Méthodes de points intérieurs et leurs applications sur des problèmes d'optimisation semi-définis

Amina Zerari

Normandie Université

THESE

Pour obtenir le diplôme de doctorat

Spécialité : **Mathématiques Appliquées**

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

Méthodes de points intérieurs et leurs applications sur des problèmes d'optimisation semi-définis

Présentée et soutenue par
Amina ZERARI

Thèse soutenue publiquement le (09/12/2020) devant le jury composé de		
Mme Sonia CAFIERI	Professeur, Ecole Nationale de l'Aviation Civile – Université de Toulouse	Examinateur
Mme Souad EL BERNOUSSI	Professeur, Université Mohammed V - Rabat	Rapporteur
M. Bachir MERIKHI	Professeur, Université Ferhat ABBAS Sétif-1	Examinateur
Mme Khadra NACHI	Professeur, Université d'Oran 1	Rapporteur
M. Djamel BENTERKI	Professeur, Université Ferhat ABBAS Sétif-1	Co-Directeur de thèse
M. Adnan YASSINE	Professeur, Université Le Havre Normandie	Directeur de thèse

Thèse en cotutelle dirigée par Adnan YASSINE du Laboratoire de Mathématiques Appliquées du Havre (LMAH) à l'Université Le Havre Normandie et Djamel BENTERKI de l'Université Ferhat ABBAS Sétif 1 en Algérie.

Logo Etablissement

UNIVERSITÉ LE HAVRE NORMANDIE

Logo Ecole Doctorale

MI ΙΣ

Logo Laboratoire

LMAH

REMERCIEMENTS

Cette thèse devient une réalité avec le soutien et l'aide de nombreuses personnes. Ces lignes me donnent l'occasion de les remercier.

Mes premiers remerciements vont aux mes directeurs de thèse :

- ***M. Dj. BENTERKI**, Professeur à l'université Ferhat Abbas sétif-1 pour le soutien continu, sa patience, sa motivation et ses conseils tout au long de ces années de recherche.*
- ***M. A. YASSINE**, Professeur à l'université Le Havre Normandie, d'avoir accepté d'encadrer cette thèse, pour ses relectures, ses remarques et ses commentaires pertinents.*

*Je remercie **M. B. MERIKHI**, Professeur à l'université Ferhat Abbas sétif-1, de m'avoir fait l'honneur de faire partie de ce jury et d'en être le président.*

Je désire remercier également :

- ***Mme. S. EL BERNOUSSI**, Professeur à l'université Mohammed V – Agdal.*
- ***Mme. S. CAFIERI**, Professeur à l'école Nationale de l'Aviation Civile, France.*
- ***Mme. K. NACHI**, Professeur à l'université Oran1 Ahmed Ben Bella.*

D'avoir accepté juger ce travail et d'en être les examinateurs.

Je voudrais adresser mes remerciements à tous les membres du laboratoire de mathématiques fondamentales et numériques, et à toute l'équipe administrative du département de mathématiques à l'université Ferhat Abbas sétif-1.

*Je remercie mes chères amies pour leurs aides continues et leurs encouragements, particulièrement : **Ahlem BENNANI, Nadia HAZZEM, Haizia Bounajaa,…***

*Je remercie **Dr. Ikram BOURAS**, tu as toujours été là quand j'avais besoin de toi.*

Je remercie ma famille : mes parents, ma belle-mère, mes sœurs et mes frères pour le soutien et pour l'encouragement.

Enfin, merci à mon mari pour son soutien, sa compréhension et son sacrifice durant l'élaboration de cette thèse.

ملخص :

تعد طرق النقطة الداخلية الاكثر شهرة و الأكثر فعالية في حل مسائل البرمجة المثالية. هذه الطرق تتميز بتقاربها الحدودي وسلوكها الجيد. في هذا البحث نهتم بالدراسة النظرية و العددية لخوارزميات النقطة الداخلية للبرمجة نصف معرَفة.

في الجزء الأول ، قدمنا خوارزمية الإسقاط للنقطة الداخلية الأولية-الثنوية مع مرحلتين ، حيث انجزنا ثلاثة طرق فعالة جديدة لحساب خطوة الانتقال.

و في الجزء الثاني، اهتممنا بطريقة من نوع المسار المركزي المعتمد على دالة النواة. لهذا اقترحنا دالتي نواة جديدتين بحاجز لوغاريتمي و اللتان تعطيان أحسن تكلفة للخوارزمية محصل عليها إلى يومنا هذا.

الكلمات المفتاحية : البرمجة نصف المعرَفة ، طريقة النقط الداخلية ، طريقة الإسقاط ، دالة النواة ، تكلفة الخوارزمية.

Abstract:

Interior point methods are well known as the most efficient to solve optimization problems. These methods have a polynomial convergence and good behavior. In this research, we are interested in a theoretical, numerical and an algorithmic study of interior-point methods for semidefinite programming.

Indeed, we present in a first part, a primal-dual projective interior point algorithm of polynomial type with two phases, where we introduced three new effective alternatives for computing the displacement step.

Then, in the second part, we are interested in a primal-dual central trajectory method via a kernel function, we proposed two new kernel functions with a logarithmic term that give the best-known complexity results.

Keywords: Semidefinite programming, Interior point method, Projective method, Kernel function, Algorithmic complexity.

Résumé :

Les méthodes de points intérieurs sont bien connues comme les plus efficaces pour résoudre les problèmes d'optimisation. Ces méthodes possèdent une convergence polynômiale et un bon comportement numérique. Dans cette recherche, nous nous sommes intéressés à une étude théorique, algorithmique et numérique des méthodes de points intérieurs pour la programmation semi-définie.

En effet, on présente dans une première partie un algorithme réalisable projectif primal-dual de points intérieurs de type polynômial à deux phases, où on a introduit trois nouvelles alternatives efficaces pour calculer le pas de déplacement.

Ensuite, dans la deuxième partie, on s'intéresse aux méthodes de type trajectoire centrale primale-duale via une fonction noyau, nous proposons deux nouvelles fonctions noyaux à terme logarithmique qui donnent la meilleure complexité algorithmique, obtenue jusqu'à présent.

Mots clés : Programmation semi-définie, Méthode de points intérieurs, Méthode projective, Fonction noyau, Complexité algorithmique.

Table des matières

Introduction

La programmation mathématique, branche de l'optimisation, s'occupe de la minimisation (ou la maximisation) sous contraintes d'une fonction à plusieurs variables, schéma très général s'appliquant à de nombreuses situations pratiques dans beaucoup de domaines (minimisation de coûts, de durées, maximisation de gains, de rentabilité, etc.).

Le problème de la programmation semi-définie SDP est un domaine important et récent de la programmation mathématique. La plupart des travaux de recherche sur SDP ont été réalisés dans les années 90, bien que ses racines remontent à quelques décennies de plus (voir par exemple Bellman et Fan [8]). Un article sur la programmation semi-définie paru en 1981 intitulé "Linear Programming with Matrix Variables" (Craven et Mond [17]), représente l'ouvrage le plus convenable et la meilleure façon d'introduire le problème SDP. L'objectif est de minimiser le produit scalaire de deux matrices symétriques, à savoir une matrice constante C et une matrice variable X semi-définie-positive ($X \succeq 0$), soumis à un ensemble de contraintes. Le problème SDP est une généralisation de la programmation linéaire PL, où les vecteurs sont remplacés par des matrices et l'orthant positif $\mathbb{R}^n_+$, est remplacé par le cône des matrices semi-définies positives. Ce qui explique donc le transport du savoir faire de la programmation linéaire à la programmation semi-définie.

Dans la dernière décennie, SDP a été l'un des domaines de recherche les plus actifs dans la programmation mathématique. Il y a deux principaux facteurs qui sont responsables de cet accru intérêt pour SDP. Premièrement, SDP intervient dans différents problèmes pratiques et mathématiques de grande importance, à savoir : la théorie du contrôle, l'optimisation combinatoire, la programmation non linéaire, le problème de coupure maximale dans la théorie des graphes et le problème de min-max des valeurs propres. Deuxièmement, la renaissance des méthodes de points intérieurs après les investigations effectuées par Karmarkar [32] pour la programmation linéaire PL.

Cependant, un nouveau type de méthodes de résolution a fait son apparition en 1984 [32] dites méthodes de points intérieurs. La plupart des idées sous-jacentes à ces méthodes proviennent du domaine de la programmation non linéaire. Parmi leurs avantages, citons :

1. Efficacité théorique : ces méthodes ont une complexité polynomiale et

s'exécutent en temps polynomial.

2. Efficacité pratique : les temps de calcul et de réponse de ces méthodes sont compétitifs.

3. Traitement de très grands problèmes : ces méthodes permettent de résoudre des problèmes de très grande taille en un temps acceptable et raisonnable.

4. Adaptation au cas non linéaire : il est possible d'adapter ces méthodes à la programmation non linéaire, plus particulièrement à la programmation convexe, ce qui permet de traiter de nouveaux types de problèmes pour lesquels on ne connaissait jusqu'à présent aucune méthode de résolution efficace.

On désigne par méthode de points intérieurs (MPI), toute procédure de résolution générant une suite de points appartenant à l'intérieur (relatif) du domaine réalisable (admissible) et convergeant vers une solution optimale du programme considéré. Il y a principalement trois grandes catégories de méthodes de points intérieurs : les méthodes affines, les méthodes de réduction du potentiel et les méthodes de trajectoire centrale.

On peut distinguer différents MPI, selon qu'il s'agit des MPI réalisables ou non-réalisables. Les MPI réalisables commencent par une matrice strictement réalisable et conservent la faisabilité pendant le processus de résolution. Cependant, pour obtenir un tel point, il est généralement aussi difficile que de résoudre le problème sous-jacent. Les MPI non-réalisables pour les problèmes de programmation semi-définis commencent par une matrice définie-positive arbitraire. La performance des MPI non réalisables existantes dépend fortement du choix du point de départ.

Le domaine des méthodes de points intérieurs pour PL a commencé avec l'algorithme d'ellipsoïde de Khachiyan en 1979, qui a permis de lier le nombre d'itérations par un polynôme, ce qui a répondu à la question de savoir si des problèmes de programmation linéaire peuvent être résolus en temps polynomial, mais les expériences pratiques par la méthode d'ellipsoïde ont été décevantes. Suivi par le célèbre article de Karmarkar [32] en 1984, qui a introduit un algorithme avec une complexité polynômiale améliorée, ceci a également été accompagnée par l'efficacité du comportement numérique de l'algorithme. Dans la décennie suivante, des milliers de documents, basés sur l'algorithme de Karmarkar, sont apparus sur ce sujet.

Les méthodes de points intérieurs ont fait récemment l'objet de plusieurs monographies dont Roos, Terlaky et Vial [56], Wright [69], Ye [70] et Nesterov et Nemirovskii [49]. Un excellent survol des méthodes de points intérieurs pour la programmation linéaire est celui de Gonzaga [26]. Il a fallu presque dix ans pour étayer les prétentions de l'efficacité des méthodes de points intérieurs. Plusieurs études ont indiqué que ces méthodes ont des performances supérieures aux algorithmes de type simpliciale. Voir par exemple, l'état de l'art sur les problèmes de grande taille (Lustig et al. [40] et plus récemment E. D. Andersen et K. D. Andersen [4]).

Ayant des liens entre PL et SDP, on n'est pas surpris que les algorithmes de points intérieurs pour PL ont été étendus avec succès à SDP. La première extension des algorithmes de points intérieurs de PL à SDP a été réalisée par Nesterov et Nemirovski [49], et indépendamment par Alizadeh [1] en 1991. Cela explique le regain d'intérêt de la recherche dans SDP dans les années 1990. Dans ses travaux [1, 2], Alizadeh a étendu la méthode projective de réduction du potentiel de Ye à partir de PL à SDP. D'autre part, Nesterov et Nemirovski [49] ont présenté une théorie profonde et unifiée des méthodes de points intérieurs pour résoudre les problèmes d'optimisation coniques plus généraux en utilisant la notion des fonctions barrières auto-concordantes.

Récemment, Kojima, Shindoh et Hara [35], Monteiro [47] et Y. Zhang [72] ont présenté des algorithmes primaux-duaux de points intérieurs pour la programmation semi-définie (ainsi pour la version de complémentarité linéaire semi-définie issue du problème SDP) qui sont généralisés à partir des algorithmes similaires conçus pour la programmation linéaire PL (ou pour le problème de complémentarité linéaire (PCL)). Leurs directions de recherche sont obtenues à partir d'une équation modifiée (en utilisant un facteur de paramétrisation) de Newton pour approcher une solution réalisable sur le chemin central. La convergence polynômiale de ces algorithmes a été étudiée et prouvée par plusieurs chercheurs [27, 31, 46, 48, 51].

En 1990, Monteiro et al. [45] ont montré la convergence polynômiale de la méthode affine primale-duale. La généralisation des méthodes de points intérieurs MPI de la programmation linéaire PL à la programmation semi-définie SDP a commencé au début des années 90. Les premiers travaux sur les MPI pour SDP ont été développés indépendamment par Alizadeh [2] et Nesterov et Nemirovski [49]. En 1995, Alizadeh [2] a proposé une méthode de réduction du potentiel pour SDP. Presque en même temps, en 1994, Nesterov et al. [49] ont prouvé que MPI sont capables de résoudre les problèmes d'optimisation conique généraux, en particulier les problèmes SDP, en un temps polynomial. En 1996, Vanderbeghe et al. [63] ont proposé un algorithme primal-dual de la méthode de réduction du potentiel. En 1997, Monteiro [47], a proposé une méthode de trajectoire centrale primale-duale. En 1999, Monteiro et al. [46] ont proposé une méthode primale-duale de type trajectoire centrale et ont prouvé que la complexité de son algorithme est polynomial. Au même temps, Ji et al. [37] ont étudié la convergence de la méthode de trajectoire centrale de type prédicteur-correcteur. En 2002, Halicka et al. [27] ont proposé une étude sur la convergence de la méthode de trajectoire centrale en programmation semi-définie. Une année après, Benterki et al. [10] ont fait une étude aménagée théorique et numérique de l'algorithme proposé par Alizadeh [2]. Les travaux se poursuivent à présent, on trouve ainsi ceux de Koulaei et al. [36] en (2007), qui se sont intéressés à l'extension de l'algorithme de type Mehrotra pour la programmation semi-définie. La même année, Benterki et al. [11] ont proposé une méthode de points intérieurs

réalisable pour résoudre le problème SDP, l'avantage de cette dernière est qu'elle fournit une solution strictement réalisable initiale pour SDP. En 2012, Liu et al. [39] ont présenté un nouveau algorithme de type correcteur de second ordre pour SDP et ont prouvé sa convergence polynômiale pour la direction de Nesterov et Todd (NT). En 2015, Kettab et al. [33] ont proposé une relaxation de la méthode barrière logarithmique pour la programmation semi-définie. La plupart de ces plus récents travaux sont concentrés sur les méthodes primales-duales.

Une extension des MPI primales-duales de PL à SDP a d'abord été réalisée par Nesterov et Nemirovski [49] et a permis d'obtenir une complexité polynômiale permettant de résoudre des problèmes coniques en introduisant les fonctions barrières auto-concordantes, constituées de la fonction barrière logarithmique. Peng et al. [53] ont proposé un nouveau paradigme dans l'MPI classique pour résoudre PL et quelques autres extensions de ce problème dans lesquelles la fonction barrière logarithmique est remplacée par les fonctions barrières Self-Regular (SR). La complexité des itérations de PL et de ses extensions, basées sur les fonctions barrières SR, les ont conduits à obtenir la meilleure complexité pour les méthodes à petit et à grand pas, $O(\sqrt{n}\log\frac{n}{\varepsilon})$ et $O(\sqrt{n}\log n\log\frac{n}{\varepsilon})$, respectivement. Notons que, basée sur la fonction barrière de la famille logarithmique, ces bornes sont $O(\sqrt{n}\log\frac{n}{\varepsilon})$ et $O(n\log\frac{n}{\varepsilon})$, respectivement [56]. De plus, une classe des MPI primales-duales pour PL basée sur une nouvelle famille de fonctions noyaux, assez générale et inclut la fonction logarithmique classique, la fonction SR et certaines fonctions noyaux non-SR, telles que son cas particulier a été proposé par Bai et al. dans [7]. De nombreux chercheurs ont proposé différents algorithmes de points intérieurs basés sur la fonction noyau pour divers problèmes d'optimisation. Roos et al. [6] définissent les fonctions noyaux élligibles et ont proposé une MPI primale-duale pour PL et ont simplifié l'analyse de complexité de Peng et al.'s dans [53]. EL Ghami et al. [23] ont proposé un algorithme primal-dual de points intérieurs pour PL basé sur une fonction noyau avec un terme barrière trigonométrique, mais ils n'ont pas obtenu le meilleur résultat de complexité connu pour la méthode à grand pas. Bouafia et al. [14] ont proposé un algorithme primal-dual de points intérieurs pour PL basé sur une fonction noyau avec un terme barrière trigonométrique et ont obtenu le meilleur résultat de complexité connu pour la méthode à petit et à grand pas.

Plusieurs méthodes de points intérieurs (MPI) pour PL ont été étendues avec succès à SDP. Wang et al. [65] ont proposé une MPI primale-duale pour SDP basé sur une version généralisée de la fonction noyau dans [6] et ont obtenu une complexité $O(q^2\sqrt{n}\log\frac{n}{\varepsilon})$, $q>1$, et $O(\sqrt{n}\log n\log\frac{n}{\varepsilon})$ pour les petit et grand pas, respectivement. EL Ghami et al. [22] ont étendu les MPI pour PL dans [6] à SDP et ont obtenu les mêmes résultats de complexité obtenus dans PL. EL Ghami et al. [24] ont proposé une MPI primale-duale pour SDP basée sur une version généralisée de la fonction noyau donnée dans [5] et ont obtenu $O(\sqrt{n}\log n\log\frac{n}{\varepsilon})$ pour la méthode à grand pas. Lee et al. [38] ont défini une nouvelle classe de

fonctions noyaux et ont obtenu les meilleurs résultats de complexité connus pour les méthodes à petit et à grand pas, pour PL et SDP. EL Ghami [21] a généralisé l'analyse présentée dans [14] pour SDP et a obtenu les meilleurs résultats de complexité connus pour la méthode à petit et à grand pas. Récement Fathi-Hafshejani et al. [25] ont proposé une nouvelle fonction noyau et ont obtenu les meilleurs résultats de complexité connus pour les méthodes à petit et à grand pas, pour SDP. Touil et al. [60] ont proposé une nouvelle fonction noyau paramétrisé et ont obtenu $O(\sqrt{n}(\log n)^{\frac{pq+1}{pq}} \log \frac{n}{\varepsilon})$ pour la méthode à grand pas. Moaberfard et al. [44] ont proposé une nouvelle fonction noyau trigonométrique et ont obtenu le meilleur résultat de complexité connus pour les méthodes à grand pas, pour PL.

Notre travail apporte sur des contributions théoriques, algorithmiques et numériques pour résoudre convenablement des problèmes d'optimisation de la programmation semi-définie linéaire. Il est réparti en deux volets :

Le premier concerne le développement d'un algorithme réalisable projectif primal-dual de points intérieurs de type polynômial à deux phases. La première phase donne une solution initiale strictement réalisable de (SDP), et la deuxième donne une solution optimale primale-duale de (SDP) et son dual (DSDP). Pour chaque phase, nous introduisons une procédure efficace pour calculer le pas de déplacement utilisant trois nouvelles alternatives différentes. Ces nouveaux résultats théoriques sont consolidés par des expérimentations numériques efficaces réalisées sur plusieurs exemples.

Dans le deuxième volet, on s'intéresse plus particulièrement, aux méthodes de type trajectoire centrale primale-duale via une fonction noyau, à cause de leurs succès numériques vis-à-vis d'autres classes de méthodes de points intérieurs. Nous avons proposé deux nouvelles fonctions noyaux à terme logarithmique permettant d'obtenir la meilleure complexité algorithmique, connue jusqu'à présent. Les tests numériques effectués sur plusieurs exemples confirment les propos théoriques.

Cette thèse est répartie en quatres chapitres. Le premier chapitre présente un rappel sur des notions fondamentales d'usage fréquent pour la suite, à savoir : L'analyse convexe, quelques résultats de programmation mathématique et les notions des principales propriétés des cônes des matrices symétriques semi-définies positives.

Le chapitre 2, est consacré à la programmation semi-définie. On donne tout d'abord, sa formulation générale, sa dualité, quelques domaines d'application et on termine par un bref aperçu historique résumant les principaux résultats durant la dernière décennie ainsi que les méthodes de résolution.

Dans le troisième chapitre, on propose une méthode réalisable primale-duale projective de points intérieurs pour résoudre SDP basée sur celle introduite par D. Benterki et al. [11] dans laquelle, on introduit trois nouvelles procédures donnant lieu au calcul effectif du pas de déplacement intervenant dans l'algorithme

et qui constituent l'originalité de notre travail. Ce chapitre est étayé par des expérimentations numériques d'ordre comparatifs sur quelques exemples connus dans la littérature.

Dans le dernier chapitre, nous proposons un algorithme de points intérieurs primal-dual pour SDP basé sur deux nouvelles fonctions noyaux de type logarithmique. Nous analysons ainsi les résultats de complexité des méthodes à petit et à grand pas. Ce chapitre est achevé par des expérimentations numériques d'ordre comparatif sur quelques exemples tests.

Enfin, nous clôturons ce travail par une conclusion générale résumant les nouveaux résultats obtenus dans cette thèse ainsi que leur nouveauté et des perspectives pour des futurs travaux de recherche suivies par une bibliographie.

Chapitre 1

Notions fondamentales

Dans ce chapitre, nous allons introduire certaines notions et résultats de calcul matriciel et quelques propriétés des matrices symétriques (en particulier, semi-définie-positive) ainsi que des notions de base de l'analyse convexe et de la programmation mathématique qui seront utiles par la suite.

1.1 Rappel matriciel

Dans cette partie, on donne quelques notions utiles concernant l'analyse matricielle. Notons tout d'abord par

$$M_n = \left\{A \in \mathbb{R}^{n\times n}\right\}$$

l'ensemble des matrices carrées d'ordre n à coefficients dans $\mathbb{R}$.

et l'ensemble

$$\mathbb{S}^n = \left\{A \in M_n \ / \ A^T = A\right\}$$

designe l'espace des matrices carrées symétriques d'ordre n à coefficients dans $\mathbb{R}$.

1.1.1 Valeurs propres et spectre

1. Soit la matrice $A \in M_n$, alors A est régulière (ou inversible) si et seulement si $det(A) \neq 0$ où $det(A)$ est le déterminant de la matrice A.
2. Pour une matrice $A \in M_n$, le polynôme à variable λ,

$$p_A(\lambda) = det(\lambda I_n - A),$$

est appelé polynôme caractéristique de A. Les racines de $p_A(\lambda)$ sont dites les valeurs propres de A.

3. Soient $\lambda_1, ..., \lambda_m$ les valeurs propres de la matrice A d'ordre n. Alors le polynôme caractéristique peut être représenté par

$$p_A(\lambda) = p_0 + p_1\lambda + ... + p_{n-1}\lambda^{n-1} + \lambda^n = (\lambda - \lambda_1)^{n_1}...(\lambda - \lambda_m)^{n_m},$$

où les n_i sont des nombres entiers positifs avec $\sum_{i=1}^{m} n_i = n$. Le nombre n_i est le multiplicateur ou le multiplicateur algébrique de la valeur propre λ_i, $i = 1, ..., m$.

4. Une valeur propre est dite simple si son multiplicateur est égal à 1.
5. Il est important de signaler que le déterminant d'une matrice A est le produit de ses valeurs propres.
6. Le spectre de $A \in M_n, n \geq 1$, est l'ensemble des valeurs propres de A

$$\sigma(A) = \{\lambda \in \mathbb{R}, \quad (\lambda I_n - A) \text{ est singulière (i.e., n'est pas inversible)}\}.$$

7. Le rayon spectral de A est la plus grande valeur propre de A en valeur absolue

$$\rho(A) = \underset{\lambda \in \sigma(A)}{max} |\lambda|$$

8. Si $A \in \mathbb{S}^n$, alors
   (a) Toutes les valeurs propres de A notées $\lambda_i(A)$ sont réelles.
   (b) Il existe une base de vecteurs orthonormés dans laquelle A est diagonalisable, i.e., il existe une matrice P orthogonale (dite de passage) telle que $A = PDP^T$ où D est une matrice diagonale. Les colonnes u_i de P sont les vecteurs propres de A et les valeurs propres (λ_i) de A sont les coefficients diagonaux de D (dans ce cas les matrices A et D sont dites semblables).
9. Pour nos besoins, il convient d'écrire les valeurs propres dans l'ordre croissant,

$$\lambda_{min}(A) = \lambda_1(A) \leq \lambda_2(A) \leq ... \leq \lambda_n(A) = \lambda_{max}(A).$$

Soient A et B deux matrices de $\mathbb{S}^n$, alors

(a) $\lambda_{min}(A + B) \geq \lambda_{min}(A) + \lambda_{min}(B)$,

(b) $\lambda_{max}(A+B) \leq \lambda_{max}(A) + \lambda_{max}(B)$.

10. Soit la matrice $A \in M_n$, alors on a

    (a) Si A est orthogonale, alors les valeurs propres de A sont -1 ou 1.

    (b) A est régulière (ou inversible) si et seulement si 0 n'est pas une valeur propre de A.

11. Soit A une matrice inversible de M_n, alors

    (a) λ est une valeur propre de A si et seulement si λ est une valeur propre de A^T.

    (b) λ est une valeur propre de A si et seulement si λ^{-1} est une valeur propre de A^{-1}.

    (c) λ est une valeur propre de A si et seulement si $\lambda + \beta$ est une valeur propre de $A + \beta I_n$.

    (d) λ est une valeur propre de A si et seulement si $\beta\lambda$ est une valeur propre de βA.

1.1.2 Trace, produit scalaire et norme

1. La trace d'une matrice $A \in M_n$ est définie par la somme de ses éléments diagonaux
$$Tr(A) = \sum_{i=1}^{n} a_{ii} \quad \forall A \in M_n,$$

2. La trace est une application linéaire et pour $A \in M_n$ elle est égale à la somme de ses valeurs propres.

3. Pour tous $A, B, C \in M_n$ et $\alpha, \beta \in \mathbb{R}$ on a les propriétés suivantes :

    (a) $Tr(\alpha A + \beta B) = \alpha Tr(A) + \beta Tr(B)$ (Linéarité),

    (b) $Tr(A^T) = Tr(A)$,

    (c) $Tr(A^2) \leq Tr(A^T A)$ (Commutativité)

    (d) $Tr(AB) = Tr(BA)$ (Invariance par permutation),

    (e) $Tr(AB) \leq \frac{1}{2}(Tr(A^2 + B^2))$ avec A, B symétriques,

    (f) $Tr(ABC) = Tr(CAB) = Tr(BCA) \neq Tr(ACB)$,

    (g) $Tr(BAB^{-1}) = Tr(A)$.

4. Le produit scalaire usuel de deux vecteurs x et y de $\mathbb{R}^n$ est défini par

$$< x, y >= \sum_{i=1}^{n} x_i y_i = x^T y$$

5. De même, on définit un produit scalaire sur l'ensemble des matrices carrées réelles. Le produit scalaire de deux matrices $A, B \in M_n$ est défini par

$$A \bullet B = Tr(A^T B) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} b_{ij} = B \bullet A$$

Enonçons maintenant les propriétés du produit scalaire $A \bullet B$: Pour cela, on considère $A, B, C \in M_n$ et $\alpha, \beta \in \mathbb{R}$ alors :

(a) $< A + B, C >=< A, C > + < B, C >$,

(b) $< \alpha A, \beta B >= \alpha\beta < A, B >$,

(c) $< A, B >=< B, A >$.

6. La norme vectorielle est une application de $\mathbb{R}^n$ dans $\mathbb{R}_+$, notée par $||.||$ et vérifie les conditions suivantes :

(a) $\forall x \in \mathbb{R}^n : ||x|| = 0 \Longleftrightarrow x = 0$,

(b) $\forall \lambda \in \mathbb{R}, \forall x \in \mathbb{R}^n : ||\lambda x|| = |\lambda| ||x||$,

(c) $\forall x, y \in \mathbb{R}^n : ||x + y|| \leq ||x|| + ||y||$.

7. Soient $A, B \in M_n$, l'application $||.|| : M_n \rightarrow \mathbb{R}_+$ est appelée norme matricielle si elle vérifie les conditions suivantes :

(a) $||A|| = 0 \Leftrightarrow A = 0$,

(b) $||\alpha A|| = |\alpha| ||A||, \alpha \in \mathbb{R}$,

(c) $||A + B|| \leq \ ||A|| + ||B||$,

(d) $||AB|| \leq ||A|| ||B||$.

8. Pour toute matrice $A \in M_n$, on a :

(a) $||A||_1 = \max\limits_{j=1}^{n} \left(\sum\limits_{i=1}^{n} |a_{ij}| \right)$,

(b) $||A||_\infty = \max\limits_{i=1}^{n} \left(\sum\limits_{j=1}^{n} |a_{ij}| \right)$,

(c) $||A||_2 = \sqrt{\max_{i=1}^{n}|\lambda_i|} = \sqrt{\rho(A^TA)} \qquad \forall A \in M_n$, est appelée la norme euclidienne, où $\lambda_i, i = 1, ..., n$, sont les valeurs propres de la matrice A^TA.

9. On utilisera également la norme de Frobenius, associée au produit scalaire

$$||A||_F = \sqrt{A \bullet A} = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}(a_{ij})^2} \quad \forall A \in M_n.$$

et on a les propriétés suivantes :

(a) $||A||_F = ||A^T||_F$,

(b) $||A||_2 \leq ||A||_F \leq \sqrt{n}||A||_2$,

(c) $||\alpha A||_F = |\alpha|||A||_F, \quad \forall \alpha \in \mathbb{R}$,

(d) $||A + B||_F \leq ||A||_F + ||B||_F$ (Inégalité triangulaire),

(e) $||AB||_F \leq ||A||_F||B||_F$,

(f) $||A + B||_F^2 + ||A - B||_F^2 = 2(||A||_F^2 + ||B||_F^2)$. (Identité de Parallélogramme),

(g) Si $< A, B >= 0$, alors $||A + B||_F^2 = ||A - B||_F^2 = ||A||_F^2 + ||B||_F^2$ (Théorème de Pythagore),

(h) $< A, B >= \frac{1}{4}(||A + B||_F^2 - ||A - B||_F^2)$,

(i) $< A, B >= \frac{1}{2}(||A + B||_F^2 - ||A||_F^2 - ||B||_F^2)$.

10. On note que si : $A \in \mathbb{S}^n$, alors on obtient facilement les résultats suivants :

$$||A||_F = \sqrt{(A^TA)} = \sqrt{Tr(A^2)} = \sqrt{\sum_{i=1}^{n}\lambda_i^2(A)},$$

et

$$||A||_2 = \sqrt{(\lambda_{\max}A^2)} = \max_{i=1}^{n}|\lambda_i(A)| = \rho(A),$$

de plus,

$$||A||_2 \leq ||A||_F \leq \sqrt{n}||A||_2,$$

et pour toute norme matricielle on a :

$$\lambda(A) \leq ||A||.$$

11. L'inégalité de Cauchy-Schwarz s'écrit alors pour toutes matrices

$$A, B \in M_n \quad |A \bullet B| \leq ||A||_F ||B||_F.$$

12. Soit $A \in \mathbb{S}^n$, on a

$$\lambda_{\max}(A) = \max_{||v||=1} \langle v, Av \rangle \qquad \text{(Rayleigh-Ritz)}$$

ici $||v||$ est la norme euclidienne du vecteur $v \in \mathbb{R}^n$.

1.1.3 Matrices semi-définies positives et leurs propriétés

Nous sommes intéressés dans ce travail par les matrices symétriques semi-définies positives.

Définition 1.1.1 *Une matrice $A \in \mathbb{S}^n$ est dite*
- Semi-définie-positive, et on écrira $A \succeq 0$, si

$$\langle x, Ax \rangle = x^T Ax \geq 0, \quad \forall x \in \mathbb{R}^n$$

et on note par $\mathbb{S}^n_+$ l'ensemble des matrices semi-définies positives.
- définie-positive, et on écrira $A \succ 0$, si

$$\langle x, Ax \rangle > 0, \ \ \forall x \in \mathbb{R}^n - \{0\}$$

et on note par $\mathbb{S}^n_{++}$ l'ensemble des matrices définies positives.

Théorème 1.1.1 *[28] Soit $A \in \mathbb{S}^n$, alors les propositions suivantes sont équivalentes :*

1. *$A \in \mathbb{S}^n_+$ (resp. $A \in \mathbb{S}^n_{++}$);*
2. *$\lambda_i(A) \geq 0$ (resp. $\lambda_i(A) > 0$) $\forall i = 1, ..., n$;*
3. *Il existe une matrice $C \in M_n$ telle que $A = C^T C$;*
4. *Tous les mineurs principaux dominants de A sont positifs (resp. strictement positifs).*

Propriétés

1. $A, B \in \mathbb{S}^n_+$, alors
   (a) $A \succeq B \Leftrightarrow A - B \succeq 0$,

(b) $A + B \succeq B$,

(c) $A^{\frac{1}{2}} B A^{\frac{1}{2}} \succeq 0$,

(d) $Tr(AB) \leq Tr(A)Tr(B)$,

(e) $Tr(AB) \geq 0$.

2. Il est facile de constater qu'une matrice $A \in \mathbb{S}^n_{++}$ si et seulement si $A^{-1} \in \mathbb{S}^n_{++}$, puisque les valeurs propres de A^{-1} sont $\frac{1}{\lambda_i(A)}$ pour tout $i = 1, ..., n$.
3. Toute sous-matrice principale d'une matrice semi-définie (resp. définie) positive est aussi semi-définie (resp. définie) positive.
4. Pour tout $A \in \mathbb{S}^n_+$, il existe $i \in \{1, ..., n\}$ tel que $a_{ii} = \max_{i,j \in \{1,...,n\}} |a_{ij}|$.
5. Si $A \in \mathbb{S}^n_+$ et $a_{ii} = 0$ pour un certain $i \in \{1, ..., n\}$ alors $a_{ij} = 0$ pour tout $j \in \{1, ..., n\}$.
6. Soit $B \in M_n$ une matrice inversible, alors $A \in \mathbb{S}^n_+$ (resp. $\mathbb{S}^n_{++}$) $\iff$ $B^T AB \in \mathbb{S}^n_+$ (resp. $\mathbb{S}^n_{++}$).
7. On a l'équivalence suivante :

$$A \succeq 0 \iff A \bullet B \geq 0, \quad \forall B \succeq 0.$$

8. On a aussi

$$A \succ 0 \iff A \bullet B > 0, \quad \forall B \succeq 0 \text{ non nulle.}$$

9. Si $A, B \in \mathbb{S}^n_+$, alors

$$A \bullet B = 0 \iff AB = 0 \Leftrightarrow \frac{1}{2}(AB + BA) = 0$$

10. Soit $A, B \in \mathbb{S}^n$, alors

(a) Si $A \succeq 0$, alors $||A||_F \leq Tr(A)$ et $n(\det(A))^{\frac{1}{n}} \leq Tr(A)$

(b) Si $C, D \in \mathbb{S}^n$ telles que $C - A \succeq 0$ et $D - B \succeq 0$ alors

$$Tr(AB) \leq Tr(CD)$$

En particulier, si $C = A$ alors $Tr(AB) \leq Tr(AD)$

(c) $A \succeq B \iff C^T AC \succeq C^T BC, \quad \forall C \in M_n$

(d) Si $A \succeq I_n$ alors A est inversible et $I_n \succeq A^{-1}$

(e) Si $B \succeq A \succ 0$ alors B est inversible ($B \succ 0$) et $A^{-1} \succeq B^{-1}$.

11. $\lambda_{\min}(A)\lambda_{\max}(B) \leq \lambda_{\min}(A)Tr(B) \leq A \bullet B \leq n\lambda_{\max}(A)Tr(B) \leq n^2\lambda_{\max}(A)\lambda_{\max}(B)$.

Théorème 1.1.2 *Si $A \in \mathbb{S}^p_{++}, C \in \mathbb{S}^n$ et $B \in \mathbb{R}^{p\times n}$ alors*

$$\begin{pmatrix} A & B \\ B^T & C \end{pmatrix} \succeq 0 \iff C - B^T A^{-1} B \succeq 0$$
$$\begin{pmatrix} A & B \\ B^T & C \end{pmatrix} \succ 0 \iff C - B^T A^{-1} B \succ 0$$

La matrice $S = C - B^T A^{-1} B$ s'appelle complément de Schur de A. En particulier, pour tout $x \in \mathbb{R}^n$ et $X \in \mathbb{S}^n$

$$\begin{pmatrix} 1 & x^T \\ x & X \end{pmatrix} \succeq 0 \iff X - xx^T \succeq 0$$

Proposition 1.1.1 *Soit $A \in \mathbb{S}^n_+$, alors il existe une unique matrice $B \in \mathbb{S}^n_+$ telle que*

$$A = BB = B^2.$$

Où B est appelée la racine carrée de A, et on note souvent $B = A^{\frac{1}{2}}$. De plus, on a $rg(A) = rg(B)$.

Théorème 1.1.3 *(Factorisation de Cholesky) Pour $A \in \mathbb{S}^n_{++}$, il existe une unique matrice triangulaire inférieure et inversible L telle que $A = LL^T$.*

Définition 1.1.2 *(Matrice à diagonale (strictement) dominante)*

1. *Une matrice $A \in M_n$ est à diagonale dominante si*

$$|a_{ii}| \geq \sum_{i\neq j=1}^{n} |a_{ij}| \text{ pour tout } i = 1, ..., n$$

2. *Une matrice $A \in M_n$ est à diagonale strictement dominante si*

$$|a_{ii}| > \sum_{i\neq j=1}^{n} |a_{ij}| \text{ pour tout } i = 1, ..., n$$

Théorème 1.1.4 *Si $A \in \mathbb{S}^n$ est à diagonale strictement dominante et si tous les éléments diagonaux sont strictement positifs. Alors A est définie-positive.*

1.2 Analyse convexe

La notion de convexité est un outil mathématique important pour l'étude théorique et numérique des problèmes d'optimisation.

1.2.1 Ensembles affines

Définition 1.2.1 *Un sous ensemble C de $\mathbb{R}^n$ est dit affine si :*

$$\forall x, y \in C, \quad \forall \lambda \in \mathbb{R} : \quad (1-\lambda)x + \lambda y \in C.$$

On dit aussi que C est une variété affine (ou linéaire).

Les ensembles affines élémentaires sont : $\emptyset, \{x\}(x \in \mathbb{R}^n)$, et chaque sous-espace vectoriel de $\mathbb{R}^n$.

Définition 1.2.2 *On appelle combinaison affine des éléments $x_1, ..., x_m$ de $\mathbb{R}^n$ tout élément de la forme :*

$$x = \sum_{i=1}^{m} \lambda_i x_i, \quad avec \quad \lambda_i \in \mathbb{R} \ et \ \sum_{i=1}^{m} \lambda_i = 1.$$

Théorème 1.2.1 *Toute partie affine de $\mathbb{R}^n$ contient ses combinaisons affines :*

$$\forall x_i \in C, \ \sum_{i=1}^{m} \lambda_i x_i \in C, \quad avec \quad \lambda_i \in \mathbb{R} \ et \ \sum_{i=1}^{m} \lambda_i = 1.$$

Définition 1.2.3 *Soit S une partie de $\mathbb{R}^n$. Alors il existe une partie affine unique $C \subseteq \mathbb{R}^n$ contenant S appelée enveloppe affine de S et notée $aff(S)$, c'est la plus petite partie affine de $\mathbb{R}^n$ contenant S.*

$$aff(S) = \cap\{C_S : S \subset C_S \text{ et } C_S \text{ affine}\}.$$

Si $S \neq \emptyset \Rightarrow aff(S) \neq \emptyset$ (puisque $S \subseteq aff(S)$).

Théorème 1.2.2 *Soit $S \subseteq \mathbb{R}^n, S \neq \emptyset$, alors :*

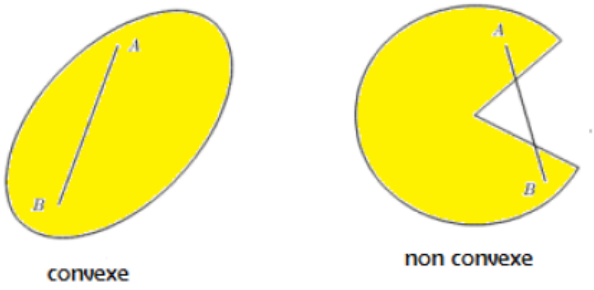
$$aff(S) = \left\{ x \in \mathbb{R}^n \ / \ x = \sum_{i=1}^{m} \lambda_i x_i, \ x_i \in S, \ \sum_{i=1}^{m} \lambda_i = 1 \right\}$$

1.2.2 Ensembles convexes

Définition 1.2.4 *Un sous ensemble C de $\mathbb{R}^n$ est dit convexe si :*

$$\forall x, y \in C, \quad \forall \lambda \in [0,1] : \quad (1-\lambda)x + \lambda y \in C.$$

Autrement dit, le segement de droite joignant deux points quelconques $x, y \in C$ c-à-d $[x,y] = \{(1-\lambda)x + \lambda y \ , \ 0 \leq \lambda \leq 1\}$ est entièrement inclus dans C.



Exemple 1.2.1

1. *L'orthant positif $\mathbb{R}^n_+$, toute boule dans un espace vectoriel normé sont convexes.*
2. *Le disque unitaire $C = \{(x_1, x_2) \in \mathbb{R}^2 : x_1^2 + x_2^2 \leq 1\}$ est convexe.*
3. *Les ensembles $\left\{x \in \mathbb{R}^n \ : b^T x \geq \beta\right\}, \left\{x \in \mathbb{R}^n \ : b^T x \leq \beta\right\}$ sont des demi-espaces fermés convexes, avec $b \in \mathbb{R}^n, \beta \in \mathbb{R}$.*
4. *Les ensembles $\left\{x \in \mathbb{R}^n \ : b^T x > \beta\right\}, \left\{x \in \mathbb{R}^n \ : b^T x < \beta\right\}$ sont des demi-espaces ouverts convexes non vide, avec $b \in \mathbb{R}^n, \beta \in \mathbb{R}$.*
5. *L'ensemble $P = \left\{x \in \mathbb{R}^n \ : Ax = b\right\}$, est un ensemble convexe fermé appelé polyèdre convexe ou tout simplement polyèdre, avec $b \in \mathbb{R}^m, A \in \mathbb{R}^{m \times n}$.*

Les opérations algébriques suivantes conservent la convexité

- L'intersection quelconque.
- Le produit cartésien.
- Les transformations affines.
- Les combinaisons linéaires $\sum_{i=1}^{m} \alpha_i C_i$, ($\alpha_i \in \mathbb{R}$ et $m \in \mathbb{N}$).
- La translation $C + a \ / \ a \in \mathbb{R}^n$.

Définition 1.2.5 *On appelle combinaison convexe de m-vecteurs de* $\mathbb{R}^n$, $x_1, ..., x_m$, *toute combinaison linéaire :*

$$\sum_{i=1}^{m} \lambda_i x_i, \ \ où\ \lambda_i \geq 0\ et\ \sum_{i=1}^{m} \lambda_i = 1$$

Définition 1.2.6 *L'enveloppe convexe de l'ensemble* $S \subset \mathbb{R}^n$ *est le plus petit convexe de* $\mathbb{R}^n$ *contenant* S.

$$conv(S) = \cap(C_i),$$

avec C_i convexe contenant S.

Et on a :

$$S \text{ convexe} \Longleftrightarrow S = conv(S).$$

Définition 1.2.7 *Soit* C *un convexe non vide de* $\mathbb{R}^n$ *et* $x \in C$.

Le point x *est dit extrémal (ou sommet de* C*) s'il n'est pas à l'intérieur d'un segment de droite contenu dans* C. *Autrement dit si :*

$$\forall x_1, x_2 \in C, \forall \lambda \in]0,1[, x = \lambda x_1 + (1-\lambda)x_2 \Rightarrow x = x_1 = x_2.$$

Remarque 1.2.1 *La définition d'un point extrémal* x *d'un convexe* C, *est équivalente à chacune des deux propriétés suivantes :*

1. $x = \frac{1}{2}x_1 + \frac{1}{2}x_2 \Rightarrow x = x_1 = x_2 \ \ (\forall x_1, x_2 \in C)$.
2. $C - \{x\} = \{y \in C \ / \ y \neq x\}$ *est convexe.*

Remarque 1.2.2 *Tout point extrémal d'un convexe* C *est un point de la frontière de* C.

1.2.3 Fonctions convexes

Soit $f : \mathbb{R}^n \to \mathbb{R}$, on associe à f les ensembles suivants :

– **Epigraphe de f :**

$$epi(f) = \{(x, \alpha) \in \mathbb{R}^n \times \mathbb{R} : f(x) \leq \alpha\}.$$

– **Domaine effectif de f :**

$$dom(f) = \{x \in \mathbb{R}^n : f(x) < +\infty\}.$$

– **Ensembles de niveau $\alpha \in \mathbb{R}$:**

$$S_\alpha(f) = \begin{cases} \{x \in \mathbb{R}^n : f(x) \leq \alpha\} \text{ (inférieur large),} \\ \{x \in \mathbb{R}^n : f(x) \geq \alpha\} \text{ (supérieur large).} \end{cases}$$

Définition 1.2.8 *f est dite propre si*

$$dom f \neq \emptyset \text{ et } f(x) > -\infty, \forall x \in \mathbb{R}^n.$$

Dans le cas contraire, on dit que f est impropre.

Définition 1.2.9 *Soit C une partie convexe non vide. La fonction f est dite :*

1. *Convexe sur C si*

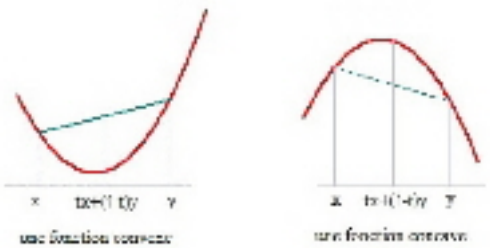
$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y) \quad \forall x, y \in C, \forall \lambda \in [0,1]$$

2. *Strictement convexe sur C si l'inégalité ci-dessus est stricte pour tous $x, y \in C, x \neq y$ et $\lambda \in]0,1[$.*
3. *Fortement convexe (ou $\alpha-$convexe) sur C s'il existe $\alpha > 0$ tel que pour tous $x, y \in C$ et $\lambda \in [0,1]$ on ait*

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y) - \frac{\alpha}{2}\lambda(1-\lambda)||x-y||^2$$

4. *Concave sur C si $(-f)$ est convexe sur C. Et on a*

$$\textit{convexité forte} \Rightarrow \textit{convexité stricte} \Rightarrow \textit{convexité}.$$



Exemple 1.2.2 *1. Toute fonction affine $f : C \to \mathbb{R}$, est à la fois convexe et concave.*

2. *Toute fonction quadratique* $f : x \to \langle x, Ax\rangle + \langle b, x\rangle + c$, *avec* A *symétrique semi-définie-positive (resp.* A *symétrique définie-positive) est convexe (resp. strictement convexe).*
3. *Toute norme est convexe non nécessairement strictement convexe, en revanche, la norme issue d'un produit scalaire est strictement convexe.*
4. *Toute fonction* $f : C \to \mathbb{R}$ *sous-linéaire i.e., vérifiant pour tous* $x, y \in C$,

$$f(x+y) \leq f(x) + f(y) \quad et \quad f(\alpha x) = \alpha f(x) \quad \forall \alpha > 0$$

est convexe.

5. *La fonction* ***log-barrière*** *définie sur* $\mathbb{R}^n_{++}$ *par*

$$f : x \to \begin{cases} -\sum_{i=1}^{n} \log x_i & si\ x_i > 0 \\ +\infty & sinon \end{cases}$$

est convexe.

6. *La fonction* $A \to (\det A)^{-1}$ *est strictement convexe sur l'ensemble des matrices définie-positives.*
7. *La fonction* ***log-déterminant*** *définie sur l'ensemble des matrices symétriques définie-positives par*

$$f(A) = \begin{cases} -\ln \det A & si\ A\ est\ définie\text{-}positive \\ +\infty & sinon \end{cases}$$

est convexe.

Propriétés :

1. La somme des fonctions convexes est convexe.
2. Le produit d'un scalaire positif par une fonction convexe est convexe.
3. La composée d'une fonction convexe et d'une fonction convexe croissante est convexe.

Théorème 1.2.3 *Si* f *est deux fois continûment différentiable sur un ouvert* Ω *et* C *une partie convexe de* Ω, *les conditions suivantes sont équivalentes :*

1. f *est convexe sur* C,

2. $\forall x, y \in C$, $f(y) \geq f(x) + < \nabla f(x), y - x >$, *où* $\nabla f(x)$ *est le vecteur gradient de* f *au point* x,
3. $\forall x \in C$, *la matrice Hessienne* $H(x) = \nabla^2 f(x)$ *est semi-définie-positive sur* C.

Lemme 1.2.1 *Une combinaison linéaire à coéfficients positifs des fonctions convexes est une fonction convexe.*

$$\begin{cases} f_1, ..., f_m \text{ convexe} \\ \lambda_1 \geq 0, ..., \lambda_m \geq 0 \end{cases} \Rightarrow f = \sum_{i=1}^{m} \lambda_i f_i \text{ convexe.}$$

Définition 1.2.10 *Soit* C *fermé non vide de* $\mathbb{R}^n$ *et* $f : C \rightarrow]-\infty, +\infty]$, f *est dite inf-compacte si ses ensembles de niveau inférieurs* $S_\lambda(f)$*sont compacts* $\forall \lambda \in \mathbb{R}$.

Lemme 1.2.2 f *est inf-compacte si et seulement si* $\lim\limits_{||x|| \rightarrow +\infty} f(x) = +\infty$.

Définition 1.2.11 *Une fonction* f *est dite coercive sur un ensemble convexe* S *si* $\lim\limits_{\substack{||x|| \rightarrow +\infty \\ x \in S}} f(x) = +\infty$.

Définition 1.2.12 *On appelle fonction barrière toute fonction f qui vérifie :*

1. $f(x)$ *finie, si* x *appartient à l'intérieur relatif du domaine réalisable (admissible).*
2. $f(x)$ *tend vers l'infini quand* m *s'approche de la frontière du domaine réalisable.*

Remarque 1.2.3 *Il est d'usage d'étendre une fonction* f *définie sur un ensemble* C *de* $\mathbb{R}^n$ *en une fonction* $g : \mathbb{R}^n \rightarrow]-\infty, +\infty]$ *comme suit :*

$$g(x) = \begin{cases} f(x) & \text{si } x \in C \\ +\infty & \text{sinon.} \end{cases}$$

Si C *est convexe alors,* f *est convexe (strictement convexe) sur* C *si et seulement si son extension* g *est convexe (strictement convexe) sur* $\mathbb{R}^n$.

Lemme 1.2.3 *Soit* C *une partie non vide de* $\mathbb{R}^n$. *Un vecteur* $d \neq 0$ *de* $\mathbb{R}^n$ *est dit direction admissible de* C *en* $a \in C$ *si*

$$\exists \overline{\alpha} > 0 \ / \ a + \alpha d \in C, \forall \alpha \in [0, \overline{\alpha}].$$

Si la propriété est vraie $\forall a \in C$, *on dit que* d *est une direction admissible pour* C.

1.2.4 Cônes

Définition 1.2.13 *On dit que K est un cône si :*

$$\lambda K \subset K \qquad \forall \lambda > 0$$

Un cône K est dit :

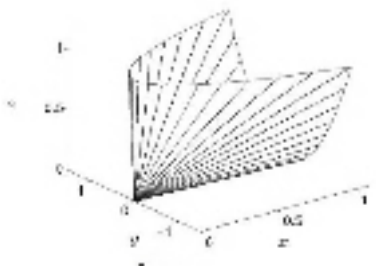
1. *Pointé si $0 \in K$,*
2. *Saillant si $K \cap (-K) \subset \{0\}$,*
3. *Solide si $IntK \neq \emptyset$,*
4. *Convexe si $K + K \subset K$.*

Proposition 1.2.1 *L'ensemble $\mathbb{S}^n_+$ est un cône convexe fermé saillant et solide. Et l'ensemble $\mathbb{S}^n_{++}$ est un cône convexe.*

Exemple 1.2.3 *Dans $\mathbb{S}^2$, le cône des matrices semi-définies positives $\mathbb{S}^2_+$ est l'ensemble des matrices symétriques*

$$\begin{aligned} A &= \begin{pmatrix} x & y \\ y & z \end{pmatrix} \quad \textit{telles que :} \\ x &\geq 0, \quad z \geq 0, \quad xz - y^2 \geq 0. \end{aligned}$$

La figure suivante illustre le cône $\mathbb{S}^2_+$:



Comme exemples de cône, on peut citer les orthants positifs $\mathbb{R}^n_+, \mathbb{R}^n_{++}$, l'ensemble des polynômes positifs, l'ensemble des fonctions réelles qui sont positives sur une partie donnée de leur ensemble de définition, les ensembles de matrices symétriques définies positives, semi-définie-positives.

Proposition 1.2.2
- *L'intersection quelconque de cônes convexes est un cône convexe.*
- *Le produit cartésien de cônes convexes est un cône convexe.*

1.3 Programmation mathématique

On donne, dans cette partie, les outils de base de l'optimisation. On rappelle quelques définitions élémentaires et les théorèmes fondamentaux d'existence et d'unicité de minima. Des conditions d'optimalité sont aussi présentées.

Un programme mathématique est en général défini comme suit :

$$(PM)\begin{cases} \min\limits_{x} f(x) \\ h_j(x) \leq 0, \quad j = 1, ..., p \\ g_i(x) = 0, \quad i = 1, ..., n \\ x \in \mathbb{R}^n. \end{cases}$$

Où : h_j, g_i sont des fonctions de $\mathbb{R}^n \to \mathbb{R}$ et l'ensemble

$$\mathcal{F} = \{x \in \mathbb{R}^n : h_j(x) \leq 0, \quad j = 1, ..., p \text{ et } g_i(x) = 0, \ i = 1, ..., n\}$$

est appelé ensemble des contraintes (ou des solutions admissibles), dit aussi domaine de faisabilité. La fonction $f : \mathbb{R}^n \to \mathbb{R}$ est appelée fonction objectif ou économique.

Définitions

- On appelle solution réalisable de (PM) tout point x vérifiant les contraintes i.e., $x \in \mathcal{F}$.
- Une solution réalisable qui optimise l'objectif sur $\mathcal{F}$ est dite solution optimale globale de (PM).
- Un point $x^* \in \mathcal{F}$ est une solution optimale locale de (PM) si :

$$\exists \vartheta \text{ (voisinage) de } x^* \text{ tel que : } f(x^*) \leq f(x); \ \forall x \in \vartheta$$

- Une contrainte d'inégalité $h_j(x) \leq 0, \forall j$, est dite saturée (active) en $x^* \in \mathcal{F}$ si $h_j(x^*) = 0$. Une contrainte d'égalité est par définition saturée en tout point x de $\mathcal{F}$.

Remarque 1.3.1 *Le problème d'optimisation précédent consiste :*

- *Soit à chercher un point optimal.*
- *Soit, si un tel point n'existe pas on cherche une borne inférieure à f.*
- *Soit à établir que f est non bornée inférieurement sur $\mathcal{F}$, auquel cas on adopte la convention $\inf\limits_{\mathcal{F}} f(x) = -\infty$. Lorsque $\mathcal{F}$ est vide, on pose par convention $\inf\limits_{\mathcal{F}} f(x) = +\infty$.*

1.3.1 Classification d'un programme mathématique

Il existe beaucoup d'algorithmes d'optimisation dans différentes applications scientifiques. Cependant beaucoup de méthodes ne sont valables que pour certains types de problèmes. Ainsi, il est important de bien connaitre les caractéristiques du problème posé afin d'identifier la technique approprieé pour sa résolution.

On classifie un (PM) à partir de deux propriétés fondamentales à savoir la convexité et la différentiabilité de la fonction objectif et les contraintes.

- (PM) est un problème différentiable si les fonctions f, g_i, h_j sont toutes différentiables.
- (PM) est un problème convexe si f et h_j sont convexes et g_i affines.

Théorème 1.3.1 *Pour un programme convexe, tout optimum local est un optimum global.*

La classe modèle des (PM) est celle des programmes convexes différentiables, les programmes non convexes ou non différentiables sont difficiles à traiter. Enfin, le cas le plus simple est celui de la programmation linéaire où f, g_i et h_j sont affines.

On peut le classé aussi en fonction des caractéristiques mathématiques de la fonction objectif, des contraintes et des variables d'optimisation (Tableau1).

Caractéristiques	**Propriétés**	**Classification**
Nombre de variables	Une seule variable	Monovariable
	Plus d'une variable	Multivariable
Type de variables	Réelles	Continue
	Entières	Discrète
	Réelles et entières	Mixte
	Entières avec permutations	Combinatoire
Type de fonction objectif	Linéaire en fonction des variables	Linéaire
	Quadratique en fonction des variables	Quadratique
	Non linéaire en fonction des variables	Non linéaire
Formulation du problème	Soumis à des limitations	Avec contraintes
	Pas de limitations	Sans contraintes

1.3.2 Qualification des contraintes

La qualification des contraintes est satisfaite pour tout $x^* \in \mathcal{F}$ dans les cas suivants :

- $\mathcal{F}$ est un polyédre convexe (toutes les contraintes sont affines).
- Les gradients des contraintes saturées en x^* sont linéairement indépendants.
- $\mathcal{F}$ est convexe et $int^1(\mathcal{F}) \neq \emptyset$.

On dit que le point x^* est régulier si les contraintes sont qualifiées en x^*.

Remarque 1.3.2 *La résolution complète de (PM) est traitée dans l'ordre des points suivants :*

- *L'existence (et éventuellement l'unicité) d'une solution optimale ;*
- *Caractérisation de la solution ;*
- *Elaboration d'algorithmes pour calculer numériquement cette solution.*

1.4 Résolution d'un programme mathématique (PM)

1.4.1 Existence et unicité d'une solution optimale d'un (PM)

Théorème 1.4.1 ***(Weirstrass)*** *Si f est une fonction continue sur $\mathcal{F} \subset \mathbb{R}^n$ et $\mathcal{F}$ est compact (fermé et borné) non vide, alors (PM) admet au moins une solution optimale $x^* \in \mathcal{F}$.*

Corollaire 1.4.1 *Si $\mathcal{F} \subset \mathbb{R}^n$ est non vide et fermé et si f est continue et coercive sur $\mathcal{F}$, alors (PM) admet au moins une solution optimale. Si f est strictement convexe et l'ensemble $\mathcal{F}$ est convexe, alors (PM) admet une solution optimale unique.*

Corollaire 1.4.2 *Si f est une fonction inf-compacte, alors (PM) admet une solution optimale globale.*

Remarque 1.4.1 *La stricte convexité n'assure pas l'existence de la solution mais assure l'unicité.*

[1] Répresente l'intérieur de $\mathcal{F}$.

1.4.2 Conditions d'optimalités

Reprenons le problème (PM) :

$$(PM)\begin{cases} \min\limits_{x} f(x) \\ h_j(x) \leq 0, \quad j = 1, ..., p \\ g_i(x) = 0, \quad i = 1, ..., n \\ x \in \mathbb{R}^n, \end{cases}$$

Le lagrangien associé à (PM) est défini par :

$$\mathcal{L}(x, \lambda, \mu) = f(x) + \sum_{j=1}^{p} \lambda_j h_j(x) + \sum_{i=1}^{n} \mu_i g_i(x),$$

où : λ_j et μ_i sont des réels dits multiplicateurs de Lagrange tel que : $\lambda_j \in \mathbb{R}^+$, $j = 1, ..., p$ et $\mu_i \in \mathbb{R}$, $i = 1, ..., n$.

La théorie de Karush-Kuhn-Tuker **(K.K.T)** permet d'écrire les conditions nécéssaires d'optimalité pour tout problème d'optimisation avec contraintes possédant une fonction objectif différentiable.

Théorème 1.4.2 *Si x^* est une solution optimale locale de (PM) satisfaisant l'une des conditions de qualifications précédentes, alors il existe des multiplicateurs $\lambda \in \mathbb{R}^p_+$ et $\mu \in \mathbb{R}^n$ tel que :*

$$\begin{cases} \nabla f(x^*) + \sum\limits_{j=1}^{p} \lambda_j \nabla h_j(x^*) + \sum\limits_{i=1}^{n} \mu_i \nabla g_i(x^*) = 0, & \textit{(condition d'optimalité),} \\ \lambda_j h_j(x^*) = 0, & j = 1, ..., p, \quad \textit{(condition de complémentarité),} \\ g_i(x^*) = 0, & i = 1, ..., n. \end{cases}$$

Remarque 1.4.2 *1- Si (PM) est convexe, les conditions de **(K.K.T)** sont à la fois nécessaires et suffisantes pour que x^* soit un minimum global.*
*2- Si les contraintes ne sont pas qualifiées en x^*, les conditions de **(K.K.T)** ne s'appliquent pas (x^* peut être optimal sans vérifier ces conditions).*

1.4.3 Cas d'un programme mathématique particulier

Considérons le programme mathématique (ou bien le problème d'optimisation) suivant :

$$(PO)\begin{cases} \min\limits_{x} f(x) \\ Cx = d, \\ Gx \geq h, \\ x \in \Omega. \end{cases}$$

où f est une fonction différentiable, Ω est un sous-ensemble ouvert de $\mathbb{R}^n$, C et G sont deux matrices de M_n et d et h deux vecteurs de $\mathbb{R}^n$. Alors les conditions nécessaires de Karush-Kuhn-Tucker du premier ordre sont données dans le cadre du théorème suivant :

Théorème 1.4.3 *[69] Supposons que x^* est une solution locale de (PO), et f est une fonction différentiable au voisinage de x^*. Alors, il existe deux vecteurs y et z vérifiant les conditions suivantes :*

$$\begin{aligned} &\nabla f(x^*) - C^T y - G^T z = 0 \\ &Cx^* = d \\ &Gx^* \geq h \\ &z \geq 0 \\ &z^T(Gx^* - h) = 0 \end{aligned}$$

où les vecteurs y et z sont les multiplicateurs de Lagrange.

Définition 1.4.1 *La solution x^* du problème (PO) et les multiplicateurs de Lagrange correspondants (y, z) qui satisfaient les conditions du théorème précédent sont dits strictement complémentaires si $z^T(Gx^* - h) > 0$. d'après cette définition, on a :*

$$\begin{aligned} z_i &= 0, \ (Gx^* - h)_i > 0 \\ z_i &> 0, \ (Gx^* - h)_i = 0. \end{aligned}$$

Solution globale et convexité

Soit Ω un ensemble ouvert convexe, alors l'ensemble des solutions réalisables de (PO) est un ensemble convexe. Si la fonction objectif f est convexe dans l'ensemble des contraintes, alors (PO) est un problème d'optimisation convexe.

1. Si le problème (PO) est convexe, alors x^* est une solution globale de (PO) s'il existe des vecteurs y et z multiplicateurs de Lagrange vérifiant les conditions du Théorème précédent.

2. Si f est une fonction strictement convexe dans l'ensemble des solutions realisables, alors toute solution locale est une solution globale.

1.4.4 Dualité lagrangienne

Le dual de (PM) est le programme mathématique (DM) suivant :

$$(DM)\begin{cases} \sup\limits_{\lambda,\mu}\inf\limits_{x\in\mathcal{F}}\mathcal{L}(x,\lambda,\mu), \\ \nabla_x\mathcal{L}(x,\lambda,\mu)=0, \\ \lambda\in\mathbb{R}^p_+,\ \mu\in\mathbb{R}^n. \end{cases}$$

La dualité joue un rôle fondamental dans l'étude et la résolution de (PM). Entre autre, elle fournit des informations supplémentaires très utiles.

1.4.5 Méthode de Newton pour résoudre un système non linéaire

La résolution du (PM) revient à résoudre le système d'équation non linéaire suivant :

$$\begin{cases} \nabla f(x)+\sum\limits_{j=1}^{p}\lambda_j\nabla h_j(x)+\sum\limits_{i=1}^{n}\mu_i\nabla g_i(x), \\ \lambda_j h_j(x), & j=1,...,p, \\ g_i(x), & i=1,...,n. \end{cases}$$

Posons

$$F(x,\lambda,\mu)=\begin{pmatrix} \nabla f(x)+\sum\limits_{j=1}^{p}\lambda_j\nabla h_j(x)+\sum\limits_{i=1}^{n}\mu_i\nabla g_i(x)=0, \\ \lambda_j h_j(x)=0, & j=1,...,p, \\ g_i(x)=0, & i=1,...,n. \end{pmatrix}$$

où

$$\begin{array}{rcl} F & : & \mathbb{R}^{2n+p}\to\mathbb{R}^{2n+p} \\ (x,\lambda,\mu) & \mapsto & \left(\nabla f(x)+\sum\limits_{j=1}^{p}\lambda_j\nabla h_j(x)+\sum\limits_{i=1}^{n}\mu_i\nabla g_i(x),\lambda_j h_j(x),g_i(x)\right) \end{array}$$

Les méthodes les plus populaires appliquées pour la résolution d'un système non linéaire est la méthode de Newton. Dans ce qui suit décrivons son principe.

Soit $g : \mathbb{R}^n \to \mathbb{R}^n$ une fonction continue, différentiable et soit $J(x)$ la matrice jacobiènne de la fonction g. Alors nous considérons le système non linéaire suivant :

$$g(x) = 0$$

A partir d'un vecteur x^0 de $\mathbb{R}^n$ et l'utilisation de la formule suivante

$$x^{k+1} = x^k - J(x^k)^{-1} g(x^k),$$

on construit une suite de point définie par

$$x^{k+1} = x^k + d^k,$$

où d^k est le vecteur de direction, solution du système

$$J(x^k)d^k = -f(x^k).$$

1.4.6 Méthodes de directions admissibles

Cette classe de méthodes résout un problème de minimisation non linéaire en se déplaçant d'un point de l'ensemble des contraintes de (PM) D vers un autre de ses points au coût inférieur. Elles fonctionnent selon le principe suivant : étant donné un élément x^k de D, une direction d^k est générée telle que pour un $\alpha^k > 0$ et suffisamment petit, les propriétés suivantes sont assurées :

1. $x^k + \alpha^k d^k$ appartient toujours à D
2. $f(x^k + \alpha^k d^k)$ est inférieure à $f(x^k)$

Une fois d^k déterminée, α^k s'obtient par minimisation monodimenssionnelle pour que le déplacement dans la direction d^k soit optimal, mais cette fois-ci il est nécessaire d'imposer une borne supérieure sur la valeur de α^k afin de ne pas sortir de D. Cela définit le nouveau point x^{k+1} et le processus est recommencé.

1.4.7 Méthode de résolution d'un programme mathématique

On peut classifier les méthodes de résolution d'un programme mathématique en trois catégories

a) Méthodes de type gradient

I. Gradient conjugué Cette méthode a été proposée par Hestenes (1952) pour résoudre un système linéaire à matrice définie-positive, puis généralisée par Fletcher et Reeves (1964) pour résoudre des problèmes d'optimisation non linéaires, elle est connue par son efficacité pour minimiser une fonction quadratique sans contraintes. Dans le cas contraint, un changement de variable simple permet de se ramener au cas sans contraintes, en effet : x^0 un point satisfaisant les contraintes ($Ax^0 = 0$) et posons $x = x^0 + P_A y$ tel que$P_A = I - A^T(AA)^{-1}A$ est l'opérateur de la projection sur le noyau de la matrice A. Le principe de cette méthode est de construire progressivement des directions $d_0, d_1, ..., d_k$ mutuellement conjuguées par rapport à la matrice Hessienne ($\nabla^2 f(x)$) de la fonction objectif du problème d'optimisation : $d_i \nabla^2 f(x) d_j = 0$, $\forall i \neq j$, $i; j = \{0, 1, ..., k\}$.

II. Gradient projeté (Rosen 1960) le principe de cette méthode est de projeter à chaque itération le gradient sur la frontière du domaine réalisable. Il faut signaler que cette méthode est conçue pour un programme plus général de la forme :

$$\begin{cases} minf(x) \\ Ax = b, \\ x \geq 0 \end{cases}$$

où f est différentiable non nécessairement convexe.

b) Méthodes simpliciales

Parmi les méthodes simpliciales on cite, celle de gradient réduit dûe à Wolfe. C'est une extension directe de la méthode du simplexe, appliquée à la programmation quadratique. De ce fait, elle présente les mêmes inconvénients à savoir complexité exponentielle.

c) Méthodes de points intérieurs

Conjointement aux méthodes décrites précédemment, il existe actuellement des méthodes de points intérieurs pour la résolution d'un problème d'optimisation convexe. Ce sont des extensions des méthodes développées pour la programmation linéaire (affines, projectives et de trajectoire centrale). Les problèmes

d'initialisation, le coût de l'itération et le choix de la direction de descente deviennent plus pesants.

On distingue trois classes fondamentales de méthodes de points intérieurs à savoir :

I. Méthodes Affines Il s'agit pratiquement de l'algorithme de Karmarkar sans fonction potentiel et sans transformation projective, on utilise une transformation affine et on remplace la contrainte de non négativité par un ellipsoide qui contient le nouvel itéré. L'algorithme est d'une structure simple, malheureusement, il n'est pas facile de démontrer la polynômialité.

II. Méthodes de réduction du potentiel La fonction potentiel joue un grand rôle dans le développement des méthodes de points intérieurs. L'algorithme de Karmarkar appliqué au programme linéaire sous forme standard utilise une fonction potentiel de la forme : $(n+1)\log(c^T x - z) - \sum_{i=1}^{n} \log(x_i)$ où z est une borne inférieure de la valeur optimale de l'objectif. Karmarkar prouve la convergence et la polynômialité de son algorithme et a montré que cette fonction potentiel est réduite à chaque itération par au moins une constante. Depuis 1987, les chercheurs introduisent des fonctions du potentiel de type primales-duales, parmi lesquelles, celle de Tanabe, Todd et Ye [69] défnie par : $\Phi_p(x,s) = p\log(x^T s) - \sum_{i=1}^{n} \log(x_i s_i)$ pour $p > n$. Cette fonction a joué un rôle très important dans le développement des algorithmes de réduction du potentiel après 1988. Les algorithmes correspondants à ces méthodes possèdent une complexité polynômiale, nécessitant $O(\sqrt{n}|\log\varepsilon|)$ itérations pour réduire le saut de dualité.

III. Méthodes de la trajectoire centrale Elles ont été introduites à la même époque que les méthodes de réduction du potentiel et pleinement développées au début des années 90. Elles possèdent de bonnes propriétés théoriques : une complexité polynômiale et une convergence superlinéaire.

Les algorithmes de la trajectoire centrale restreintent les itérés à un voisinage du chemin central, ce dernier est un arc de points strictement réalisables.

Plusieurs chercheurs essaient de généraliser le principe de ces méthodes pour la programmation non linéaire. En 1989, Megiddo [43] a proposé un algorithme primal-dual de trajectoire centrale pour la programmation linéaire avec une gé-

néralisation pour le problème de complémentarité linéaire (LCP). Kojima et al. [34] ont développé un algorithme primal-dual pour la programmation linéaire, une extension pour le (LCP) est proposée par les mêmes chercheurs en 1989 avec la complexité $O(\sqrt{n}\log\frac{1}{\varepsilon})$ itérations.

Chapitre 2

Programmation semi-définie

Dans ce chapitre, nous aborderons l'étude des problèmes de minimisation de fonctions linéaires à contraintes linéaires, où la variable est une matrice carrée symétrique $X \in \mathbb{S}^n$ semi-définie-positive.

2.1 Formulation du problème

Un programme semi-défini est un programme mathématique avec :
- Un ensemble des variables x_{ij}.
- Une fonction objectif linéaire.
- Un ensemble des contraintes linéaires.
- Une contrainte de semi-défini positivité.

Définition 2.1.1 *Un programme semi-défini sous forme standard s'écrit comme suit :*

$$(SDP)\begin{cases} \min C \bullet X = \langle C, X\rangle = Tr(CX) \\ A_i \bullet X = b_i, \quad i = 1, ..., m \\ X \succeq 0, \end{cases}$$

où : $b = (b_1, b_2, ..., b_m) \in \mathbb{R}^m$, C, X *et* A_i $(i = 1, ..., m)$ *sont des matrices dans* $\mathbb{S}^n$.

Proposition 2.1.1 *Un programme semi-défini est un programme convexe.*

Pour la démonstration, il est facile de voir que la fonction objectif est une fonction linéaire, donc démontrer qu'un programme semi-défini est convexe, revient à démontrer que l'ensemble des contraintes du problème (SDP) *est un ensemble convexe. Pour plus de détails voir [28].*

Définition 2.1.2 *1- Une matrice $X \in \mathbb{S}^n$ est dite réalisable de (SDP) si*

$$X \succeq 0, \quad A_i \bullet X = b_i, \quad i = 1, ..., m.$$

on désigne par $\mathcal{F}_p = \{X \in \mathbb{S}^n, \ X \text{ réalisable}\}$ l'ensemble des solutions réalisables primales de (SDP).
2- Une matrice $X \in \mathbb{S}^n$ est dite strictement réalisable de (SDP) si

$$X \succ 0, \quad A_i \bullet X = b_i, \quad i = 1, ..., m$$

on désigne par $\mathcal{F}_p^+ = \{X \in \mathbb{S}^n, \ X \text{ strictement réalisable}\}$ l'ensemble des solutions strictement réalisables primales de (SDP).

Définition 2.1.3 *1- La valeur optimale primale de (SDP) est définie par :*

$$p^* = \inf(C \bullet X) \ tq \ \ X \in \mathbb{S}_+^n; \ A_i \bullet X = b_i \ \ i = 1, ..., m.$$

2- X^* est une solution optimale primale de (SDP) si

$$X^* \in \mathcal{F}_p \ et \ C \bullet X^* = p^*.$$

2.2 Cas particuliers

Plusieurs programmes mathématiques peuvent se formuler en un problème (SDP). Nous présentons ici quelques uns.

2.2.1 Programmation linéaire

En pratique, la transformation d'un programme linéaire en un programme (SDP) n'est pas intéressante du point de vue numérique, malgré qu'un programme linéaire peut s'écrire théoriquement sous forme de (SDP). En effet :

Un programme linéaire (PL) est un problème sous la forme :

$$(PL) \begin{cases} \min \ c^T x \\ Ax = b \\ x \geq 0, \end{cases}$$

où $x, c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ et $A \in \mathbb{R}^{m \times n}$.

Pour transformer le programme linéaire (PL) en un programme (SDP), on définit

$$X = diag(x), \quad C = diag(c) \text{ et } A_i = diag(a^i) \ \ \forall i \in \{1, ..., m\}$$

où a^i désigne la $i^{éme}$ ligne de la matrice A. Nous reécrivons alors la fonction objectif linéaire sous la forme

$$c^T x = C \bullet X$$

et pour les contraintes, on a

$$Ax = b \iff A_i \bullet X = b_i \quad i = 1, ..., m$$

de même

$$x_i \geq 0, \ i = 1, ..., n \iff X \succeq 0.$$

Ainsi, (PL) en variable x peut être réécrit sous la forme standard d'un (SDP) en variable matricielle X comme suit :

$$(SDP) \begin{cases} \min C \bullet X \\ A_i \bullet X = b_i \quad i = 1, ..., m \\ X \succeq 0 \end{cases}$$

2.2.2 Programmation quadratique

Les problèmes d'optimisation quadratiques peuvent aussi se formuler en programme (SDP).

Un programme quadratique convexe (PQC) sous contraintes quadratiques convexes est défini par :

$$(PQC) \begin{cases} \min \ x^T Q_0 x + b_0^T x + c_0 \\ x^T Q_i x + b_i^T x + c_i \leq 0, \ i = 1, ..., m \\ x \in \mathbb{R}^n. \end{cases}$$

où $Q_i \in \mathbb{S}_+^n$, $b_i \in \mathbb{R}^n$ et $c_i \in \mathbb{R}$, $i = 0, ..., m$.

Pour transformer le programme (PQC) en un programme (SDP), on procède de la maniére suivante :

Pour la fonction objectif on rajoutant une variable auxiliaire réelle y de sorte que

$$x^T Q_0 x + b_0^T x + c_0 \leq y, \quad y \in \mathbb{R}$$

Donc la résolution de (PQC) revient à la résolution du problème

$$(PQC)\begin{cases} \min\ y \\ x^TQ_0x + b_0^Tx + c_0 \leq y \\ x^TQ_ix + b_i^Tx + c_i \leq 0,\ i = 1, ..., m \\ x \in \mathbb{R}^n. \end{cases}$$

On transforme ensuite les contraintes en s'appuyant sur le théorème de Schur. En effet,

Comme $Q_i \in \mathbb{S}_+^n$, elle possède une unique racine carrée $M_i \in \mathbb{S}_+^n$ d'où $Q_i = M_i^2$, et on a

$$x^TQ_0x + b_0^Tx + c_0 \leq y \iff \begin{pmatrix} I & M_0x \\ x^TM_0^T & -c_0 - b_0^Tx + y \end{pmatrix} \succeq 0,$$

$$x^TQ_ix + b_i^Tx + c_i \leq 0 \iff \begin{pmatrix} I & M_ix \\ x^TM_i^T & -c_i - b_i^Tx \end{pmatrix} \succeq 0$$

Le problème (PQC) se transforme alors comme suit

$$\begin{cases} \min\ y \\ \begin{pmatrix} I & M_0x \\ x^TM_0^T & -c_0 - b_0^Tx + y \end{pmatrix} \succeq 0 \\ \begin{pmatrix} I & M_ix \\ x^TM_i^T & -c_i - b_i^Tx \end{pmatrix} \succeq 0 \quad i = 1, ..., m \\ x \in \mathbb{R}^n, \quad y \in \mathbb{R}. \end{cases}$$

où il y a $(n + 1)$ variables inconnues (x, y) et $(m + 1)$ contraintes d'inégalité matricielles linéaires.

2.3 Domaines d'application

La programmation semi-définie est un domaine vaste qui traduit de nombreuses applications en ingénierie, statistiques, graphes,... etc. Quelques unes de ces applications sont présentées dans la partie suivante.

2.3.1 Optimisation des valeurs propres

Il s'agit des plus anciens problèmes traités à l'aide de la programmation semi-définie.

a) Recherche des valeurs propres extrêmes

Considérons A une matrice symétrique réelle, le problème de recherche de la valeur propre minimale d'une matrice symétrique peut se formuler comme suit :

$$\lambda_{\min}(A) = \min_{||x||=1} x^T A x = \min_{x\in\mathbb{R}^m, x\neq 0} \frac{x^T A x}{||x||^2} \qquad (VE)$$

Posant $X = xx^T$, alors

$$Tr(X) = \sum_{i=1}^{n} x_i^2 = ||x||^2 = 1.$$

Le problème (VE) devient alors

$$SDP \begin{cases} \min A \bullet X = \lambda_{\min}(A) \\ Tr(X) = 1 \\ X \succeq 0 \end{cases}$$

De même, le problème de la plus grande valeur propre peut s'écrire comme :

$$\lambda_{\max}(A) = \max_{||x||=1} x^T A x = \max_{x\in\mathbb{R}^m, x\neq 0} \frac{x^T A x}{||x||^2}$$

et sa formulation en (SDP) est :

$$\begin{cases} \max A \bullet X = \lambda_{\max}(A) \\ Tr(X) = 1 \\ X \succeq 0 \end{cases}$$

b) Problèmes de Min-Max des valeurs propres

Ce problème a été étudié vers les années 1950 en algèbre linéaire comme suit :
Chercher une valeur optimale

$$\lambda^* = \min_{x\in\mathbb{R}^n} \lambda_{\max}(C + A(x))$$

Où : $C \in M_n$ et

$$\begin{aligned} A : \quad & \mathbb{R}^n \to M_n \\ & x \longmapsto A(x) = \textstyle\sum_{i=1}^{n} x_i A_i \end{aligned}$$

On a la propriété $A \preceq \lambda I_n \iff \lambda_{\max}(A) \leq \lambda$ tel que $A \in \mathbb{S}^n$ et $\lambda \in \mathbb{R}$.

Le problème min-max s'écrit sous forme d'un problème (SDP) comme suit :

$$\begin{cases} \min \lambda \\ \lambda_{\max}(C + A(x)) \leq \lambda \\ x \in \mathbb{R}^n \end{cases} \iff \begin{cases} \min \lambda \\ \lambda I - C - A(x)) \succeq 0 \\ x \in \mathbb{R}^n \end{cases}$$

c) Norme spectrale d'une matrice

Soit $A \in M_n$. On considère sa norme spectrale

$$\max_{\lambda \in sp(A^T A)} \sqrt{\lambda} = ||A||_2$$

Cette norme peut être calculer à l'aide du problème (SDP) suivant :

$$\begin{cases} ||A||_2 = \min\limits_{t \geq 0} t \\ \begin{bmatrix} tI & A \\ A^T & tI \end{bmatrix} \succeq 0 \end{cases}$$

On utilise ici le théorème du complément de Schur.

d) Minimisation de la somme des r plus grandes valeurs propres

Pour minimiser la somme des r plus grandes valeurs propres de la matrice symétrique $A(x)$ dépend affinement de x i.e., $A(x) = A_0 + x_1 A_1 + ... + x_k A_k$, avec $A_i \in \mathbb{S}^n, i = 0, ..., k$, on résout le problème de programmation semi-définie suivant

$$\begin{cases} \min rt + tr(X) \\ tI_n + X - A(x) \succeq 0 \\ X \succeq 0. \end{cases}$$

Où $X \in \mathbb{S}^n$ et $t \in \mathbb{R}$.

Pour la démonstration et plus de détail voir [49]

2.3.2 Optimisation combinatoire

Les relaxations (SDP) sont utilisées en lieu de la relaxation linéaire (ou continue) pour obtenir des bonnes bornes pour les problèmes d'optimisation en variables entières.

a) Théorie des graphes

On s'intéresse à la détermination du nombre de clique et du nombre chromatique de G, en introduisant la fonction υ de Lovàsz.

la fonction υ de Lovàsz.

L'exemple le plus célèbre d'application de programmation (SDP) est probablement la fonction υ de Lovàsz. La fonction υ de Lovàsz est une application υ d'un graphe complet $G = (V, E)$ dans $\mathbb{R}_+$ de sorte que :

$$\omega(G) \leq \upsilon(G) \leq \chi(G).$$

Cette relation est justement connue comme le "Théorème Sandwich". Ce théorème implique que $\upsilon(G)$ peut être vu comme une approximation des deux nombres $\omega(G)$ et $\chi(G)$ à la fois.

On montre que la fonction υ de Lovàsz est en fait donnée comme la valeur optimale du problème (SDP) suivant :

$$\begin{cases} \upsilon(G) = \max Tr(ee^T X) = e^T X e \\ Tr(X) = 1 \\ x_{ij} = 0 \text{ si } (i,j) \notin E \text{ et } i \neq j \\ X \succeq 0 \end{cases}$$

où $e = (1, ..., 1)^T \in \mathbb{R}^n$. Par conséquent, on a exprimé notre problème de détermination du nombre de clique $\omega(G)$ et du nombre chromatique $\chi(G)$ en un problème (SDP) où la détermination de la valeur $\upsilon(G)$ peut se faire de maniére efficace, alors que dans le cas général, celle de $\omega(G)$ ou de $\chi(G)$ est délicate. On obtient alors une approximation de ces deux nombres. Le cas le plus favorable est celui d'un graphe parfait, pour lequel on a l'égalité $\omega(G) = \chi(G)$. Cependant, la difficulté réside dans la reconnaissance de tels graphes.

b) Problème de coupure maximale (Max_cut)

Nous disposons d'un graphe $G(X, U)$ orienté et nous noterons n le nombre de sommets de ce graphe. De plus, nous le supposerons valué, c-à-d qu'à chaque arc G (du noeud i au noeud j) est associé un poids P_{ij}.

Le problème que nous souhaitons résoudre est de trouver une coupe de capacité maximale, c-à-d trouver comment séparer les sommets en deux ensembles S et $\overline{S}$, de manière à ce que la capacité de cette coupe soit maximale.

A rappeler que la capacité d'une coupe étant définie comme la somme des capacités des arcs interconnectant S et $\overline{S}$.

Mathématiquement, le problème peut être formulé de la manière suivante :

Soit A la matrice $n \times n$ adjacente du graphe.

On définit la matrice :

$$\begin{cases} L = diag(Ae) - A \\ e = (1, ..., 1)^T \in \mathbb{R}^n. \end{cases}$$

La matrice L est dite la matrice laplacienne associée au graphe.

On représente la coupe S par le vecteur x tel qu'à chacun des sommets i du graphe, nous associons une valeur x_i de la manière suivante :

$x_i = 1$ si le sommet i est dans S

$x_i = -1$ si le sommet i est dans $\overline{S}$.

On aura donc la formulation suivante du problème :

$$(MC) \begin{cases} \max \frac{1}{4} x^T L x \\ x = \{-1, 1\}^n \end{cases}$$

Si on pose : $X = \frac{1}{4} x x^T$, on obtient un problème de la forme (SDP) suivant :

$$\begin{cases} \max L \bullet X \\ A_i \bullet X = \frac{1}{4} \quad i = 1, ..., m \\ X \in \mathbb{S}^n_+. \end{cases}$$

Où : $A_i[j,k] = \begin{cases} 1 & i = j = k \\ 0 & ailleurs. \end{cases}$

2.3.3 Approximation logarithmique de Tchebychev

Supposons qu'on cherche à résoudre approximativement le système d'équations $a_i^T x = b_i$ pour $i = 1, ..., m$.

Le terme approximativement a été utilisé parce que la résolution exacte est très difficile (trop de contraintes). On peut dès lors choisir plusieurs critères pour déterminer la meilleure solution. Le plus courant est la minimisation de la somme des carrés des écarts $(a_i^T x - b_i)$; cependant, dans l'approximation de Chebychev, on minimise la norme du résidu l_∞ ; i.e., on minimise l'écart maximal en valeur absolue, selon

$$\min \max_i |a_i^T x - b_i| \tag{2.1}$$

Ce problème peut se mettre sous la forme du programme linéaire suivant

$$\begin{cases} \min t \\ -t \leq a_i^T x - b_i \leq t, \quad i = 1, ..., m, \end{cases}$$

où x est une variable et t est une variable auxiliaire.

Dans certaines applications les b_i sont des quantités s'exprimant dans une unité de puissance ou d'intensité et généralement de logarithme. Dans ce cas, il est préférable d'un point de vue purement physique de minimiser le maximum des écarts entre les logarithmes de $a_i^T x$ et b_i, ce qui se traduit par

$$\min \max_i |\log a_i^T x - \log b_i| \tag{2.2}$$

On suppose que $b_i > 0$ et $a_i^T x > 0, \forall i = 1, ..., m$. Ce problème est appelé problème d'approximation logarithmique de Chebychev et peut être transformé en un programme semi-défini. On transforme alors les contraintes en

$$-\log t \leq \log \frac{a_i^T x}{b_i} \leq \log t, \quad \text{d'où } \frac{1}{t} \leq \frac{a_i^T x}{b_i} \leq t.$$

Le problème (2.2) est équivalent au problème semi-défini suivant

$$\begin{cases} \min t \\ \frac{1}{t} \leq \frac{a_i^T x}{b_i} \leq t, \quad i = 1, ..., m, \end{cases}$$

ou

$$\begin{cases} \min t \\ \begin{pmatrix} t - \frac{a_i^T x}{b_i} & 0 & 0 \\ 0 & \frac{a_i^T x}{b_i} & 1 \\ 0 & 1 & t \end{pmatrix} \succeq 0, \quad i = 1, ..., m. \end{cases}$$

Le premier terme de la diagonale est responsable de l'inégalité $\frac{a_i^T x}{b_i} \leq t$, tandis que $\frac{a_i^T x}{b_i} \geq \frac{1}{t}$ se traduit par la matrice 2×2 inférieure droite. En effet, pour une matrice 2×2, le fait d'être semi-définie-positive implique la non-négativité du déterminant, d'où on déduit $\frac{a_i^T x}{b_i} t - 1 \geq 0$ et finalement l'inégalité recherchée.

Donc, cet exemple illustre deux points importants. Premièrement, il montre que la programmation semi-définie inclut beaucoup de problèmes d'optimisation qui ne semblent pas pour la première vue. Deuxièment, il montre que le problème est plus général qu'un programme linéaire, malgré la proche analogie.

2.3.4 Problèmes géométriques en formes quadratiques

Plusieurs problèmes géométriques impliquant des fonctions quadratiques peuvent être exprimés sous forme des programmes semi-définis. On présente ici un exemple simple.

Supposons, qu'on donne k ellipsoïdes $\xi_1, ..., \xi_k$ décrivent comme l'ensemble des sous niveaux des fonctions quadratiques

$$f_i(x) = x^T A_i x + 2b_i^T x + c_i, \quad i = 1, ..., k,$$

i.e., $\xi_i = \{x / f_i(x) \leq 0\}$. Le but est de trouver la plus petite sphère contenant l'ensemble des k ellipsoïdes donnés. La condition qu'un ellipsoïde contient un autre, peut être exprimée en termes d'une inégalité matricielle.

Supposons que les ellipsoïdes $\xi = \{x : f(x) \leq 0\}$ et $\widetilde{\xi} = \left\{x : \widetilde{f}(x) \leq 0\right\}$, avec

$$f(x) = x^T Ax + 2b^T x + c, \quad \widetilde{f}(x) = x^T \widetilde{A} x + 2\widetilde{b}^T x + \widetilde{c},$$

ont un intérieur non vide. Alors, on peut montrer que $\widetilde{\xi}$ contient ξ si et seulement s'il existe un $\tau \geq 0$ tel que

$$\begin{pmatrix} A & b \\ b^T & c \end{pmatrix} \preceq \tau \begin{pmatrix} \widetilde{A} & \widetilde{b} \\ \widetilde{b}^T & \widetilde{c} \end{pmatrix}.$$

Revenons à notre problème, considérons la sphère S représentée par

$$f(x) = x^T x - 2x_c^T x + \gamma \leq 0$$

S contient les ellipsoïdes $\xi_1, ..., \xi_k$ si et seulement s'il existe $\tau_1, ..., \tau_k$ positifs tels que

$$\begin{pmatrix} I & -x_c \\ -x_c^T & \gamma \end{pmatrix} \preceq \tau_i \begin{pmatrix} A_i & b_i \\ b_i^T & c_i \end{pmatrix}, \quad i = 1, ..., k.$$

Notre but est de minimiser le rayon de la sphère S qui est $r = \sqrt{x_c^T x_c - \gamma}$. Pour cela, on exprime la condition $r^2 \leq t$ par l'inégalité matricielle suivante

$$\begin{pmatrix} I & x_c \\ x_c^T & t + \gamma \end{pmatrix} \succeq 0,$$

et on minimise la variable t.

Donc, la recherche de la plus petite sphère qui contient les ellipsoïdes $\xi_1, ..., \xi_k$, revient à la résolution du problème de programmation semi-définie suivant

$$\begin{cases} \min t \\ \begin{pmatrix} I & -x_c \\ -x_c^T & \gamma \end{pmatrix} \preceq \tau_i \begin{pmatrix} A_i & b_i \\ b_i^T & c_i \end{pmatrix}, \quad i = 1, ..., k, \quad \tau_i \geq 0, \\ \begin{pmatrix} I & x_c \\ x_c^T & t + \gamma \end{pmatrix} \succeq 0, \end{cases}$$

où $x_c, \tau_1, ..., \tau_k$ et t sont les variables du problème.

Cet exemple montre encore l'ampleur des problèmes qui peuvent être reformulés en programmation semi-définie et que cette reformulation n'est pas simple.

2.3.5 Optimisation quadratique non convexe

La programmation semi-définie joue un rôle très utile dans l'optimisation non convexe ou combinatoire. Considérons par exemple, le problème d'optimisation quadratique suivant

$$\begin{cases} \min f_0(x) \\ f_i(x) \leq 0, \quad i = 1, ..., L, \end{cases} \tag{2.3}$$

où $f_i(x) = x^T A_i x + 2b_i^T x + c_i, i = 0, ..., L$. Les matrices A_i peuvent être indéfinies, dans ce cas le problème (2.3) est un problème d'optimisation non convexe très difficile à résoudre. Par exemple, il inclut tous les problèmes d'optimisation de fonction objectif polynômiale et contraintes polynômiales (voir [49, 57]).

Dans la pratique, par exemple l'algorithme de branch-and-bound, il est important d'avoir des bonnes et moins couteuses estimations inférieures de la valeur optimale de (2.3) qui soient calculables efficacement. Shor [57] et Poljak et al. [54] ont proposé le calcul des estimations inférieures par la résolution du programme semi-défini suivant

$$\begin{cases} \max t \\ \begin{pmatrix} A_0 & b_0 \\ b_0^T & c_0 - t \end{pmatrix} + \tau_1 \begin{pmatrix} A_1 & b_1 \\ b_1^T & c_1 \end{pmatrix} + ... + \tau_L \begin{pmatrix} A_L & b_L \\ b_L^T & c_L \end{pmatrix} \succeq 0, \\ \tau_i \geq 0, \quad i = 1, ..., L. \end{cases} \tag{2.4}$$

Où t et τ_i sont les variables du problème. On peut vérifier facilement que ce programme semi-défini (2.4) offre des estimations inférieures pour (2.3).

Supposons que x est réalisable pour le problème non convexe (2.3), i.e.,

$$f_i(x) = \begin{pmatrix} x \\ 1 \end{pmatrix}^T \begin{pmatrix} A_i & b_i \\ b_i^T & c_i \end{pmatrix} \begin{pmatrix} x \\ 1 \end{pmatrix} \leq 0,$$

Alors, si les variables t et $\tau_1, ..., \tau_L$, satisfont les contraintes du programme semi-

défini (2.4) on déduit que

$$\begin{aligned} 0 &\leq \begin{pmatrix} x \\ 1 \end{pmatrix}^T \left[\begin{pmatrix} A_0 & b_0 \\ b_0^T & c_0 - t \end{pmatrix} + \tau_1 \begin{pmatrix} A_1 & b_1 \\ b_1^T & c_1 \end{pmatrix} + ... + \tau_L \begin{pmatrix} A_L & b_L \\ b_L^T & c_L \end{pmatrix} \right] \begin{pmatrix} x \\ 1 \end{pmatrix} \\ &= f_0(x) - t + \tau_1 f_1(x) + ... + \tau_L f_L(x) \\ &\leq f_0(x) - t \end{aligned}$$

Par conséquent $t \leq f_0(x)$ pour tout point x réalisable dans (2.3). Le problème (2.4) peut être aussi obtenu via la dualité du lagrangien, pour plus de détails, voir Shor [57] ou Poljak et al. [54].

Par ailleurs, le problème dual associé au programme semi-défini (2.4) est donné par

$$\begin{cases} \min Tr(A_0 X) + 2b_0^T x + c_0 \\ Tr(A_i X) + 2b_i^T x + c_i \leq 0, \quad i = 1, ..., L \\ \begin{pmatrix} X & x \\ x^T & 1 \end{pmatrix} \succeq 0, \end{cases} \tag{2.5}$$

où les variables sont $X \in \mathbb{S}^n$ et $x \in \mathbb{R}^n$.

On voit que les deux problèmes (2.4) et (2.5) donnent la même estimation.

Notons que la contrainte

$$\begin{pmatrix} X & x \\ x^T & 1 \end{pmatrix} \succeq 0,$$

est équivalente à $X \succeq xx^T$. Par conséquent, le problème de programmation semi-défini (2.5) peut directement se considéré comme une relaxation du problème original (2.3),

qui peut s'écrire comme suit

$$\begin{cases} \min Tr(A_0 X) + 2b_0^T x + c_0 \\ Tr(A_i X) + 2b_i^T x + c_i \leq 0, \quad i = 1, ..., L \\ X = xx^T. \end{cases} \tag{2.6}$$

La seule différence entre les deux problèmes (2.6) et (2.5) est le remplacement de la contrainte non convexe $X = xx^T$ par la relaxation de la contrainte convexe $X \succeq xx^T$.

Il est intéressant de noter que la formulation (2.6) est équivalente au problème (2.3)

Par exemple, nous considérons le programme $(-1,1)$ quadratique

$$\begin{cases} \min x^T Ax + 2b^T x \\ x_i^2 = 1, \quad i = 1, ..., k, \end{cases} \tag{2.7}$$

qui est un problème non convexe difficile. La contrainte entière $x_i \in \{-1, 1\}$ peut être écrite comme une contrainte d'égalité quadratique $x_i^2 = 1$ ou l'équivalence de deux inégalités $x_i^2 \leq 1$ et $x_i^2 \geq 1$. En appliquant (2.5), on trouve que le programme semi-défini

$$\begin{cases} \min Tr(AX) + 2b^T x \\ x_{ii} = 1, \quad i = 1, ..., k, \\ \begin{pmatrix} X & x \\ x^T & 1 \end{pmatrix} \succeq 0, \end{cases}$$

donne une estimation inférieure de la valeur optimale du problème (2.7), avec $X = X^T$ et x sont des variables.

2.3.6 Quelques problèmes non linéaires

Exemple 2.3.1 *Considérons le problème non linéaire suivant :*

$$\begin{cases} \min(x_1^3 + x_2) \\ x_1^3 x_2 \geq 1 \\ x_1 \geq 0, \ x_2 \geq 0 \end{cases}$$

ce problème s'écrit sous forme d'un programme (SDP) comme suit :

$$(SDP) \begin{cases} \min C \bullet X \\ A_1 \bullet X = b_1 \\ X \succeq 0 \end{cases}$$

avec $C = I, \ X = \begin{pmatrix} x_1^3 & x_3 \\ x_3 & x_2 \end{pmatrix}, \quad A_1 = \begin{pmatrix} 0 & 0.5 \\ 0.5 & 0 \end{pmatrix}$ *et* $b_1 = 1.$

Exemple 2.3.2 *Considérons le problème non linéaire suivant :*

$$\begin{cases} \min \frac{(c^T x)^2}{d^T x} \\ Ax + b \geq 0, \end{cases}$$

où on suppose que $d^T x > 0$. *Ce problème se reformule alors de la manière suivante :*

$$\begin{cases} \min t \\ Ax + b \geq 0, \\ \frac{(c^T x)^2}{d^T x} \leq t \iff \begin{pmatrix} t & c^T x \\ c^T x & d^T x \end{pmatrix} \succeq 0 \end{cases}$$

On obtient alors le problème (SDP) *suivant :*

$$\begin{cases} \min t \\ \begin{pmatrix} diag(Ax+b) & 0 & 0 \\ 0 & t & c^T x \\ 0 & c^T x & d^T x \end{pmatrix} \succeq 0 \end{cases}$$

Exemple 2.3.3 *Considérons le problème non linéaire suivant :*

$$\begin{cases} \min \ \ g(\theta) = \frac{10}{\cos(\theta)} + 6 - 5\tan(\theta) \\ \theta \in]0; \frac{\pi}{2}[\end{cases}$$

ce problème s'écrit sous forme d'un programme (SDP) *comme suit :*

$$(SDP) \begin{cases} \min C \bullet X \\ A \bullet X > 1 \\ X \succeq 0 \end{cases}$$

avec

$$X = \begin{pmatrix} \frac{1}{\cos(\theta)} & 0 & 0 \\ 0 & \tan(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}, C = \begin{pmatrix} 10 & 0 & 0 \\ 0 & -5 & 0 \\ 0 & 0 & 6 \end{pmatrix} \text{ et } A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

2.4 Dualité

La dualité en (SDP) linéaire est très similaire à la dualité en (PL) classique à quelques différences près.

Soit le problème (SDP) linéaire standard

$$(SDP) \begin{cases} \min C \bullet X \\ A_i \bullet X = b_i, \ i = 1, ..., m \\ X \succeq 0 \end{cases}$$

Pour obtenir le problème dual de (SDP), on considère la fonction lagrangienne

$$\begin{aligned} q(y) &= \min_{X\in\mathbb{S}^n_+}\left[C\bullet X+\sum_{i=1}^m(b_i-A_i\bullet X)y_i, \quad y\in\mathbb{R}^m,\right] \\ &= \min_{X\in\mathbb{S}^n_+}\left[(C-\sum_{i=1}^m y_iA_i)\bullet X+b^Ty, \qquad y\in\mathbb{R}^m,\right] \end{aligned}$$

et on a :

$$\max_{y\in\mathbb{R}^m} q(y) = \begin{cases} \max b^Ty & \text{si } C-\sum_{i=1}^m y_iA_i\succeq 0 \\ -\infty & \text{ailleurs.} \end{cases}$$

Donc, par convention le dual de (SDP) est

$$(DSDP)\begin{cases} \max b^Ty \\ C-\sum\limits_{i=1}^m y_iA_i\succeq 0, \quad i=1,...,m \\ y\in\mathbb{R}^m. \end{cases}$$

C'est un programme semi-défini qui s'écrit aussi :

$$(DSDP)\begin{cases} \max b^Ty \\ S+\sum\limits_{i=1}^m y_iA_i=C, \quad i=1,...,m \\ y\in\mathbb{R}^m, \quad S\succeq 0. \end{cases}$$

Définition 2.4.1 *Un couple $(y,S)\in\mathbb{R}^m\times\mathbb{S}^n$ est dit solution réalisable de $(DSDP)$ si*

$$S\succeq 0, \qquad \sum_{i=1}^m y_iA_i+S=C, \qquad i=1,...,m$$

on designe par $\mathcal{F}_d=\{(y,S)\in\mathbb{R}^m\times\mathbb{S}^n, \quad (y,S)$ réalisable$\}$, l'ensemble des solutions réalisables duales de $(DSDP)$.
De même (y,S) est dit strictement réalisable de $(DSDP)$ si

$$S\succ 0, \qquad \sum_{i=1}^m y_iA_i+S=C, \qquad i=1,...,m$$

on designe par $\mathcal{F}_d^+=\{(y,S)\in\mathbb{R}^m\times\mathbb{S}^n, \quad (y,S)$ strictement réalisable$\}$, l'ensemble des solutions strictement réalisables duales de $(DSDP)$.

Définition 2.4.2 - *La valeur optimale de* $(DSDP)$ *est définie par :*

$$d^* = \sup_y [b^T y \ tq \quad C - \sum_{i=1}^{m} y_i A_i \in \mathbb{S}_+^n \quad i = 1, ..., m, \quad y \in \mathbb{R}^m].$$

- (y^*, S^*) *est une solution optimale duale de* $(DSDP)$ *si*

$$(y^*, S^*) \in \mathcal{F}_d, \text{ et } b^T y^* = d^*.$$

2.4.1 Relations primales-duales pour la programmation semi-définie

Comme pour la programmation linéaire, le problème (SDP) peut s'écrire sous plusieurs formes, le tableau suivant présente les différents types de problèmes duaux correspondants

Minimisation	Maximisation
Variables	Contraintes
matrice ou scalaire ≥ 0	matrice ou scalaire $\leq$
matrice ou scalaire ≤ 0	matrice ou scalaire $\geq$
matrice $\succeq 0$	matrice $\preceq$
matrice $\preceq 0$	matrice $\succeq$
matrice ou scalaire non astreint	matrice ou scalaire $=$
Contraintes	Variables
matrice ou scalaire $\geq$	matrice ou scalaire ≥ 0
matrice ou scalaire $\leq$	matrice ou scalaire ≤ 0
matrice $\succeq$	matrice $\succeq 0$
matrice $\preceq$	matrice $\preceq 0$
matrice ou scalaire $=$	matrice ou scalaire non astreint

2.4.2 Dualité faible

Proposition 2.4.1 *Soit* $X \in \mathcal{F}_p$ *et* $(y, s) \in \mathcal{F}_d$, *alors*

$$C \bullet X - b^T y = S \bullet X \geq 0, \text{ et } p^* \geq d^*.$$

Preuve. 1)-On a

$$\begin{aligned} C \bullet X - b^T y &= C \bullet X - \sum_{i=1}^{m} b_i y_i \\ &= C \bullet X - \sum_{i=1}^{m} y_i (A_i \bullet X) \\ &= (C - \sum_{i=1}^{m} y_i A_i) \bullet X \\ &= S \bullet X \geq 0, \text{ car } S \succeq 0 \text{ et } X \succeq 0 \end{aligned}$$

donc

$$C \bullet X - b^T y \geq 0, \quad \forall X \in \mathcal{F}_p \text{ et } \forall (y, s) \in \mathcal{F}_d.$$

2)- Nous avons

$$\begin{array}{rrcll} & C \bullet X & \geq & b^T y & \forall X \in \mathcal{F}_p \text{ et } \forall y \in \mathcal{F}_d \\ \Rightarrow & C \bullet X^* & \geq & b^T y & \forall (y, s) \in \mathcal{F}_d \\ \Rightarrow & p^* = C \bullet X^* & \geq & b^T y^* = d^* & \\ \Rightarrow & p^* & \geq & d^* & \end{array}$$

d'où la dualité faible. ■

2.4.3 Dualité forte

Bien que la programmation linéaire et la programmation (SDP) aient une structure très similaire, mais certains résultats de la dualité en (PL) ne sont pas valable en (SDP) en particulier, la dualité forte qui n'est pas vérifiée sauf si la stricte réalisabilité de l'un des deux problèmes est préservée, comme le montre le théorème suivant :

Théorème 2.4.1 *1) Si le problème (SDP) est strictement réalisable i.e., $\exists X \in \mathcal{F}_p^+$ alors $p^* = d^*$. Si en autre, p^* est fini, alors l'ensemble des solutions optimales duales de $(DSDP)$ est compact non vide.*
2) Si le problème $(DSDP)$ est strictement réalisable i.e., $\exists (y, S) \in \mathcal{F}_d^+$ alors $p^* = d^*$. Si en autre, d^* est fini, alors l'ensemble des solutions optimales primales de (SDP) est compact non vide.

2.5 Complémentarité en SDP

A l'image de ce qui se passe en programmation linéaire, on peut exprimer la condition pour X^* et y^* d'être des solutions optimales de (SDP) et $(DSDP)$ sous la forme d'une condition dite de complémentarité.

Théorème 2.5.1 *Soient $X^* \in \mathcal{F}_P$ et $(y, Z) \in \mathcal{F}_D$ de saut de dualité*

$$< C, X^* > -b^T y^* =< X^*, Z^* > .$$

Alors X^* et (y^*, Z^*) sont des solutions optimales pour (SDP) et $(DSDP)$ respectivement si et seulement si $X^*Z^* = 0$.

Le problème de complémentarité s'écrit alors :

$$(Pc) \begin{cases} < A_i, X >= b_i \qquad i = 1, ..., m, \quad X \succeq 0 \\ Z + \sum_{i=1}^{m} y_i A_i = C, \qquad Z \succeq 0 \\ XZ = 0. \end{cases}$$

Remarque 2.5.1 *Comme (SDP) est un problème convexe, la solution du problème de complémentarité est un optimum global pour (SDP).*

2.6 Méthodes de résolution de SDP

La similarité de SDP avec la programmation linéaire PL, a motivé les chercheurs d'appliquer des techniques ayant fait leurs preuves pour les PL, en particulier les méthodes de trajectoire centrale de type primal-dual de points intérieurs.

La généralisation des méthodes de points intérieurs de PL à SDP remonte au début des années 1990. Les premiers algorithmes dans ce sens ont été introduites de façon indépendante par Alizadeh [2] et Nesterov et Nemirovskii [49]. Alizadeh [2] a étendu l'algorithme projectif de réduction du potentiel de Ye à partir de PL à SDP et affirme que de nombreux algorithmes de points intérieurs connus pour PL peuvent être transformés en des algorithmes pour résoudre SDP. Nesterov et Nemirovski [49] ont présenté la théorie profonde des méthodes de points intérieurs qui est basée sur des fonctions barrières auto-concordantes.

2.6.1 Méthodes de points intérieurs

La méthode de points intérieurs est l'une des méthodes les plus utilisées et les plus efficaces pour la résolution des problèmes SDP, elle est relativement nouvelle et s'apparente à la méthode projective de Karmarkar pour la programmation linéaire.

Derière le terme points intérieurs découlent trois différents types de méthodes

1. Les méthodes affines.
2. Les méthodes de réduction du potentiel.
3. Les méthodes de trajectoire centrale.

a) Méthodes affines

L'idée remonte à Dikin (1967), Puis reprise et développée par plusieurs chercheurs au milieu des années 80. Il s'agit pratiquement de l'algorithme de Karmarkar sans fonction potentielle et sans transformation projective, on utilise une transformation affine puis on remplace la contrainte de non négativité par un ellipsoïde qui contient le nouvel itéré.

L'algorithme correspondant à cette méthode est d'une structure simple, malheureusement, il n'est pas facile de démontrer la polynomialité. À l'égard de cette dernière, Dikin [20], en 1967 a prouvé la convergence de la méthode affine primale sous des hypothèses de non dégénérescence. En 1990, Monteiro et al. [45] ont démontré que la méthode affine primale-duale est de complexité polynômiale ($O(nL^2)$), où L représente le nombre de bits requis pour stocker (et traiter) les données, par contre, la convergence pour les méthodes affines primales ou duales est restée une question ouverte.

b) Méthodes de réduction du poteniel

Ces méthodes sont le fruit direct d'une grande partie des études acharnées menées par plusieurs chercheurs vers la fin des années 80. En 1995, Alizadeh a proposé une méthode primale-duale de réduction du potentiel [2].

En 2003 et inspirant des travaux de Alizadeh, Benterki [9] a fait une étude algorithmique et numérique sur des méthodes de reduction du potentiel pour résoudre des problèmes de programmation semi-définie.

Les ingredients principaux utilisés dans ce type d'algorithme sont :

1. Une transformation projective T_k qui permet de ramener (SDP) à la forme réduite plus maniable (ressemble à la forme réduite de Karmarkar en PL).

Plaçons nous à l'étape k, où l'on dispose de l'itéré $X_k \in \mathbb{S}^n_{++}$. La factorisation de Cholesky de X_k donne une matrice triangulaire inférieure L_k dont les éléments diagonaux sont strictements positifs tel que $L_k L_k^T = X_k$. Alors, la tranformation

projective est définie comme suit :

$$\begin{aligned} T_k : \ & \mathbb{S}^n \to \mathbb{S}^n \times \mathbb{R}^r \\ & X \mapsto (\overline{X}, \overline{x}) \\ & T_k(X) = (\overline{X}, \overline{x}), \end{aligned}$$

où

$$\overline{X} = \frac{(n+r)L_k^{-1} X L_k^{-T}}{r + < X_k^{-1}, X >} \ , \ \overline{x} = \frac{(n+r)}{r + < X_k^{-1}, X >} e_r.$$

Ici $e_r = (1, 1, ..., 1)^T \in \mathbb{R}^r$. Notons que $X = T_k^{-1}(\overline{X}, \overline{x})$.

Le problème (SDP) sera équivalent au problème semi-défini suivant :

$$(TSDP) \begin{cases} \min < \overline{C}, \overline{X} > + \overline{c}^T(z)\overline{x} \\ \overline{\overline{\mathcal{A}}} \begin{pmatrix} vec\overline{X} \\ \overline{x} \end{pmatrix} = \begin{pmatrix} 0 \\ n+r \end{pmatrix} \\ ||\overline{X} - I||^2 + ||\overline{x} - e_r||_2^2 \leq \beta^2 \end{cases}$$

où

– $\overline{C} = L_k^T C L_k \in \mathbb{S}^n$.

– $\overline{c}(z) = -\left(\frac{x}{r}\right) e_r$ est un vecteur de $\mathbb{R}^r$.

– $\overline{A_i} = L_k^T A_i L_k \in \mathbb{S}^n$.

– $\overline{\mathcal{A}} = \mathcal{A}(L_k \otimes L_k)$ est une $m \times n^2$ matrice.

– $\overline{A} = \left(\frac{-1}{r}\right) b e_r^T$ est une $m \times r$ matrice.

– $\overline{\overline{\mathcal{A}}} = \begin{pmatrix} \overline{\mathcal{A}} & \overline{A} \\ (vecI)^T & e_r^T \end{pmatrix}$ est une $(m+1) \times (n^2 + r)$ matrice.

La solution optimale $(\overline{X}^*, \overline{x}^*)$ de $(TSDP)$ est facile à obtenir ; il suffit par exemple d'écrire les conditions d'optimalité, on trouve :

$$\begin{pmatrix} vec\overline{X}^* \\ \overline{x}^* \end{pmatrix} = \begin{pmatrix} vecI \\ e_r \end{pmatrix} - \beta \frac{p(z)}{||p(z)||},$$

où $p(z)$ est la projection du vecteur coût $\begin{pmatrix} vec\overline{C} \\ \overline{c}(z) \end{pmatrix}$ sur le noyau de la matrice $\overline{\overline{\mathcal{A}}}$.

D'où $X = T_k^{-1}(\overline{X}^*, \overline{x}^*)$ est une solution réalisable de (SDP) candidate à l'optimalité.

2. Fonction potentielle primale-duale :

$$\psi(X, y) = (n+r) \log < X, C - \sum_{i=1}^{m} y_i A_i > - \log \det(X(C - \sum_{i=1}^{m} y_i A_i)),$$

définie pour tout $X \in \mathbb{S}^n_{++}$ et tout $y \in \mathbb{R}^m$ vérifiant $C - \sum_{i=1}^m y_i A_i \in \mathbb{S}^n_{++}$.

Cette fonction permet de prouver la convergence polynômiale de l'algorithme proposé par Alizadeh [2].

c) Méthodes de trajectoire centrale de type primal-dual

Les méthodes de trajectoire centrale primale-duale ont été introduites à la même époque que les méthodes de réduction du potentiel et pleinement développées au début des années 90. Elles ont attiré une grande intention de la part des chercheurs dans le monde entier et elles montrent en général un excellent comportement pratique et théorique (une complexité polynômiale et une convergence super-linéaire). On trouve en 1996, les travaux de Helmberg et al. [30] qui ont proposé un algorithme primal-dual de trajectoire centrale pour SDP. En 1997, Monteiro [47] a proposé une méthode primale-duale de trajectoire centrale et a montré la convergence polynômiale de l'algorithme à court et long pas. En 1999, Ji et al. [37] ont étudié la convergence de la méthode de trajectoire centrale de type predicteur-correcteur. Aussi, Monteiro et al. [46] ont proposé une méthode primale-duale de trajectoire centrale et ont montré la convergence polynômiale vers une solution optimale. Ces travaux se poursuivent à ce jour, on trouve ainsi ceux de Halicka et al. (2002) [27] qui ont proposé une étude sur la convergence de la méthode de trajectoire centrale en optimisation semi-définie. En 2007, Koulaei et al. [36] ont proposé une extension de l'algorithme de Mehrotra predicteur-correcteur basé sur la direction de Nesterov et todd (NT). En 2012, Liu et al. [39] ont présenté un nouveau algorithme de type correcteur de second ordre pour SDP et ont prouvé la convergence polynômiale de ce dernier pour la direction de Nesterov et Todd (NT). En 2015, Kettab et al. [33] ont proposé une relaxation de la méthode barrière logarithmique pour la programmation semi-définie.

Le principe de ces méthodes est de minimiser le saut de dualité en résolvant le système suivant :

$$\begin{cases} < A_i, X >= b_i \qquad i = 1, ..., m, \\ Z + \sum_{i=1}^{m} y_i A_i = C, \\ XZ = \mu I. \quad X, Z \succeq 0 \end{cases} \tag{2.8}$$

qui est le système paramétrisé des conditions d'optimalité des problèmes (SDP)

et $(DSDP)$. Le système (2.8) admet une solution unique sous les hypothèses, A_i, $i = 1, ..., n$ sont linéairement indépendantes, et $\mathcal{F}^\circ \neq \emptyset$. La solution du système est notée par $(X(\mu), y(\mu), Z(\mu))$ pour $\mu > 0$ fixé.

L'ensemble des solutions $(X(\mu), y(\mu), Z(\mu))$ pour tout $\mu > 0$ définie la trajectoire centrale qui converge vers la solution optimale quand μ tend vers 0.

Pour résoudre le système non linéaire (2.8), on utilise la méthode de Newton. L'objectif est d'obtenir des directions primales et duales $\triangle X, \triangle y$ et $\triangle Z$, respectivement en résolvant le système linéaire :

$$\begin{cases} < A_i, \triangle X >= 0 \qquad i = 1, ..., m, \\ \sum_{i=1}^{m} \triangle y_i A_i + \triangle Z = 0, \\ X \triangle Z + Z \triangle X = \mu I - XZ. \quad X, Z \succ 0. \end{cases} \tag{2.9}$$

L'itération de Newton complète est définie par :

$$\begin{aligned} X^+ &= X + \triangle X, \\ y^+ &= y + \triangle y, \\ Z^+ &= Z + \triangle Z. \end{aligned}$$

et doit être strictement réalisable (i.e., $X^+ \in \mathcal{F}_p^\circ$ et $(y^+, Z^+) \in \mathcal{F}_D^\circ$).

Malheureusement, X^+ n'est pas toujours symétrique, pour remédier ce problème, plusieurs chercheurs ont proposé dans la litérature des directions symétriques. On cite par exemples les travaux de Zhang [72] , Helemberg et al., Kojima et al. et Monteiro [30, 35, 47], Alizadeh-Heaberly-Overton [3] et Nesterov-Todd [59].

Actuellement, plusieurs chercheurs [16, 24, 64, 66] ont étendu l'approche des fonctions noyaux de la programmation linéaire à la programmation semie-définie pour trouver de nouvelles directions et améliorer la complexité des algorithmes. Une étude détaillée sur ce type de méthodes sera présentée dans le chapitre 4.

2.6.2 Autres méthodes de résolution d'un programme semi-définie

Dans la littérature, il existe d'autres méthodes de résolution d'un programme semi-défini, moins connues que la méthode de points intérieurs.

Considérons maintenant la méthode des faisceaux [29, 30, 52]. Si l'on suppose que la condition suivante est vérifiée

$$\exists \overline{y} \text{ tel que } I = \mathcal{A}^T \overline{y}. \tag{2.10}$$

La méthode des faisceaux est basée sur le problème de valeur propre suivant, équivalent à (DSDP) :

$$\max_{y} \alpha \lambda_{\min}(C - \mathcal{A}^T \overline{y}) + b^T y \tag{2.11}$$

L'hypothèse (2.10) revient à supposer que les solutions primales admissibles de (SDP) ont une trace constante, notée α, i.e. $tr(X) = \alpha \; \forall X \in \{X \succeq 0, \mathcal{A}X = b\}$. Il est important de noter qu'un grand nombre de programmes semi-définis peuvent s'écrire ainsi.

Par ailleurs, la fonction $\lambda_{min}(C - A^T \overline{y})$ est une fonction concave et non différentiable. Les méthodes de faisceaux permettent de donner une solution approchée de la valeur propre minimale.

D'autre part, Burer et al. [15] ont résolu une classe des programmes semi-définis via la programmation non linéaire.

Chapitre 3

Méthode projective primale-duale de points intérieurs pour SDP

3.1 Introduction

L'ensemble $\mathbb{S}_+^n$ est un cône non-polyédrique, la notion de sommet n'est donc plus valable pour les problèmes SDP, ce qui favorise d'avantage l'extension des méthodes de points intérieurs en programmation linéaire pour SDP. Plusieurs travaux ont été réalisés depuis les années 90**,** et le grand nombre d'articles parus dans des revues internationales en témoignent. Tout particulièrement, les travaux de Shapiro, Fletcher-Craven (1996) qui s'intéressent aux conditions d'optimalité du problème SDP, les travaux de Ramana et Wolkowic (1997) qui ont étudié la dualité forte pour ces problèmes. Du point de vue algorithmique, on rencontre les méthodes de points intérieurs qui sont relativement nouvelles et qui s'apparentent à la méthode projective de Karmarkar pour la programmation linéaire. Ces dernières années, plusieurs chercheurs ont proposé des méthodes pour résoudre les problèmes SDP qui sont généralement des extensions des méthodes de points intérieurs pour la programmation linéaire. En effet, en 1994, Farid Alizadeh [2] est le premier chercheur qui a fait une étude profonde sur SDP et a proposé un algorithme primal-dual de points intérieurs du type projectif appelé **"Algorithme de réduction du potentiel"**. En 2004, Dj. Benterki et al. [10] se sont intéressés à l'étude numérique de l'algorithme projectif de Alizadeh.

Toutes ces méthodes de différents types à savoir les méthodes de trajectoire centrale réalisable et non-réalisable, primale, duale et primale-duale ont un problème majeur commun celui de l'initialisation.

En 2007, Dj. Benterki et al. [11] ont proposé une méthode de points intérieurs réalisable pour résoudre SDP.

L'avantage principal de cette méthode est qu'elle fournisse une solution réalisable initiale de SDP.

Dans ce chapitre, et motivés par leurs travaux, nous présentons un algorithme efficace primal-dual de points intérieurs à deux phases sans recherche linéaire pour les problèmes d'optimisation semi-définis. Nous proposons trois nouvelles alternatives qui donnent des pas de déplacement efficaces et améliorent le comportement de l'algorithme. De plus, nous présentons également quelques expérimentations numériques pour consolider nos propos théoriques [71].

3.2 Position du problème

Soit le programme semi-défini suivant :

$$\begin{cases} z^* = \min C \bullet X \\ A_i \bullet X = b_i, \quad i = 1, ..., m \\ X \succeq 0 \end{cases} \qquad (SDP)$$

et son dual :

$$\begin{cases} w^* = \max b^T y \\ S = C - \sum_{i=1}^{m} y_i A_i, \\ S \succeq 0. \end{cases} \qquad (DSDP)$$

où : $b = (b_1, b_2, ..., b_m) \in \mathbb{R}^m$; C, X et A_i $(\forall i = 1, ..., m)$ sont des matrices dans $\mathbb{S}^n$.

Supposons que :

1. $C \neq 0$.
2. A_i $(i = 1, ..., m)$ sont linéairement indépendantes.
3. Chacun de deux problèmes, problème primal (SDP) et respectivement le problème dual $(DSDP)$, possède une solution strictement réalisable i.e., $\mathcal{F}_p^+ \neq \emptyset$ et $\mathcal{F}_d^+ \neq \emptyset$.

Dans la suite, on décrit le passage de l'itération X_k à l'itération X_{k+1} à travers un algorithme de réduction de potentiel de type projectif.

Supposons qu'à l'itération k, on dispose d'une solution strictement réalisable X_k ($X_k \in \mathbb{S}^n_{++}$), donc la factorisation de Cholesky nous donne une matrice triangulaire inférieure L_k telle que : $X_k = L_k L_k^T$.

3.3 Transformation projective

On utilise la transformation projective donnée par Benterki et al. [11] pour ramener l'itération actuelle au centre $(I, 1)$ de l'ensemble $\widetilde{K}$. On définie

$$\begin{aligned} T_k : \quad & \mathbb{S}^n_+ \to \widetilde{K} \\ & X \mapsto T_k(X) = (\overline{X}, \overline{x}) \end{aligned}$$

où :

$$\overline{X} = \overline{x} L_k^{-1} X L_k^{-T} \quad , \quad \overline{x} = \frac{(n+1)}{1 + \langle X_k^{-1}, X \rangle}$$

et

$$\widetilde{K} = \{(\overline{X}, \overline{x}) \in \mathbb{S}^+_n \times]0, +\infty[\ : \ \langle I, \overline{X} \rangle + \overline{x} = n+1\},$$

T_k est inversible et on a :

$$X = T_k^{-1}(\overline{X}, \overline{x}) = \frac{1}{\overline{x}} L_k \overline{X} L_k^T$$

3.3.1 Problème transformé de (SDP)

Le problème transformé de (SDP) par T_k est le problème semi-défini suivant :

$$0 = \min_{(\overline{X}, \overline{x})} \left[\langle C_k, \overline{X} \rangle - z^* \overline{x} : \begin{array}{l} b_i \overline{x} - \langle A_i^{(k)}, \overline{X} \rangle = 0, i = 1, \ldots, m \\ \langle I, \overline{X} \rangle + \overline{x} = n+1, \\ \overline{X} \in \mathbb{S}^n_+, \ \overline{x} \geq 0. \end{array} \right] \qquad (TSDP1)$$

où

1. $C_k = L_k^T C L_k$.
2. $A_i^k = L_k^T A_i L_k \qquad \forall i = 1, ..., m.$

Preuve. 1) Commençant par l'objectif. On a :

$$\begin{aligned} \langle C, X \rangle - z^* = 0 \quad & \Rightarrow \quad \left\langle C, \tfrac{1}{\overline{x}} L_k \overline{X} L_k^T \right\rangle - z^* = 0 \\ & \Rightarrow \quad \left\langle L_k^T C L_k, \overline{X} \right\rangle - z^* \overline{x} = 0 \\ & \Rightarrow \quad \left\langle C_k, \overline{X} \right\rangle - z^* \overline{x} = 0 \end{aligned}$$

2) Pour les contraintes, on a :

$$\begin{aligned} \langle A_i, X\rangle = b_i &\Rightarrow \left\langle A_i, \tfrac{1}{\overline{x}} L_k \overline{X} L_k^T \right\rangle = b_i \\ &\Rightarrow \left\langle L_k^T A_i L_k, \overline{X} \right\rangle = \overline{x} b_i \\ &\Rightarrow \left\langle A_i^k, \overline{X} \right\rangle = \overline{x} b_i \end{aligned}$$

3) De même, on a :

$$\overline{X} = \overline{x} L_k^{-1} X L_k^{-T} \in \mathbb{S}_+^n$$

car

$$\left\langle \overline{X} t, t \right\rangle = \left\langle \overline{x} L_k^{-1} X L_k^{-T} t, t \right\rangle = \overline{x} \left\langle X L_k^{-T} t, L_k^{-T} t \right\rangle \geq 0$$

car $X \in \mathbb{S}_+^n$ ■

Remarque 3.3.1 $(I, 1)$ *est une solution strictement réalisable pour* $(TSDP1)$ *car :*

$$\begin{aligned} \left\langle A_i^k, I \right\rangle - b_i &= \left\langle L_k^T A_i L_k, I \right\rangle - b_i \\ &= \langle A_i, X_k \rangle - b_i = 0 \quad \forall i \end{aligned}$$

et

$$\langle I, I \rangle + 1 = n + 1,$$

3.3.2 Approximation de la valeur optimale z*

Comme la valeur optimale z^* est inconnue, on utilise une borne supérieure de z^*, en prenant à chaque itération k,

$$z_k = \langle C, X_k \rangle = \langle C_k, I \rangle$$

alors $z_k > z^*$.

Le problème $(TSDP1)$ devient :

$$r_k = \min_{(\overline{X}, \overline{x})} \left[\langle C_k, \overline{X} \rangle - z_k \overline{x} \ : \ \begin{array}{l} b_i \overline{x} - \langle A_i^{(k)}, \overline{X} \rangle = 0, i = 1, \ldots, m \\ \langle I, \overline{X} \rangle + \overline{x} = n + 1, \\ \overline{X} \in \mathbb{S}_n^+, \ \overline{x} \geq 0. \end{array} \right] \qquad (TSDP2)$$

De manière similaire à la méthode de Karmarkar pour les problèmes de programmation linéaire, on relaxe le problème $(TSDP2)$ en un problème d'optimisation convexe. Donc $(TSDP2)$ devient :

$$r_k(\beta) = \min_{(\overline{X}, \overline{x})} \left[\langle C_k, \overline{X} \rangle - z_k \overline{x} : \ \begin{array}{l} b_i \overline{x} - \langle A_i^{(k)}, \overline{X} \rangle = 0, i = 1, \ldots, m \\ \langle I, \overline{X} \rangle + \overline{x} = n + 1, \\ \|\overline{X} - I\|^2 + (\overline{x} - 1)^2 \leq \beta^2, \end{array} \right] \qquad (TSDP3)$$

avec $\beta > 0$. $(I, 1)$ est aussi une solution strictement réalisable pour $(TSDP3)$ et donc $r_k(\beta) \leq 0$. de plus, si $\beta \in [0, 1]$, l'ensemble des solutions réalisables de $(TSDP3)$ est inclu dans l'ensemble des solutions réalisables de $(TSDP2)$ et donc

$$r_k(\beta) \geq r_k$$

il est clair que si $0 < \beta < \beta'$ alors

$$0 \geq r_k(\beta) \geq r_k(\beta').$$

Proposition 3.3.1 *[11]*

$$r_k(\beta) < 0, \qquad \forall \beta > 0$$

3.3.3 Forme réduite du problème $(TSDP3)$

Pour simplifier la forme de $(TSDP3)$, Benterki et al. [11] ont fait le changement de variable suivant : $V = \overline{X} - I$ et $v = \overline{x} - 1$, donc le problème $(TSDP3)$ devient équivalent au problème d'optimisation convexe :

$$\min_{(V,v)} \left[\begin{array}{ll} & b_i v - \langle A_i^{(k)}, V \rangle = 0, \ i = 1, \ldots, m, \\ \langle C_k, V \rangle - z_k v : & \langle I, V \rangle + v = 0, \\ & \|V\|^2 + v^2 \leq \beta^2. \end{array} \right] \qquad (TSDP4)$$

3.4 Conditions d'optimalité et algorithme de résolution

3.4.1 Conditions d'optimalité

En utilisant les conditions nécessaires et suffisantes d'optimalité du problème convexe $(TSDP4)$, la solution optimale (V^*, v^*) est donée par

$$V^* = -\beta P_k \ \text{ et } \ v^* = -\beta p_k$$

où

$$P_k = \frac{V_k}{\tau}, \quad p_k = \frac{v_k}{\tau} \ \text{ et } \ \tau = (\|V_k\|^2 + v_k^2)^{\frac{1}{2}}.$$

avec

$$V_k = C_k - \sum_{i=1}^{m} y_i A_i^{(k)} \ \text{ et } \ v_k = \sum_{i=1}^{m} b_i y_i - z_k$$

et $y = (y_1, y_2, \ldots, y_m) \in \mathbb{R}^m$ est l'itérée duale du problème $(DSDP)$ qui est la solution d'un systeme linéaire $m \times m$

$$My = d, \tag{3.1}$$

où $\forall i, j = 1, \ldots, m$

$$M_{ij} = \langle A_i^{(k)}, A_j^{(k)} \rangle + b_i b_j,$$
$$d_i = \langle C_k, A_i^{(k)} \rangle + b_i z_k.$$

Remarque 3.4.1 *La matrice M est symétrique et définie-positive. Donc le système linéaire (3.1) peut être résolu par des méthodes directes (cas de faible ou moyenne dimension par exemple, Cholesky), ou par des méthodes itératives (cas de dimension élevée par exemple, gradient conjugué). Donc, V_k et v_k sont facilement obtenus.*

Revenant au problème $(TSDP3)$, sa solution est donnée par :

$$\begin{pmatrix} \overline{X}(\beta) \\ \overline{x}(\beta) \end{pmatrix} = \begin{pmatrix} I \\ 1 \end{pmatrix} - \beta \begin{pmatrix} P_k \\ p_k \end{pmatrix}$$

On choisit le pas de déplacement β de telle manière que la matrice $\overline{X}(\beta)$ reste définie-positive et le scalaire $\overline{x}(\beta)$ reste strictement positif.
Alors, le candidat de la nouvelle itération primale X est donné par :

$$X = T_k^{-1}(\overline{X}(\beta), \overline{x}(\beta))$$

et celui de la nouvelle itération duale (y, S) est donné par :

$$S = C - \sum_{i=1}^{m} y_i A_i$$

Notons que V_k, P_k, $\overline{X}(\beta)$ et X sont symétriques par construction.

Résumons maintenant l'algorithme.

3.4.2 Algorithme primal-dual

Initialisation :

1. $k = 0$, X_0 est une solution strictement réalisable du problème (SDP)
2. On choisit une précision $\varepsilon > 0$

Itération k :

1. Prendre $z_k = \langle C, X_k \rangle$.
2. Déterminer L_k telle que $X_k = L_k L_k^T$,
3. Calculer
   - $C_k = L_k^T C L_k$,
   - $A_i^{(k)} = L_k^T A_i L_k$, $i = 1, ..., m$.
   - $M_{ij} = \langle A_i^{(k)}, A_j^{(k)} \rangle + b_i b_j, \quad i, j = 1, ..., m$
   - $d_i = b_i z_k + \langle C_k, A_i^{(k)} \rangle, \quad i = 1, ..., m$
4. Résoudre le système $My = d$.
5. Calculer :
   - $V_k = (C_k - \sum_{i=1}^m y_i A_i^{(k)})$,
   - $v_k = (\sum_{i=1}^m b_i y_i - z_k)$,
   - $\tau = (\|V_k\|^2 + v_k^2)^{\frac{1}{2}}$.
6. Calculer :
   - $X_{k+1} = X_k - \dfrac{\beta_k}{\tau - \beta_k v_k} L_k (V_k - v_k I) L_k^T$.
   - $y_{k+1} = y$.
   - $S_{k+1} = C - \sum_{i=1}^m y_i^{k+1} A_i$.

Test d'arrêt :

1. Si $\beta_k[\langle C_k, V_k \rangle - z_k v_k] \leq \tau\varepsilon$ ou $< X_{k+1}, S_{k+1} > \leq \varepsilon$: **STOP,**
2. Sinon, poser $k = k + 1$ et revenir à **Itération k**

La proposition suivante, montre que pour tout $\beta > 0$ l'objectif de (SDP) se réduit d'une itération à l'autre ce qui assure la convergence de l'algorithme.

Proposition 3.4.1 *[11] Pour tout pas de déplacement $\beta > 0$ on a :*

$$X_{k+1} = X_k - \frac{\beta}{1 - \beta p_k}(L_k P_k L_k^T - p_k X_k),$$

$$r_k(\beta) = -\beta[\langle C_k, P_k \rangle - z_k p_k],$$

et

$$\langle C, X_{k+1}\rangle - \langle C, X_k\rangle = \frac{1}{1-\beta p_k} r_k(\beta) < 0.$$

3.5 Calcul d'une solution initiale strictement réalisable

Le problème de la faisabilité stricte de (SDP) consiste à trouver une matrice X vérifiant :

$$\left\{X \in \mathbb{S}^n_{++} : \langle A_i, X\rangle = b_i, \quad i = 1, ..., m\right\} \qquad (F)$$

En utilisant la technique de la variable artificielle, le problème (F) devient équivalent au problème suivant :

$$\min_{(X,\lambda)} \left[\lambda : \begin{array}{l} \langle A_i, X\rangle + \lambda(b_i - \langle A_i, F_0\rangle) = b_i, \quad i = 1, \ldots, m, \\ X \in \mathbb{S}^n_{++},\ \lambda \geq 0,\ F_0 \in \mathbb{S}^n_{++} \text{ (quelconque)} \end{array}\right] \qquad (AP)$$

Théorème 3.5.1 *[11] (X^*, λ^*) est une solution optimale de (AP) avec $\lambda^* \leq \varepsilon$ si et seulement si X^* est une solution de (F).*

Remarque 3.5.1 *Si $\lambda^* \nrightarrow 0$ alors (F) est vide.*

(AP) est un problème semi-défini qui s'écrit sous la forme suivante :

$$\min_{X'} [\ \langle C', X'\rangle\ :\ X' \in \mathbb{S}^n_{++},\ \langle A'_i, X'\rangle = b_i \text{ pour } i = 1, \ldots, m], \qquad (AP\_SDP)$$

où : la matrice $C' \in \mathbb{S}^{(n+1)\times(n+1)}$ définie comme suit :

$$C'[i,j] = \begin{cases} 1 & \text{si } i = j = n+1, \\ 0 & \text{sinon.} \end{cases}$$

et les matrices $A'_i \in \mathbb{S}^{(n+1)\times(n+1)}$ $(i = 1, ..., m)$ sont définies comme suit :

$$A'_i = \begin{pmatrix} A_i & 0 \\ 0 & b_i - \langle A_i, F_0\rangle \end{pmatrix}$$

Finalement, la matrice $X' \in \mathbb{R}^{(n+1)\times(n+1)}$ est définie comme suit :

$$X' = \begin{pmatrix} X & 0 \\ 0 & \lambda \end{pmatrix}$$

Remarque 3.5.2 $X'_0 = \begin{pmatrix} F_0 & 0 \\ 0 & 1 \end{pmatrix}$ *est une solution strictement réalisable pour (AP_SDP) pour toute matrice F_0 définie-positive, par exemple $F_0 = I_n$.*

3.6 Calcul du pas de déplacement

Les alternatives suivantes donnent un pas de déplacement qui assure la faisabilité stricte des itérés X_{k+1} du problème (SDP).

Lemme 3.6.1 *[68] Soit A une matrice de M_n et $\lambda = (\lambda_1, \lambda_2, ..., \lambda_n) \in \mathbb{R}^n$ ces valeurs propres. On définit :*

$$\bar{\lambda} = \frac{1}{n}\sum_{i=1}^{n}(A)_{ii} \quad et \quad \sigma^2 = \frac{1}{n}\sum_{i,j=1}^{n}(A)_{ij}^2 - \bar{\lambda}^2$$

alors on a :

$$\begin{aligned} \bar{\lambda} - \sigma\sqrt{n-1} &\leq \min_i \lambda_i \leq \bar{\lambda} - \frac{\sigma}{\sqrt{n-1}} \\ \bar{\lambda} + \frac{\sigma}{\sqrt{n-1}} &\leq \max_i \lambda_i \leq \bar{\lambda} + \sigma\sqrt{n-1} \end{aligned}$$

3.6.1 Première alternative

Lemme 3.6.2 *[11] On définit :*

$$\widetilde{\beta}_k = \left[\max\left\{p_k, \bar{\lambda} + \sigma\sqrt{n-1}\right\}\right]^{-1}$$

avec : $\bar{\lambda} = \frac{1}{n}\sum_{i=1}^{n}(P_k)_{ii}$,

et $\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n}(P_k)_{ij}^2 - \bar{\lambda}^2$.

Alors X_{k+1} est une solution strictement réalisable de (SDP) pour tout :

$$\beta_k = \varrho\widetilde{\beta}_k \quad \forall \varrho \in]0,1[$$

3.6.2 Deuxième alternative

Appliquons un pas de déplacement basé sur une condition suffisante pour qu'une matrice symétrique soit définie-positive.

Lemme 3.6.3 *On définit :*

$$\widetilde{\beta}_k = \left[\max\left\{p_k, \sum_{i\neq j=1}^{n} |(P_k)_{ij}| + (P_k)_{ii}\right\}\right]^{-1}$$

Alors X_{k+1} est une solution strictement réalisable de (SDP) pour tout :

$$\beta_k = \varrho\widetilde{\beta}_k \quad \forall \varrho \in]0,1[$$

Preuve. On cherche β telle que $\overline{X}(\beta) \succ 0$ et $\overline{x}(\beta) > 0$.

Rappellons qu'une matrice symétrique A est définie-positive si $A_{ii} > \sum_{i\neq j=1}^{n} |A_{ij}|$ $\forall i = 1, ..., n$.

Alors $\overline{X}(\beta) \succ 0$ et $\overline{x}(\beta) > 0$ si

$$\Leftrightarrow \begin{cases} 1 - \beta_k (P_k)_{ii} > \sum_{i\neq j=1}^{n} \beta_k |(P_k)_{ij}| & \forall i = 1, \ldots, n \\ 1 - \beta_k p_k > 0. \end{cases}$$

$$\Leftrightarrow \begin{cases} \dfrac{1}{\beta_k} > (P_k)_{ii} + \sum_{i\neq j=1}^{n} |(P_k)_{ij}| & \forall i = 1, \ldots, n \\ \dfrac{1}{\beta_k} > p_k \end{cases}$$

donc

$$\frac{1}{\beta_k} > \max \left\{ p_k, (p_k)_{ii} + \sum_{i\neq j=1}^{n} |(P_k)_{ij}| \right\}$$

si on pose

$$\widetilde{\beta}_k = \left[\max \left\{ p_k, \sum_{i\neq j=1}^{n} |(P_k)_{ij}| + (P_k)_{ii} \right\} \right]^{-1}$$

donc

$$\beta_k = \varrho \widetilde{\beta}_k \quad \forall \varrho \in]0, 1[$$

■

3.6.3 Troisième alternative

Lemme 3.6.4 *On définit :*

$$\widetilde{\beta}_k = \left[\max \left\{ p_k, \bar{\lambda} + \sigma \sqrt{n-1} + p \right\} \right]^{-1}$$

avec : $\bar{\lambda} = \frac{1}{n} \sum_{i=1}^{n} (P_k - p_k I)_{ii}$,

et $\sigma^2 = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{n} (P_k - p_k I)_{ij}^2 - \bar{\lambda}^2$.

Alors X_{k+1} est une solution strictement réalisable de (SDP) pour tout :

$$\beta_k = \varrho \widetilde{\beta}_k \quad \forall \varrho \in]0, 1[$$

Preuve. Il faut montrer que la matrice X_{k+1} est définie-positive. On a

$$\begin{aligned} X_{k+1} &= X_k - \frac{\beta_k}{1-\beta_k p_k}(L_k P_k L_k^T - p_k X_k), \\ &= L_k L_k^T - \frac{\beta_k}{1-\beta_k p_k}(L_k P_k L_k^T - p_k L_k L_k^T), \\ &= L_k(I - \frac{\beta_k}{1-\beta_k p_k}(P_k - p_k I))L_k^T, \end{aligned}$$

Soit $\lambda = (\lambda_1, ..., \lambda_n)$ les valeurs propres de la matrice $(P_k - p_k I)$.

Alors

$$X_{k+1} \succ 0 \iff I - \frac{\beta_k}{1-\beta_k p_k}(P_k - p_k I) \succ 0$$

$$\iff \begin{cases} I - \dfrac{\beta_k}{1-\beta_k p_k}(P_k - p_k I) \succ 0 \\ \text{avec} \\ 1 - \beta_k p_k > 0 \end{cases} \Leftrightarrow \begin{cases} 1 - \dfrac{\beta_k}{1-\beta_k p_k}\max_i \lambda_i > 0 \\ \text{avec} \\ \beta_k < \dfrac{1}{p_k} \end{cases}$$

ce qui donne d'après [68]

$$\Rightarrow \begin{cases} 1 - \dfrac{\beta_k}{1-\beta_k p_k}\bar{\lambda} - \dfrac{\beta_k}{1-\beta_k p_k}\sigma\sqrt{n-1} > 0 \\ \dfrac{1}{\beta_k} > p_k \end{cases} \Leftrightarrow \begin{cases} \dfrac{1}{\beta_k} > \bar{\lambda} + \sigma\sqrt{n-1} + p_k \\ \dfrac{1}{\beta_k} > p_k \end{cases}$$

donc

$$\frac{1}{\beta_k} > \max\left\{p_k, \bar{\lambda} + \sigma\sqrt{n-1} + p_k\right\}$$

posons

$$\widetilde{\beta}_k = \left(\max\left\{p_k, \bar{\lambda} + \sigma\sqrt{n-1} + p_k\right\}\right)^{-1}$$

donc

$$\beta_k = \varrho\widetilde{\beta}_k \quad \forall \varrho \in]0,1[$$

■

3.6.4 Quatrième alternative

Lemme 3.6.5 *On définit :*

$$\widetilde{\beta}_k = \left[\max\left\{\frac{v_k}{\tau}, \frac{\bar{\mu} + \sigma\sqrt{n-1} + v_k}{\tau}\right\}\right]^{-1}$$

avec : $\bar{\mu} = \frac{1}{n}\sum_{i=1}^{n}(CX_k + \sum_{i=1}^{m}\lambda_i A_i X_k - vI)_{ii}$,

et $\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n}(CX_k + \sum_{i=1}^{m}\lambda_i A_i X_k - vI)_{ij}^2 - \bar{\mu}^2$.

Alors X_{k+1} *est une solution strictement réalisable de (SDP) pour tout :*

$$\beta_k = \varrho\widetilde{\beta}_k \qquad \forall \varrho \in]0,1[$$

Preuve. Il faut montrer que la matrice X_{k+1} est définie-positive.

On a :

$$\begin{aligned} X_{k+1} &= X_k - \frac{\beta_k}{\tau - \beta_k v_k} L_k(V_k - v_k I)L_k^T, \\ &= X_k\left[I - \frac{\beta_k}{\tau - \beta_k v_k} X_k^{-1}(L_k(V_k - v_k I)L_k^T)\right], \\ &= X_k^{\frac{1}{2}}\left[I - \frac{\beta_k}{\tau - \beta_k v_k}(L_k^{-T}(V_k - v_k I)L_k^T)\right]X_k^{\frac{1}{2}}, \end{aligned}$$

Soit $\mu = (\mu_1, ..., \mu_n)$ les valeurs propres de la matrice $(L_k^{-T}(V_k - v_k I)L_k^T)$.

Alors

$$X_{k+1} \succ 0 \iff I - \frac{\beta_k}{\tau - \beta_k v_k}(L_k^{-T}(V_k - v_k I)L_k^T) \succ 0$$

$$\iff \begin{cases} I - \frac{\beta_k}{\tau - \beta_k v_k}(L_k^{-T}(V_k - v_k I)L_k^T) \succ 0 \\ \text{avec} \\ \tau - \beta_k v_k > 0 \end{cases} \Leftrightarrow \begin{cases} 1 - \frac{\beta_k}{\tau - \beta_k v_k}\max_i \mu_i > 0 \\ \text{avec} \\ \beta_k < \frac{\tau}{v_k} \end{cases}$$

ce qui donne d'après [68]

$$\begin{cases} 1 - \frac{\beta_k}{\tau - \beta_k v_k}\bar{\mu} - \frac{\beta_k}{\tau - \beta_k v_k}\sigma\sqrt{n-1} > 0 \\ \frac{1}{\beta_k} > \frac{v_k}{\tau} \end{cases} \Leftrightarrow \begin{cases} \frac{1}{\beta_k} > \frac{\bar{\mu} + \sigma\sqrt{n-1} + v_k}{\tau} \\ \frac{1}{\beta_k} > \frac{v_k}{\tau} \end{cases}$$

donc

$$\frac{1}{\beta_k} > \max\left\{\frac{v_k}{\tau}, \frac{\bar{\mu} + \sigma\sqrt{n-1} + v_k}{\tau}\right\}$$

posons

$$\widetilde{\beta}_k = \left(\max\left\{\frac{v_k}{\tau}, \frac{\bar{\mu} + \sigma\sqrt{n-1} + v_k}{\tau}\right\}\right)^{-1}$$

avec

$$\bar{\mu} = \frac{1}{n}\sum_{i=1}^{n}(L_k^{-T}(V_k - v_k I)L_k^T)_{ii}, = \frac{1}{n}\sum_{i=1}^{n}(CX_k + \sum_{i=1}^{m}\lambda_i A_i X_k - v_k I)_{ii}.$$

de même

$$\begin{aligned}\sigma^2 &= \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n}(L_k^{-T}(V_k - v_k I)L_k^T)_{ij}^2 - \bar{\mu}^2 \\ &= \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n}(CX_k + \sum_{i=1}^{m}\lambda_i A_i X_k - vI)_{ij}^2 - \bar{\mu}^2.\end{aligned}$$

donc

$$\beta_k = \varrho\widetilde{\beta}_k \qquad \forall \varrho \in]0,1[$$

■

3.7 Complexité de l'algorithme

Le pas de déplacement calculé par les lemmes ci-dessus assure la convergence de l'algorithme de la même manière que celle d'Alizadeh [2] donné dans le théorème ci-dessous.

Avant cela, rappelons la fonction potentielle primale-duale définie par Alizadeh [2] :

$$\psi(X,S) = (n+1)\ln(< X,S >) - \ln\det(XS),$$

définie pour tout $(X,S) \in \mathcal{F}_p^\circ \times \mathcal{F}_d^\circ$.

Théorème 3.7.1 *[2] Soient $(X_0, y_0, S_0) \in \mathcal{F}_p^\circ \times \mathcal{F}_d^\circ$, et supposon que $\psi(X_0,S_0) \leq O(\sqrt{n}E)$ pour une constante E. Si l'algorithme génère une séquence des points intérieurs primaux-duaux (X_j, y_j, S_j) tels que $\psi(X_j,S_j) \geq \psi(X_{j+1},S_{j+1}) + \delta$ pour un nombre fixe δ alors, l'algorithme converge après $k = O(\sqrt{n}|\log\varepsilon|)$ itérations et nous avons*

$$< C, X_k > - b^t y_k < 2^E \varepsilon.$$

3.8 Tests numériques

L'algorithme a été testé sur des problèmes de références issues de la bibliothèque de problèmes tests SDPLIB [13], moyennant les quatre alternatives du calcul du pas de déplacement β.

Nous avons utilisé le Langage Matlab avec une précision $\varepsilon = 10^{-6}$ et $\varrho = 0.9$.

On note par :

– iter : Le nombre d'itérations nécessaire pour chaque phase.
– CPU : Le temps d'exécution nécessaire pour chaque phase.
– alt i : l'alternative i du calcul du pas de déplacement, pour $i = 1, 2, 3$ et 4.

Notez que alt1 correspond le pas de déplacement du Lemme (3.1) qui est donné dans l'algorithme décrit dans [11]. Dans le tableau suivant, nous présentons les résultats comparatifs en nombre d'itérations et en temps de calcul entre notre algorithme utilisant les trois alternatives (alt i, i = 2,3,4) et l'algorithme de Benterki et al. présenté dans [11] en utilisant l'alternative 1 (alt1).

		Phase 1		Phase 2	
		iter	CPU	iter	CPU
*control*3 (136, 45)	alt1	31	26.4577	302	266.1783
	alt2	174	147.0561	2526	266.4696
	alt3	31	26.6781	302	270.9406
	alt4	52	0.8691	2139	140.4147
h inf 1 (13, 14)	alt1	22	0.0708	180	0.4128
	alt2	33	0.1304	820	1.8836
	alt3	22	0.0647	180	0.4180
	alt4	22	0.0986	2670	47.1297
h inf 14 (73, 34)	alt1	27	4.3479	292	43.0457
	alt2	74	12.0115	1023	149.9512
	alt3	27	4.4036	292	43.2073
	alt4	27	4.3957	5082	681.6062
h inf 15 (91, 37)	alt1	28	7.4879	258	71.3782
	alt2	70	18.5666	834	230.1724
	alt3	28	7.5175	305	55.4506
	alt4	28	7.6222	875	244.6610
*qap*5 (136, 26)	alt1	32	12.7076	29	10.6264
	alt2	202	19.5387	86	32.3189
	alt3	30	10.3752	28	11.6038
	alt4	32	12.3809	28	27.6333
*truss*1 (6, 13)	alt1	40	0.0414	243	0.1700
	alt2	94	0.1045	458	0.3224
	alt3	40	0.0376	242	0.1692
	alt4	821	0.6033	2298	1.6402
*truss*3 (27, 31)	alt1	32	0.6415	255	37.6889
	alt2	70	1.3480	127	2.5499
	alt3	32	0.6878	250	2.4235
	alt4	142	2.6931	434	12.6619

3.9 Conclusion

Dans cette partie, nous avons introduit un algorithme primal-dual de points

intérieurs pour résoudre un problème d'optimisation semi-défini en utilisant quatre alternatives pour calculer le pas de déplacement. L'algorithme obtenu confirme le but théorique et donne effectivement une solution strictement réalisable de (SDP) et une solution primale-duale de (SDP) et $(DSDP)$. De plus, les tests numériques montrent que l'alternative 1 correspondant à l'algorithme donné dans [11] et l'alternative 3 sont plus efficaces que les alternatives 2 et 4 en termes de nombre d'itérations et de temps de calcul.

Chapitre 4

Méthode primale-duale de points intérieurs via une nouvelle fonction noyau logarithmique pour SDP

4.1 Introduction

Les méthodes primales-duales de points intérieurs MPI pour la programmation semi-définie SDP ont été largement étudiées, le lecteur est renvoyé à Klerk [19] pour plus d'éclaircissement. Récemment, une méthode de points intérieurs non réalisa ble de programmation linéaire PL a été présentée par Roos [55].

Certaines extensions des méthodes de points intérieurs ont été réalisées par Mansouri et Roos [42]. Mansouri et Roos [41] ont étendu cet algorithme à la programmation semi-définie en utilisant une étape de faisabilité spécifique. La fonction barrière est déterminée par une fonction simple unidimensionnelle, appelée fonction noyau. Bai et al. [5] introduit une nouvelle fonction barrière qui n'est pas une fonction barrière dans le sens habituel du terme.

Dans ce chapitre, nous proposons un algorithme primal-dual de points intérieurs pour l'optimisation semi-définie basé sur deux nouvelles fonctions noyaux avec un terme barrière logarithmique efficace. Nous montrons que le meilleur résultat de complexité peut être atteint, à savoir $O(\sqrt{n}\log n\log\frac{n}{\varepsilon})$, pour les méthodes à grand pas et $O(\sqrt{n}\log\frac{n}{\varepsilon})$ pour les méthodes à petit pas, ce qui améliore

de manière significative les résultats de complexité obtenus jusqu'à présent, basés sur une fonction noyau logarithmique. Les tests numériques effectués montrent l'efficacité de ce nouvel algorithme.

4.2 Déscription de l'algorithme

4.2.1 Méthode de la trajectoire centrale

Nous considérons le problème d'optimisation semi-défini SDP, dont la forme primale est donnée par :

$$\begin{cases} \min C \bullet X \\ A_i \bullet X = b_i, \quad i = 1, ..., m \\ X \succeq 0 \end{cases} \tag{SDP}$$

et son dual :

$$\begin{cases} \max b^T y \\ S = C - \sum_{i=1}^{m} y_i A_i, \\ S \succeq 0. \end{cases} \tag{DSDP}$$

où : $b = (b_1, b_2, ..., b_m)$, $y \in \mathbb{R}^m$; C et A_i $(\forall i = 1, ..., m)$ sont des matrices dans $\mathbb{S}^n$.

Tout au long de ce chapitre, nous formulons les hypothèses suivantes :

1. Les matrices A_i, $i = 1, ..., m$ sont linéairement indépendantes.
2. Les problèmes (SDP) et $(DSDP)$ satisfont la condition de point intérieur (CPI), i.e.,
$$\exists\ X \in \mathcal{F}_P^{\circ}, (y, S) \in \mathcal{F}_D^{\circ}$$

Il est bien connu que la recherche d'une solution optimale (X^*, y^*, S^*) de (SDP) et $(DSDP)$ est équivalent à résoudre le système suivant :

$$\begin{cases} A_i \bullet X = b_i, \quad i = 1, ..., m,\ X \succeq 0, \\ \sum_{i=1}^{m} y_i A_i + S = C,\ S \succeq 0, \\ XS = 0. \end{cases} \tag{4.1}$$

L'idée de base des méthodes de points intérieurs (MPIs) de type primale-duale consiste à remplacer la troisième équation (la condition de complémentarité)

dans le système (4.1) par l'équation paramétrisée $XS = \mu I$ avec $\mu > 0$, donc on considère le système suivant :

$$\begin{cases} A_i \bullet X = b_i, \quad i = 1, ..., m, \ X \succ 0, \\ \sum_{i=1}^{m} y_i A_i + S = C, \ S \succ 0, \\ XS = \mu I. \end{cases} \tag{4.2}$$

Si les problèmes (SDP) et $(DSDP)$ satisfaient les conditions de point intérieur (CPI), alors pour chaque $\mu > 0$ le système (4.2) admet une solution unique $(X(\mu), y(\mu), S(\mu))$ (voir [35, 49, 58, 62]), que l'on appelle $\mu - centre$. L'ensemble de $\mu - centre$, construit la trajectoire centrale de (SDP) et de $(DSDP)$ qui converge vers la solution optimale des problèmes (SDP) et $(DSDP)$ quand μ tend vers 0 [67]. En général, les méthodes de points intérieurs (MPIs) pour SDP se composent de deux stratégies : La première stratégie consiste à déterminer une solution paramétrisée strictement réalisable et vérifie certaines conditions (condition de proximité) et la seconde consiste à diminuer le paramètre μ à $\mu_+ = (1 - \theta)\mu$, $0 < \theta < 1$ pour déterminer la qualité de la solution.

4.2.2 Direction de descente

Appliquons la méthode de Newton au système (4.2), nous obtenons le système de Newton comme suit :

$$\begin{cases} A_i \bullet \Delta X = 0, \quad i = 1, ..., m, \\ \sum_{i=1}^{m} \Delta y_i A_i + \Delta S = 0, \\ X\Delta S + \Delta X S = \mu I - XS. \end{cases} \tag{4.3}$$

Puisque A_i sont linéairement indépendants et $X \succ 0, S \succ 0$, le système (4.3) a une direction unique $(\Delta X, \Delta y, \Delta S)$. A partir de la deuxième équation de (4.3), on note que ΔS est symétrique, mais que ΔX peut ne pas être symétrique. Différentes méthodes de symétrisation de la troisième équation de (4.3) sont proposées afin que le nouveau système symétrisé ait une solution symétrique unique. Dans ce chapitre, nous utilisons le schéma de symétrisation NT [50]. On définit la matrice :

$$P = X^{1/2}(X^{1/2}SX^{1/2})^{-1/2}X^{1/2} = S^{-1/2}(S^{1/2}XS^{1/2})^{1/2}S^{-1/2}$$

et

$$D = P^{1/2}, \quad V = \frac{1}{\sqrt{\mu}} D^{-1} X D^{-1} = \frac{1}{\sqrt{\mu}} DSD = \frac{1}{\sqrt{\mu}} (D^{-1} X S D)^{1/2}. \tag{4.4}$$

On remarque que les matrices D et V sont semi-définies positives et symétriques. Utilisant (4.4) le système de Newton (4.3) devient :

$$\begin{cases} \overline{A}_i \bullet D_X = 0, \quad i = 1, ..., m, \\ \sum_{i=1}^{m} \Delta y_i \overline{A}_i + D_S = 0, \\ D_X + D_S = V^{-1} - V. \end{cases} \tag{4.5}$$

avec

$$\overline{A}_i = \frac{1}{\sqrt{\mu}} D A_i D, \ i = \overline{1, m}, \ D_X = \frac{1}{\sqrt{\mu}} D^{-1} \Delta X D^{-1}, \ D_S = \frac{1}{\sqrt{\mu}} D \Delta S D. \tag{4.6}$$

Le système (4.5) détermine une direction NT symétrique unique dont les matrices D_X et D_S sont orthogonales et il est évident que

$$Tr(D_X D_S) = Tr(D_S D_X) = 0.$$

4.2.3 Algorithme générique de points intérieurs pour SDP

Pour $V = Q^T diag(\lambda_1(V), \lambda_2(V), ..., \lambda_n(V)) Q$, la décomposition spectrale de $V \in \mathbb{S}^n_{++}$, on généralise une fonction $\psi(t) : \mathbb{R}_{++} \to \mathbb{R}_+$ à la fonction matricielle $\psi(V) : \mathbb{S}^n_{++} \to \mathbb{S}^n$ comme suit :

$$\psi(V) = Q^T diag(\psi(\lambda_1(V)), \psi(\lambda_2(V)), ..., \psi(\lambda_n(V))) Q, \tag{4.7}$$

$$\psi'(V) = Q^T diag(\psi'(\lambda_1(V)), \psi'(\lambda_2(V)), ..., \psi'(\lambda_n(V))) Q. \tag{4.8}$$

Remplaçant le côté droit $V^{-1} - V$ de la troisième équation de (4.5) par $-\psi'(V)$, on obtient le système linéaire :

$$\begin{cases} \overline{A}_i \bullet D_X = 0, \quad i = 1, ..., m, \\ \sum_{i=1}^{m} \Delta y_i \overline{A}_i + D_S = 0, \\ D_X + D_S = -\psi'(V). \end{cases} \tag{4.9}$$

où $\psi(t)$ est une fonction noyau donnée, et $\psi(V), \psi'(V)$ sont les fonctions matricielles associées définies en (4.7), le système (4.9) a une solution symétrique unique.

Pour toute fonction noyau $\psi(t)$, on définie $\Psi(V) : \mathbb{S}^n_{++} \to \mathbb{R}_+$ par

$$\Psi(V) = Tr(\psi(V)) = \sum_{i=1}^{n} \psi(\lambda_i(V)). \tag{4.10}$$

Alors $\Psi(V)$ est strictement convexe avec $V \succ 0$ et atteint son minimum global $V = I$ et $\Psi(I) = 0$. Puisque D_X et D_S sont orthogonaux, pour $\mu > 0$,

$$\Psi(V) = 0 \Leftrightarrow V = I \Leftrightarrow D_X = D_S = 0 \Leftrightarrow X = X(\mu), S = S(\mu).$$

Par conséquent, on peut utiliser $\Psi(V)$ comme fonction de proximité pour mesurer la distance entre l'itération actuelle et le $\mu - centre$ correspondant.

L'algorithme proposé dans ce chapitre se déroule comme suit : supposons que l'on donne un point strictement réalisable (X, y, S) qui est dans un $\tau-$voisinage de $\mu - centre$ et $\tau \geq 1$[19]. Ensuite, nous diminuons μ à $\mu^+ = (1-\theta)\mu$, pour $\theta \in (0, 1)$, puis on résout le système (4.9) et (4.6) pour obtenir la direction de descente. La condition de positivité d'une nouvelle itération est assurée avec le bon choix du pas qui est défini dans la section suivante. Cette procédure est répétée jusqu'à ce qu'on trouve une nouvelle itération (X_+, y_+, S_+) qui est dans un $\tau-$voisinage de $\mu^+ - centre$ et puis on pose $\mu = \mu^+$ et $(X, y, S) = (X_+, y_+, S_+)$. Ce processus est répété jusqu'à ce que μ soit suffisamment petit, i.e., $n\mu < \varepsilon$.

Les paramètres τ, θ et le pas de déplacement α doivent être choisi de telle sorte que l'algorithme est optimisé dans le sens où le nombre d'itérations requises soit le plus petit possible. Le choix de paramètre θ joue un rôle important dans la théorie et la pratique des MPIs. Si θ est une constante indépendante de la dimension n du problème, par exemple $\theta = \frac{1}{2}$, nous appelons l'algorithme de grand pas. Si θ dépend de la dimension du problème, telles que $\theta = \frac{1}{\sqrt{n}}$, alors l'algorithme est appelé algorithme de petit pas.

L'algorithme générique de points intérieurs primal-dual pour SDP est donné comme suit :

Primal-Dual MPI pour SDP

Initialisation

un paramètre de seuil $\tau \geq 1$;

un paramètre de précision $\varepsilon > 0$;

un paramètre θ, $0 < \theta < 1$;

une solution initiale strictement réalisable (X^0, S^0) et $\mu^0 = 1$

tels que $\Psi(X^0, S^0, \mu^0) \leq \tau$;

Début

$X = X^0; S = S^0; \mu = \mu^0;$

Tant que $n\mu > \varepsilon$ **faire**

Début

$\mu = (1 - \theta)\mu;$

Tant que $\Psi(X, S, \mu) > \tau$ **faire**

Début

Résoudre le système (4.9) puis (4.6) pour obtenir (ΔX, Δy, ΔS);

Déterminer le pas de déplacement α et prendre

$X = X + \alpha\Delta X;$

$y = y + \alpha\Delta y;$

$S = S + \alpha\Delta S;$

Fin tant que.

Fin tant que.

Fin.

4.3 Première fonction noyau et ses propriétés

Dans cette partie, nous présentons une nouvelle fonction noyau paramétrée et nous donnons ses propriétés qui sont essentielles à notre analyse de la complexité.

Définition 4.3.1 *On appelle $\psi : \mathbb{R}_{++} \to \mathbb{R}_+$ une fonction noyau, si elle est deux fois dérivable et satisfait les conditions suivantes :*

$$\psi'(1) = \psi(1) = 0,\ \psi''(t) > 0,\ t > 0, \quad \lim_{t \to 0^+} \psi(t) = \lim_{t \to \infty} \psi(t) = \infty. \tag{4.11}$$

À partir des deux premières conditions, ψ est strictement convexe et minimale à $t = 1$, et ψ est exprimé en fonction de sa dérivée seconde comme suit :

$$\psi(t) = \int_1^t \int_1^\xi \psi''(\zeta) d\zeta d\xi. \tag{4.12}$$

Cependant, la troisième condition indique la propriété barrière de ψ.

Maintenant, on considère notre nouvelle fonction noyau ψ_1 comme suit :

$$\psi_1(t) = \frac{p(t^2-1)}{2} + \frac{p(t^{-pq+1}-1)}{(pq-1)(q+1)} - \frac{pq}{(q+1)}\log t,\ p \geq 1, q > 1, t > 0. \quad (4.13)$$

Il est facile de vérifier que ψ_1 est une fonction noyau barrière et ses trois dérivées successives sont :

$$\begin{aligned} \psi_1'(t) &= pt - \frac{p}{q+1}t^{-pq} - \frac{pq}{(q+1)}t^{-1}, \quad (4.14)\\ \psi_1''(t) &= p + \frac{p(pq)}{q+1}t^{-pq-1} + \frac{pq}{(q+1)}t^{-2},\\ \psi_1'''(t) &= -\frac{p(pq)(pq+1)}{q+1}t^{-pq-2} - \frac{2pq}{(q+1)}t^{-3}. \end{aligned}$$

d'après (4.14), on a

$$\psi_1''(t) > p, \quad t > 0. \quad (4.15)$$

Lemme 4.3.1 *Soit ψ_1 définie dans (4.13), alors*
(i) $t\psi_1''(t) + \psi_1'(t) > 0, \quad 0 < t < 1,$
(ii) $t\psi_1''(t) - \psi_1'(t) > 0, \quad t > 1,$
(iii) $\psi_1'''(t) < 0, \ \ t > 0.$
(iv) $\psi_1''(t)\psi_1'(\beta t) - \beta\psi_1'(t)\psi_1''(\beta t) > 0, \quad t > 0, \quad \beta > 1.$

Preuve. Pour (i), utilisant (4.14), on a

$$t\psi_1''(t) + \psi_1'(t) = 2pt + \frac{p(pq-1)t^{-pq}}{q+1} > 0,$$

pour tout $p \geq 1$, $q > 1$ et $t > 0$.

Pour (ii), on a

$$t\psi_1''(t) - \psi_1'(t) = \frac{p(pq+1)t^{-pq}}{q+1} + \frac{2pq}{(q+1)t} > 0,$$

pour tout $p \geq 1$, $q > 1$ et $t > 0$.

Pour (iii), il est clair de (4.14) que

$$\psi_1'''(t) < 0, \text{ pour } t > 0$$

Pour (iv), la preuve est similare à celui du lemme 2.4 dans [6]. ■

Lemme 4.3.2 *Pour ψ_1, on a pour $p \geq 1$ et $q > 1$,*
(i) $\frac{p}{2}(t-1)^2 \leq \psi_1(t) \leq \frac{1}{2p}[\psi_1'(t)]^2, \ \ t > 0,$
(ii) $\psi_1(t) \leq \frac{1}{2}\psi_1''(1)(t-1)^2, \ \ t \geq 1.$

Preuve. Pour (i), utilisant la première condition de (4.11) et (4.15), on a

$$\psi_1(t) = \int_1^t \int_1^\xi \psi_1''(\zeta)d\zeta d\xi \geq \int_1^t \int_1^\xi p d\zeta d\xi = \frac{p}{2}(t-1)^2,$$

la deuxième inégalité est obtenue comme suit :

$$\begin{aligned} \psi_1(t) = \int_1^t \int_1^\xi \psi_1''(\zeta)d\zeta d\xi &\leq \frac{1}{p}\int_1^t \int_1^\xi \psi_1''(\xi)\psi_1''(\zeta)d\zeta d\xi \\ &= \frac{1}{p}\int_1^t \psi_1''(\xi)\psi_1'(\xi)d\xi \\ &= \frac{1}{p}\int_1^t \psi_1'(\xi)d\psi_1'(\xi)d\xi = \frac{1}{2p}[\psi_1'(t)]^2. \end{aligned}$$

Pour (ii), utilisant le théorème de Taylor, la première condition de (4.11) et Lemme 4.3.1 (iii), on a

$$\begin{aligned} \psi_1(t) &= \psi_1(1) + \psi_1'(1)(t-1) + \tfrac{1}{2}\psi_1''(1)(t-1)^2 + \tfrac{1}{3!}\psi_1'''(c)(t-1)^3 \\ &= \tfrac{1}{2}\psi_1''(1)(t-1)^2 + \tfrac{1}{3!}\psi_1'''(c)(t-1)^3 \\ &< \tfrac{1}{2}\psi_1''(1)(t-1)^2, \end{aligned}$$

tel quet $1 \leq c \leq t$. Ceci complète la preuve. ■

Lemme 4.3.3 *Soit $\varrho : [0, \infty) \to [1, \infty)$ la fonction inverse de ψ_1 pour $t \geq 1$. Alors on a*

$$1 + \sqrt{\frac{2s}{\psi_1''(1)}} \leq \varrho(s) \leq 1 + \sqrt{\frac{2s}{p}}, \quad p \geq 1, \ q > 1, \ s \geq 0.$$

Preuve. Soit $s = \psi_1(t)$ pour $t \geq 1$, i.e., $\varrho(s) = t$. Par la définition de ψ_1,

$$s = \frac{p(t^2-1)}{2} + \frac{t^{-pq+1}-1}{q(q+1)} - \frac{pq}{(q+1)}\log t, \ p \geq 1, \ q > 1, t > 0.$$

Utilisant Lemme 4.3.2 (i), on a

$$s = \psi_1(t) \geq \frac{p}{2}(t-1)^2$$

implique que

$$t = \varrho(s) \leq 1 + \sqrt{\frac{2s}{p}}.$$

Pour la deuxième inégalité utilisant Lemme 4.3.2 (ii), alors

$$s = \psi_1(t) \leq \frac{1}{2}\psi_1''(1)(t-1)^2,\ t \geq 1.$$

alors

$$t = \varrho(s) \geq 1 + \sqrt{\frac{2s}{\psi_1''(1)}}.$$

Ceci complète la preuve. ■

Lemme 4.3.4 *Soit* $\rho : [0, \infty) \to (0, 1]$ *la fonction inverse de* $-\frac{1}{2}\psi_1$ *pour* $0 < t \leq 1$. *Alors on a*

$$\rho(z) \geq \left(\frac{p}{2(q+1)z + p}\right)^{\frac{1}{pq}}, \quad p \geq 1,\ q > 1,\ z \geq 0.$$

Preuve. Soit $z = -\frac{1}{2}\psi_1(t)$ pour $0 < t \leq 1$. A partir du définition de ρ, $\rho(z) = t$, pour $z \geq 0$. Alors on a

$$z = \frac{1}{2}(\frac{p}{q+1}t^{-pq} - pt + \frac{pq}{(q+1)}t^{-1}) \geq \frac{1}{2}(\frac{p}{q+1}t^{-pq} - p + \frac{pq}{q+1}),$$

d'où

$$t^{-pq} \leq \frac{2(q+1)z + p}{p}.$$

par conséquent, nous obtenons

$$t = \rho(z) \geq \left(\frac{p}{2(q+1)z + p}\right)^{\frac{1}{pq}}.$$

Ceci complète la preuve. ■

Pour l'analyse de l'algorithme, nous utilisons également la mesure de proximité basée sur la norme $\delta(V)$ comme suit :

$$\delta(V) = \frac{1}{2}||\psi'(V)|| = \frac{1}{2}\sqrt{\sum_{i=1}^{n}(\psi'(\lambda_i(V)))^2} = \frac{1}{2}||D_X + D_S||, \quad V \in \mathbb{S}^n_{++}. \quad (4.16)$$

Dans le lemme suivant, on donne une relation entre deux mesures de proximité.

Lemme 4.3.5 *Soit $\delta(V)$ et $\Psi_1(V)$ définient dans (4.16) et (4.10), respectivement. Alors, on a*

$$\delta(V) \geq \sqrt{\frac{p}{2}\Psi_1(V)}, \quad V \in \mathbb{S}^n_{++}.$$

Preuve. Utilisant (4.16) et la deuxième inégalité du Lemme 4.3.2 (i),

$$\delta^2(V) = \frac{1}{4}\sum_{i=1}^{n}(\psi_1'(\lambda_i(V)))^2 \geq \frac{p}{2}\sum_{i=1}^{n}\psi_1(\lambda_i(V)) = \frac{p}{2}\Psi_1(V).$$

Par conséquent, on a $\delta(V) \geq \sqrt{\frac{p}{2}\Psi_1(V)}$. ■

Remarque 4.3.1 *Tout au long de ce chapitre, nous supposons que $\tau \geq 1$. En utilisant le lemme 4.3.5 et en supposant que $\Psi_1(V) \geq \tau$, nous avons $\delta(V) \geq \frac{1}{\sqrt{2}}$.*

Dans ce qui suit, utilisant la remarque 4.3.1, nous estimons l'effet d'un $\mu-$update (mise à jour) sur la valeur de $\Psi(V)$.

Lemme 4.3.6 *(Lemma 4.16 dans [64]) Soit ϱ définie dans Lemme 4.3.3. Alors, on a*

$$\Psi(\beta V) \leq n\psi\left(\beta\varrho\left(\frac{\Psi(V)}{n}\right)\right), \quad V \in \mathbb{S}^n_{++}, \quad \beta \geq 1.$$

Lemme 4.3.7 *Soit $0 \leq \theta < 1$ et $V_+ = \frac{V}{\sqrt{1-\theta}}$. Si $\Psi_1(V) \leq \tau$, alors pour $q > 1$ on a*

(i) $\Psi_1(V_+) \leq \frac{p(2pq+q+1)}{2(q+1)(1-\theta)}\left(\sqrt{n\theta} + \sqrt{\frac{2\tau}{p}}\right)^2$,

(ii) $\Psi_1(V_+) \leq \dfrac{np\theta + 2\tau + 2\sqrt{2\tau np}}{2(1-\theta)}$.

Preuve. Pour (i), comme $\frac{1}{\sqrt{1-\theta}} \geq 1$ et $\varrho\left(\frac{\Psi_1(V)}{n}\right) \geq 1$, on a $\frac{\varrho\left(\frac{\Psi_1(V)}{n}\right)}{\sqrt{1-\theta}} \geq 1$. Utilisant Lemme 4.3.6 avec $\beta = \frac{1}{\sqrt{1-\theta}}$, Lemme 4.3.2 (ii), Lemme 4.3.3 et $\Psi_1(V) \leq \tau$, on a

$$\begin{aligned}
\Psi_1(V_+) &\leq n\psi_1\left(\tfrac{1}{\sqrt{1-\theta}}\varrho\left(\tfrac{\Psi_1(V)}{n}\right)\right) \\
&\leq \tfrac{np}{2}\left(\tfrac{2pq+q+1}{q+1}\right)\left(\frac{\varrho\left(\frac{\Psi_1(V)}{n}\right)}{\sqrt{1-\theta}} - 1\right)^2 \\
&\leq \tfrac{np(2pq+q+1)}{2(q+1)}\left(\frac{1+\sqrt{\frac{2\tau}{np}}-\sqrt{1-\theta}}{\sqrt{1-\theta}}\right)^2 \\
&\leq \tfrac{np(2pq+q+1)}{2(q+1)}\left(\frac{\theta+\sqrt{\frac{2\tau}{np}}}{\sqrt{1-\theta}}\right)^2 \\
&= \tfrac{p(2pq+q+1)}{2(q+1)(1-\theta)}\left(\sqrt{n\theta} + \sqrt{\tfrac{2\tau}{p}}\right)^2,
\end{aligned}$$

où la dernière inégalité est vérifiée de $1-\sqrt{1-\theta}=\frac{\theta}{1+\sqrt{1-\theta}}<\theta,\ \ 0\leq\theta<1.$

Pour (ii), pour $t\geq 1$, on a

$$\psi_1(t)\leq\frac{p(t^2-1)}{2}.$$

Utilisant Lemme 4.3.6 avec $\beta=\frac{1}{\sqrt{1-\theta}}$, Lemme 4.3.3 et $\Psi_1(V)\leq\tau$, on a

$$\begin{aligned}
\Psi_1(V_+) &\leq n\psi_1\left(\tfrac{1}{\sqrt{1-\theta}}\varrho\left(\tfrac{\Psi_1(V)}{n}\right)\right)\\
&\leq \tfrac{np}{2}\left(\left[\tfrac{1}{\sqrt{1-\theta}}\varrho\left(\tfrac{\Psi_1(V)}{n}\right)\right]^2-1\right)\\
&= \tfrac{np}{2(1-\theta)}\left(\varrho\left(\tfrac{\Psi_1(V)}{n}\right)^2-(1-\theta)\right)\\
&\leq \tfrac{np}{2(1-\theta)}\left(\left[1+\sqrt{\tfrac{2\Psi_1(V)}{np}}\right]^2-(1-\theta)\right)\\
&\leq \tfrac{np}{2(1-\theta)}\left(\theta+2\tfrac{\tau}{np}+2\sqrt{\tfrac{2\tau}{np}}\right)\\
&= \tfrac{np\theta+2\tau+2\sqrt{2\tau np}}{2(1-\theta)},
\end{aligned}$$

Ceci complète la preuve. ■

Définissant pour $p\geq 1$, $q>1$ et $0\leq\theta<1$,

$$(\widetilde{\Psi}_1)_0=\frac{p(2pq+q+1)}{2(q+1)(1-\theta)}\left(\sqrt{n}\theta+\sqrt{\frac{2\tau}{p}}\right)^2,\quad(\overline{\Psi}_1)_0=\frac{np\theta+2\tau+2\sqrt{2\tau np}}{2(1-\theta)}. \tag{4.17}$$

On utilise $(\widetilde{\Psi}_1)_0$ et $(\overline{\Psi}_1)_0$ comme des bornes supérieures de $\Psi_1(V)$ pour les méthodes à petit pas et à grand pas, respectivement.

Remarque 4.3.2 *Pour la méthode à petit pas, on prend $\tau=O(1)$ et $\theta=\Theta(\frac{1}{\sqrt{n}})$, et pour la méthode à grand pas, on prend $\tau=O(n)$ et $\theta=\Theta(1)$.*

4.3.1 Analyse de complexité

Dans cette section, nous calculons un pas de déplacement α efficace, la réduction de la fonction de proximité au cours d'une itération interne et améliorons les résultats de la complexité de l'algorithme, pour $\mu>0$.

Si nous prenons un pas α dans la direction $(\Delta X,\Delta y,\Delta S)$, nous obtenons une nouvelle itération (X_+,y_+,S_+), où

$$X_+=X+\alpha\Delta X,\quad y_+=y+\alpha\Delta y,\quad S_+=S+\alpha\Delta S,\quad\alpha>0.$$

Utilisant (4.6), nous pouvons réécrire X_+ et S_+ comme suit :

$$X_+ = \sqrt{\mu}D(V + \alpha D_X)D, \quad S_+ = \sqrt{\mu}D^{-1}(V + \alpha D_S)D^{-1}. \tag{4.18}$$

D'après (4.5), on a $V_+ = \frac{1}{\sqrt{\mu}}(D^{-1}X_+S_+D)^{1/2}$ et d'après (4.18), on a

$$V_+^2 = (V + \alpha D_X)(V + \alpha D_S).$$

Comme $V + \alpha D_X \in \mathbb{S}^n_{++}$, $V + \alpha D_S \in \mathbb{S}^n_{++}$, V_+^2 est similare à la matrice

$$(V + \alpha D_X)^{\frac{1}{2}}(V + \alpha D_S)(V + \alpha D_X)^{\frac{1}{2}}.$$

Cela implique que les valeurs propres de V_+ sont les mêmes que celles de

$$\left((V + \alpha D_X)^{\frac{1}{2}}(V + \alpha D_S)(V + \alpha D_X)^{\frac{1}{2}}\right)^{\frac{1}{2}}.$$

Alors on a

$$\Psi(V_+) = \Psi\left(\left((V + \alpha D_X)^{\frac{1}{2}}(V + \alpha D_S)(V + \alpha D_X)^{\frac{1}{2}}\right)^{\frac{1}{2}}\right).$$

Lemme 4.3.8 *(Proposition 5.2.6 dans [53]) soient $V_1, V_2 \in \mathbb{S}^n_{++}$. Alors on a*

$$\Psi\left(\left[V_1^{\frac{1}{2}}V_2V_1^{\frac{1}{2}}\right]^{\frac{1}{2}}\right) \leq \frac{1}{2}(\Psi(V_1) + \Psi(V_2)).$$

À partir du lemme 4.3.8, on obtient

$$\Psi(V_+) \leq \frac{1}{2}(\Psi(V + \alpha D_X) + \Psi(V + \alpha D_S)). \tag{4.19}$$

4.3.2 Calcul du pas de déplacement et diminution de la fonction de proximité

On définie pour $\alpha > 0$,

$$\begin{aligned} f(\alpha) &= \Psi(V_+) - \Psi(V), \\ f_1(\alpha) &= \frac{1}{2}(\Psi(V + \alpha D_X) + \Psi(V + \alpha D_S)) - \Psi(V). \end{aligned}$$

D'après (4.19), $f(\alpha) \leq f_1(\alpha)$ et $f(0) = f_1(0) = 0$.

Maintenant, pour estimer la diminution de la proximité au cours d'une étape, nous avons besoin des deux dérivées successives de $f_1(\alpha)$ par rapport à α. On a :

$$\begin{aligned} f_1'(\alpha) &= \frac{1}{2}Tr(\psi'(V+\alpha D_X)D_X + \psi'(V+\alpha D_S)D_S), \\ f_1''(\alpha) &= \frac{1}{2}Tr(\psi''(V+\alpha D_X)D_X^2 + \psi''(V+\alpha D_S)D_S^2). \end{aligned}$$

Il est évident que $f_1''(\alpha) > 0$, à moins que $D_X = D_S = 0$.

À partir de la troisième équation du système (4.9) et (4.16), nous avons

$$f_1'(0) = \frac{1}{2}Tr(\psi'(V)(D_X + D_S)) = \frac{1}{2}Tr(-(\psi'(V))^2) = -2\delta^2(V).$$

Pour la simplicité des notations, soit $\delta = \delta(V)$ et $\Psi = \Psi(V)$.

Pour trouver le pas de déplacement, nous avons besoin des lemmes suivants.

Lemme 4.3.9 *(Lemma 5.19 dans [64]) Soit δ défini dans* (4.16)*. Alors on a*

$$f_1''(\alpha) \leq 2\delta^2\psi''(\lambda_n(V) - 2\alpha\delta).$$

Lemme 4.3.10 *(Lemma 4.2 dans [6]) Si le pas α satisfait*

$$-\psi'(\lambda_n(V) - 2\alpha\delta) + \psi'(\lambda_n(V)) \leq 2\delta,$$

alors

$$f_1'(\alpha) \leq 0.$$

Lemme 4.3.11 *(Lemma 4.4 dans [6]) Soit ρ et $\overline{\alpha}$ définient dans Lemme 4.3.4 et Lemme 4.3.10 ,respectivement. Alors*

$$\overline{\alpha} \geq \frac{1}{\psi''(\rho(2\delta))}.$$

Lemme 4.3.12 *Soit ρ et $\overline{\alpha}$ définient dans Lemme 4.3.11. Si $1 \leq \tau \leq \Psi_1(V)$, alors on a*

$$\overline{\alpha} \geq \frac{1}{p + \left(\frac{pq(p+1)}{q+1}\right)\left(\frac{4(q+1)\delta}{p} + 1\right)^{\frac{pq+1}{pq}}}.$$

Preuve. Utilisant Lemme 4.3.11, Lemme 4.3.4 et (4.14) on a

$$\begin{aligned} \overline{\alpha} &\geq \frac{1}{\psi_1''(\rho(2\delta))} \\ &\geq \frac{1}{\psi_1''((\frac{p}{4(q+1)\delta+p})^{\frac{1}{pq}})} \\ &= \frac{1}{p+\frac{p(pq)}{(q+1)}(\frac{p}{4(q+1)\delta+p})^{\frac{-pq-1}{pq}} + \frac{pq}{q+1}(\frac{p}{4(q+1)\delta+p})^{\frac{-2}{pq}}} \\ &\geq \frac{1}{p+\left(\frac{pq(p+1)}{q+1}\right)\left(\frac{4(q+1)\delta}{p}+1\right)^{\frac{pq+1}{pq}}}. \end{aligned}$$

Cela complète la preuve. ■

Définissons le pas par défaut $\widetilde{\alpha}$ comme suit :

$$\widetilde{\alpha} = \frac{1}{p + \left(\frac{pq(p+1)}{q+1}\right)\left(\frac{4(q+1)\delta}{p} + 1\right)^{\frac{pq+1}{pq}}}, \tag{4.20}$$

avec $\widetilde{\alpha} \leq \overline{\alpha}$.

Lemme 4.3.13 *(Lemma 4.5 dans [6]) Si le pas α est tel que $\alpha \leq \overline{\alpha}$, alors*

$$f(\alpha) \leq -\alpha\delta^2.$$

Lemme 4.3.14 *Soit $\widetilde{\alpha}$ définie dans* (4.20). *Alors on a*

$$f(\widetilde{\alpha}) \leq -\frac{1}{26\sqrt{2}pq(p+1)(q+1)^{\frac{1}{pq}}}\Psi_1^{\frac{pq-1}{2pq}}.$$

Preuve. Utilisant Lemme 4.3.13 avec $\alpha = \widetilde{\alpha}$ et (4.20), on a

$$\begin{aligned}
f(\widetilde{\alpha}) &\leq -\widetilde{\alpha}\delta^2 \\
&= -\frac{\delta^2}{p+\left(\frac{pq(p+1)}{q+1}\right)\left(\frac{4(q+1)\delta}{p}+1\right)^{\frac{pq+1}{pq}}} \\
&\leq -\frac{\delta^2}{p(2\delta)^{\frac{pq+1}{pq}}+pq(p+1)(q+1)^{\frac{1}{pq}}\left(\frac{4\delta}{p}+\frac{2\delta}{q+1}\right)^{\frac{pq+1}{pq}}} \\
&= -\frac{\delta^2}{4p+25pq(p+1)(q+1)^{\frac{1}{pq}}\delta^{\frac{pq+1}{pq}}} \\
&\leq -\frac{\delta^{\frac{pq-1}{pq}}}{26pq(p+1)(q+1)^{\frac{1}{pq}}} \\
&\leq -\frac{\Psi^{\frac{pq-1}{2pq}}}{26\sqrt{2}pq(p+1)(q+1)^{\frac{1}{pq}}}.
\end{aligned}$$

Cela complète la preuve. ■

4.3.3 Les bornes de l'itération

Nous avons besoin de compter le nombre d'itérations internes nécessaires pour revenir à la situation où $\Psi(V) \leq \tau$ après une mise à jour de μ.

Lemme 4.3.15 *(Proposition 1.3.2 dans [53]) Supposons qu'une séquence $t_k > 0$, $k = 0, 1, 2, ..., K$ satisfaisant l'inégalité suivante :*

$$t_{k+1} \leq t_k - \eta t_k^{\gamma}, \quad \eta > 0, \quad \gamma \in [0, 1[, \quad k = 0, 1, 2, ..., K.$$

Alors

$$K \leq \left\lceil \frac{t_0^{1-\gamma}}{\eta(1-\gamma)} \right\rceil.$$

Notons la valeur de Ψ après $\mu-$update par Ψ_0 et les valeurs subséquentes dans la même itération externe sont notées Ψ_l, $l = 0, 1, 2, ..., K$, où K représente le nombre total d'itérations internes par une itération externe. On a alors $\Psi_0 \leq (\widetilde{\Psi}_1)_0$ et $\Psi_0 \leq (\overline{\Psi}_1)_0$, où $(\widetilde{\Psi}_1)_0$ et $(\overline{\Psi}_1)_0$ sont définis dans (4.17). Alors nous avons $\Psi_{K-1} > \tau$ et $0 \leq \Psi_K \leq \tau$.

Théorème 4.3.1 *Soit* $(\widetilde{\Psi}_1)_0$ *et* $(\overline{\Psi}_1)_0$ *définies dans* (4.17) *et soit* K_1 *et* K_2 *le nombre total d'itérations internes dans une itération externe pour les méthodes à petit et grand pas, respectivement. Alors pour* $q \geq 1$, *on a*

(i) $K_1 \leq \left[26\sqrt{2}pq(p+1)(q+1)^{\frac{1}{pq}}(\widetilde{\Psi}_1)_0^{\frac{pq+1}{2pq}}\right]$,

(ii) $K_2 \leq \left[26\sqrt{2}pq(p+1)(q+1)^{\frac{1}{pq}}(\overline{\Psi}_1)_0^{\frac{pq+1}{2pq}}\right]$.

Preuve. Pour (i), combinant Lemme 4.3.14 et Lemme 4.3.15 avec $\eta = \frac{1}{26\sqrt{2}pq(p+1)(q+1)^{\frac{1}{pq}}}$ et $\gamma = \frac{pq-1}{2pq}$, on a

$$K_1 \leq \left[26\sqrt{2}pq(p+1)(q+1)^{\frac{1}{pq}}(\widetilde{\Psi}_1)_0^{\frac{pq+1}{2pq}}\right].$$

Pour (ii), de la même manière, on a

$$K_2 \leq \left[26\sqrt{2}pq(p+1)(q+1)^{\frac{1}{pq}}(\overline{\Psi}_1)_0^{\frac{pq+1}{2pq}}\right].$$

■

Maintenant, nous estimons le nombre total d'itérations de notre algorithme.

Le nombre d'itérations externes est borné par $\left[\frac{1}{\theta}\log\frac{n}{\varepsilon}\right]$ [56]. En multipliant le nombre d'itérations externes par le nombre d'itérations internes, le nombre total d'itérations pour les méthodes à petit et grand pas est borné par :

$$26\sqrt{2}pq(p+1)(q+1)^{\frac{1}{pq}}(\widetilde{\Psi}_1)_0^{\frac{pq+1}{2pq}}\frac{1}{\theta}\log\frac{n}{\varepsilon}$$

et

$$26\sqrt{2}pq(p+1)(q+1)^{\frac{1}{pq}}(\overline{\Psi}_1)_0^{\frac{pq+1}{2pq}}\frac{1}{\theta}\log\frac{n}{\varepsilon}$$

respectivement.

Remarque 4.3.3 *En utilisant la remarque 4.3.2, si* $p = 1$, $q =$*constante, on obtient* $O(\sqrt{n}\log\frac{n}{\varepsilon})$ *itérations pour les méthodes à petit pas. De même, en choisissant* $p = 1$, $q = \log n$, *on obtient* $O(\sqrt{n}\log n\log\frac{n}{\varepsilon})$ *itérations pour les méthodes à grand pas. Ce sont les meilleurs résultats de complexité connus pour de telles méthodes.*

4.4 Deuxième fonction noyau

Dans cette partie, nous présentons une autre nouvelle fonction noyau paramétrée et nous donnons ses propriétés qui sont essentielles à notre analyse de la complexité.

4.4.1 Propriétés de la fonction noyau

On considère notre nouvelle fonction noyau ψ_2 [12] comme suit :

$$\psi_2(t) = t^2 - 1 + \frac{t^{1-q} - 1}{q-1} - \log t, \qquad q > 1, \quad t > 0. \tag{4.21}$$

Il est facile de vérifier que ψ_2 est une fonction noyau barrière et nous donnons les trois premières dérivées par rapport à t :

$$\begin{aligned} \psi_2'(t) &= 2t - t^{-q} - \frac{1}{t}, \\ \psi_2''(t) &= 2 + qt^{-q-1} + \frac{1}{t^2}, \\ \psi_2'''(t) &= -q(q+1)t^{-q-2} - \frac{2}{t^3}. \end{aligned} \tag{4.22}$$

d'après (4.22), on a

$$\psi_2''(t) > 1, \quad t > 0. \tag{4.23}$$

Lemme 4.4.1 *Soit ψ_2 définie dans (4.21), alors*
(i) $t\psi_2''(t) + \psi_2'(t) > 0, \quad 0 < t < 1,$
(ii) $t\psi_2''(t) - \psi_2'(t) > 0, \quad t > 1,$
(iii) $\psi_2'''(t) < 0, \ \ t > 0.$

Preuve. Pour (i), utilisant (4.22), on a

$$t\psi_2''(t) + \psi_2'(t) = 4t + (q-1)t^{-q} > 0,$$

pour tout $q > 1$ et $t > 0$.

Pour (ii), on a

$$t\psi_2''(t) - \psi_2'(t) = (q+1)t^{-q} + \frac{2}{t} > 0,$$

pour tout $q > 1$ et $t > 0$.

Pour (iii), il est clair de (4.22) que

$$\psi_2'''(t) < 0, \text{ pour } t > 0$$

■

Lemme 4.4.2 *Pour ψ_2, on a pour $q > 1$,*
(i) $\frac{1}{2}(t-1)^2 \leq \psi_2(t) \leq \frac{1}{2}[\psi_2'(t)]^2, \ \ t > 0,$
(ii) $\psi_2(t) \leq \frac{1}{2}(3+q)(t-1)^2, \ \ t \geq 1.$

Preuve. Pour (i), utilisant la première condition de (4.11) et (4.23), on a

$$\psi_2(t) = \int_1^t \int_1^\xi \psi_2''(\zeta) d\zeta d\xi \geq \int_1^t \int_1^\xi p d\zeta d\xi = \frac{1}{2}(t-1)^2,$$

la deuxième inégalité est obtenue comme suit :

$$\begin{aligned} \psi_2(t) = \int_1^t \int_1^\xi \psi_2''(\zeta) d\zeta d\xi &\leq \int_1^t \int_1^\xi \psi_2''(\xi)\psi_2''(\zeta) d\zeta d\xi \\ &= \int_1^t \psi_2''(\xi)\psi_2'(\xi) d\xi \\ &= \int_1^t \psi_2'(\xi) d\psi_2'(\xi) d\xi = \tfrac{1}{2}[\psi_1'(t)]^2. \end{aligned}$$

Pour (ii), utilisant le théorème de Taylor, la première condition de (4.11) et Lemme 4.4.1 (iii), on a

$$\begin{aligned} \psi_2(t) &= \psi_2(1) + \psi_2'(1)(t-1) + \tfrac{1}{2}\psi_2''(1)(t-1)^2 + \tfrac{1}{3!}\psi_2'''(c)(t-1)^3 \\ &= \tfrac{1}{2}\psi_2''(1)(t-1)^2 + \tfrac{1}{3!}\psi_2'''(c)(t-1)^3 \\ &< \tfrac{1}{2}\psi_2''(1)(t-1)^2, \end{aligned}$$

tel quet $1 \leq c \leq t$. Ceci complète la preuve. ■

Lemme 4.4.3 *Soit $\varrho : [0, \infty) \to [1, \infty)$ la fonction inverse de ψ_2 pour $t \geq 1$. Alors on a*

$$1 + \sqrt{\frac{2s}{3+q}} \leq \varrho(s) \leq 1 + \sqrt{2s}, \quad q > 1, \ s \geq 0.$$

Preuve. Soit $s = \psi_2(t)$ pour $t \geq 1$, i.e., $\varrho(s) = t$. Par la définition de ψ_2,

$$s = (t^2 - 1) + \frac{t^{1-q} - 1}{q-1} - \log t, \ q > 1, t > 0.$$

Utilisant Lemme 4.4.2 (i), on a

$$s = \psi_2(t) \geq \frac{1}{2}(t-1)^2$$

implique que

$$t = \varrho(s) \leq 1 + \sqrt{2s}.$$

Pour la deuxième inégalité utilisant Lemme 4.4.2 (ii), alors

$$s = \psi_2(t) \leq \frac{1}{2}(3+q)(t-1)^2, \ t \geq 1.$$

alors

$$t = \varrho(s) \geq 1 + \sqrt{\frac{2s}{3+q}}.$$

Ceci complète la preuve. ■

Lemme 4.4.4 *Soit* $\rho : [0, \infty) \to (0, 1]$ *la fonction inverse de* $-\frac{1}{2}\psi_2$ *pour* $0 < t \leq 1$. *Alors on a*

$$\rho(z) \geq \left(\frac{1}{2z+1}\right)^{\frac{1}{q-1}}, \quad q > 1, \ z \geq 0.$$

Preuve. Soit $z = -\frac{1}{2}\psi_2(t)$ pour $0 < t \leq 1$. A partir du définition de ρ, $\rho(z) = t$, pour $z \geq 0$. Alors on a

$$z = -\frac{1}{2}(2t - t^{-q} - \frac{1}{t}) \Longleftrightarrow t^{-q+1} = 2t(z+t) - 1,$$

d'où

$$t^{-q+1} \leq 2(z+1) - 1.$$

par conséquent, nous obtenons

$$t = \rho(z) \geq \left(\frac{1}{2z+1}\right)^{\frac{1}{q-1}}.$$

Ceci complète la preuve. ■

Lemme 4.4.5 *Soit* $\delta(V)$ *et* $\Psi_2(V)$ *définient dans (4.16) et (4.10), respectivement. Alors on a*

$$\delta(V) \geq \sqrt{\frac{1}{2}\Psi_2(V)}, \quad V \in \mathbb{S}^n_{++}.$$

Preuve. Utilisant (4.16) et la deuxième inégalité du Lemme 4.4.2 (i),

$$\delta^2(V) = \frac{1}{4}\sum_{i=1}^{n}(\psi_2'(\lambda_i(V)))^2 \geq \frac{1}{2}\sum_{i=1}^{n}\psi_2(\lambda_i(V)) = \frac{1}{2}\Psi_2(V).$$

Par conséquent, on a $\delta(V) \geq \sqrt{\frac{1}{2}\Psi_2(V)}$. ■

Lemme 4.4.6 *Soit* $0 \leq \theta < 1$ *et* $V_+ = \frac{V}{\sqrt{1-\theta}}$. *Si* $\Psi_2(V) \leq \tau$, *alors pour* $q > 1$ *on a*

(i) $\Psi_2(V_+) \leq \frac{3+q}{2(1-\theta)} \left(\sqrt{n\theta} + \sqrt{2\tau}\right)^2$,

(ii) $\Psi_2(V_+) \leq \dfrac{n\theta + 2\tau + 2\sqrt{2\tau n}}{1-\theta}$.

Preuve. Pour (i), comme $\frac{1}{\sqrt{1-\theta}} \geq 1$ et $\varrho\left(\frac{\Psi_2(V)}{n}\right) \geq 1$, on a $\frac{\varrho\left(\frac{\Psi_2(V)}{n}\right)}{\sqrt{1-\theta}} \geq 1$. Utilisant Lemme 4.3.6 avec $\beta = \frac{1}{\sqrt{1-\theta}}$, Lemme 4.4.2 (ii), Lemme 4.4.3 et $\Psi_2(V) \leq \tau$, on a

$$\begin{aligned}
\Psi_2(V_+) &\leq n\psi_2\left(\tfrac{1}{\sqrt{1-\theta}}\varrho\left(\tfrac{\Psi_2(V)}{n}\right)\right) \\
&\leq \tfrac{n(3+q)}{2}\left(\frac{\varrho\left(\frac{\Psi(V)}{n}\right)}{\sqrt{1-\theta}} - 1\right)^2 \\
&\leq \tfrac{n(3+q)}{2}\left(\frac{1+\sqrt{\frac{2\tau}{n}}-\sqrt{1-\theta}}{\sqrt{1-\theta}}\right)^2 \\
&\leq \tfrac{n(3+q)}{2}\left(\frac{\theta+\sqrt{\frac{2\tau}{n}}}{\sqrt{1-\theta}}\right)^2 \\
&= \tfrac{(3+q)}{2(1-\theta)}\left(\sqrt{n}\theta + \sqrt{2\tau}\right)^2,
\end{aligned}$$

où la dernière inégalité est vérifiée de $1 - \sqrt{1-\theta} = \frac{\theta}{1+\sqrt{1-\theta}} < \theta$, $0 \leq \theta < 1$.

Pour (ii), pour $t \geq 1$, on a

$$\psi_2(t) \leq t^2 - 1.$$

Utilisant Lemme 4.3.6 avec $\beta = \frac{1}{\sqrt{1-\theta}}$, Lemme 4.4.3 et $\Psi_2(V) \leq \tau$, on a

$$\begin{aligned}
\Psi_2(V_+) &\leq n\psi_2\left(\tfrac{1}{\sqrt{1-\theta}}\varrho\left(\tfrac{\Psi_2(V)}{n}\right)\right) \\
&\leq n\left(\left[\tfrac{1}{\sqrt{1-\theta}}\varrho\left(\tfrac{\Psi_2(V)}{n}\right)\right]^2 - 1\right) \\
&= \tfrac{n}{(1-\theta)}\left(\varrho\left(\tfrac{\Psi_2(V)}{n}\right)^2 - (1-\theta)\right) \\
&\leq \tfrac{n}{(1-\theta)}\left(\left[1 + \sqrt{\tfrac{2\Psi_2(V)}{np}}\right]^2 - (1-\theta)\right) \\
&\leq \tfrac{n}{(1-\theta)}\left(\theta + 2\tfrac{\tau}{n} + 2\sqrt{\tfrac{2\tau}{n}}\right) \\
&= \tfrac{n\theta+2\tau+2\sqrt{2\tau n}}{1-\theta},
\end{aligned}$$

Ceci complète la preuve. ■

Définissant pour $q > 1$ et $0 \leq \theta < 1$,

$$(\widetilde{\Psi}_2)_0 = \frac{p(2pq+q+1)}{2(q+1)(1-\theta)}\left(\sqrt{n}\theta + \sqrt{\frac{2\tau}{p}}\right)^2, \quad (\overline{\Psi}_2)_0 = \frac{np\theta + 2\tau + 2\sqrt{2\tau np}}{2(1-\theta)}. \tag{4.24}$$

On utilise $(\widetilde{\Psi}_2)_0$ et $(\overline{\Psi}_2)_0$ comme des bornes supérieures de $\Psi_2(V)$ pour les méthodes à petit pas et à grand pas, respectivement.

4.4.2 Analyse de complexité

Dans cette section, nous calculons un pas de déplacement α efficace, la réduction de la fonction de proximité au cours d'une itération interne et améliorons les résultats de la complexité de l'algorithme, pour $\mu > 0$.

Lemme 4.4.7 *Soit ρ et $\overline{\alpha}$ définient dans Lemme 4.3.11. Si $1 \leq \tau \leq \Psi_2(V)$, alors on a*

$$\overline{\alpha} \geq \frac{1}{2 + (q+1)(4\delta+1)^{\frac{q+1}{q-1}}}.$$

Preuve. Utilisant Lemme 4.3.11, Lemme 4.4.4 et (4.14) on a

$$\begin{aligned}
\overline{\alpha} &\geq \frac{1}{\psi_2''(\rho(2\delta))} \\
&= \frac{1}{2+\frac{1}{\rho(2\delta)^2}+q(\rho(2\delta))^{-q-1}} \\
&\geq \frac{1}{2+(4\delta+1)^{\frac{2}{q-1}}+q(4\delta+1)^{\frac{q+1}{q-1}}} \\
&\geq \frac{1}{2+(q+1)(4\delta+1)^{\frac{q+1}{q-1}}}.
\end{aligned}$$

Cela complète la preuve. ■

Définissons le pas par défaut $\widetilde{\alpha}$ comme suit :

$$\widetilde{\alpha} = \frac{1}{2+(q+1)(4\delta+1)^{\frac{q+1}{q-1}}}, \tag{4.25}$$

avec $\widetilde{\alpha} \leq \overline{\alpha}$.

Lemme 4.4.8 *Soit $\widetilde{\alpha}$ définie dans* (4.25). *Alors on a*

$$f(\widetilde{\alpha}) \leq -\frac{1}{40\sqrt{2}(q+1)}\Psi^{\frac{q-3}{2(q-1)}}.$$

Preuve. Utilisant Lemme 4.3.13 avec $\alpha = \widetilde{\alpha}$ et (4.25), on a

$$\begin{aligned}
f(\widetilde{\alpha}) &\leq -\widetilde{\alpha}\delta^2 \\
&= -\frac{\delta^2}{2+(q+1)(4\delta+1)^{\frac{q+1}{q-1}}} \\
&\leq -\frac{\delta^2}{2(2\delta)^{\frac{q+1}{q-1}}+(q+1)(4\delta+(2\delta))^{\frac{q+1}{q-1}}} \\
&\leq -\frac{\delta^{2-\frac{q+1}{q-1}}}{(8+(q+1)36)} \\
&\leq -\frac{\delta^{\frac{q-3}{q-1}}}{40(q+1)} \\
&\leq -\frac{\Psi^{\frac{q-3}{2(q-1)}}}{40\sqrt{2}(q+1)}.
\end{aligned}$$

Cela complète la preuve. ■

4.4.3 Les bornes de l'itération

Nous avons besoin de compter le nombre d'itérations internes nécessaires pour revenir à la situation où $\Psi_2(V) \leq \tau$ après une mise à jour de μ.

Théorème 4.4.1 *Soit* $(\widetilde{\Psi}_2)_0$ *et* $(\overline{\Psi}_2)_0$ *définies dans* (4.24) *et soit* K_1 *et* K_2 *le nombre total d'itérations internes dans une itération externe pour les méthodes à petit et grand pas, respectivement. Alors pour* $q \geq 1$, *on a*

(i) $K_1 \leq \left[80\sqrt{2}(q-1)(\widetilde{\Psi}_2)_0^{\frac{q+1}{2(q-1)}}\right]$,

(ii) $K_2 \leq \left[80\sqrt{2}(q-1)(\overline{\Psi}_2)_0^{\frac{q+1}{2(q-1)}}\right]$.

Preuve. Pour (i), combinant Lemme 4.3.15 et Lemme 4.4.8 avec $\eta = \frac{1}{40\sqrt{2}(q+1)}$ et $\gamma = \frac{q-3}{2(q-1)}$, on a

$$K_1 \leq \left[80\sqrt{2}(q-1)(\widetilde{\Psi}_2)_0^{\frac{q+1}{2(q-1)}}\right].$$

Pour (ii), de la même manière, on a

$$K_2 \leq \left[80\sqrt{2}(q-1)(\overline{\Psi}_2)_0^{\frac{q+1}{2(q-1)}}\right].$$

■

Maintenant, nous estimons le nombre total d'itérations de notre algorithme. En multipliant le nombre d'itérations externes par le nombre d'itérations internes, le nombre total d'itérations pour les méthodes à petit et grand pas est borné par :

$$80\sqrt{2}(q-1)(\widetilde{\Psi}_2)_0^{\frac{q+1}{2(q-1)}}\frac{1}{\theta}\log\frac{n}{\varepsilon}$$

et

$$80\sqrt{2}(q-1)(\overline{\Psi}_2)_0^{\frac{q+1}{2(q-1)}}\frac{1}{\theta}\log\frac{n}{\varepsilon}$$

respectivement.

Remarque 4.4.1 *En utilisant la remarque 4.3.2, si* q *=constante, on obtient* $O(\sqrt{n}\log\frac{n}{\varepsilon})$ *itérations pour les méthodes à petit pas. De même, en choisissant* $q = 1 + \log n$, *on obtient* $O(\sqrt{n}\log n\log\frac{n}{\varepsilon})$ *itérations pour les méthodes à grand pas. Ce sont les meilleurs résultats de complexité connus pour de telles méthodes.*

4.5 Résultats numériques

Dans cette section, notre objectif principal est de fournir des expérimentations numériques concernant les performances pratiques de la nouvelle fonction noyau proposée dans (4.13) par rapport à la fonction noyau donnée dans [25].

$$\psi_f(t) = \frac{t^2-1}{2} - \int_1^t \left(\frac{e-1}{e^x-1}\right)^p dx, \qquad p \geq 1,$$

et de montrer que notre fonction noyau (4.13) était bien prometteuse dans la pratique par rapport à certaines autres fonctions noyaux considérées.

Il a était montré que les deux fonctions sont très proches en terme de performance numérique, mais on a remarqué que la fonction $\psi_1 = \psi_{new}$ domine dans quelques instances. Donc on l'a utilisé pour comparer avec la fonction ψ_f donnée précédemment.

Dans tous les tableaux, nous désignons par "*iter*" et "CPU" le nombre total d'itérations et le temps requis en secondes, respectivement. De plus, $\psi_{new} = \psi_1$ représente la nouvelle fonction noyau proposée dans (4.13). Nous avons utilisé les paramètres suivants :

$$\begin{aligned} \varepsilon &= 10^{-8}, \qquad \tau = \sqrt{n}, \qquad \mu_0 = 1, \\ \theta &\in \{0.1, 0.5, 0.9\}, \quad p = 1, \quad q = \log n/2. \end{aligned}$$

Exemple 4.5.1 *[64] : On considère le problème (SDP) suivant :*

$$A_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -2 & -1 \\ 0 & -1 & 1 & -1 & -2 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 0 & 0 & -2 & 2 & 0 \\ 0 & 2 & 1 & 0 & 2 \\ -2 & 1 & -2 & 0 & 1 \\ 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 & 2 \end{bmatrix},$$

$$A_3 = \begin{bmatrix} 2 & 2 & -1 & -1 & 1 \\ 2 & 0 & 2 & 1 & 1 \\ -1 & 2 & 0 & 1 & 0 \\ -1 & 1 & 1 & -2 & 0 \\ 1 & 1 & 0 & 0 & -2 \end{bmatrix}, \quad C = \begin{bmatrix} 3 & 3 & -3 & 1 & 1 \\ 3 & 5 & 3 & 1 & 2 \\ -3 & 3 & -1 & 1 & 2 \\ 1 & 1 & 1 & -3 & -1 \\ 1 & 2 & 2 & -1 & -1 \end{bmatrix},$$

$$b = \begin{bmatrix} -2 \\ 2 \\ -2 \end{bmatrix}, \quad X_0 = I, \quad S_0 = I, \quad y_0 = (1,1,1)^T.$$

Le tableau suivant résume les résultats obtenus

	ψ_f		ψ_{new}	
θ	$iter$	CPU	$iter$	CPU
0.1	207	22.1097	207	0.4766
0.5	45	4.8909	45	0.1545
0.9	26	2.9296	26	1.0956

Exemple 4.5.2 *(Exemple 1 dans [61]) : Considérons le problème (SDP) suivant :*

$$\begin{aligned} A_1 &= \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \quad A_2 = I, \quad C_{ij} = -1, \qquad \forall i,j \in \{1,2\} \\ b &= \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad X_0 = diag(0.5, 0.5), \quad S_0 = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}, \quad y_0 = \begin{bmatrix} 0 \\ -3 \end{bmatrix}. \end{aligned}$$

Le tableau suivant résume les résultats obtenus

	ψ_f		ψ_{new}	
θ	$iter$	CPU	$iter$	CPU
0.1	199	14.3190	198	0.2147
0.5	43	3.4335	43	0.1623
0.9	25	1.8708	24	0.1059

Exemple 4.5.3 *(Exemple 2 dans [61]) : Dans cet exemple, on a*

$$\begin{aligned} C &= diag(5,8,8,5), \quad A_4 = I, \quad b = (1,1,1,2)^T \\ \text{et } A_k, \quad k &= 1,2,3 \text{ sont définies comme suit :} \\ A_k(i,j) &= \begin{cases} 1 & si \;\; i = j = k \;\; ou \; i = j = k+1; \\ -1 & si \;\; i = k, \;\; j = k+1 \;\; ou \; i = k+1, \;\; j = k; \\ 0 & sinon. \end{cases} \\ X_0 &= \frac{1}{2} I, \quad S_0 = \begin{bmatrix} 2 & 1.5 & 0 & 0 \\ 1.5 & 3.5 & 1.5 & 0 \\ 0 & 1.5 & 3.5 & 1.5 \\ 0 & 0 & 1.5 & 2 \end{bmatrix}, \quad y_0 = (1.5, 1.5, 1.5, 1.5)^T. \end{aligned}$$

Le tableau suivant résume les résultats obtenus

	ψ_f		ψ_{new}	
θ	$iter$	CPU	$iter$	CPU
0.1	203	18.8729	203	0.3841
0.5	44	4.1067	44	0.1323
0.9	25	2.3846	25	0.1120

Maintenant, on passe à deux exemples à tailles variables.

Exemple 4.5.4 *(Exemple 4 dans [61]) : Ce problème (SDP) est défini comme suit :*

$$\begin{aligned} A_k(i,j) &= \begin{cases} 1 & si\ i=j=k; \\ 1 & si\ i=j\ et\ i=k+m; \quad k=1,...,m, \\ 0 & sinon. \end{cases} \\ C &= -I,\ b(i)=2,\ i=1,..,m. \\ X_0 &= \begin{cases} 1.5 & i\le j; \\ 0.5 & i>j; \end{cases}, \quad S_0=I, \quad y_0(i)=-2, \quad i=1,...,m. \end{aligned}$$

Les résultats obtenus pour $m \in \{10, 25, 50, 100\}$ *, sont présentés dans les tableaux suivants :*

pour $m = 10$

	ψ_f		ψ_{new}	
θ	$iter$	CPU	$iter$	CPU
0.1	244	66.6537	242	1.4856
0.5	61	16.8074	61	0.6697
0.9	33	9.0325	33	0.5430

pour $m = 25$

	ψ_f		ψ_{new}	
θ	$iter$	CPU	$iter$	CPU
0.1	264	169.4989	259	11.8889
0.5	64	38.3909	63	4.8082
0.9	35	25.2266	34	4.5558

pour $m = 50$

	ψ_f		ψ_{new}	
θ	$iter$	CPU	$iter$	CPU
0.1	275	481.4549	273	152.8225
0.5	65	174.0386	65	79.8312
0.9	37	127.1565	37	70.5128

pour $m = 100$

	ψ_f		ψ_{new}	
θ	$iter$	CPU	$iter$	CPU
0.1	301	$6.7508e^{+3}$	294	$4.7225e^{+3}$
0.5	67	$2.8665e^{+3}$	67	$2.4653e^{+3}$
0.9	44	$2.9516e^{+3}$	44	$2.4652e^{+3}$

Exemple 4.5.5 *(Exemple cube dans [18]) : Considérons le problème (SDP) suivant :*

pour $k = 1, ..., m$.

$$A_k(i,j) = \begin{cases} 1 & si\ i = j = k\ ou\ i = j = k+m; \\ 25 & si\ i = j = k+1\ ou\ i = j = k+m+1; \\ -5 & si\ i = k,\ j = k+1\ ou\ i = k+m,\ j = k+m+1; \\ -5 & si\ \ i = k+1,\ j = k\ ou\ i = k+m+1,\ j = k+m; \\ 0 & sinon. \end{cases},$$

$$C = -2I,\ b(i) = 2,\ i = 1, .., m, \quad X_0 = S_0 = I, \quad y_0 = (0, 0, ..., 0)^T.$$

L'algorithme est exécuté avec $m \in \{100, 170, 250\}$ *et* $\theta = 0.9$, *le tableau suivant résume les résultats obtenus*

	ψ_f		ψ_{new}	
m	$iter$	CPU	$iter$	CPU
100	44	$2.3945e^{+3}$	44	$1.9658e^{+3}$
170	45	$3.7642e^{+4}$	45	$2.6814e^{+4}$
250	50	$2.1550e^{+5}$	50	$1.9942e^{+5}$

Exemple 4.5.6 *(Problèmes tests de SDPLIB [13]) : L'algorithme a été testé sur des problèmes de référence de la bibliothèque des problèmes tests SDPLIB. Ils commencent avec le point initial* $X_0 = S_0 = I$ *et* $y_0 = 0$ *et avec* $\theta = 0.9$.

Le tableau suivant résume les résultats obtenus

			ψ_f		ψ_{new}	
	m	n	$iter$	CPU	$iter$	CPU
$control1$	21	15	27	6.1521	27	0.5439
$control2$	66	30	27	17.3590	27	7.6585
$control3$	136	45	27	84.3097	27	74.1237
$control4$	231	60	27	423.7427	27	356.1534
$control5$	351	75	27	$1.4224e^{+3}$	27	$1.2821e^{+3}$
$gpp100$	101	100	28	232.8007	28	204.1248
$hinf1$	13	16	29	11.5243	29	0.2655
$hinf2$	13	16	30	11.7002	30	0.2843
$hinf3$	13	16	32	8.8578	32	0.2993
$hinf4$	13	16	30	8.4722	30	0.2828
$hinf5$	13	16	32	9.8052	32	0.3148
$hinf6$	13	16	31	9.2530	31	0.3049
$hinf7$	13	16	30	8.0903	30	0.3229
$hinf8$	13	16	33	9.5336	33	0.2958
$hinf9$	13	16	32	8.7731	32	0.3122
$hinf10$	21	18	30	8.6530	30	0.5635
$hinf11$	31	22	30	12.0619	30	1.4668
$hinf12$	43	24	30	12.6508	30	2.6228
$hinf13$	57	30	35	22.0333	35	8.0645
$hinf14$	73	34	30	26.9598	30	14.3958
$hinf15$	91	37	32	44.1425	32	28.4352
$mcp100$	100	100	34	322.1379	33	246.0535
$qap5$	136	26	27	34.2571	27	24.6527
$theta1$	104	50	32	79.2318	30	54.4034
$truss1$	6	13	28	6.0656	28	0.0695
$truss2$	58	133	28	171.3178	28	125.5758
$truss3$	27	31	30	12.2743	29	1.5593
$truss4$	12	19	28	7.3415	28	0.1798

d'après les résultats obtenus, on conclut les remarques suivantes :

– Notre nouvelle fonction noyau ψ_{new} produit un meilleur temps d'exécution que la fonction noyau ψ_f.

– Dans certains cas, notre nouvelle fonction noyau ψ_{new} réduit le nombre d'itérations par rapport à l'autre fonction noyau ψ_f.
– Le nombre d'itérations de l'algorithme dépend de la valeur du paramètre θ, dans la plupart des cas, le plus grand θ donne un meilleur nombre d'itérations.
– Les résultats montrent une croissance très lente quand n augmente ce qui est précisément ce que l'on espère pour les MPIs.

4.6 Conclusion

Dans ce chapitre, nous avons proposé deux nouvelles fonctions noyaux efficaces, munie d'un paramètre logarithmique. Nous avons montré que le meilleur résultat de complexité pour les méthodes à petit et à grand pas peut être atteint, à savoir $O(\sqrt{n}\log n\log\frac{n}{\varepsilon})$ pour les méthodes à grand pas et $O(\sqrt{n}\log\frac{n}{\varepsilon})$ pour celles à petit pas. Les résultats obtenus dans ce chapitre représentent des contributions importantes pour améliorer l'analyse de la convergence et de la complexité des MPIs primales-duales pour SDP. De plus, les résultats numériques ont été présentés pour illustrer les avantages de nos fonctions noyaux et montrer leurs efficacité.

Conclusion générale

Dans cette thèse, on s'est intéressé à résoudre le problème de programmation semi-définie SDP par les deux familles des méthodes de points intérieurs les plus connues en optimisation :

- La réduction du potentiel projective.
- La trajectoire centrale (TC).

De première part, nous avons proposé un algorithme efficace primal-dual de points intérieurs du type projectif à deux phases. Pour calculer le pas de déplacement, plusieurs alternatives ont été proposées. Nous avons remarqué, à titre d'exemple, que celles de la recherche linéaire sont les plus coûteuses. Pour remédier ce problème, on a proposé une nouvelle approche composée de trois nouvelles alternatives pour calculer le pas de déplacement par une technique simple, facile et moins coûteuse. Pour valoriser notre contribution, illustrer l'efficacité de notre approche et confirmer la convergence des trois alternatives vers la solution optimale du problème SDP, nous avons présenté des simulations numériques qui prouvent, confirment et consolident les résultats théoriques trouvés. Les résultats obtenus, compatibles avec les propos théoriques, ont montré la supériorité de la deuxième alternative par rapport aux autres, mesurée par le nombre d'itérations.

D'autre part, nous nous sommes intéressés à la forme analytique de la fonction noyau dans l'algorithme de trajectoire centrale pour résoudre un problème semi-défini. Nous avons proposé deux fonctions noyaux logarithmiques avec paramètre qui sont les premières fonctions de ce type donnant la meilleure complexité algorithmique pour les méthodes à grand pas. Ces propositions représentent des nouvelles contributions d'ordre algorithmique, théorique et numérique.

Les résultats obtenus sont très encourageants, ce qui nous motive à généraliser nos travaux à plusieurs problèmes d'optimisation et donnent lieu à d'autres perspectives dans le domaine de l'optimisation numérique (les problèmes quadratique semi-défini, les problèmes de complémentarité, les problèmes coniques,

etc.).

Ce travail est étayé par les publications suivantes :

*D. Benterki, A. Yassine, A. Zerari, Interior-point algorithm for semidefinite programming based on a logarithmic kernel function, Bull. Math. Soc. Sci. Math. Roumanie. à paraître.

*A. Zerari, D. Benterki, A new efficient primal-dual projective interior point method for semidefinite programming, Journal of Nonlinear Functional Analysis. 2019 (2019) 1–12.

Bibliographie

[1] F. Alizadeh, Combinatorial Optimization with Interior-Point Methods and Semi-Definite Matrices, Ph.D. Thesis, Computer Science Department, University of Minnesota, Minneapolis, MN, (1991).

[2] F. Alizadeh, Interior point methods in semidefinite programming with applications to combinatorial optimization, SIAM J. Optim. 5 (1995) 13–51.

[3] F. Alizadeh, J. P. A. Haeberly, M. L. Overton, Primal-dual interior-point methods for semidefinite programming : Convergence rates, stability and numerical results, SIAM J. Optim. 8 (1998) 746–768.

[4] E. D. Andersen, K. D. Andersen, The MOSEK interior point optimizer for linear programming : an implementation of the homogeneous algorithm, In H. Frenk, K. Roos, T. Terlaky, and S. Zhang, editors, High performance optimization, pages 197–232. Kluwer Academic Publishers, (2000).

[5] Y. Q. Bai, M. El Ghami, C. A. Roos, new efficient large-update primal-dual interior-point method based on a finite barrier, SIAM J. Optim. 13 (2003) 766–782.

[6] Y. Q. Bai, M. El Ghami, C. A. Roos, comparative study of kernel functions for primal-dual interior-point algorithms in linear optimization, SIAM J. Optim. 15 (2004) 101–128.

[7] Y. Q. Bai, G. Lesaja, C. Roos, Y. Q. Wang, M. El Ghami, A Class of Large-update and Small-update Primal-dual Interior-point Algorithms for Linear Optimization, J. Optim. Theory Appl. 13 (2008) 341–359.

[8] R. Bellman, K. Fan, On systems of linear inequalities in hermitian matrix variables, In V.L. Klee, editor, Convexity, Proc. Sympos. Appl. Math. 7 (1963) 1–11.

[9] D. Benterki, Résolution des problème de programmation semi-définie par des méthodes de réduction du potentiel, thèse de Doctorat d'état, département de Mathématiques, Université Ferhat Abbas, Sétif-1, Algérie, (2004).

[10] D. Benterki, J. P. Crouzeix, B. Merikhi, A numerical implementation of an interior point method for semidefinite programming, Pesquisa Operacional, Janeiro. 23 (1) (2003) 49–59.

[11] D. Benterki, J.P. Crouzeix and B. Merikhi, A numerical feasible interior point method for lineair semidefinite programs, RAIRO Oper. Res. 41 (2007) 49–59.

[12] D. Benterki, A. Yassine, A. Zerari, Interior-point algorithm for semidefinite programming based on a logarithmic kernel function, Bull. Math. Soc. Sci. Math. Roumanie. à paraître.

[13] B. Borchers, SDPLIB 1.2, A library of semidefinite programming test problems, Optim Methods Softw. 11 (1999) 683–690.

[14] M. Bouafia, D. Benterki, A. Yassine, An efficient primal-dual interior point method for linear programming problems based on a new kernel function with a trigonometric barrier term, J. Optim. Theory Appl. 170 (2016) 528–545.

[15] X. Burer, R.D.C Monteiro, Y. Zhang. Solving a class of semidefinite programs via nonlinear programming, Math. Program. 93 (2002) 97–122.

[16] B. K. Choi, G. M. Lee, On complexity analysis of the primal-dual interior-point method for semidefinite optimization problem based on a new proximity function, Nonlinear Anal. 71 (2009) 2628–2640.

[17] B. Craven, B. Mond, Linear programming with matrix variables, Linear Algebra Appl. 38 (1981) 73–80.

[18] J. P. Crouzeix, B. Merikhi, A logarithm barrier method for semidefinite programming, RAIRO. Oper. Res. 42 (2008) 123–139.

[19] E. De Klerk, Aspects of semidefinite programming : interior point algorithms and selected applications, Kluwer Academic Publishers, (2002).

[20] I. I. Dikin, Iterative solution of problems of linear and quadratic programming, Dokl. Akad. Nauk. 174 (1967) 747–748.

[21] M. El Ghami, Primal-dual algorithms for semidefinite optimization problems based on generalized trigonometric barrier function, International Journal of Pure and Applied Mathematics. 114 (2017) 797–818.

[22] M. El Ghami, Y. Q. Bai, C. Roos, Kernel-function based algorithms for semidefinite optimization, RAIRO. Oper. Res. 43 (2009) 189–199.

[23] M. El Ghami, Z. A. Guennoun, S. Bouali, T. Steihaug, Interior-point methods for linear optimization based on a kernel function with a trigonometric barrier term, J. Comput. Appl. Math. 236 (2012) 3613–3623.

[24] M. El Ghami, C. Roos, T. Steihauga, A generic primal-dual interior-point method for semidefinite optimization based on a new class of kernel functions, Optim. Methods Softw. 25 (3) (2010) 387–403.

[25] S. Fathi-hafshejani, A. Fakharzadeh, An interior-point algorithm for semidefinite optimization based on a new parametric kernel function, Journal of Nonlinear Functional Analysis. 2018 (2018) 1–24.

[26] C. C. Gonzaga, Path following methods for linear programming, SIAM Review. 34 (1992) 167–227.

[27] M. Halicka, E. De Klerk, C. Roos, On the convergence of the central path in semidefinite optimization, SIAM J. Optim. 12 (2002) 1090–1099.

[28] C. Helmberg, Semidefinite programming for combinatorial optimization, Konrad-Zuse-Zentrum für Informations technik Berlin, Takustrabe 7, D-14195 Berlin, Germany, (2000).

[29] C. Helmberg, K.C. Kiwiel, A spectral bundle method with bounds. Math. Program., 93 (2) (2002) 173–194.

[30] C. Helmberg, F. Rendl, R. J. Vanderbei, H.Wolkowicz, An interior-point method for semidefinite programming, SIAM J. Optim. 6 (1996) 342–361.

[31] J. Ji, F. A. Potra, R. Sheng, On the local convergence of a predictor-corrector method for semidefinite programming, SIAM J. Optim. 10 (1999) 195–210.

[32] N. Karmarkar, A new polynomial-time algorithm for linear programming, Combinatorica. 4 (1984) 373–395.

[33] S. Kettab, D. Benterki, A relaxed logarithmic barrier method for semidefinite programming, RAIRO. Oper. Res. 49 (2015) 555–568.

[34] M. Kojima, S. Mizuno, A. Yoshise, A primal-dual interior point algorithm for linear programming, in : N. Megiddo (Ed.), Progress in Mathematical Programming : Interior Point and Related Methods, Springer Verlag, New York, (1989) 29–47.

[35] M. Kojima, S. Shindoh, S. Hara, Interior-point methods for the monotone semidefinite linear complementarity problem in symmetric matrices, SIAM J. Optim.7 (1997) 86–125.

[36] M. H. Koulaei, T. Terlaky, On the extension of a Mehrotra-type algorithm for semidefinite optimization, Technical Report 2007/4, Advanced optimization Lab. Department of Computing and Software, McMaster University, Hamilton, Ontario, Canada.

[37] I. Krim, Optimization sous contraintes de semi-définie positivité, Thèse de Magister, Université d'Oran, Algérie, (2014).

[38] Y. H. Lee, Y. Y. Cho, J. H. Jin, G. M. Cho, Interior-point algorithms for LO and SDO based on a new class of kernel functions. J. Nonlinear Convex Anal. 13 (2012) 555–573.

[39] C. Liu, H. Liu, A new second-order corrector interior-point algorithm for semidefinite programming, Math. Methods Oper. Res. 75 (2012) 165–183.

[40] I. J. Lustig, R. E. Marsten, D. F. Shanno, Interior point methods : Computational state of the art, ORSA Journal on Computing. 6 (1994) 1–15.

[41] H. Mansouri, C. Roos, A new full-Newton step O(n) infeasible interior-point algorithm for semidefinite optimization, Numer. Algorithms. 52 (2) (2009) 225–255.

[42] H. Mansouri, C. Roos, Simplified O(nL) infeasible interior-point algorithm for linear optimization using full Newton steps, Optim. Methods Softw. 22 (3) (2007) 519–530.

[43] N. Megiddo, Pathways to the optimal set in linear programming, in : N. Megiddo (Ed.), Progress in Mathematical Programming : Interior Point and Related Methods, Springer Verlag, New York, (1989) 313–158.

[44] Z. Moaberfard, S. Fathi-Hafshejani, A. J. Fakharzadeh, An interior-point method for linear optimization based on a trigonometric kernel function, Journal of Nonlinear Functional Analysis. (2019) 1–17.

[45] R. D. C. Monteiro, I. Adler, M. G. C. Resende, A polynomial time primal dual affine scaling algorithm for linear and convex quadratic programming and its power series extension, Math. Oper. Res. 15 (1990) 191–214.

[46] R. D. C. Monteiro, T. Tsuchiya, Polynomial convergence of a new family of primal-dual algorithms for semidefinite programming, SIAM J. Optim. 9 (1999) 551–577.

[47] R. D. C. Monteiro, Primal-dual path-following algorithms for semidefinite programming, SIAM J. Optim. 7 (1997) 663–678.

[48] R. D. C. Monteiro, Polynomial convergence of primal-dual algorithms for semidefinite programming based on Monteiro and Zhang family of directions, SIAM J. Optim. 8 (1998) 797–812.

[49] Y. E. Nesterov, A. S. Nemirovsky, Interior-polynomial methods in convex programming, vol. 13, Studies in applied mathematics, Society for Industrial and applied mathematics, Philadelphia, PA, (1994).

[50] Y. Nesterov, M.J. Todd, Self-scaled barriers and interior-point methods for convex programming. Math. Oper. Res. 22 (1997) 1–42.

[51] A. M. Nunez, R. M. Freund, Condition-measure bounds on the behavior of the central trajectory of semidefinite program, SIAM J. Optim. 12 (2001) 818–836.

[52] F. Oustry, A second order bundle method to minimize the maximum eigenvalue function, Math. Program. 89 (2000) 1–33.

[53] J. Peng, C. Roos, T. Terlaky. Self-Regularity, A new paradigm for primal-dual interior-point algorithms, Princeton University Press. (2002).

[54] S. Poljak, F. Rendl, H. Wolkowicz, A recipe for best semidefinite relaxation for (0,1)-quadratic programming, J. Global Optim. 7 (1995) 51–73.

[55] C. Roos, A full-Newton step O(n) infeasible interior-point algorithm for linear optimization, SIAM J. Optim. 16 (4) (2006) 1110–1136 :

[56] C. Roos, T. Terlaky, J. Ph. Vial, Theory and Algorithms for Linear Optimization : An interior point approach, John Wiley & Sons, New York, (1997).

[57] N. Z. Shor, Quadratic optimization problems, Soviet J. Circuits Systems Sciences. 25 (1987) 1–11.

[58] J. Sturm, Theory and algorithms of semidefinite programming, in : H. Frenk, C. Roos, T. Terlaky, S. Zhang (Eds.), High Performance Optimization, Kluwer Academic Publishers, (2000) 3–20.

[59] M. J. Todd, K. C. Toh, R. H. Tütüncü, On the Nesterov-Todd direction in semidefinite programming, SIAM J. Optim. 8 (1998) 769–796.

[60] I. Touil, D. Benterki, A primal-dual interior-point method for the semidefinite programming problem based on a new kernel function, Journal of Nonlinear Functional Analysis. (2019) 1–17.

[61] I. Touil , D. Benterki, A. Yassine, A feasible primal-dual interior point method for linear semidefinite programming, J. Comput. Appl. Math. 312 (2016) 216–230.

[62] L. Tuncel, Potential reduction and primal dual methods, in : H. Wolkowicz, R. Saigal, L. Vandenberghe (Eds.), Theory, Algorithms and Applications, in : Handbook of Semidefinite Programming, Kluwer Academic Publishers, Boston, MA, (2000).

[63] L. Vandenberghe, S. Boyd, Primal-dual potential reduction method for problems involving matrix inequalities, Math. Programming. Series B. 69 (1995) 205–236.

[64] G. Q. Wang, Y. Q. Bai, C. Roos, Primal-Dual Interior-Point Algorithms for Semidefinite Optimization Based on a Simple Kernel Function, Journal of Mathematical Modelling and Algorithms. 4 (2005) 409–433.

[65] G. Q. Wang, Y. Q. Bai, A class of polynomial primal-dual interior-point algorithm for semidefinite optimization. J. Shanghai Univ. Nat. Sci.. 10 (2006) 198–207.

[66] G. Q. Wang, Y. Q. Bai, A new primal-dual path-following interior-point algorithm for semidefinite optimization, J. Math. Anal. Appl. 353 (2009) 339–349.

[67] H. Wolkowicz, R. Saigal, L. Vandenberghe. Handbook of semidefinite programming, theory, algorithms and applications, Kluwer Academic Publishers. (2000).

[68] H. Wolkowicz, G.P.H. Styan, Bounds for eigenvalues using traces, Linear Algebra Appl. 29 (1980) 471–506.

[69] S.J. Wright, Primal-dual interior point methodes, SIAM, Philadelphia, USA. (1997).

[70] Y. Ye, Interior point algorithms, Discrete Mathematics and Optimization. Wiley-Interscience, New York. (1997).

[71] A. Zerari, D. Benterki, A new efficient primal-dual projective interior point method for semidefinite programmig, Journal of Nonlinear Functional Analysis. 2019 (2019) 1–12. doi.org/10.23952/jnfa.2019.12

[72] Y. Zhang, On extending some primal-dual interior-point algorithms from linear programming to semidefinite programming, SIAM J. Optim. 8 (1998) 365–386.