA (August 2001, revised Februrary 2004).

[152] A. Wächter and L.T. Biegler, *Line Search Filter Methods for Nonlinear Programming : Local Convergence* Research Report RC 23033, IBM T. J. Watson Research Center, Yorktown, USA (August 2001, revised Februrary 2004).

[153] C. Audet and J.E. Dennis, *A pattern search filter method for nonlinear programming without derivatives*, Technical Report TR00-09, Department of Computational and Applied Mathematics, Rice University, 2000.

[154] G.H. Golub and C.F. Van Loan, *Matrix Computations*, Johns Hopkins University Press Baltimore Maryland, 1989.

[155] J.E. Dennis, C. Price, I. Coope, *Direct Search Methods for Nonlinearly Constrained Optimization Using Filters and Frames* Optimization and Engineering, 2003.

[156] C.C. Gonzaga, E. Karas and M. Vanti, *A globally convergent filter method for nonlinear programming*, Technical report, Dept. of Mathematics, Federal University of Santa Catarina, Brazil, revised August 2002.

[157] E. Anderson, and Z. Bai, and C. Bischof, and S. Blackford, and J. Demmel, and J. Dongarra, and J. Du Croz, and A. Greenbaum, and S. Hammarling, and A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, Third Edition, Society for Industrial and Applied Mathematics, 1999, Philadelphia, PA, 0-89871-447-8 (paperback).

[158] R.T. Rockafellar, *Convex Analysis*, Princeton Mathematics Ser. 28. Princeton University Press, Princeton, New Jersey, 1993.

[159] R.T. Rockafellar, *Lagrange multipliers and optimality*, SIAM Rev. 35, (1993) 183 238.

[160] I. Bongartz and A. Conn and N. Gould and P. Toint, *Constrained and unconstrained testing environment*, ACM Transactions on Mathematical Software 21 (1995).

[161] B. Murtagh and M. Saunders, *Mathematical Programming Studies* 16 (1982), pp. 84-117.

[162] R. Byrd, Guanghui Liu, and J. Nocedal, *On the Local Behavior of an Interior Point Method for Nonlinear Programming* in Numerical Analysis, D.F. Griffiths and D.J. Higham, eds, pp.37-56 (1997) Addison Wesley Longman.

[163] J. Nocedal, S.J. Wright, *Numerical Optimization,* Springer Series in Operations Research, Springer-Verlag, 1999.

[164] J.-C. Gilbert, C.C. Gonzaga, E. Karas, *Examples of Ill-Behaved Central Paths in Convex Optimization* Tech. Rep. 4179, l'INRIA-Rocquencourt, may 2001.

[165] Galilée, *Discours concernant deux Sciences nouvelles,* trad. M. Clavelin, A. Colin, 1970, p. 131.

[166] J.-C. Gilbert, *Optimisation Différentiable - Théorie et Algorithmes* Notes de cours, Octobre 1998.

[167] P.S. Bradley, Usama, M. Fayyad and O. L. Mangasarian *Mathematical Programming for Data Mining : Formulations and Challenges.* Mathematical Programming Technical Report 98-01, January 1998. Revised July 1998 . INFORMS Journal on Computing 11, 1999, 217-238.

[168] O. L. Mangasarian *Data Mining via Support Vector Machines* Data Mining Institute Technical Report 01-05, May 2001. IFIP Conference on System Modelling and Optimization, Trier, Germany, July 23-27, 2001. "System Modeling and Optimization XX", E. W. Sachs and R. Tichatschke, editors, Kluwer Academic Publishers, Boston 2003, 91-112.

[169] O. L. Mangasarian *Mathematical Programming in Data Mining* Mathematical Programming Technical Report 96-05, August 1996 – Revised November 1996 and March 1997, Data Mining and Knowledge Discovery, 1(2), 1997, 183-201.

[170] V. N. Vapnik *The Nature of Statistical Learning Theory* Berlin, Springer-Verlag, 1995

[171] V. N. Vapnik *Statistical Learning Theory,* John Wiley, 1998, NY, p.732.

[172] V. N. Vapnik *An overview of statistical learning theory,* IEEE transactions on Neural Networks 10, 5, 1999, pp. 988-1000.

[173] T. Friess, N. Cristianini, C. Campbell, *The Kernel-Adatron : a Fast and Simple Learning Procedure for Support Vector Machines* in : Shavlik, J. (ed) Proceeding of the Fifteenth International Conference on Machine Learning (ICML),1998, pg. 188-196

[174] N. Cristianini, C. Campbell, C. Burges, *Kernel Methods : Current Research and Future Directions* Machine Learning 46(1/3) : 5-9, January 2002

[175] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines* Cambridge University Press ; 2000.

[176] B. Scholkopf, *The Kernel Trick for Distances,* NIPS, 301-307, 2000.

[177] J.-B. Hiriart-Urruty, C. Lemarechal, *Convex Analysis and Minimization Algorithms*, Grundlehren der Mathematischen Wissenschaften 305-306. Springer-Verlag, 1993.

[178] S. Benson, Y. Ye, and X. Zhang, *Solving large-scale sparse semidefinite programs for combinatorial optimization,* SIAM J. Optim. 10 (2000) 443-461.

[179] M.X. Goemans and D.P. Williamson, *Improved Approximation Algorithms for Maximum Cut and Satis ability Problems Using Semidefinite Programming.* J. Assoc. Comput. Mach. 42 (1995), pp. 1115-1145. 11.

[180] S. Burer and R.D.C. Monteiro and Y. Zhang,, *Rank-Two Relaxation Heuristics for Max-Cut and Other Binary Quadratic Programs,* SIAM. J. Optim. 12(2001)pp. 503-521.

[181] B. Haasdonk, *Feature Space Interpretation of SVMs with Indefinite Kernels.* Internal Report 1/03, IIF-LMB, University Freiburg, Germany, October 2003.

[182] M.H. Wright, *What, if anything, is new in Optimization ?,* Technical report 00-4-08, Bell laboratories at Murray Hill, 2000.

[183] F. Facchinei, A. Fischer, C. Kanzow, *On the identification of zero variables in an interior-point framework* Complete numerical results. Mathematical Programming Technical Report 98-06-results, Computer Sciences Department, University of Wisconsin - Madison, Madison, WI, May 1998.

[184] R.A. Tapia, A.S. El-Bakry, Y. Zhang, *A Study of Indicators for Identifying Zero Variables in Interior-point Methods,* SIAM Review, 36 (1994), 45-72.

[185] F. Facchinei, A. Fischer, C. Kanzow, *On the identification of zero variables in an interior-point framework* SIAM Journal on Optimization 10, 2000, pp. 1058-1078.

[186] A.R. Conn, N.I.M. Gould, and P.L. Toint, *A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds,* SIAM J. Numer. Anal. 28 (1991), pp. 545–572.

[187] A.R. Conn, N.I.M. Gould, and P.L. Toint, *LANCELOT : A Fortran package for large-scale nonlinear optimization (Release A),* Springer Series in Computational Mathematics 17, Springer–Verlag, 1992.

[188] M. Le Hoai, *Programmation Quadratique Convexe pour Support Vector Machine.* Mémoire de DEA Sciences de l'Ingénieur, Insa-Rouen, Juillet 2004

[189] G. Palubeckis, *Multistart tabu search strategies for the unconstrained binary quadratic optimization problem.* Annals of Operations Research, Vol. 131, No. 1-4, pp. 259-282, 2004.

[190] R.A. Tapia, *On the role of slack variables in quasi-Newton methods for constrained optimization,* Numerical Optimization of Dynamic Systems , In L. C. W. Dixon and G.P. Szego, eds., North Holland, pp. 235–246, 1980.

[191] J.R. Shewchuk, *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain,* August 1994.

[192] Y. Saad, *Iterative Methods for Sparse Linear Systems,* 2nd edition. SIAM, Philadelpha, PA, 2003.

[193] G.B. dantzig, *Linear Programming and Extensions.* Princeton University Press, 1998.

[194] J. Neumann, Ch. Schnörr and G. Steidl, *SVM-based feature selection by direct objective minimisation,* In : Pattern Recognition, Lecture Notes in Computer Science, C. E. Rasmussen, H. H. Büulthoff, M. A. Giese and B. Schölkopf (Eds.), Springer, Berlin, 212 - 219 (2004).

[195] A.R. Conn, N.I.M. Gould, and P.L. Toint, *Trust-Region Methods* SIAM, Philadelphia, PA, USA, 2000.

[196] R. Barrett and M. Berry and T.F. Chan and J. Demmel and J. Donato and J. Dongarra and V. Eijkhout and R. Pozo and C. Romine and H. Van der Vorst, *Templates for the Solution of Linear Systems : Building Blocks for Iterative Methods,* 2nd Edition, SIAM, 1994, Philadelphia, PA.

[197] A. George, J. Liu, *Computer solution of Large Sparse Positive Definite Systems*, Printice-Hall, 1981.

Cette thèse est principalement consacrée à l'association des méthodes de points intérieurs et des techniques de l'optimisation DC et DCA pour résoudre les problèmes d'optimisation non convexes de grande taille. La thèse comporte trois parties :
la première partie est consacrée aux techniques d'optimisations locales et s'articule autour des méthodes de points intérieurs et de la programmation DC. Nous y développons deux algorithmes. Après une présentation non exhaustive de la programmation DC, des méthodes de points intérieurs et des propriétés essentielles de la classe des matrices quasi-définies au chapitre un, nous présentons au chapitre deux un nouvel algorithme basé sur une reformulation des conditions d'optimalité de Karush-Kuhn-Tucker. Le troisième chapitre est consacré à l'intégration des techniques d'optimisation DC dans un schéma de points intérieurs, c'est l'algorithme IPDCA.
La seconde partie de la thèse est consacrée aux solutions globales de problèmes de programmation quadratique. Dans le premier chapitre de cette partie nous explorons l'intégration d'IPDCA dans un schéma B&B. Le second chapitre de la partie est consacré à la résolution de problèmes quadratiques à variables 0-1 par un schéma B&B dans lequel nous faisons intervenir IPDCA. Le troisième chapitre est quant à lui consacré à l'optimisation monotone due au Professeur Tuy. Nous examinons plus particulièrement son intégration dans un B&B dans lequelle DCA est appelé pour améliorer la borne supérieure. Le quatrième et dernier chapitre de cette partie est consacré à une procédure de redémarrage de DCA.
La dernière partie de la thèse est consacrée aux applications industrielles. Nous y appliquons les deux algorithmes développés dans la première partie de la thèse à un problème de mécanique de structure de grande dimension et à un problème en Data Mining.

This Ph.D Thesis is mainly devoted to the combination of DC Programming techniques and the Interior Point programming approaches to deal with large scale nonconvex programs.
After a brief review of the general DC theory and algorithms, the interior point scheme is presented, the quasidefinite matrices class and its essential properties are reminded. In the second chapter we introduce a new algorithm for nonlinear problems.The algorithm is based on Chen-Harker-Kanzow-Smale regularization technique and filter approach. In the third chapter we present our main contribution, a new primal-dual interior point algorithm for solving nonconvex programming problem. The new algorithm IPDCA, follows the DCA scheme to handle nonconvexity. The fourth chapter is devoted to the integration of IPDCA in a B&B to globally solve nonconvex quadratic programming problems. In the fifth chapter we integrate IPDCA in a B&B to globally solve a 0-1 quadratic programming problems. In the sixth chapter we integrate DCA in a B&B procedure in which the lower bound is computed using monotonic optimization introduced by H. Tuy. In the seventh chapter we proposed a restarting procedure for DCA based on global optimality conditions for convex quadratic maximization problems. In the eighth chapter the two algorithms developed in chapter two and in chapter three are applied to solve a large scale structural mechanics problem. In the last chapter we use IPDCA for a machine learning with non positive kernels. The problem is a large scale nonconvex quadratic programming problem with a dense hessian matrix.