République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITE MOHAMED KHIDER, BISKRA

FACULTE des SCIENCES EXACTES et des SCIENCES de la NATURE et de la VIE

DEPARTEMENT DE MATHEMATIQUES

Mémoire présenté par

Kharfallah Amina

En vue de l'obtention du Diplôme de

MASTER en Mathématiques

Option : **Analyse**

Titre

Méthodes de points intérieurs pour la programmation linéaire

Membres du Comité d'Examen

Pr.	KHLIL NACER	UMKB	Président
Dr.	KACI FATMA	UMKB	Encadreur
Dr.	ADOUANE SAIDA	UMKB	Examinateur

Juin 2021

Dédicace

J'ai l'immense plaisir de dédier ce modeste travail à :

Mes très chers parents : Ferhat et Houria

Mes frères : Oussama, Yahia et Mon Chèr Mohamed

Mes soeurs : Insaf, Asma, Nahla et Sara, Ahlem

A tous les membres de ma grande

famille,oncles, tantes, cousins et cousines.

Mes amis(es)

REMERCIEMENTS

Je remercie tout d'abord « DIEU » tout puissant de m'avoir donné la santé et le courage pour effectuer ce projet de fin d'étude dans les meilleures conditions. Comme je tiens à adresser toute la reconnaissance et ma gratitude à :

Mon promoteur : Mme . Kaci fatma pour ses précieux conseils et son suivi.

Je remercie les membres du jury Khlil Naceur et Adouane Saida d'avoir accepter d'examiner mon travail.

Enfin, je tiens à remercier tous ceux qui ont contribués de prés et de loin à la réalisation de ce travail.

Table des matières

Table des figures

Liste des tableaux

Introduction

La programmation mathématique est un ensemble des méthodes ou processus mathématiques dont le but est de trouver un optimum au moins local (on ne peut pas faire mieux avec des décisions proches) voire global (on ne peut pas faire mieux tout court, quelques soient les décisions) pour un problème décisionnel donné.

La programmation mathématique englobe des outils comme :

- La programmation linéaire (encore appelée optimisation linéaire) : elle est utilisé lorsque de grands volumes sont en jeu et lorsque les relations entre ces quantités sont linéaires.
- La programmation linéaire en nombres entiers : c'est une extension de la méthode précédente qui permet de travailler aussi sur des petites quantités entières et décisions binaires.
- La programmation non linéaire : elle est utilisée lorsque les relations entre les décisions ne peuvent pas du tout être exprimées de façon linéaire, même avec des hypothèses simplificatrices. Elle est plus générale que la programmation linéaire mais a le défaut de ne pas garantir d'avoir les meilleures décisions.
- La programmation dynamique : adaptée aux cas où le problème décisionnel possède une propriété de sous-optimalité, c'est-à-dire que des décisions bonnes pour le problème global sont aussi bonnes pour des sous-problèmes.

Depuis la formulation et le développement de la méthode du simplexe pour la résolution d'un programme linéaire vers la fin des années **40**, la programmation linéaire demeure le

modèle d'optimisation le plus utilisé par les décideurs, ce qui est certainement dù à la robustesse et la stabilité des algorithmes disponibles. Lorsque Klee et Minty (**1972**) ont trouvé un exemple qui montrait pour la première fois que la méthode du simplexe pouvait prendre un nombre exponentiel d'itérations. A cet égard, la recherche d'un algorithme polynomial pour la programmation linéaire était lancée. La réponse est venue de Khachiyan en (**1979**) lorsqu'il a montré que la méthode des ellipsoïdes (un algorithme de points intérieurs développé au début des années **70** pour la programmation non linéaire) était de complexité globale $O(n^4L)$ lorsqu'elle est appliquée à la programmation linéaire, où **L** est le nombre de bits requis pour stocker (et traiter) les données.

En (**1984**), Karmarkar a proposé un algorithme polynomial de complexité $(n^{3,5}L)$ eficace en pratique, basé sur une méthode de points intérieurs. L'intérêt pour ces méthodes principalement développées depuis les années **60** voir Dikin (**1967**) et Fiacco et McCormik **(1968)**, a connu un renouveau pour les problèmes non linéaires et a ouvert un nouveau domaine pour les problèmes linéaires. Les méthodes de points intérieurs s'avèrent eficaces pour les problèmes d'optimisation à grande taille à cause de leur caractère polynomial. Den Herty **(1994)** a classé les méthodes points intérieurs en trois catégories : méthodes affines, méthodes de réduction de potentiel et les méthodes de trajectoire centrale.

Le mémoire est organisé comme suit

Le premier chapitre contient une présentation générale de quelques notions de base nécessaires par la suite telle que l'analyse convexe (ensembles, dérivées, la convexité des ensembles et fonction convexe), la programmation mathématique, les conditions d'optimalités et la programmation linéaire.

Le deuxiéme chapitre contient l'essentiel de notre travail, nous nous intéressons à la résolution d'un programme linéaire par deux méthodes de types points intérieur, il s'agit de la méthode affine et de la méthode de karmarkar qui est de type projectif. Pour chaque méthode, nous donnons les principes de la méthode ainsi que leurs convergences.

A la fin, de ce chapitre nous présentons le solveur Matlab déstiné spécialement à la réso-

lution des programmes linéaires par les méthodes des points intérieurs.

Chapitre 1

Notions de la programmation linéaire

1.1 Notions de convexité

La notion de convexité prend deux formes : un ensemble convexe, et une fonction convexe.

Définition 1.1.1 *Un ensemble D est convexe si pour toute paire de points x et y appartenant à D alors*

$$tx \ + (1-t)y \in D \quad ,\forall\ 0 \leq t \leq 1$$

c'est-a-dire le segment de droite reliant deux points de D est entièrement contenu dans D.

- Un ensemble est dit affine si pour toute paire de points x et y appartenant à D, $t \in \mathbb{R}$ alors

$$tx \ + (1-t)y \in D$$

- Un ensemble D est un polyèdre s'il s'écrit comme suit

$$D = \{x \in \mathbb{R}^n : p_i^t x \leq \alpha_i \ , i = 1, ..., m\}$$

où p_i est un vecteur non nul de $\mathbb{R}^n$ et α_i est un scalaire pour $i = 1, ..., m$.

- Une fonction f est convexe sur un ensemble convexe D si

$$f\ (tx+(1-t)y) \leq t\ f\ (x)+(1-t)\ f\ (y), \quad \forall\ 0 \leq t \leq 1$$

- Une fonction f est (strictement) concave sur un ensemble convexe D si $(-f\ )$ est (strictement) convexe sur D.
- Le gradient d'une fonction $f : \mathbb{R}^n \to \mathbb{R}$ continûment différentiable évalué au point $x \in \mathbb{R}^n$ s'écrit

$$\nabla f(x) = (\frac{\partial f(x)}{\partial x_1}\ \frac{\partial f(x)}{\partial x_2}...\frac{\partial f(x)}{\partial x_n})^t$$

et l'élément de la i^{eme} ligne et la j^{eme} colonne de la matrice Hessienne s'écrit

$$\left[\nabla^2 f(x)\right]_{i\ j} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j}$$

Définition 1.1.2 ***(Combinaison linéaire convexe)***

Un vecteur $y \in D \subset \mathbb{R}^n$ est une combinaison linéaire convexe des points $\{x_1, ..., x_p\}$ s'il existe des coefficient réels $\lambda_i \geq 0,\ \ i \in \{1, ..., p\}$, tels que

$$y = \sum_{i=1}^{p} \lambda_i x_i \ \text{ avec } \sum_{i=1}^{p}\ \lambda_i = 1.$$

Définition 1.1.3 ***(Enveloppe convexe)***

L'enveloppe convexe d'un ensemble $S \subset \mathbb{R}^n$, est l'ensemble des points de $\mathbb{R}^n$ qui s'écrivent comme combinaison convexe des points de S.

Elle est notée par

$$conv(S) = \left\{ x \in \mathbb{R}^n : x = \sum_{i=1}^{p} \lambda_i x_i, \quad x_i \in S, \quad \lambda_i \geq 0,\ \forall i = 1, p \text{ et } \sum_{i=1}^{p} \lambda_i = 1 \right\}$$

Définition 1.1.4 ***(Point extrême)***

Soit D un convexe non vide de $\mathbb{R}^n$, x est dit point extrême ou sommet de D si

$$x = \lambda x_1 + (1-\lambda)x_2 \ \forall x_1, x_2 \in D \ et \ \lambda \in]0,1[\ alors \ x = x_1 = x_2$$

Tout point extrémal d'un convexe D est un point de la frontière de D.

Proposition 1.1.1 *Soit D un convexe non vide de $\mathbb{R}^n$ alors x est un sommet de D si et seulement si $D/\{x\}$ est un convexe.*

1.2 Ensemble et fonction affine

Définition 1.2.1 ***(Ensemble affine)***

Un sous-ensemble D de $\mathbb{R}^n$ est dit affine si

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

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

Définition 1.2.2 ***(Combinaison affine)***

Soient $x_1, ..., x_m$ des points quelconques de $\mathbb{R}^n$, on dit que $x \in \mathbb{R}^n$ est une combinaison affine de $x_1, ..., x_m$, s'il existe $(\lambda_1, \lambda_2, ..., \lambda_m) \in \mathbb{R}^m$ tels que

$$x = \sum_{i=1}^{m} \lambda_i x_i \ avec \ \sum_{i=1}^{m} \lambda_i = 1$$

Proposition 1.2.1 *Un sous-ensemble $D \in \mathbb{R}$ est dit affine si et seulement s'il contient toutes les combinaisons affines de ses éléments.*

Définition 1.2.3 ***(Fonctions affines)***

Une fonction f définie de $\mathbb{R}^n$ dans $\mathbb{R}^m$ est dite affine si

$$f\left[(1-\lambda)x+\lambda y\right]=(1-\lambda)\ f(x)+\lambda\ f(y),\ \forall\ x,y\in\mathbb{R}^n,\ \forall\ \lambda\in\mathbb{R}.$$

Théorème Toute fonction affine f de $\mathbb{R}^n$ dans $\mathbb{R}^m$ est de la forme $f\ (x)=Ax+b$, où $A\in$ à l'ensemble des matrices réelles de taille $(m\times n)M_{m,n}$ et $b\in\mathbb{R}^m$.

Proposition 1.2.2 *Une fonction affine f de $\mathbb{R}^n$ dans $\mathbb{R}^m$ est linéaire si et seulement si $f(0)=0$.*

1.3 Programmation mathématique

La programmation mathématique, et plus particulièrement l'optimisation vise à résoudre des problèmes ou l'on cherche à déterminer parmi un grand nombre de solution candidates celles qui donne le meilleur rendement.

D'une façon générale un programme mathématique dans $\mathbb{R}^n$ est un problème d'optimisation sous la forme suivante

$$(P1)\begin{cases} Minf(x)\text{ où }Maxf(x)\\ s.c.\\ g_i(x)\leq 0,\ i:\ 1..m\\ h_j(x)=0,\ j:\ 1..p\\ x\in s\subseteq\mathbb{R}^n \end{cases}$$

où $f:\mathbb{R}^n\longrightarrow\mathbb{R}$ est une fonction continument différentiable appelée fonction objectif, et l'ensemble des conditions $\{g_i(x)\leq 0,\ i:1..m;\ h_j(x)=0,\ j:1..p,\ x\in s\subseteq\mathbb{R}^n\}$ sont les contraintes du problème $(P1)$.

L'ensemble $D=\{x\in s\subseteq\mathbb{R}^n/g_i\leq 0,\ i:1..m;h(x)=0,\ j:1..p\}$ est appelé l'ensemble des solutions réalisable du problème $(P1)$

1.3.1 Notion de base

Définition 1.3.1 *On appelle solution réalisable du problème (P1), tout point vérifiant les contraintes de ce problème (P1) (appartenant à D)*

Définition 1.3.2 *Un point $x^* \in D$ qui minimise la fonction objectif sur D dit solution optimale du problème (P1) si et seulement si*

$$\forall x \in D, f(x^*) \leq f(x)$$

On note par $argmin_D f(x)$, l'ensemble des solutions optimales du problème .

Définition 1.3.3 *Un point $x^* \in D$ est une solution optimale locale de (P1), s'il existe un voisinage V de x^*, et un $\varepsilon \geq 0$ tel que*

$$\begin{cases} f(x^*) \leq f(x), \forall x \in V(x^*) \\ où \\ V(x^*) = \{x \in D / \parallel x - x^* \parallel < \varepsilon\} \end{cases}$$

Définition 1.3.4 *(x appartient au voisinage de $V(x^*)$) et on note par $locmin_D f(x)$ l'ensemble des solutions optimales locales de $(P1)$*

Et de plus si f et D sont convexes, le minimum local est global pour le problème$(P1)$

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

- soit à chercher un point optimal (local, global).
- soit, si un tel point n'existe pas on cherche une borne inférieure à la fonction f.
- soit, à établir que f est non borné inférieurment sur D, auquel cas on adopte la convention $\inf_D f(x) = -\infty$
- lorsque D est vide on pose par convention $\inf_D f(x) = +\infty$

La classification de (PM) et son traitement numérique sont établis a partir des propriétés fondamentales des fonctions f, g_i et h_j à savoir la convexité, la différentiabilité et la linéarité. Parmi les cas particuliers les plus étudiés on note

- La programmation linéaire (f linéaire, g_i, h_j affine, D ortant positif).
- La programmation convexe (f, g_i, convexe, h_j affine, D convexe).
- La programmation en nombres entiers (D discret).

1.4 Existence et unicité

Théorèm 1.4.1 [1] **(Weierstrass)**

Si D est un compact non vide de $\mathbb{R}^n$ et si f est continue sur D alors le problème mathématique (PM) admet au moin une solution optimale globale $x^* \in D$.

Théorèm 1.4.2 [1]

Si D est non vide et fermé de $\mathbb{R}^n$, f est continue et coercive sur D, (c'est-a-dire $\lim\limits_{\|x\| \to +\infty} f(x) = +\infty$ alors (PM) admet au moins une solution optimale globale.

Théorèm 1.4.3 [8] (Karush-kuhn-tucker. contraintes linéaires)

Soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction différentiable sur D . Si x un minimum local du problème mathématique, alors, il éxiste un vecteur $y \in \mathbb{R}^m$ et $\lambda \in \mathbb{R}^n_+$ tel que :

$$\begin{cases} \nabla f(x^*) + \sum\limits_{i=1}^{n} \lambda_i \nabla g_i(x^*) + \sum\limits_{j=1}^{m} y_j \nabla h_j(x^*) = 0 \\ \lambda_i g_i(x^*) = 0, \ i = 1, ..., n \\ y_j h_j(x^*) = 0, \ j = 1, ..., m \end{cases}$$

1.5 Programmation linéaire

La programmation linéaire est un cadre mathématique générale permettant de modéliser et de résoudre certains problèmes d'optimisation.

historiquement, la programmtion linéaire à été développée et utilisée en 1947 par George Bernard Dazing, Marshall Wood et leurs collaborateurs au USA departement of the Air Force.

Définition 1.5.1 (2) *On considère le programme linéaire primal (P)*

$$(P)\begin{cases} \min\limits_{x} c^T x \\ s.c.\ Ax = b \\ x \geq 0 \end{cases}$$

Ainsi formulé, on dira qu'un programme linéaire est sous forme standard. Notons que tout programme linéaire c'est -à-dire tout problème d'optimisation où la fonction objectif et les fonctions définissant les contraintes sont affines, peut se mettre sous la forme standard par des manipulations algébriques simples. La matrice A est de dimension $m \times n$ avec $m \leq n$. On pose l'hypothèse que la matrice A est de plein rang. Sur des modélisations pratiques, cette hypothèse est rarement satisfaite. La matrice A peut toutefois être réduite à une matrice de plein rang par une décomposition QR ou une élimination de Gauss. L'ensemble des solutions réalisables

$$S = \{x \in \Re^n Ax = b, x \geq 0\}$$

Forme un polyèdre convexe, c'est-à-dire l'intersection d'un nombre fini de demi-espaces. Un programme linéaire est réalisable si l'ensemble S est non vide. Un programme réalisable est borné si l'objectif est borné sur S. On dit que x est un point extrème de S, s'il ne peut pas s'écrire comme une combinaison convexe stricte deux points de S. On présente ici, sans preuve, les principaux résultats de la programmation linéaire. Le lecteur les retrouvera dans l'excellente monographie de Chvàtal (1983).

Remarque : Un programme linéaire sans contrainte d'inégalité n'est d'aucun intérèt.

1.6 Formes usuelles d'un programme linéaire

Les programmes linéaires (PL) se présentent sous deux formes différentes

1.6.1 Forme standard

Un programme linéaire (PL) est sous forme standard si toutes les contraintes sont des égalités, c'est-à-dire qu'il s'écrit sous la forme

$$(PL)\begin{cases} Opt\ (c^T x) \\ sous\ les\ contraintes \\ Ax = b \\ x \geq 0. \end{cases}$$

1.6.2 Forme canonique

Un programme linéaire (PL) est sous forme canonique si toutes les contraintes sont des inégalités, c'est-à-dire qu'il s'écrit sous la forme :

$$(PL)\begin{cases} Opt\ (c^T x) \\ sous\ les\ contraintes \\ Ax(\leq ou \geq)b \\ x \geq 0. \end{cases}$$

1.7 Exemple de Modèlisation

Problème [5] la direction d'une usine de meubles a constaté qu'il y a des temps morts dans chacun des départements de l'usine. Pour remédier à cette situation, elle décide d'utiliser ces temps morts pour fabriquer deux nouveaux modèles de bureaux, M1 et M2. Les temps de réalisation pour chacun de ces modèles dans les ateliers de sciage, d'assemblage et de sablage ainsi que les temps libres dans chacun de ces atteliers sont donnés dans le

TAB. 1.1 – Table Caption

	M1	M2	Temps libres
Sciage	1	2	20
Assemblage	2	1	22
Sablage	1	1	12

tableau ci-dessous. Ces temps représentent le nombre d'heures nécessaires à un homme pour effectuer le travail. Les profits que la compagnie peut réaliser pour chacun de ces modèles sont de 300€ pour M1 et 200€ pour M2 .

La direction déterminer combien de bureaux de chaque modèle elle doit fabriquer pour maximiser son profit .

Solution 1.7.1 *Posons x_1 : le nombre de bureaux du modèle M1*

x_2 : le nombre de bureaux d'un modèle M2

les temps libres de chaque département imposent des contraintes qu'il faut respecter.

les contraintes est

$$\begin{cases} x_1 + 2x_2 \leq 20 \\ 2x_1 + x_2 \leq 22 \\ x_1 + x_2 \leq 12 \end{cases}$$

IL s'ajoute à ces contraintes des contraintes de non-négativité puisque le nombre de bureaux ne peut étre négatif, on a donc

$$x_1 \geq 0 \; et \; x_2 \geq 0$$

1.8 Résolution par voie graphique

Prenons notre exemple de modèlisation

Graphiquement les solutions réalisables sont du polygone convexe de la figure suivante

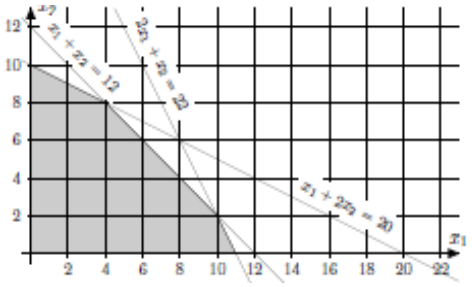

FIG. 1.1 – Figure 1 :Résolution graphique

La direction veut maximiser son profit, c'est à dire maximiser la fonction

$$f(x_1, x_2) = 300x_1 + 200x_2$$

Pour chacune des ces solutions, c'est à dire pour chacune des points du polygone convex, la compagne fera un profit positif si la compagne fabrique trois exemplaires d'un modèle M2, le profit sera

$$f(3, 2) = 300 \cdot 3 + 200 \cdot 2 = 1300(€)$$

IL ne saurait être question de calculer le profit réalisable pour chacun des points du polygone convexe pour avoir une vision globale du problème, représentons le profit réalisé par le paramètre P on a

$$300x_1 + 200x_2 = P$$

qui réprésente une famille de droite parallèles. En isolant x_2 on obtient

$$x_2 = -\frac{300}{200}x_1 + \frac{P}{200}$$
$$x_2 = -\frac{3}{2}x_1 + \frac{p}{200}$$

IL s'agit donc d'une famille de droites de pente $\frac{-3}{2}$ et dont l'ordonnée à l'origine est $\frac{P}{200}$. parmi les droites de cette famille, seules celles ayant des points communs avec le polygone convexe nous intéressent la fonction $f(x_1, x_2)$ atteindra sa valeur maximale lorsque

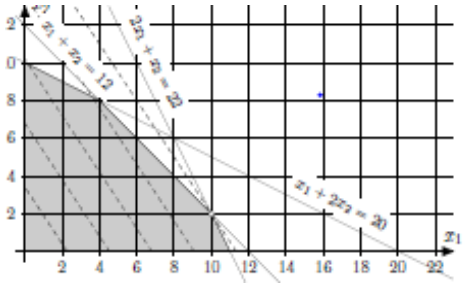

FIG. 1.2 – Figure 2 :Résolution graphique

l'ordonnée à l'origine $\frac{P}{200}$ de la droite

$$x_2 = -\frac{3}{2}x_1 + \frac{p}{200}$$

Atteindra sa valeur maximaum tout en passant par au moins un des points du polygone convexe.

Graphiquement on constate que la droite respectant ces conditions semble être la droite de la famille passant par le point sommet $(10; 2)$. Le profit est alors

$$f(10, 2) = 300.10 + 200.2 = 3400(€)$$

IL reste à sassurer algébriquement des coordonnées du points sommet en résolvant le système

$$\begin{cases} 2x_1 + x_2 = 22 \\ x_1 + x_2 = 12 \end{cases} \boxed{\Rightarrow} \begin{cases} x_1 = 10 \\ x_2 = 2 \end{cases}$$

1.9 Dualité en programmation linéaire

La notion de dualité est un concept fondamental en programmation linéaire qui conduit à un résultat de grande portée théorique et pratique (le théorème de dualité faible et le théorème de dualité forte). Ainsi, étant donné un problème de programmation linéaire (PL) appelé primal, on peut toujours lui associer un autre problème appelé dual (noté

par (DL)).

Soit le problème primal

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

Son dual s'écrit sous la forme

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

Les solutions des programmes primal et dual sont liées par les théorèmes de la dualité faible et forte.

Théorème 1.9.1 [7] (Dualité faible)

Si x et y sont des solutions réalisables de problèmes (PL) et (DL) respectivement, alors

$$c^t x > b^t y$$

Preuve

On a x et y sont réalisables, alors du problème (DL) on a

$$c \geq A^t y \Rightarrow c^t x \geq y^t A x = (b^t y)^t = b^t y$$

Théorème 1.9.2 [7] (Dualité forte)

Si x^* et y^* sont des solutions réalisables de (PL) et (DL) respectivement tels que

$$b^t y^* = c^t x^*$$

alors x^* et y^* sont des solutions optimales des problèmes (PL) et (DL) respectivement .

Preuve

On veut démontrer que x^* et y^* sont des solutions optimales des problèmes (PL) et (DL) respectivement c'est- a-dire, on démontre que

$$c^t x^* = min c^t x \; et \; b^t y^* = max b^t y.$$

D'aprés le théorème précédent on a

$$c^t x \geq b^t y \; donc \; min c^t x \geq max b^t y$$

en particulier,

$$c^t x^* \geq min c^t x \geq max b^t y \geq b^t y^*.$$

Et comme

$$b^t y^* = c^t x^*$$

alors

$$\begin{cases} c^t x^* \geq min \; c^t x \geq max \; b^t y \geq c^t x^* \Rightarrow c^t x^* \geq min \; c^t x \geq c^t x^* \\ \qquad\qquad et \\ b^t y^* \geq min \; c^t x \geq max \; b^t y \geq b^t y^* \Rightarrow b^t y^* \geq max \; b^t y \geq b^t y^*. \end{cases}$$

D'où le résultat.

Chapitre 2

Méthodes de points intérieurs

2.1 Méthode de résolution d'un programme linéaire

2.1.1 Méthode du simplexe

Elle a été développée à la fin des années 40 par G.Dantzig [4]. Elle tient compte systématiquement des résultats établis précédemment. Elle évolue sur la frontière du domaine réalisable de sommet en sommet adjacent, en réduisant la valeur de l'objectif jusqu'à l'optimum. Un critère d'optimalité simple permet de reconnaître le sommet optimal. Le nombre de sommet étant fini, l'algorithme ainsi défini converge en un nombre fini d'itérations n'excédant pas le nombre $C_n^m = \frac{n!}{m(n-m)!}$ sous l'hypothèse que tous les sommets visités sont non dégénérés.

En général, la méthode du simplexe possède un comportement numérique très satisfaisant confirmé par ses applications multiples dans la résolution d'une large classe de problèmes pratiques.

2.1.2 Méthodes modernes de points intérieurs

Ces méthodes ont été développées dans les années 60. Leur utilisation pour la programmation linéaire n'a pas reçu autant d'enthousiasme à cause de la dominance quasi totale de la méthode du simplexe à cette époque. Après l'apparition de l'algorithme de Karmarkar en 1984 pour la programmation linéaire, les méthodes de points intérieurs ont connu une véritable révolution, on enregistre plus de 3000 publications en quelques années. On distingue trois classes fondamentales de méthodes de points intérieurs à savoir :

1. Méthodes afinnes (Dikin)
2. Méthodes de réduction du potentiel
3. Méthodes de trajectoire centrale (TC)

2.2 Méthode affine

L'idée remonte a Dikin (1967). On présente la méthode affine primale comme introduction aux méthodes de points intérieurs. Cette méthode se caractérise par sa simplicité et sa relative efficacité pratique. Par contre, la complexité polynomiale n'est pas connue pour l'approche primale (ou duale), alors qu'elle est établie pour l'approche primale-duale. De plus l'analyse de convergence est davantage complexe que pour les algorithmes de chemin central. L'idée consiste à utiliser à partir d'un point intérieur la direction projetée de plus forte descente dans un espace transformé. La direction de plus forte descente est parmi toutes les directions réalisables celle qui donne le plus grand taux de décroissance. Cette propriété est utile si le point courant se trouve éloigné des frontières définies par les bornes de non négativité. Dans le cas contraire, le pas effectué dans cette direction sera petit donnant une décroissance nette petite de l'objectif. Ainsi, l'approche consiste à calculer une direction de descente qui n'approche pas trop rapidement de la frontière. Cette direction de plus forte descente est calculée à partir d'un problème mis à l'échelle qui centre la solution courante et qui est ensuite transformée dans l'espace original. L'algorithme

comprend les trois étapes suivantes :

1. calcul de la direction de descente
2. calcul de pas
3. calcul de la tranformation affine

On considère le programme linéaire suivant

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

Avec la matrice A de plein rang. On note S^0 l'intérieur de S c'est à dire

$$S^0 = \{x \in \Re^n : Ax = b, x > 0\}$$

On suppose que l'ensemble S^0 n'est pas vide.

Calcul de la direction de descente

Soit x_c le point courant tel que

$$\begin{cases} Ax_c = b \\ x_c > 0 \end{cases}$$

On cherche

$$x^+ = x_c + \alpha \Delta x$$

tel que

$$\begin{cases} c^T x^+ \ \leq \ c^T x_c \\ Ax^+ = b, \quad x^+ > 0 \end{cases}$$

Le déplacement doit donc vérifier

$$\begin{cases} c^t \Delta x \leq 0 \\ Ax^+ = A(x_c + \alpha \Delta x) = b \end{cases}$$

Sous l'hypothèse que $\alpha > 0$, Δx doit étre dans le noyau de A c'est à dire

$$\Delta x \in N(A) = \{x \in \Re^n / Ax = 0\}$$

La direction de plus forte descente est donnée par

$$\begin{cases} \min\limits_{\Delta x} c^T \Delta x \\ s.c. A\Delta x = 0 \\ \|\Delta\| = 1 \end{cases}$$

dont la solution est

$$\Delta x = \frac{proj_A(-c)}{\|proj_A(-c)\|}$$

où $Proj_A()$ étant la matrice orthogonale de projection sur le noyau de A. Comme A est de plein rang et en oubliant la normalisation, on a

$$\begin{aligned} \Delta x &= proj_A(-c) \\ &= -(I - A^T(AA^T)^{-1}A)c. \end{aligned}$$

Remarque 2.2.1 *Une matrice orthogonale de projection P sur le noyau de A vérifie les*

propriétés suivantes

$$AP = 0$$
$$P = P^T$$
$$P^2 = P$$

Il est alors facile de vérifier que

$$\begin{aligned} c^T \Delta x &= -c^T proj_A(c) \\ &= -c^T proj_A^2(c) \\ &= -\|proj_A(c)\|^2 \leq 0 \end{aligned}$$

Longueur de pas

Puisque nous sommes dans le cas linéaire, le taux de décroissance est constant dans la direction Δx. Le pas maximal est limité par les contraintes de non négativité. De plus, la transformation affine qui nous permettra de centrer le point n'est pas définie sur la frontière. Donc il suffit de choisir

$$\alpha = \gamma \alpha_{max} \; avec \; 0 < \gamma < 1,$$

et

$$\alpha_{max} = \min_{\Delta x_i < 0} \frac{-(x_c)_i}{\Delta x_i}$$

En pratique, on choisit $\gamma = 0.995$. Si $\Delta x \geq 0$ et $\Delta x \neq 0$, le problème est alors non borné.

Transformation affine

On fait une mise à l'échelle afin que le nouveau point x^+ soit loin de la frontière définie par $x \geq 0$. Un point idéal serait le vecteur unitaire e. Ainsi, on cherche une x transformation affine qui transforme x^+ en e. La matrice de transformation est simplement l'inverse de

la matrice diagonale dont les composantes sont les mêmes que celles de x^+. Cette matrice est inversible puisque $x^+ > 0$. Notons cette matrice par X.

On a alors

$$\begin{cases} X^{-1}x^+ = e \\ X^{-1}x = \bar{x}. \end{cases}$$

Le programme linéaire devient (dans l'espace transformé)

$$\begin{cases} \min\limits_{\bar{x}} c^T X\ \bar{x} = \bar{c}^T \bar{x} \\ s.c.\ AX\ \bar{x} = \bar{A}\bar{x} = b \\ \bar{x} \geq 0. \end{cases}$$

Dans l'espace $\bar{x}$ et étant donné le nouveau point courant $\bar{x}_c = e$, on obtient :

$$\begin{aligned} \Delta\bar{x} &= proj_{\bar{A}}(-\bar{c}) \\ &= -(I - A^{-t}(AA^t)^{-1}\bar{A})\bar{c} \\ &= (I - XA^t(AX^2A^t)^{-1}AX)Xc \end{aligned}$$

Et

$$\bar{x}^+ = \bar{x}_c + \alpha\Delta\bar{x}$$

1. alors

$$\begin{aligned} x^+ &= X\bar{x}^+ \\ &= x_c + \alpha X\Delta\bar{x} \\ &= x_c - \alpha X(I - XA^t(AX^2A^t)^{-1}AX)Xc \end{aligned}$$

2.2.1 Propriétés et commentaires de la méthode affine

1. C'est une méthode primale, performante en pratique, quoique sensible au choix initial.
2. Elle est simple au sens qu'elle ne demande ni transformation projective ni fonction potentiel, ni même la préparation du problème (forme canonique).
3. La démonstration de la convergence est relativement simple dans le cas dégénéré,elle est fastidieuse en cas d'échéant .
4. Il n y'a pas de résultat de polynomialité, on pense plutôt que l'algorithme est d'une complexité exponentielle.

2.2.2 Algorithme de la méthode affine primale

Initialisation x_0 le point initial, $0 < y < 1$, Δ_c la décroissance de la fonction objective, ε le critére d'arrêt ;

Début

$x_c := x_0$;

$\Delta_c := \varepsilon(c^T x_c) + 1$;

Tant que $\frac{\boldsymbol{\Delta_c}}{\boldsymbol{c^T x_c}} > \varepsilon$ **faire**

$X := X_c$;

$\Delta_x := -X(I - XA^T(AX^2A^t)^{-1}AX)Xc$;

$\alpha = \gamma \underset{\Delta_{x_i}<0}{min} \frac{-(x_c)_i}{\Delta x_i}$

$x^+ := x_c + \alpha\Delta x$

$\Delta x := c^T x_c - c^T x^+$;

$x_c := x^+$;

fin Tan que

fin

2.2.3 Convergence

On a le résultat suivant

Théorème [9] Sous les hypothèses que A est une matrice de plein rang, qu'il existe un point strictement intérieur et que la fonction objectif est non constante sur le domaine réalisable, alors

1. Si le problème primal et le problème dual sont non dégénérés, alors pour tout $\gamma < 1$, la suite générée par l'algorithme converge vers une solution optimale.
2. Pour $\gamma \leq 2/3$, la suite générée par l'algorithme converge vers une solution optimale.

2.2.4 Critère d'arrêt

Le critère d'arrêt devrait être relié à la satisfaction des conditions d'optimalité.

Considérons d'abord le cas non dégnéré (primal et dual) et soit le programme dual (DL)

$$(DL)\begin{cases} \underset{y,s}{max}\ b^T y \\ s.c.\ A^T y + s = c \\ y \in \mathbb{R}^m,\ s \geq 0 \end{cases}$$

Soit x_k la solution courante et considérons le programme

$$\begin{cases} \underset{y,c}{max} \|X_k s\| \\ s.c.\ A^T y + s = c \\ y \in \mathbb{R}^m,\ s \geq 0 \end{cases}$$

La solution de ce programme est donnée par

$$\begin{cases} y_k = (AX_k^2 A^T)^{-1} AX_k^2 c \\ s_k = c - A^T y_k \end{cases}$$

On en déduit

$$\nabla x_k = -X_k^2 s_k$$

et le vecteur dual est obtenu comme sous-produit du calcul de la direction de descente.

Théorème Sous l'hypothèse de non dégénérescence primales et duales, la solution (y_k, s_k) converge vers la solution optimale duale du problème (DL) lorsque x_k converge vers la solution optimale primale du problème primale (PL).

Dans ce cas, un critère d'arrêt peut être défini sur la norme du vecteur de complémentarité.

2.3 Méthode Karmarkar

En 1984 Karmarkar propose un algorithme de résolution d'un programme linéaire de la forme

$$\omega^* = min\{c^T x : Ax = 0, x \in S_n\} \qquad (KP)$$

pour lequel on connait à priori la valeur optimale $\omega^* = 0$ et une solution réalisable $\alpha = \frac{e_n}{n}$ (centre du simplexe S_n). A est une matrice de $\Re^{m \times n}$ et de rang plein.

$S_n = \left\{ x \in \Re^n : x \geq 0, \sum_{i=1}^{n} x_i = 1 \right\}$ est le simplexe de dimension $(n-1)$

On suppose que :

(Hypothèse 1) la matrice A est de plein rang $(rgA = m < n)$.

(Hypothèse 2) On dispose d'un point x^0 strictement réalisable $(Ax^0 = b, x^0 > 0)$.

(Hypothèse 3) La valeur optimale est connue au départ.

2.3.1 Principe de la méthode

A partir d'une solution initiale $x^0 = a$ l'algorithme construit une suite de points intérieurs qui converge vers la solution optimale du problème. Dans le but de ramener la valeur objectif à zéro on le minimise localement sur une sphère inscrite dans l'orthant réalisable. Pour chaque itération k, l'itéré $x^k > 0$ est ramené au centre de simplexe S_n pour la

transformation suivante

$$T_k : S_n \to S_n$$

$$x \mapsto T_k(x) = y$$

où

$$T_k(x) = \frac{D_k^{-1}x}{e_n^t D_k^{-1}x} = y \ et \ T_k^{-1}(y) = \frac{D_k y}{e_n^t D_k y} \ avec \ D_k = diag(x^k)$$

Le problème (KP) s'écrit comme suit

$$min\left\{\frac{c^t D_k\ y}{e_n^t D_k\ y} : \frac{AD_k\ y}{e_n^t D_k y} = 0, \sum_{i=1}^{n} y_i = 1, y > 0\right\}$$

Les hypothèses de départ permettent d'obtenir le programme de Karmarkar simplifié

$$min\left\{c^t D_k\ y : AD_k y = 0, \sum_{i=1}^{n} y_i = 1, y > 0\right\} \qquad (01)$$

Pour la résolution de ce problème on a le Lemme suivant

Lemme 2.3.1 *Si pour un programme linéaire donné on connait une solution réalisable* y^0 *tel que* $(\ y_i^0 > 0, i = 1, ..., n+1)$

alors l'ellipsoide :

$$ell = \left\{y \in \Re^{n+1} : \sum_{i=1}^{n+1} \frac{(\ y_i - \boldsymbol{y}_i^0)^2}{(y_i^0)^2} \leq \beta^2, 0 < \beta < 1\right\}$$

est dans l'intérieur de l'orthant positif de $\mathbb{R}^{n+1}$

D'après le Lemme, si on ajoute au problème (01) la contrainte :

$$y \in \mathbb{R}^n : \|y - \alpha\| \leq \alpha r \ où \ 0 < \alpha < 1 \ et \ r = \frac{1}{\sqrt{n(n-1)}}$$

On obtient alors le problème suivant

$$min\left\{c^t D_k\ y : AD_k y = 0, \sum_{i=1}^{n} y_i = 1, \|y-\alpha\|^2 \leq (\alpha r)^2\right\} \qquad (02)$$

Ceci est un problème d'optimisation sur une sphère, dont la solution optimale est donnée à l'aide du théorème suivant

Théorèm 2.3.1 La solution optimale du problème (02) est donnée explicitement par

$$y^k = \alpha - ard^k,\ où\ d^k = \frac{p^k}{\|p^k\|}\ avec\ p^k = p_{B_k}(D_k c),\ B_k = \begin{bmatrix} AD_k \\ e_n^t \end{bmatrix}$$

A chaque itération k, on revient à la variable initiale x en appliquant la transformation inverse T_k^{-1} et ainsi de suite jusqu'à ce que le test d'optimalité ($c^t x^k \leq \varepsilon$) soit vérifie.

2.3.2 Algorithme de Karmarkar

Début Algorithme

Initialisation

$\varepsilon > 0$ est un paramètre précision

$x^0 = \alpha = \frac{e_n}{n}$ est une solution initiale, $e_n = (1, ..., 1) \in \mathbb{R}^n$

$k = 0$

Tant que ($c^t x^k \geq \varepsilon$) **Faire**

1 Construire

$$D_k = diag(x^k),\ A_k = AD_k,\ B_k = \begin{bmatrix} A_k \\ e_n^t \end{bmatrix}$$

2 Calculer

$p^k = p_{B_k}(D_k c) = [I - B_k^t(B_k B_k^t)^{-1} - B_k]\, D_k c$, (la projection sur le noyau de B_k)

$d^k = \frac{\boldsymbol{p}^k}{\|\boldsymbol{p}^k\|}$

$y^k = \alpha - ard^k\ où\ r = \frac{1}{\sqrt{n(n-1)}},\ 0 < \alpha < 1$

3 Prendre

$$x^{k+1} = T_k^{-1}(y^k) = \frac{D_k\ y^k}{e_n^t D_k\ y^k}$$

$$k = k + 1$$

Fin Tant que

Fin Algorithme

2.3.3 Etude de la convergence

Théorème 2.3.1. [6] A chaque itération k, le point y^k vérifie

$$\frac{c^t D_k y^k}{c^t D_k \alpha} \leq 1 - \frac{\alpha}{n-1}$$

Pour établir la convergence de l'algorithme, Karmarkar introduit à l'objectif la fonction potentiel suivante$f(x) = \sum_{i=1}^{n} ln(\frac{c^t x}{x_i})$, définie sur : $\left\{ x \in \mathbb{R}^n :\ x > 0,\ Ax = 0, \sum_{i=1}^{n} x_i = 1 \right\}$.

Lemme 2.3.2 *Soit x^k le $k^{iéme}$ itéré de l'algorithme, alors*$\frac{c^t x^k}{c^t x^0} \leq (exp(f(x^k) - f(x^0))^{\frac{1}{n}}$.

Karmarkar montre que la convergence de l'algorithme est réalisée en $O(nq + n\ ln\ n)$ itérations pour $0 < \alpha < \frac{1}{4}$

Théorème 2.3.2. [6] Si $0 < \alpha < \frac{1}{4}$ et $x^0 = \frac{e_n}{n}$, l'algorithme calcule la solution optimale après $O(nq + n\ ln\ n)$ itérations tel que

1. $c^t x^k = 0$
2. $\frac{c^t x^k}{c^t x^0} \leq \varepsilon = 2^{-q}$ où q est une précision fixée.

2.4 Optimisation linéaire avec Matlab

La commande **linprog** est un solveur qui permet de trouver le minimum d'un programme linéaire. Avant d'utiliser la commande, les contraintes doivent être écrites sous la forme $\leq$

ou = .

La forme complète de la commande **linprog** est :

[x,fval,exitflag,output,lambda] = *linprog* (c, A, b, Aeq, beq, lb, ub, x0,options)

Les sorties renvoyées sont :

x : La solution optimale du problème.

fval : La valeur optimale de la fonction objective.

exitflag : Si la valeur de exitflag est supérieure ou égale à 1 ceci signifie que l'algorithme a bien convergé, sinon l'algorithme n'a pas convergé.

output: Contient des informations sur le processus d'optimisation.

lambda: Contient le multiplicateur de lagrange a la solution x.

Les arguments en entrées sont :

c : Le coefficient de la fonction objective (sous forme d'un vecteur ligne).

A : Les coefficients des contraintes $\leq$ (sous forme d'une matrice).

b : Les deuxièmes membres des contraintes$\leq$ (sous forme d'un vecteur colonne).

Aeq : Les coefficients des contraintes = (sous forme d'une matrice).

beq : Les deuxièmes membres des contraintes = (sous forme d'un vecteur colonne).

lb, ub : La borne sup et inf des variables x_i (sous forme d'un vecteur colonne).

x0 : Les valeurs initiales des variables des décisions.

options : Contient des paramètres qui décrivent comment *linprog* doit s'exécuter.

Remarque : Le problème peut ne pas impliquer tous les paramètres de la comande *linprog*, donc on devrait utiliser des vecteurs vides [ ] au lieu de paramètres non utilisés.

Avant d'exécuter *linprog*, on peut spécifier comment il doit s'exécuter, en manipulant les options paramètre. Par exemple, si on veut forcer *linprog* a utiliser l'algorithme simplexe, alors on écrit

options = **optimoptions**('*linprog*', '*Algorithm*', '*simplex*') ;

Et si on veut forcer *linprog* à utiliser l'algorithme de point intérieur, on écrit

options = optimoptions('*linprog*', '*Algorithm*', '*interior − point*') ;

Exemple 2.4.1 *On considère le programme linéaire suivant*

$$\begin{cases} 2x_1 + 6x_2 = Z(Min) \\ x_1 + x_2 \leq 40 \\ x_1 - x_2 = 30 \\ x_1 - 4x_2 \leq 160 \\ x_1 \geq 0 \\ x_2 \geq 0 \end{cases}$$

On cherche la solution optimale de ce problème par la commande *linprog*, on écrit alors

c=[2 6] ;

A=[1 1 ; 1 -4] ;

b=[40 ;160] ;

Aeq=[1 -1] ;

beq=[30] ;

lb=[0 ;0] ;

ub=[] ;

x0=[0 ;0] ;

Options = Optimoptions('*linprog*', '*Algorithm*', '*interior − point*') ;

[x,fval,exitflag,output]=*linprog* (*c*, *A*, *b*, *Aeq*, *beq*, *lb*, *ub*, x0,options).

Le résultat s'affiche comme suit

The interior-point algorithm uses a built-in starting point ; ignoring user-supplied X0.

Optimization terminated.

x =

30.0000

0.0000

fval =

60.0000

exitflag =

1

output =

iterations : 4

algorithm : 'interior-point'

cgiterations : 0

message :'Optimization terminated'.

constrviolation : 3.5527e-15

firstorderopt : 1.7308e-07

La solution optimale obtenue par la méthode de point intérieur après 4 itérations est $x_1 = 30$ et $x_2 = 0$.

La valeur de la fonction objective est 60. La méthode utilisé est convergente puisque exitflag=1.

Conclusion

Les méthodes de points intérieurs sont connues par leur efficacité, rapidité de convergence, simplicité algorithmique et capacité de résoudre des problèmes de grandes tailles. L'inconvénient principal dans ce type de méthodes est l'initialisation, c'est-à-dire la détermination d'un point initial qui se trouve à l'intérieur du domaine réalisable.

Théoriquement, on suppose que ce point est connu, mais numériquement l'obtention de ce point initial présente un handicap majeur pour ces méthodes importantes.

La recherche de ce point initial pour la méthode affine nécessite par exemple l'intégration des méthodes de type projectif, la forme réduit de la méthode de Karmarkar .

Bibliographie

[1] Bazarra, S. (1993) H.D. Sherali and C.M. Shetty, "Nonlinear programming theory and algorithms", Second edition.

[2] Chvatal, V. (1983). Linear Programming, W.H.Freeman and Company, New York.

[3] Djeffal, A. (2013). Etude de quelques algorithmes de points intérieurs pour la programmation convexe (Doctoral dissertation, Université de Batna 2).

[4] Dantzig, G. B. (1990). Origins of the simplex method. In A history of scientific computing (pp. 141-151).

[5] Javet, J.P. (2020), Programation linéaire, CADEV-n^0 30'075.

[6] Karmarkar,N. 4(1984). A new polynomial - time algorithm for linear programming, Combinatorica, pp.373-395.

[7] Menniche, L. (2017). Etude théorique et numérique d'une classe des méthodes de points intérieur pour la programmation linéair, thèse de doctorat, Université Ferhat Abbas, Sétif-1,Algérie.

[8] Nocedal, J. and Wright, S. (2006). *Numerical optimization*. Springer Science.

[9] Saigal, R. (1995). Linear programming, A Modern Integrated Aanalysis, International Series in Operation Research and Management Science, Vol. 1, Kluwer Academic Publishers, November.

Annexe A : Abréviations et Notations

Δ	la direction d'amélioration
∇	le gradient
$\partial f(x)$	la premiere dérivé de la fonction f
$s.c.$	sous contraintes
x^*	la solution optimale du problème
Δx	le gradient projeté
(KP)	karmarkar problème
ell	ellipsoide

تلخيص

هدفنا في هذه الأطروحة هو تقديم طريقتين للنقطة الداخلية المستخدمة لحل البرنامج الخطي ، وهما طريقة افين وطريقة كارماركار. نبدأ بقليل من التذكير حول مفهوم التحدب ، وتعريف البرنامج الرياضي بالإضافة الي شرط ولا جود و التفرد للحد الادنى من النقاط. لكل طريقة تمت دراستها ، نقدم خوارزمياتهم ونظريات التقارب الخاصة بهم. تم تقديم أداة حل مطلاب المخصصة لهذه الأساليب في نهاية هذه الأطروحة.

الكلمات المفتاحية :

النقاط الداخلية ، البرنامج الخطي ، الطريقة الأفينية ، كارماركار ، الخوارزميات

Résumé :

Notre objective dans ce mémoire est de présenter deux méthodes des points intérieurs utilisées pour la résolution d'un programme linéaire, il s'agit de la méthode affine et la méthode de Karmarkar. Nous commençons par un petit rappel sur la notion de convexité, la définition d'un programme mathématique ainsi que la condition d'existence et d'unicité d'un point minimum pour ces programmes. Pour chaque méthode étudiée, nous présentons leurs algorithmes et leurs théorèmes de convergence. Le solveur Matlab destinés pour ces méthodes a été donné à la fin de ce mémoire.

Mots clé

Points intérieurs, programme linéaire, méthode affine, Karmarkar, algorithmes

Abstract :

Our objective in this thesis is to present two interior point methods used for the resolution of a linear program, these are the affine method and the Karmarkar method. We start with a little reminder on the notion of convexity, the definition of a mathematical program as well as the condition of existence and uniqueness of a minimum point for these programs. For each method studied, we present their algorithms and their convergence theorems. The Matlab solver intended for these methods was given at the end of this thesis.

Keywords :

Interior points, linear program, affine method, Karmarkar, algorithms