République Algérienne Démocratique et Populaire
Ministère de l'enseignement Supérieur et de la Recherche Scientifique
Université Mohammed Seddik Ben Yahia - Jijel

Faculté des Sciences Exactes et Informatique
Département de Mathématiques

№ d'ordre : .....
№ de série : .....

Mémoire de fin d'études

Présenté pour l'obtention du diplôme de

Master

Spécialité : Mathématiques.
Option : EDP et Applications.

Thème

Problème d'initialisation et mise en œuvre d'une méthode projective pour la programmation semi-définie linéaire

Présenté par :

Amouche Messaouda

Devant le jury :

Président	: **Boufanouche Razika**	M.C.B Université de Jijel.
Encadreur	: **Menniche Linda**	M.C.B Université de Jijel.
Examinateur	: **Bekkouche Fatiha**	M.C.B Université de Jijel.

Promotion **2019/2020**

⁕ *Remerciements* ⁕

Avant tout je remercie ALLAH tout puissant qui nous a donné la force et la volonté pour pouvoir finir ce mémoire de fin d'étude.

Je tiens à exprimer mes sincères remerciements à mon encadreur ***Mme. L. Menniche*** *pour ses multiples conseils et pour m'avoir guidé qui m'a fait sur le doit chemin afin de réaliser ce modeste travail.*

Je remercie le ***Mme. R. Boufanouche*** *qui m'a fait l'honneur en acceptant de présider notre jury de mémoire. Je remercie également* ***Mme. F. Bekkouche****, pour l'honneur qu'il nos a fait en acceptant de participer à ce jury.*

Mes vifs remerciements à tous les enseignants du Département de Mathématique en général, et l'équipe de EDP et Applications en particulier.

je tiens à remercier plus personnellement mes parents qui me courager et tout ma famille, mes amis, ainsi que tous ceux qui m'ont aidé, de prés ou de loin.

TABLE DES MATIÈRES

INTRODUCTION

La programmation semi-définie (SDP) est l'un des problèmes d'optimisation qui a connu un fantastique regain d'intérêt depuis les années 90, entre autres par ce que l'on a disposé depuis, d'algorithmes efficaces permettant de les resoudre : il s'agit des algorithmes des méthodes de points intérieurs.

L'évolution rapide et le succés des méthodes de points intérieurs depuis leur relance par karmarkar (1984) [1] dans le domaine de programmation linéaire, ont incité les chercheures du monde entier à développer tout un arsenal de méthodes permuttant de traiter convenablement plusieurs classes de problème considérées jadis difficiles à résoudre, parmi les quelles la programmation semi-définie. Grâce à ces méthodes la programmation semi-définie a connue une évolution considérables sur tous les aspects : théorique, algorithmique et numérique.

Un tel succés constitue un bon stimulant pour d'autre développements. On parle déjà de programmation semi-définie quadratique, non linéaire et programmation semi-définie linéaire.

La programmation semi-définie linéaire est une généralisation de la programmation linéaire au sens que, chaque vecteur $x \in \mathbb{R}^n$ est remplacé par une $(n \times n)$ matrice symétrique semi-définie positive, le coût est remplacé par la trace du produit de deux matrices et la contrainte $x \geq 0$ est remplacée par la matrice semi-définie positive. Ce qui explique d'ailleur le transport du savoir fair de la programmation linéaire à la programmation semi-définie. L'aspect algorithmique pose un problème majeur celui de l'initialisation, qui est déjà étudié au niveau de la programmation linéaire.

Nous proposons dans ce travail un remède assez simple pour la méthode projective de Ye-lustig [2].

Une comparaison numérique intéressant est présentée dans le chapitre 3 de ce mémoire. Les deux premiers chapitres sont consacrés respectivement à une variante de la méthode projective de karmarkar pour la programmation linéaire et pour (SDP).

CHAPITRE 1

MÉTHODE PROJECTIVE DE POINT INTÉRIEUR EN PROGRAMMATION LINÉAIRE

1.1 Introduction

Dans ce chapitre, nous présentons une méthode projective (modèle) de type réduction du potentiel. C'est une variante principale de l'algorithme de Karmarkar (1984), jouissant de propriétés théoriques et numériques attractives.

L'accent est mis sur les aspects fondamentaux ayant fait l'objet de nombreuses études de recherche, et qui serviront d'appui pour les extensions développées au chapitres suivants.

1.2 Préliminaires

Dans ce paragraphe, on présente un rappel des notions fondamentales de la programmation linéaire qui serviront d'appuis pour la suite.

1.2.1 Définition d'un programme mathématique

Un problème de programmation mathématique est représenté comme suit :

$$(P_m)\begin{cases}\min f(x)\\ x \in D.\end{cases}$$

Où
$f : D \to \mathbb{R}$ appelée fonction objectif de (P_m).
$D \subset \mathbb{R}^n$ représente l'ensemble des solutions réalisables.
• Un point x de D est appelé solution réalisable (ou admissible) de (P_m).
• On appelle solution optimale de (P_m), toute solution réalisable x^* réalisant le minimum de f sur D, $f(x^*)$ sera appelée valeur optimale de (P_m).
• Si $D = \mathbb{R}^n$ on dit que (P_m) est un problème d'optimisation sans contraintes.

1.2.2 Définition d'un programme linéaire

Un programme linéaire est un programme mathématique dans lequel :
• L'ensemble D des solutions réalisables est défini par un ensemble d'équations ou d'inéquations linéaires.
• La fonction f est linéaire.

1.2.3 Forme canonique et forme standard d'un programme linéaire

Un programme linéaire est écrit sous forme canonique si l'ensemble des solutions réalisables est défini par un système d'inéquations linéaires et s'écrit sous la forme suivante :

$$(P)\begin{cases}\min c^t x\\ Ax \geq b,\\ x \geq 0.\end{cases}$$

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

Si l'ensemble des solutions réalisables est défini par un système d'équation linéaires, on dit que le programme est écrit sous forme standard.

On peut toujours mettre un programme linéaire quelconque sous forme standard en introduisant des variables supplémentaires appelées "variables d'écart".

Pour cette raison, on ne considérera dans ce qui suit, que des programmes linéaires sous forme standard :

$$(P)\begin{cases}\min c^t x = z^* \\ Ax = b, \\ x \geq 0.\end{cases}$$

Où

- $A \in \mathbb{R}^{m\times n}$, est la matrice des contraintes supposée de plein rang i.e., $Rang(A) = m < n$.
- $c \in \mathbb{R}^n$, est le vecteur coût.
- $b \in \mathbb{R}^m$, est le second membre.
- $c^t x = \sum_{j=1}^{n} c_j x_j$, est la fonction objectif.

Remarque 1.2.1. *:*

Il n'est pas restrictif de supposer $x \geq 0$. En effet, si la variable x n'est pas contrainte en signe on pourra remplacer x par la différence
$x = x^+ - x^-$, $x^+ \geq 0$, $x^- \geq 0$, en fait $x^+ = max[0, x]$, $x^- = max[0, -x]$.

1.3 Dualité en programmation linéaire

La notion de dualité est un consept fondamental en programmation linéaire, qui conduit à un résultat de grand portée théorique et pratique (le théorème de dualité faible et le théorème de dualité forte).

Définition 1.3.1. (***Définition du dual dans le cas général***) *:*

Evidemment, on peut définir le dual d'un problème linéaire quelconque (pas nécessairement sous forme standard), le tableau suivant résume les correspondances entre le problème primal et son dual et permet d'écrire le dual d'un problème linéaire quelconque.

Primal	**Dual**
Fonction objectif (min)	Second membre
Second membre	Fonction objective (max)
A matrice des contraintes	A^t matrice des contraintes
Contrainte $i \geq$	Variable $y_j \geq 0$
Contrainte $i =$	Variable $(y_j \geq, \leq 0)$
Variable $x_j \geq 0$	Contrainte $j \leq$
Variable $(x_j \geq, \leq 0)$	Contrainte $j =$

1.3.1 Dual d'un programme linéaire

Considérons le programme linéaire sous forme standard :

$$(P)\begin{cases}\min c^t x = z^* \\ Ax = b, \\ x \geq 0.\end{cases}$$

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

On appelle problème dual de (P) le programme linéaire suivant :

$$(D)\begin{cases}\max b^t y \\ A^t y \leq c, \\ y \in \mathbb{R}^m.\end{cases}$$

Exemple 1.3.1. *Soit le programme linéaire sous forme standard :*

$$(P)\begin{cases}\min(x_1 + x_2) \\ x_1 - x_2 = 0, \\ x_1 + x_2 + x_3 = 1, \\ x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0.\end{cases}$$

Le dual de ce programme s'écrit :

$$(D)\begin{cases}\max y_2 \\ y_1 + y_2 \leq 1, \\ -y_1 + y_2 \leq 1, \\ y_2 \leq 0.\end{cases}$$

1.3.2 Résultats fondamentaux

Théoréme 1.3.1. (***Dualité faible***) *:*

Si x et y sont respectivement des solutions réalisables de (P) et (D) alors,

$$c^t x \geq b^t y.$$

Théoréme 1.3.2. (***Dualité forte***) *:*

Si x^* et y^* sont respectivement des solutions réalisables pour (P) et (D) et $c^t x^* = b^t y^*$ alors x^* et y^* sont respectivement des solutions optimales pour leurs problèmes (P) et (D).

Théoréme 1.3.3. (***Théorème de dualité***) *:*

• *Si l'un ou l'autre des problèmes* (P) *et* (D) *admet une solution optimale finie, il en est de même pour l'autre, et leur valeurs optimales correspondantes sont égales.*

• *Si l'un des deux problèmes à une valeur optimale non bornée, l'autre n'a pas de solution optimale.*

1.3.3 Application de la dualité

En plus de l'éclairage qu'elle apporte sur certains points d'ordre théorique permettant de mieux caractériser les propriétés du problème donné, les résultats précédents montrent qu'en résolvant un programme linéaire (P), on résoudre en même temps son problème dual (D).

1.4 Méthode projective de point intérieur de Ye-Lustig

Soit le programme linéaire sous forme standard :

$$(P)\begin{cases}\min c^t x = z^* \\ Ax = b, \\ x \geq 0.\end{cases}$$

Où

c, $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$, tel que $rang(A) = m < n$.

Dans ces méthodes, à chaque itération k, on utilise une transformation projective qui ramène la région admissible polyèdrique $\{Ax = b, x \geq 0\}$ à un simplexe

$S_{n+1} = \{x \in \mathbb{R}^{n+1}_+, \sum\limits_{i=1}^{n+1} x_i = 1\}$.

Cette transformation est définie par : $T_a : \mathbb{R}^n \to S_{n+1}$

$$T_a(x^k) = y = \begin{cases} y_i = \dfrac{\frac{x_i^k}{a_i}}{1+\sum\limits_{i=1}^{n}\frac{x_i^k}{a_i}},\ i = 1, ..., n, \\ y_{n+1} = 1 - \sum\limits_{i=1}^{n} y_i,\ a \in \mathbb{R}^n_+. \end{cases}$$

Le problème (P) devient alors

$$P(z^*)\begin{cases}\min(D_k c, -z^*)^t y \\ A_k y = 0, \\ y \in S_{n+1} = \{y \in \mathbb{R}^{n+1},\ y \geq 0,\ \sum\limits_{i=1}^{n+1} y_i = 1\}.\end{cases}$$

Où

$A_k = [AD_k - b] \in \mathbb{R}^{(n+1)\times m}$, $D_k = diag(x^k)$, matrice diagonale.

Comme le calcul d'une solution optimale d'un programme linéaire, sur une sphère est évident, on introduit à chaque itération la plus grande sphère inscrite dans le simplexe S_{n+1}. On obtient le sous problème de $P(z^*)$ suivant :

$$P_s(z^*)\begin{cases}\min(D_k c, -z^*)^t y \\ A_k y = 0, \\ \|y - e_{n+1}\|^2 \leq r^2 < 1.\end{cases}$$

Où
$r = \frac{1}{\sqrt{n(n+1)}}$, le rayon de la sphère.
$e_{n+1} = (1, ..., 1)^t \in \mathbb{R}^{n+1}$, le centre de la sphère.

Comme la valeur de z^* est généralement inconnue, Ye-Lustig approxime z^* à chaque itération par des bornes supérieures $z^k = c^t x^k > z^*$.
Le problème $P_s(z^*)$ est remplacé par :

$$(P_k)\begin{cases}\min(D_k c, -c^t x^k)^t y \\ A_k y = 0, \\ \|y - e_{n+1}\|^2 \leq r^2 < 1.\end{cases}$$

En utilisant les conditions d'optimalités de Karush-Kuhn-Tucker (**K.K.T**) relative au problème (P_k), la solution optimale est donnée par :

$$y_k^* = \frac{e_{n+1}}{n+1} - \alpha^k r d^k,$$

où

• α^k est le pas de déplacement, $0 < \alpha_k < 1$.
• $d^k = \frac{p^k}{\|p^k\|}$, p^k est la projection du vecteur coût $(D_k c, -c^t x^k)^t$ sur le noyau de la matrice A_k i.e., $p^k = (I - A_k^t (A_k A_k^t)^{-1} A_k)(D_k c, -c^t x^k)^t$.

Remarque 1.4.1. *:*

La vitesse de convergence de cette méthode dépend du calcul de la projection p^k. Opération qui domine le coût de l'itération, mais aussi du pas de déplacement.

1.4.1 Calcul de la projection

Nous avons

$$p^k = (I - A_k^t (A_k A_k^t)^{-1} A_k)(D_k c, -c^t x^k)^t.$$

Posons

$$u_k = (A_k A_k^t)^{-1} A_k (D_k c, -c^t x^k)^t.$$

Donc

$$p^k = (D_k c, -c^t x^k) - A_k^t u_k.$$

Le calcul de p^k dépend ainsi de la manière par laquelle, on résoudre le système linéaire :

$$A_k A_k^t u_k = A_k(D_k c, -c^t x^k)^t,$$

dont la matrice $A_k A_k^t$ est symétrique définie positive.
On distingue à ce propos, deux types de méthodes :
• **Méthodes directes** : basées essentiellement sur la factorisation de Cholesky et concernent les systèmes à matrices pleines et de tailles moyennes.
• **Méthodes itératives** : concernent surtout les systèmes à grande dimension à matrice plutôt creuse, et sont en général de type gradient conjugué.

Remarque 1.4.2. *:*

Une étude comparative intéressante entre ces deux stratégies est effectuée dans [2].

1.4.2 Calcul du pas de déplacement

Théoriquement la méthode converge pour tout $0 < \alpha^k < 1$ voir [10], par contre la convergence est d'autant plus rapide que α^k est grand. Plusieurs alternatives sont proposées dans la littérature entraînant des améliorations considérables. Ces alternatives utilisent les méthodes de recherche linéaire. De même en 1989, A. Keraghel dans [10] a proposé une procédure efficace pour améliorer le pas de déplacement α^k, c'est une procédure moins coûteuse que les méthodes de recherche linéaire. Il s'agit de chercher α^k qui vérifié :
• $c^t x^{k+1} < c^t x^k$ (la monotonie).
• $y^k > 0$ (la stricte faisabilité).

Ce qui donne :

$$\alpha^k = \begin{cases} \beta\alpha_{max} & \text{si} \quad \mu < 0, \\ -\beta\alpha_{max} & \text{si} \quad \mu > 0. \end{cases}$$

Avec
$\mu = d_{n+1}^k c^t x^k - c^t D^k d_{[n]}^k \quad 0 < \beta < 1.$

$\alpha_{max} = \frac{1}{(n+1)r\, max_i \in I_+(d_i^k)} \quad I_+ = \{i \in \{1, 2, ..., (n+1)\}/d_i^k > 0\}.$

Remarque 1.4.3. *:*

Des tests numériques sont donnés dans [10] qui offrent des pas de déplacement $\alpha^k > 1$, en accélérant davantage la convergence de l'algorithme.

Algorithme de Ye-Lustig

Début

Initialisation : x° un point strictement réalisable ($Ax^\circ = b$, $x^\circ > 0$), ε est une précision fixée, $k = 0$,

Tant que $\frac{\|d^k\|}{|c^t x^\circ|} > \varepsilon$ **faire**

- $A_k = [AD_k - b], D_k = diag(x^k)$,
- $p^k = (I - A_k^t(A_k A_k^t)^{-1} A_k)(D_k c, -c^t x^k)^t$,
- $d^k = \frac{p^k}{\|p^k\|}$,
- $y^{k+1} = \frac{e_{n+1}}{n+1} - \alpha^k r d^k$,
- $x^{k+1} = T_a^{-1}(y^{k+1}) = D_k y^{k+1}[n]/y_{n+1}^{k+1}, k = k + 1$,

Fin tant que

Fin.

1.5 Calcul d'une solution réalisable initiale

Le calcul d'une solution réalisable initiale, est un problème difficile qui se manifeste dans les différentes variantes des méthodes de points intérieurs. Pour remidier ce problème plusieurs alternatives sont proposées :

1. Une procédure de variable artificielle.
2. Une variante de la méthode de big M.
3. Une Méthode démarrent d'un point non nécessairement réalisable ($x^\circ > 0$).

Dans notre cas, c'est la première procédure qui convient le mieux.

En effet, trouver une solution strictement réalisable pour (P) revient à résoudre le problème :

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

Équivalent au problème auxiliaire :

$$(AP)\begin{cases} \min \lambda \\ Ax + \lambda(b - Ax^\circ) = b,\ x^\circ \in \mathbb{R}_+^n, \\ (x, \lambda)^t \geq 0. \end{cases}$$

Le problème (AP) peut s'écrire aussi sous la forme :

$$(AP)\begin{cases} \min \bar{c}\bar{x} \\ B\bar{x} = b, \\ \bar{x} \geq 0. \end{cases}$$

Où

• $\bar{c} = (0, 0, ..., 1)^t \in \mathbb{R}^{n+1}$.

• $B = \begin{bmatrix} A & b - Ax^\circ \end{bmatrix} \in \mathbb{R}^{(n+1)\times m}$.

• $\bar{x} = (x, \lambda)^t \geq 0$, tel que $\bar{x} \in \mathbb{R}^{n+1}$.

Le problème (AP) possède une solution strictement réalisable triviale $\bar{x} = (x^\circ, 1)^t$, $x^\circ \in \mathbb{R}^n_+$ (l'orthant positif), exemple $x^\circ = (1, 1, ..., 1)^t$.

Le problème (AP) est équivalent au problème (F), au sens :

Lemme 1.5.1. *:*

$(x^*, lambda)$ est une solution optimale de (AP) ssi x^*est une solution de (F), (avec $\lambda < \varepsilon$, ε étant une précision donnée). En pratique, si λ reste loin de zéro, en concluse que le problème (F) n'est pas de solutions et alors (P) est non réalisable.

1.5.1 Accélération de la convergence de l'algorithme de Ye-Lustig

Rappelons que le calcul de la projection p^k domine le coût de l'itération, pour cela D. Benterki et B. Merikhi dans [5] suggèrent d'améliorer les performances de cet algorithme en modifiant la phase 1. Cette modification permet de réduire le coût et le nombre d'itérations. L'algorithme modifié obtenu est le suivant :

Algorithme modifié [5]

Début

a) Initialisation : $x^\circ > 0$, $\lambda^\circ = 1$, $k = 0$,

Si $\|Ax^k - b\| \leq \varepsilon$ **Stop** : x^k est une solution strictement réalisable de (P),

Sinon Trouver u° solution de système : $AA^t u^\circ = \lambda^\circ(b - Ax^\circ)$,

• **b)**

Si $\lambda^k \leqslant \varepsilon$ **Stop** : x^k est une solution strictement réalisable de (P),

Sinon Prendre

• $u^k = \lambda^k u^\circ$,

• $z^k = -diag[(x^k)^{-1}]A^k u^k$,

Si $max|z^k|_i < 1$ **Alors** : $x^k + A^t u^k$ est une solution réalisable de (P),

Sinon Calculer l'itération (x^{k+1}, λ^{k+1}) par l'algorithme de Ye-Lustig et aller à **c)**.

• **c)** poser $k = k + 1$ est aller à **b)**,

Fin.

Exemple 1.5.1. *Reprenons, à titre indicatif, quelques tests numériques présentés dans [5]*

1. ***Hypercube*** (*matrice creuse*)
   $n = 2m$, $A[i,j] = 0$ *si* $i \neq j$ *ou* $(i+1) \neq j$.
   $A[i,i] = A[i,i+m] = 1$, $b[i] = 2$ *si* $i, j = 1, ..., m$.

Dimension $m \times n$	*Nbr d'itérations Ye-Lustig*	*Nbr d'itérations algorithme modifié*
50 × 100	*3*	*1*
100 × 200	*3*	*1*
150 × 300	*3*	*1*
200 × 400	*3*	*1*

2. ***Hilbert*** (*matrice pleine*)
   $n = 2m$, $A[i,j] = \frac{1}{i+j}$, $A[i,i+m] = 1$, $b[i] = \sum_{j=1}^{m} \frac{1}{i+j}$, $\forall\, i, j = 1, ..., m$.

Dimension $m \times n$	*Nbr d'itérations Ye-Lustig*	*Nbr d'itérations algorithme modifié*
50 × 100	*3*	*1*
100 × 200	*3*	*1*
150 × 300	*3*	*1*
200 × 400	*3*	*1*

CHAPITRE 2

MÉTHODE PROJECTIVE DE POINT INTÉRIEUR POUR LA PROGRAMMATION (SDP)

2.1 Introduction

Ce chapitre est consacré à l'étude théorique et numérique d'une extension de la méthode de Ye-Lustig présentée au chapitre 1, en prenant en considération tous les aménagements conduisant à une performance meilleure.

Un problème de programmation semi-définie linéaire (SDP) s'écrit comme suit :

$$(SDP)\begin{cases}\min tr(CX) = \min \sum_i \sum_j c_{ij}x_{ij} \\ tr(A_iX) = b_i,\ i = 1, ..., m, \\ X \text{ symétrique semi-définie positive.}\end{cases}$$

Où

C, A_i, sont des matrices symétiques données.

Nous supposons que les matrices A_i, $i = 1, ..., m$, sont linéairement indépendantes : $(\sum_{i=1}^{m} \lambda_i A_i = 0 \Rightarrow \lambda_i = 0 \,\forall\, i)$.

Notons par :

$E = \{A \in \mathbb{R}^{n\times n} \ / \ A \text{ symétrique}\}$.

$K = \{A \in E \ / \ A \text{ semi-définie positive}\}$.

$int(K) = \{A \in E \ / \ A \text{ définie positive}\}$.

2.2 Exemples de problèmes convertibles en (SDP)

On donne à titre indicatif, quelques problèmes qui se ramènent à la forme d'un programme semi-défini.

2.2.1 Problème de programmation non linéaire

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 problème semi-défini (SDP) comme suit :

$$(SDP) \begin{cases} \min tr(CX) \\ tr(A_1 X) = b_1, \\ X \in K. \end{cases}$$

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

2.2.2 Problème de min-max des valeurs propres

Ce problème a été étudié comme suit : $\lambda^* = \min_{y \in \mathbb{R}}[\lambda_{max}(C + A(y))]$ où $C \in E$, A : $\mathbb{R}^n \longrightarrow E$ est un opérateur linéaire.
Ce problème s'écrit sous la forme :

$$m_p = \min[\lambda,\ \lambda I - C - A(y) \in K,\ y \in \mathbb{R}^n,\ \lambda \in \mathbb{R}].$$

Son dual est le problème semi-définie suivant :

$$m_d = \max[tr(CX) : X \in K,\ A^t(X) = 0,\ tr(X) = 1].$$

Remarque 2.2.1. *:*

Plusieurs problèmes peuvent se formuler en (SDP). Le lecteur peut consulter à ce propos [3], [1] pour plus d'information.

2.3 L'état d'art

Dans la dernière décennie, les problèmes de programmation semi-définie (SDP) ont fait la une de la recherche dans le domaine de la programmation mathématique, le grand nombre d'articles parus dans les revues internationales en témoigne, tout particulièrement les travaux de Shapiro, Fletcher-Craven (1996) qui s'intéressent au conditions d'optimalité du problème (SDP), les travaux de Ramana et Wolkowic (1997), qui ont étudié la dualité forte pour ces problèmes.

Du point du 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. On cite par exemple :

• F. Alizadeh (1995) [1], propose une méthode projective primale-duale.

• Vanderbeghe et Boyd (1996) [14], ont proposé un algorithme primal-dual.

• Monteiro (1997) [12], propose une méthode de trajactoire centrale.

• Todd et al. (1998) [13], ont proposé des variantes newtoniennes pour résoudre le problème de complémentarité linéaire.

• J. Ji et al. (1999) [8], ont étudié la convergence de la méthode prédicteur-correcteur.

• M. Halicka et autres (2002) [7], ont étudié la convergence de la méthode de trajectoire centrale.

• D. Benterki (2004) [3], s'est intéressé à l'etude théorique et numérique de l'algorithme de Alizadeh [1].

Toutes ces méthodes de différent types ont un problème majeur commun celui de l'initialisation.

• En 2004 D. Benterki, J. P. Crouzeix et B. Merikhi [4], ont proposé une méthode de point intérieur réalisable pour résoudre (SDP).

L'avantage principal de cette méthode est le calcul d'une solution réalisable initiale pour (SDP).

L'objectif principal de notre travail dans ce chapitre, est la mise en œuvre de cette méthode à travers des exemples de tailles différentes.

2.4 Dualité en programmation semi-définie

Soit le problème de programmation semi-définie **primal** :

$$(SDP)\begin{cases}\min tr(CX)\\ tr(A_iX)=b_i,\ i=1,...,m,\\ X\in K.\end{cases}$$

Son **dual** est un programme semi-défini comme suit :

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

Notons que le dual a une forme plus simple que le primal, ce qui peut entrainer certains avantages.

Dualité faible :

Soit (X,y) des solutions réalisables pour (SDP) et $(DSDP)$ respectivement, alors :

$$b^ty\leq tr(CX).$$

Dualité forte :

Si l'un des deux problèmes (SDP) ou $(DSDP)$ admet une solution strictement réalisable i.e.,

$$X\in int(K)\text{ et }tr(A_iX)=b_i,\ i=1,..,m.$$

Où :

$$y\in\mathbb{R}^m/C-\sum_{i=1}^{m}y_iA_i\in int(K).$$

Alors :

$$tr(CX)=b^ty.$$

Pour plus de détails sur les résultats de dualité voir[3].

2.5 Méthode projective pour (SDP)

Ces méthodes utilisent à chaque itération une transformation projective ramenant le problème (SDP) à une forme réduite. En effet, supposons à l'itération k, qu'on dispose d'une solution strictement réalisable $X_k \in int(K)$.

Comme X_k est symétrique et définie positive, il existe une matrice triangulaire inférieure L_k tel que : $X_k = L_k L_k^t$.

La transformation projective est définie comme suit :

$$\begin{aligned} T_k : \quad & E \longrightarrow E \times \mathbb{R}^r \\ & X \longrightarrow (Y, y) \end{aligned}$$

où

$$Y = \frac{(n+r)L_k^{-1} X L_k^{-t}}{r + tr(X_k^{-1}X)},\ y = \frac{(n+r)}{r + tr(X_k^{-1}X)} e_r,\ e_r = (1, ..., 1)^t \in \mathbb{R}^r.$$

Propriétés de T_k :

- $T_k(X_k) = (I, e_r)$.
- T_k est bijective et on a : $X = T_k^{-1}(Y, y) = r\frac{l_k Y L_k^t}{e_r^t y}$.

Appliquons T_k sur le problème :

$$(SDP) \begin{cases} \min tr(CX) = z^* \\ tr(A_i X) = b_i,\ i = 1, ..., m, \\ X \in K. \end{cases}$$

On obtient le problème de programmation semi-définie suivant :

$$(TSDP) \begin{cases} \min[tr(C_k Y) - z^* e_r^t y/r] = 0 \\ tr(A_i^k Y) - b_i e_r^t y/r = 0,\ i = 1, ..., m, \\ tr(Y) + e_r^t y = n + r, \\ Y \in K,\ y \geq 0. \end{cases}$$

Où

- $C_k = L_k^t C L_k$.
- $A_i^k = L_k^t A_i L_k,\ i = 1, ..., m$.

Comme la valeur optimale z^* est généralement inconnue, on l'approxime par une borne supérieure $z^* \leq z_k = tr(CX_k)$.

Le problème $(TSDP)$ devient :

$$(TSDP)_k \begin{cases} \min[tr(C_k Y) - z_k e_r^t y/r] \\ tr(A_i^k Y) - b_i e_r^t y/r = 0,\ i = 1, ..., m, \\ trY + e_r^t y = n + r, \\ Y \in K,\ y \geq 0. \end{cases}$$

On relaxe $y \geq 0$ et $Y \in K$ par l'apartenance à la boule de centre (I, e_r) et de rayon β. Donc le problème $(TSDP)_k$ s'écrit :

$$(TSDP)_k \begin{cases} \min[tr(C_k Y) - z_k e_r^t y/r] \\ tr(A_i^k Y) - b_i e_r^t y/r = 0,\ i = 1, ..., m, \\ trY + e_r^t y = n + r, \\ \|Y - I\|^2 + \|y - e_r\|_2^2 \leq \beta^2 < 1. \end{cases}$$

C'est un programme mathématique convexe et différentiable, donc les conditions d'optimalités de K.K.T sont nécessaire et suffisante pour le problème et donnent la solution optimale :

$$\begin{cases} Y = I - \beta P_k, \\ y = (1 - \beta p_k)\, e_r. \end{cases}$$

Où

$P_k = V_k/(\|V_k\|^2 + r\alpha_k^2)^{1/2}, \quad p_k = \alpha_k/(\|V_k\|^2 + r\alpha_k^2)^{1/2}.$

Avec

$V_k = C_k + \sum_{i=1}^{m} \lambda_i A_i^k\ , \quad \alpha_k = -\frac{1}{r}\left(\sum_{i=1}^{m} b_i \lambda_i + z_k \right).$

$\lambda \in \mathbb{R}^m$ est la solution du système linéaire symétrique, défini positif suivant :

$$M\lambda = d.$$

Où

- $M_{ij} = tr(A_i^k A_j^k) + b_i b_j/r,\ i, j = 1, ..., m.$
- $d_i = -b_i z_k/r - tr(C_k A_i^k),\ i = 1, ..., m.$

Le nouveau itéré est donc $X_{k+1} = T_k^{-1}(Y, y)$.

2.5.1 Calcul du pas de déplacement

Les méthodes de recherche linéaire sont connue par leur utilisation pour calculer le pas de deplacement optimal. Les methode de type recheche linéaire les plus connue sont celles de Goldestein-Armija, Fibonnaci,...,etc. Malheureusement, elle sont coûteuse en volume calculatoire, voir même inapplicables pour les problèmes de programmation semi-définie (SDP). Pour remidier cette difficulté, on exploite l'idée proposée par A. Keraghel [10] et D. Benterki [3] qui et moins coûteuse et plus pratique que les methode de recherche liniéaire. En effet, rappellons que l'itération de notre algorithme est définie par :

$$\begin{cases} Y = I - \beta P_k, \\ y = (1 - \beta p_k)\, e_r. \end{cases}$$

À chaque itération, on cherche β qui vérifie :

1) $Y \in int(K)$ (définie positive).

2) $y > 0$.

Proposition 2.5.1. *:*

Soit A une matrice carrée symétrique vérifiant :
$a_{ii} > \sum_{i\neq j=1}^{n} |a_{ij}|$, $i = 1, ..., n$ *alors A est définie positive.*

Appliquons ce résultat sur la condition **1)** on a :
$Y = I - \beta\, P_k$ est définie positive si :
$Y_{ii} > \sum_{i\neq j=1}^{n} |Y_{ij}|, \forall i = 1, ..., n.$
Ce qui implique :

$$1 - \beta(P_k)_{ii} > \beta \sum_{i\neq j=1}^{n} |(P_k)_{ij}|, \forall i = 1, ..., n,$$

d'où

$$1/\beta > (P_k)_{ii} + \sum_{i\neq j=1}^{n} |(P_k)_{ij}|, \forall i = 1, ..., n, \qquad (c_1)$$

d'autre part : $y = (1 - \beta p_k)e_r > 0$,
donc $1 - \beta p_k > 0$,
d'où

$$1/\beta > p_k. \qquad (c_2)$$

De (c_1) et (c_2) on obtient : $1/\beta_{max} = \max\{p_k, (P_k)_{ii} + \sum_{i\neq j=1}^{n} |(P_k)_{ij}|, \forall i = 1, ..., n\}$.
On prend $\beta = \rho\beta_{max}$, $0 < \rho < 1$.

Lemme 2.5.1. *:*
On montre dans [3] que :
- $X_{k+1} = X_k - (\beta/(1 - \beta p_k))(L_k P_k L_k^t - p_k X_k)$.
- $m_k(\beta) = -\beta(tr(C_k P_k) - z_k p_k) < 0$, $\forall\ 0 < \beta < \beta_{max}$.
- $tr(CX_{k+1}) - tr(CX_k) = (1/(1 - \beta p_k))m_k(\beta) < 0$.

Donnons maintenant l'algorithme détaillé proposé par D. Benterki dans [4].

Algorithme de point intérieur pour (SDP) [4]

Début $r = [\sqrt{n}]$, $\rho \in]0,1[$, $\varepsilon > 0$, X_0 un point initial, $k = 0$,

Répéter

• $z_k = tr(CX_k)$,

• Déterminer L_k (matrice triangulaire inférieure) telle que : $X_k = L_k L_k^t$,

• Calculer :

$C_k = L_k^t C_k L_k$, $A_i^k = L_k^t A_i L_k$, $i = 1, ..., m$,

• Calculer la matrice M et le vecteur d tels que :

$M_{ij} = tr(A_i^{(k)} A_j^{(k)}) + \frac{1}{r} b_i b_j$, $i, j = 1, ..., m$,

$d_i = \frac{-b_i}{r} z_k - tr(C_k A_i^k)$, $i = 1, ..., m$,

• Résoudre le système : $M\lambda = d$,

• Calculer :

$V_k = C_k + \sum_{i=1}^{m} \lambda_i A_i^k$,

$\alpha_k = \frac{-1}{r} \left(\sum_{i=1}^{m} b_i \lambda_i + z_k \right)$,

$\tau = (\|V_k\|^2 + r\alpha_k^2)^{1/2}$,

$\beta_{max}(k) = \tau / \max \{\alpha_k, (V_k)_{ii} + \sum_{i \neq j=1}^{n} |(V_k)_{ij}|, i = 1, .., n\}$,

$\beta = \rho \beta_{max}(k)$,

• Calculer :

$X_{k+1} = X_k - \frac{\beta}{\tau - \beta\alpha_k} L_k (V_k - \alpha_k I) L_k^t$,

Jusqu'à : $|tr(C_k V_k) - z_k \alpha_k| \leq \varepsilon$,

Fin.

2.6 Calcul dune solution réalisable initiale

On cherche une $(n \times n)$ matrice X telle que :

$$X \in int(K),\ tr(A_i X) = b_i,\ i = 1, ..., m \qquad (F)$$

Le problème (F) est équivalent au problème suivant :

$$(AP) \begin{cases} \min \lambda \\ tr(A_i X) + \lambda(b_i - tr((A_i X_0)) = b_i,\ i = 1, ..., m, \\ X \in K,\ \lambda \geq 0. \end{cases}$$

Le problème (AP) est se transforme en programme semi-défini :

$$(SDP)' \begin{cases} \min tr(C'X') \\ tr(A'_i X') = b_i \,, i = 1, ..., m, \\ X' \in K. \end{cases}$$

Où
C' est une $(n+1) \times (n+1)$ matrice symétrique définie par :

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

et A'_i sont des $(n+1) \times (n+1)$ matrices définie par :

$$A'_i = \begin{pmatrix} A_i & 0_{n\times 1} \\ 0^t_{1\times n} & b_i - tr(A_i X_0) \end{pmatrix}, \; i = 1, ..., m.$$

Avantage du problème $(SDP)'$:

1. Il possède une solution strictement réalisable triviale.

$$X' = \begin{pmatrix} X_0 & 0 \\ 0 & 1 \end{pmatrix}, \forall X_0 \in int(K) (par\ exemple\ X_0 = I).$$

2. $X' = \begin{pmatrix} X^* & 0 \\ 0 & \varepsilon \end{pmatrix}$ est une solution optimale de $(SDP)'$ ssi X^* est une solution de (F) avec $X^* \in int(K)$ et ε suffisamment petit.

Remarque 2.6.1. *:*

Le problème $(SDP)'$ peut être résolu par l'algorithme [4], ce qui donne une solution strictement réalisable pour (SDP).

2.7 Implémentation numérique

Dans cette section, nous présentons des tests numériques. Les exemples testés sont réalisés en langage MATLAB R2008b. La précision considérée dans les exemples est comprises entre 10^{-4} et 10^{-6}.

Exemple 2.7.1. $m = 2 \,, n = 3 \,, \varepsilon = 10^{-4} \,, \rho = 0.99.$

$$C = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \; A_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \; A_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \; b = (0,1)^t.$$

Après 2 itérations l'algorithme trouve une solution réalisable initiale :

$$X_0 = \begin{pmatrix} 0.274833 & 0 & 0 \\ 0 & 0.274857 & 0 \\ 0 & 0 & 0.450435 \end{pmatrix}$$

Après 2 itérations l'algorithme trouve une solution optimale :

$$X^* = \begin{pmatrix} 0.000011 & 0 & 0 \\ 0 & 0.274857 & 0 \\ 0 & 0 & 1.000018 \end{pmatrix}$$

La valeur optimale est : $z^* = 0.004453$.
Les valeurs du pas de déplacements obtenus varient entre 2.246569 *et* 2.971212.

Exemple 2.7.2. $m = 3\,, n = 5\,, \rho = 0.99\,, \varepsilon = 10^{-4}$.

$$C = \begin{pmatrix} -4 & 0 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, A_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, A_2 = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$A_3 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, b = (4, 8, 4)^t.$$

Après 2 itérations l'algorithme trouve une solution réalisable initiale :

$$X_0 = \begin{pmatrix} 1.142030 & 0 & 0 & 0 & 0 \\ 0 & 0.332052 & 0 & 0 & 0 \\ 0 & 0 & 2.526051 & 0 & 0 \\ 0 & 0 & 0 & 6.047727 & 0 \\ 0 & 0 & 0 & 0 & 2.194264 \end{pmatrix}$$

Après 4 itérations l'algorithme trouve une solution optimale :

$$X^* = \begin{pmatrix} 3.999984 & 0 & 0 & 0 & 0 \\ 0 & 0.00002 & 0 & 0 & 0 \\ 0 & 0 & 0.000012 & 0 & 0 \\ 0 & 0 & 0 & 0.000018 & 0 \\ 0 & 0 & 0 & 0 & 0.000026 \end{pmatrix}$$

La valeur optimale est : $z^* = -15.995908$.

Les valeurs du pas de déplacements obtenus varient entre 2.267461 *et* 2.65256.

Exemple 2.7.3. $m = 3\,,\ n = 6\,, \rho = 0.99\,, \varepsilon = 10^{-4}$.

$$C = \begin{pmatrix} -4 & 0 & 0 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & -2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},\ A_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$A_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},\ A_3 = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$b = (6, 2, 4)^t$.

Après 3 itérations la solution réalisable initiale :

$$X_0 = \begin{pmatrix} 0.377853 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.006844 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.308560 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4.320431 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.306784 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.922096 \end{pmatrix}$$

Après 4 itérations la solution optimale :

$$X^* = \begin{pmatrix} 1.999877 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.000095 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.000036 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4.000182 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.000005 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.000036 \end{pmatrix}$$

La valeur optimale est : $z^* = -7.995594$.
Les valeurs du pas de déplacements obtenus varient entre 1.600668 *et* 2.238247.

Exemple 2.7.4. $m = 5\,, n = 11\,, \rho = 0.99\,, \varepsilon = 10^{-5}$.
$C = diag(-18, 7, -12, -5, 0, -8, 0, 0, 0, 0, 0)^t$.
$A_1 = diag(2, -6, 2, 7, 8, 1, 0, 0, 0, 0)^t$.
$A_2 = diag(-3, -1, 4, -3, 1, 2, 0, 1, 0, 0, 0)^t$.
$A_3 = diag(8, -3, 5, -2, 0, 2, 0, 0, 1, 0, 0)^t$.
$A_4 = diag(4, 0, 8, 7, -1, 3, 0, 0, 0, 1, 0)^t$.
$A_5 = diag(5, 2, -3, 6, -2, -1, 0, 0, 0, 0, 1)^t$.
$b = (1, -2, 4, 1, 5)^t$.
Après 6 itérations la solution réalisable initiale trouvée est :
$X^0 = diag(1.190999, 2.283859, 0.002792, 0.000572, 3.833378, 0.003548, 0.783124, 0.006945, 1.303703, 0.032493, 2.152547)^t$.
Après 7 itérations la solution optimale trouvée est :
$X^* = diag(2.000773, 4.002109, 0.000003, 0.000032, 7.003544, 0.000042, 0.000075, 1.000919, 0.00018, 0.000292, 0.998889)^t$.
La valeur optimale est : $z^* = -7.997726$.
Les valeurs du pas de déplacements obtenus varient entre 1.058669 *et* 7.040505.

Exemple 2.7.5. $\rho = 0.99$, $\varepsilon = 10^{-6}$.
$C = -I$, $b[i] = 2, \forall\, i = 1, ..., m$.

$$A_i[j,k] = \begin{cases} 1 & si\,(j = k = i)\; ou\,(j = k = m+1), \\ 0 & ailleurs. \end{cases}$$

Dans le tableau suivant, on donne la valeur optimale et le nombre d'itérations pour différentes tailles du problème :

Taille (m,n)	Nbr d'itérations pour trouver X^0	Nbr d'itérations pour trouver X^*	Valeur optimale
(5,10)	3	7	-10
(10,20)	3	9	-20
(20,40)	3	9	-40

Les solutions réalisables initiales trouvées sont respectivement :

Taille (5,10) : $X^0 = diag(0.317016, 0.601344, 0.836672, 1.019614, 1.157178, 1.682286, 1.398658, 1.163331, 0.980390, 0.842826)^t$.

Taille (10,20) : $X^0 = diag(0.237028, 0.458143, 0.659156, 1.230030, 1.321231, 1.398072, 1.463346, 1.762975, 1.541860, 1.340847, 1.162634, 1.008848, 0.878848, 0.769975, 0.678775, 0.601934, 0.536661)^t$.

Taille (20,40) : $X^0 = diag(0.180971, 0.354012, 0.518463, 0.673671, 0.818681, 0.952463, 1.074393, 1.184548, 1.283631, 1.372705, 1.452931, 1.525420, 1.591153, 1.650980, 1.705618, 1.755677, 1.801673, 1.844049, 1.883184, 1.919406, 1.819033, 1.645993, 1.481541, 1.326334, 1, 181324, 1.047543, 0.925613 , 0.815458, 0.716375, 0.627302, 0.547076, 0.474588, 0.408855, 0.349029, 0.294391, 0.244333, 0.198337, 0.155961, 0.116827, 0.080604)^t$.

La solution optimale trouvée pour toutes valeurs de m et n est :

$$X^*[j,k] = \begin{cases} 2 & \text{si } j = k, \\ 0 & \text{ailleurs.} \end{cases} \qquad j,k = 1,..,m,$$

Les pas de déplacements trouvés sont tels que :

Taille(5,10) : $1.291972 \leq \beta_k \leq 2.322314$.

Taille (10,20) : $1.242302 \leq \beta_k \leq 3.368356$.

Taille (20,40) : $1.786243 \leq \beta_k \leq 4.725747$.

Commentaires :

• On remarque que les résultats théoriques de l'algorithme sont confirmés, à savoir : l'algorithme offre une solution initiale strictement réalisable et une solution optimale.

• On remarque aussi que l'algorithme converge vers la solution optimale pour n'importe quel choix de la matrice $X' = \begin{pmatrix} X_0 & 0 \\ 0 & 1 \end{pmatrix}, \forall X_0 \in int(K)$.

• Les pas de déplacements trouvées sont supérieure à 1, ce qui favorise davantage l'utilisation du procédé [4].

2.8 Conclusion

La procédure qui à proposée dans [4] à réglé le problème d'initialisation au niveau de la programmation semi-définie.

D'autre part, les performances de cette méthode peuvent être améliorées en regardant de près le calcul de la projection, qui domine le coût de l'itération. A cet effet, on présente

dans le chapitre 3, une modification au niveau de la phase 1, afin d'accélérer la convergence de l'algorithme en question.

CHAPITRE 3

L'ALGORITHME PROJECTIF MODIFIÉ POUR LA PROGRAMMATION SEMI-DÉFINIE (SDP)

3.1 Introduction

Notre travail dans ce chapitre, concerne l'étude des performances de l'algorithme [4] au niveau du problème de réalisabilité d'un programme linéaire semi-défini. On propose une technique moins coûteuse en terme de calcul et en nombre d'itérations, cette technique permet d'éviter le calcul coûteu de la projection à chaque itération.

3.2 Calcul d'une solution réalisable

On cherche une $(n \times n)$ matrice symétrique, définie positive vérifiant :

$$\{tr(A_i X) = b_i\,,\ i = 1, ..., m,\ X \in int(K)\} \qquad (F).$$

Qui est équivalent au problème suivant :

$$(AP)\begin{cases} \min\lambda \\ tr(A_i X) + \lambda(b_i - tr(A_i X_0)) = b_i,\ i = 1, ..., m, \\ X \in K,\ \lambda \geq 0. \end{cases}$$

(AP) est un problème de programmation semi-définie de la forme :

$$(SDP)' \begin{cases} \min tr(C'X') \\ tr(A'_i X') = b_i,\ i = 1, ..., m, \\ X' \in K. \end{cases}$$

Où

$$X' = \begin{pmatrix} X_{n\times n} & 0_{n\times 1} \\ 0^t_{1\times n} & \lambda_{1\times 1} \end{pmatrix}, \quad A'_i = \begin{pmatrix} A_i & 0 \\ 0^t & b_i - tr(A_i X_0) \end{pmatrix}, i = 1, ..., m.$$

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

Le problème $(SDP)'$ admet une solution strictement réalisable triviale :

$$X' = \begin{pmatrix} X_0 & 0 \\ 0^t & 1 \end{pmatrix}, \forall X_0 \in int(K).$$

Donc le problème $(SDP)'$ peut être résolu par l'algorithme [4] du chapitre 2.
Supposons à l'itération k, on dispose d'une solution strictement réalisable

$$X'_k = \begin{pmatrix} X_k & 0 \\ 0^t & \lambda_k \end{pmatrix}, \textit{pour } (SDP)' \textit{ i.e.,}:$$

$$tr(A_i X_k) + \lambda_k(b_i - tr(A_i X_0)) = b_i\,, i = 1, ..., m,\ X_k \in int(K) \text{ et } \lambda_k > 0.$$

Pour calculer l'itération suivante, $X'_{k+1} = \begin{pmatrix} X_{k+1} & 0 \\ 0^t & \lambda_{k+1} \end{pmatrix}$, on cherche une matrice Z_k vérifiant :

$$\begin{cases} tr(A_i(X_k + Z_k)) = b_i\,,\ i = 1, ..., m, \\ (X_k + Z_k) \in int(K). \end{cases} \tag{3.1}$$

Notons que le produit scalaire de deux matrices symétriques A et B est définie par la trace de leurs produit i.e : $\langle A, B\rangle = tr(AB)$. Le problème (3.1) s'écrit aussi sous la forme :

$$\begin{cases} \langle A_i, X_k + Z_k\rangle = b_i,\ i = 1, ..., m \\ (X_k + Z_k) \text{ est défine positive.} \end{cases} \tag{3.2}$$

De (3.2) on a : $\langle A_i, X_k + Z_k\rangle = bi,\ i = 1, ..., m.$
Ce qui donne :

$$\langle A_i, X_k\rangle + \langle A_i, Z_k\rangle = b_i\,,\ i = 1, ..., m. \tag{3.3}$$

Comme (X_k, λ_k) est une soluion strictement réalisable de $(SDP)'$ alors :

$$\langle A_i, X_k\rangle + \lambda_k(b_i - \langle A_i, X_0\rangle) = b_i,\ i = 1, ..., m.$$

Ce qui donne : $b_i - \langle A_i, X_k\rangle + \lambda_k q_i$ où $q_i = b_i - \langle A_i, X_0\rangle \quad i = 1, ..., m.$
Finalement, l'equation(3.3) devient :

$$\langle A_i, Z_k\rangle = \lambda_k q_i,\ i = 1, ..., m. \tag{3.4}$$

La projection de l'espace $\{Z_k, \langle A_i, Z_k\rangle = \lambda_k q_i\ ,\ i = 1, ..., m\}$ sur l'origine revient à résoudre le problème de minimisation équivalent suivant :

$$\{\min\|Z_k\|^2 / \langle A_i, Z_k\rangle = \lambda_k q_i,\ i = 1, ..., m\}. \tag{3.5}$$

En utilisant les conditions d'optimalités sur la fonction lagrangienne de (3.5) :
$L(Z_k, \theta^k) = \|Z_k\|^2 + < \sum_{i=1}^{m} \theta_i^k A_i, Z_k > -\lambda_k \sum_{i=1}^{m} \theta_i^k q_i,$

Le gradient, $\nabla L(Z_k, \theta^k) = \begin{pmatrix} \frac{\partial L}{\partial Z_k} \\ \frac{\partial L}{\partial \theta_i^k} \end{pmatrix}_{i=1,...,m}$ doit être nul, on obtient alors :

$$\begin{cases} Z_k + \sum_{i=1}^{m} \theta_i^k A_i = 0, \\ \langle A_i, Z_k\rangle - \lambda_k q_j = 0,\ j = 1, ..., m. \end{cases} \tag{3.6}$$

Du système (3.6) on a :

$$Z_k = -\sum_{i=1}^{m} \theta_i^k A_i \quad \text{et} \quad \langle A_j, \sum_{i=1}^{m} \theta_i^k A_i\rangle = -\lambda_k q_i,\ j = 1, ..., m.$$

Ce qui donne :

$$\sum_{i=1}^{m} \langle A_j, A_i\rangle \theta_i^k = -\lambda_k q_j,\ j = 1, ..., m.$$

Qui est équivalent au système linéaire suivant :

$$A'\theta^k = b'. \tag{3.7}$$

Où

- $A'_{ij} = \langle A_i, A_j\rangle \quad i, j = 1, ..., m.$
- $b'_i = \lambda_k(\langle A_i, X_0\rangle - b_i),\ i = 1, ..., m.$

Propriétés du système (3.7)

- A' et une $(m \times m)$ matrice symétrique par construction car :

$$A'_{ij} = \langle A_i, A_j\rangle = \langle A_j, A_i\rangle = A'_{ji}\ , i, j = 1, ..., m.$$

• A' est définie positive car pour tout $x \neq 0$ on a :

$$\begin{aligned}
\langle A'x, x\rangle &= \sum_{i=1}^{m}\sum_{j=1}^{m} A'_{ij}x_ix_j \\
&= \sum_{i=1}^{m}\sum_{j=1}^{m} \langle A_i, A_j\rangle x_ix_j \\
&= \sum_{i=1}^{m}\sum_{j=1}^{m} \langle x_iA_i, x_jA_j\rangle \\
&= \sum_{i=1}^{m} \langle x_iA_i, \sum_{j=1}^{m} x_jA_j\rangle \\
&= \langle \sum_{i=1}^{m} x_iA_i, \sum_{j=1}^{m} x_jA_j\rangle \\
&= \langle B, B\rangle = \|B\|^2 \quad / \ B = \sum_{i=1}^{m} x_iA_i.
\end{aligned}$$

Notons que B ne peut pas être nulle, car A_i sont linéairement indépendantes. En effet, si $B = 0$, alors : $\sum_{i=1}^{m} x_iA_i = 0$ d'où $x_i = 0 \ \forall\, i$ (contradiction).
Donc $\langle A'x, x\rangle > 0$, $\forall\, x \neq 0$, d'où la matrice A' est définie positive.

Remarque 3.2.1. *:*

1. *Pour les mêmes raisons, citées dans le chapitre 1 page (7), on résoudre le système linéaire (3.7) par la factorisation de Cholesky.*
2. *Comme la matrice A' est constante à chaque itération, on résoud le système (3.7) une seule fois, pour avoir θ^0(à l'itération $k = 0$). Les autres solutions θ^k (à l'itération k) s'obtiennent facilement par la relation :*

$$\theta^k = \frac{\lambda_k}{\lambda_0}\theta^0.$$

3.3 Algorithme projectif modifié

Début

a) **Initialisation** : $\lambda_0 = 1$, $X_0 \in int(K)$ arbitraire, ε est une précision donnée, $k = 0$,
Si $\left(\sum_{i=1}^{n} |\langle A_i, X_k\rangle - b_i|^2\right)^{1/2} \leq \varepsilon$, **Stop** : X_k est une solution réalisable pour (SDP),
Sinon : Résoudre le système $A'\theta^k = b'$ avec
- $A'_{ij} = \langle A_i, A_j\rangle$, $i, j = 1, ..., m$.
- $b'_i = \langle A_i, X_0\rangle - b_i$, $i = 1, ..., m$.

b) **Si** $\lambda_k \leq \varepsilon$, **Stop** : X_k est une solution réalisable pour (SDP),
Sinon : Prendre
- $\theta_k = \lambda_k \theta^0$,
- $Z_k = -\sum_{i=1}^{m} \theta_i^k A_i$,

c) **Si** $(X_k + Z_k)$ est définie positive, **Stop :** $X_{k+1} = (X_k + Z_k)$ est une solution réalisable pour (SDP),
Sinon : Calculer l'itération (X_{k+1}, λ_{k+1}) par l'algorithme [4],

d) • Prendre $k = k + 1$ et retour à **b)**,

Fin.

3.4 Implémentation numérique

On termine ce chapitre, par des tests numériques dans un cadre comparatif entre l'algorithme [4] et l'algorithme modifié. On affiche dans un tableau le nombre d'itérations et le temps de calcul.

Exemple 3.4.1. $m = 2$, $n = 3$, $\rho = 0.99$, $\varepsilon = 10^{-4}$.

$$C = \begin{pmatrix} 1 & -1 & 1 \\ -1 & 2 & -2 \\ 1 & -2 & 2 \end{pmatrix},\ A_1 = \begin{pmatrix} 1 & -1 & 1 \\ -1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix},\ A_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix},\ b = (0, 1)^t.$$

Algorithme [4] :

Solution réalisable trouvée après 2 itérations est :

$$X^0 = \begin{pmatrix} 0.283927 & 0.070974 & -0.070974 \\ 0.070974 & 0.358069 & 0.003168 \\ -0.070974 & 0.003168 & 0.35869 \end{pmatrix}$$

Algorithme modifié :

Solution réalisable trouvée après 1 itération est :

$$X^0 = \begin{pmatrix} 0.285714 & 0.071429 & -0.071429 \\ 0.071429 & 0.357143 & 0 \\ -0.071429 & 0 & 0.355743 \end{pmatrix}$$

Exemple 3.4.2. $m = 3$, $n = 5$, $\rho = 0.99$, $\varepsilon = 10^{-4}$.

$$C = \begin{pmatrix} -4 & 0 & 0 & 0 & 0 \\ 0 & -5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, A_1 = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$A_2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, A_3 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, b = (8, 7, 3)^t.$$

Algorithme [4] :

Solution réalisable trouvée après 2 itérations est :

$$X^0 = \begin{pmatrix} 2.323819 & 0 & 0 & 0 & 0 \\ 0 & 1.883079 & 0 & 0 & 0 \\ 0 & 0 & 1.462662 & 0 & 0 \\ 0 & 0 & 0 & 0.909984 & 0 \\ 0 & 0 & 0 & 0 & 1.116902 \end{pmatrix}$$

Algorithme modifié :

Solution réalisable trouvée après 1 itération est :

$$X^0 = \begin{pmatrix} 2.2682213 & 0 & 0 & 0 & 0 \\ 0 & 1.8466151 & 0 & 0 & 0 \\ 0 & 0 & 1.615385 & 0 & 0 \\ 0 & 0 & 0 & 1.038462 & 0 \\ 0 & 0 & 0 & 0 & 1.153846 \end{pmatrix}$$

Exemple 3.4.3. $m = 3$, $n = 6$, $\rho = 0.99$, $\varepsilon = 10^{-4}$.

$$C = \begin{pmatrix} 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \; A_1 = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$A_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix}, \; A_3 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \; b = (0, 0, 1)^t.$$

Algorithme [4] :

Solution réalisable trouvée après 2 itérations est :

$$X_0 = \begin{pmatrix} 0.00658994 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.134014 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.133632 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.265743 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.133532 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.267234 \end{pmatrix}$$

Algorithme modifié :

Solution réalisable trouvée après 1 itération est :

$$X_0 = \begin{pmatrix} 0.066667 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.133333 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.133333 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.266667 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.133333 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.266667 \end{pmatrix}$$

Exemple 3.4.4. $m = 6$, $n = 12$, $\rho = 0.99$, $\varepsilon = 10^{-5}$.

$C = diag(-4, -5, -1, -3, 5, -8, 0, 0, 0, 0, 0, 0)^t$.

$A_1 = diag(1, 0, -4, 3, 1, 1, 1, 0, 0, 0, 0, 0)^t$.

$A_2 = diag(5, 3, 1, 0, -1, 3, 0, 1, 0, 0, 0, 0)^t$.

$A_3 = diag(4, 5, -3, 3, -4, 1, 0, 0, 1, 0, 0, 0)^t$.

$A_4 = diag(0, -1, 0, 2, 1, -5, 0, 0, 0, 1, 0, 0)^t$.

$A_5 = diag(-2, 1, 1, 1, 2, 2, 0, 0, 0, 0, 1, 0)^t$.
$A_6 = diag(2, -3, 2, -1, 4, 5, 0, 0, 0, 0, 0, 1)^t$.
$b = (1, 4, 4, 5, 7, 5)^t$.

Algorithme [4] :
Solution réalisable trouvée après 3 itérations est :

$$X^0 = diag(0.176545, 0.618584, 1.688245, 2.161219, 0.799969, 0.121946, 0.170873, 0.007439, 1.859922, 1.105881, 1.041210, 1.477802)^t.$$

Algorithme modifié :
Solution réalisable trouvée après 1 itération est :

$$X^0 = diag(0.171344, 0.690133, 1.585007, 1.982386, 0.844980, 0.100245, 0.276302, 0.032119, 1.951498, 1.381607, 1.194712, 1.658939)^t.$$

Exemple 3.4.5. $m = 11$, $n = 25$, $\rho = 0.99$, $\varepsilon = 10^{-5}$.
$C = diag(2, -1, -3, 5, -2, 0, 4, 1, 2, -1, 1, -1, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)^t$.
$A_1 = diag(1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)^t$.
$A_2 = diag(0, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0)^t$.
$A_3 = diag(0, 0, 3, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0)^t$.
$A_4 = diag(0, 0, 0, 2, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0)^t$.
$A_5 = diag(0, 0, 0, 0, 6, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0)^t$.
$A_6 = diag(0, 0, 0, 0, 0, -1, 2, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0)^t$.
$A_7 = diag(0, 0, 0, 0, 0, 0, 4, -1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0)^t$.
$A_8 = diag(0, 0, 0, 0, 0, 0, 0, 0, 3, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0)^t$.
$A_9 = diag(0, 0, 0, 0, 0, 0, 0, 0, 3, -1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0)^t$.
$A_{10} = diag(0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0)^t$.
$A_{11} = diag(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, l)^t$.
$b = (8, 4, 6, 2, 5, 1, 2, 6, 3, 9, 4)^t$.

Algorithme [4] :
Solution réalisable trouvée après 3 itérations est :

$$X^0 = diag(3.401705, 1.196524, 1.530009, 0.126172, 0.229663, 1.227159, 0.560081, 1.178426, 0.932970, 2.172183, 1.344368, 1.941695, 0.320980, 0.708173, 3.401735, 0.076943, 1.536143, 0.829009, 1.167709, 1.106997, 0.938102, 1.057795, 1.028907, 1.600056, 0.708173)^t.$$

Algorithme modifié :

Solution réalisable trouvée après 2 itérations est :

$$X^0 = diag(3.392266, 1.215467, 1.496910, 0.126204, 0.216688, 1.255075, 0.565353, 1.291262, 0.905959, 2.206433, 1.412867, 1.837511, 0.333978, 0.747266, 3.392266, 0.072155, 1.635473, 0.880839, 1.189721, 1.124368, 1.029848, 1.160555, 1.075690, 1.705677, 0.747266)^t.$$

Résumons les résultats trouvés dans le tableau suivant :

Exemple	Taille (m, n)	Itér1	Temps de calcul Algorithme originel (s)	Itér2	Temps de calcul Algorithme modifié (s)
1	(2,3)	2	0.47	1	0.15
2	(3,5)	3	0.94	1	0.16
3	(3,6)	2	0.62	1	0.16
4	(6,12)	3	0.235	1	0.63
5	(11,25)	3	0.797	2	0.469

Où :

- Itér1 désigne le nombre d'itérations de l'algorithme originel.
- Itér2 désigne le nombre d'itérations de l'algorithme modifié.

Exemple 3.4.6. $\rho = 0.99$, $\varepsilon = 10^{-6}$.

$$C[i,j] = \begin{cases} -1 & si\ i = j = 1, ..., m, \\ 0 & sinon\,. \end{cases}$$

$$A_i[j,k] = \begin{cases} 1 & si \quad (j = k = i)\ ou\ (j = k = m + i), \\ 0 & ailleurs. \end{cases}$$

$b[i] = 2, \forall\, i = 1, ..., m.$

***Algorithme** [4] :*

Solution réalisable trouvée après 3 itérations est :

Taille (5,10) : $X^0 = diag(0.317016, 0.601344, 0.836672, 1.019614, 1.157178, 1.682286, 1.398658, 1.163331, 0.980390, 0.842826)^t.$

Algorithme modifié :

Solution réalisable trouvée après 2 itérations est :

Taille (5,10) : $X^0 = diag(0.351604, 0.638960, 0.855135, 1.017415, 1.141264, 1.648396, 1.361040, 1.144865, 0.982585, 0.858736)^t.$

***Algorithme**[4] :*
Solution réalisable trouvée après 3 itérations est :

Taille (10,20) : $X^0 = diag(0.237028, 0.458143, 0.659156, 0.837370, 0.991157, 1.121157, 1.230030, 1.321231, 1.398072, 1.463346, 1.762975, 1.541860, 1.340847, 1.162634, 1.008848, 0.878848, 0.769975, 0.678775, 0.601934, 0.536661)^t$.

Algorithme modifié :
Solution réalisable trouvée après 2 itérations est :

Taille (10,20) : $X^0 = diag(0.246876, 0.477153, 0.676504, 0.846774, 0.991655, 1.115106, 1.220732, 1.311606, 1.390267, 1.458785, 1.753124, 1.522847, 1.323496, 1.153226, 1.008345, 0.884894, 0.779286, 0.688394, 0.609733, 0.541215)^t$.

***Algorithme** [4] :*
Solution réalisable trouvée après 3 itérations est :

Taille (20,40) : $X^0 = diag(0.180971, 0.354012, 0.518463, 0.673671, 0.818681, 0.952463, 1.074393, 1.184548, 1.283631, 1.372705, 1.452931, 1.525420, 1.591153, 1.650980, 1.705618, 1.755677, 1.801673, 1.844049, 1.883184, 1.919406, 1.819033, 1.645993, 1.481541, 1.32925613, 0.815458, 0.716375, 0.627302, 0.547076, 0.474588, 0.408855, 0.349029, 0.294391, 0.244333, 0.198337, 0.155961, 0.116827, 0.080604)^t$.

Algorithme modifié :
Solution réalisable trouvée après 2 itérations est :

Taille (20,40) : $X^0 = diag(0.175255, 0.356594, 0.524983, 0.680288, 0.822905, 0.953530, 1.073014, 1.182270, 1.282204, 1.373686, 1.457531, 1.534484, 1.605226, 1.670368, 1.730462, 1.7859, 97, 1.837415, 1.885106, 1.929420, 1.970668, 1.824745, 1.643406, 1.475017, 1.31926986, 0.817730, 0.717796, 0.626314, 0.542469, 0.465516, 0.3947740, 329632, 0.269538, 0.214003, 0.162585, 0.114894, 0.070580, 0.029332)^t$.

Résumons les résultats trouvés dans le tableau suivant :

Taille (m, n)	*Itér1*	*Temps de calcul Algorithme originel (s)*	*Itér2*	*Temps de calcul Algorithme modifié (s)*
(5,10)	*3*	*0.187*	*2*	*0.94*
(10.20)	*3*	*0.610*	*2*	*0.395*
(20.40)	*3*	*3.797*	*2*	*2.344*

Exemple 3.4.7. $\rho = 0.99$, $\varepsilon = 10^{-6}$.

$C[i,j] = -1 \ \forall i,j = 1,...,n.$

pour $k = 1,...,n-1.$

$A_n = I_n$

$$A_k[i,j] = \begin{cases} 1 & si \quad i = j = k \ et \ \ i = j = k+1 \\ -1 & si \quad i = k \ \ et \ \ \ j = k+1 \\ -1 & si \quad i = k+1 \ et \ \ j = k \\ 0 & sinon. \end{cases} \qquad i,j = 1,...,n,$$

$$b[i] = \begin{cases} 2 & si \quad i = 1,...,n-1, \\ n & si \quad i = n. \end{cases}$$

Résumons les résultats trouvés dans le tableau suivant :

Taille (m,n)	Itér1	Temps de calcul Algorithme originel (s)	Itér2	Temps de calcul Algorithme modifié (s)
(5,5)	3	0.63	2	0.46
(10.10)	3	0.235	2	0.125
(20.20)	3	0.734	2	0.515
(30.30)	3	3.32	2	1.813

3.4.1 Commentaires

Ces tests montrent clairement l'impact de notre modification sur le comportement numérique de l'algorithme, exprimé par la réduction du nombre d'itération et presque la moitié du temps de calcul.

CONCLUSION

Notre travail a été réalisé en deux étapes fondamentales :

1. La mise en œuvre d'une variante proposée par D. Benterki de l'algorithme projectif de Lustig. Les résultats obtenus sont conformes à l'etude théorique établie et traduisent les propriétés numériques souhaitées (globalité, stabilité, robustesse,...).
2. La modification de la phase 1 dans le même algorithme, en concervant les résultats théoriques tout en améliorant considérable le comportement numérique.

BIBLIOGRAPHIE

[1] **F. Alizadeh**, Interior point methods in semidefinite programming with applications to combinatorial optimization, SIAM Journal on Optimization, 5 (1995), pp. 13-51.

[2] **D. Benterki**, Sur l'étude des performances de l'algorithme de Karmarkar pour la programmation linéaire, Thèse de magister, Département de mathématique, Université de Annaba (1992).

[3] **D. Benterki**, Résolution des problèmes de programmation semi-définie par des méthodes de réduction du potentiel, Thèse de doctorat, Département de mathématique, Université Ferhat Abbas, Sétif (2004).

[4] **D. Benterki, J. P. Crouzeix, B. Merikhi**, A feasible Primal algorithm for linear semidefinite programming, Modelling, Computation and optimization in information Systems and managements sciences (2004), pp. 114-120.

[5] **D. Benterki, B. Merikhi**, A modified algorithm for the strict feasibility problem, RAIRO Oper. Res. 35 (2001), pp. 395-399.

[6] **A. Coulibaly**, Méthode de points intérieurs en programmation linéaire, Thèse de doctorat, Université Biaise Pascal, Clermont-Ferrand (1994).

[7] **M. Halicka, E. De Klerk, C. Roos**, On the convergence of the central path in semidefinite optimization, SIAM Journal on Optimization, 12 (2002), pp. 1090-1099.

[8] **J. Ji, F. A. Potra, R. Sheng**, On the local convergence of a predictor-corrector method for semidefinite programming, SIAM Journal on Optimization, 10 (1999), pp. 195-210.

[9] **N. K. Karmarkar**, A new polynomial time algorithm for linear programming, combinatorica. 4 (1984) 373-395.

[10] **A. Keraghel**, Etude adaptative et comparative des principales variantes dans l'algorithme de Karmarkar, Thèse de doctorat, Université Joseph Fourier, Grenoble, France (1989).

[11] **M. Minoux**, Programmation mathématique : théorie et algorithmes, tomel, Dunod, Paris (1983).

[12] **R. D. C. Monteiro**, Primal-dual path-following algorithms for semidefinite programming, SIAM Journal on Optimization, 7 (1997), PP. 663-678.

[13] **M. J. Todd, K. C. Toh, R. H. Tütüncü**, On the Nesterov-Todd direction in semidefinite programming, SIAM Journal on Optimization, 8 (1998), pp. 769-796.

[14] **L. Vanderbeghe, S. Boyd**, Positive definite programming, SIAM Review, 38 (1996), pp. 49-95.