Institutional Repository - Research Portal
Dépôt Institutionnel - Portail de la Recherche
researchportal.unamur.be
University of Namur

THESIS / THÈSE

MASTER EN SCIENCES MATHÉMATIQUES

Une méthode primale pour la résolution de systèmes surdéterminés suivant la norme de Tchebycheff

ISTACE, Marie-Paule

Award date:
1988

Awarding institution:
Universite de Namur

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 04. May. 2026

UNE METHODE PRIMALE POUR LA RESOLUTION DE SYSTEMES SURDETERMINES SUIVANT LA NORME DE TCHEBYCHEFF

Promoteur : THIRAN J-Pierre

Assistante : THIRY Suzanne

ISTACE Marie-Paule

UNE METHODE PRIMALE POUR LA RESOLUTION DE SYSTEMES SURDETERMINES SUIVANT LA NORME DE TCHEBYCHEFF.

ISTACE Marie-Paule

Résumé

Un algorithme primal de résolution de systèmes d'équations linéaires surdéterminés suivant la norme de Tchebycheff, est analysé. Cet algorithme, proposé par R.Bartels, A.Conn et Y.Li, est une combinaison de méthode de pénalité exacte et de méthode du gradient projeté. La comparaison avec des méthodes classiques de résolution, qui sont de type dual, montre que l'algorithme primal présente des résultats intéressants dans le cas de problèmes aléatoires. Pour les problèmes d'approximation discrète, il est toutefois moins efficace.

Abstract

We analyze a primal algorithm for solving overdetermined linear systems of equations in the ℓ_∞ sense. This algorithm, proposed by R.Bartels, A.Conn and Y.Li, is a combination of exact penalty method and projected gradient method. The comparison with classical methods of resolution, which are dual ones, shows that the primal algorithm offers interesting results in the case of random problems. For discrete approximation problems, it is less helpful.

Mémoire de licence en Sciences Mathématiques
Juin 1988
Unité d'analyse numérique
Promoteur : Prof.J-P.THIRAN

La réalisation de ce travail a pu être menée à bien grâce au soutien efficace de Monsieur le Professeur J.P. Thiran, promoteur, et de son assistante Madame S. Thiry. Je leur adresse mes plus vifs remerciements.

Ma gratitude va également à tous ceux qui m'ont permis de disposer des ouvrages nécessaires.

A ma famille, à Mr et Mme Rolin qui m'ont apporté une aide précieuse, je leur dis également ma reconnaissance.

TABLE DES MATIERES

0. INTRODUCTION

1. Présentation du problème général 0.1
2. Caractérisation des solutions 0.1
3. Unicité de la solution 0.3
4. Algorithmes de résolution du problème (1) 0.4
5. Lien avec la programmation linéaire 0.8
6. Cas particulier 0.9

I. L'ALGORITHME DU GRADIENT PROJETE

1. Introduction I.1
2. Algorithme I.2
3. Etude du problème (PLE) I.4
4. Description des étapes de l'algorithme I.8

II. LES METHODES DE PENALITE

1. Introduction II.1
2. Définitions II.1
3. Algorithme II.3
4. Propriétés II.3
5. Problèmes liés à l'algorithme II.7
6. Méthodes de pénalité exacte II.8

III. LA METHODE PRIMALE DE RESOLUTION D'UN SYSTEME D'EQUATIONS LINEAIRES SURDETERMINES AU SENS DE TCHEBYCHEFF

1. Introduction III.1
2. Première méthode primale III.1
3. Seconde méthode primale III.5
4. Implémentation III.12

IV. RESULTATS

1. Choix de la tolérance et du paramètre de pénalité .. IV.1
2. Les points de départ IV.5
3. Deux problèmes d'expérimentation IV.7
4. Tests numériques et commentaires IV.9
5. Situation de dégénérescence IV.33

CONCLUSIONS

ANNEXES

Annexe 1. Conditions nécessaires et suffisantes d'optimalité
Annexe 2. Les matrices de projection
Annexe 3. La factorisation QR
Annexe 4. Programme

BIBLIOGRAPHIE

AVANT-PROPOS

La résolution de systèmes d'équations linéaires surdéterminés suivant la norme de Tchebycheff a déjà fait l'objet de nombreuses publications [2]-[3]-[4]-[6]-[16]. De tels systèmes peuvent, par exemple, être obtenus par discrétisation de problèmes d'approximation uniforme d'une fonction objectif par des fonctions approximantes linéaires sur un compact donné.

Il est généralement admis que les meilleurs algorithmes de résolution sont de type dual, tel que l'algorithme d'échange et ses diverses variantes [6]-[16].

Dans une publication récente [4] Bartels, Conn and Li ont présenté un algorithme primal de résolution de sytèmes surdéterminés suivant la norme ℓ_∞ en particularisant un algorithme proposé, il y a plusieurs années, par Conn dans le cadre général d'optimisation sans contraintes et de programmation linéaire [9]. Cet algorithme combine la minimisation d'une fonction de pénalité exacte avec l'emploi d'une méthode de gradient projeté. Dans le rapport précité, les auteurs soutiennent une thèse opposée à la supériorité des méthodes duales en présentant des résultats en faveur de leur algorithme.

Le but de ce mémoire est d'analyser l'algorithme de Bartels, Conn et Li et de le comparer à un algorithme d'échange mis au point récemment aux Facultés.
Le mémoire se structure de la manière suivante :
Le chapitre introductif définit le problème à résoudre et résume les aspects essentiels de la théorie classique de l'approximation uniforme dans le cas linéaire.

Au chapitre I et II, sont présentées les méthodes de pénalité exacte et du gradient projeté dans le cadre général d'optimisation avec contraintes. Ceci facilite la description de l'algorithme de Bartels, Conn et Li au chapitre III.
Le chapitre IV présente les résultats numériques de l'algorithme et leur comparaison avec la méthode d'échange.

CHAPITRE 0 : INTRODUCTION

1. PRESENTATION DU PROBLEME GENERAL

On se propose de résoudre le problème d'approximation discret au sens de la norme de Tchebycheff suivant.

Soit $A \in R^{m \times n} (m > n \geq 2)$ et un vecteur ℓ de R^m, on recherche un vecteur x^* de R^n qui minimise le vecteur résidu :

$$r(x) = \ell - A\,x \text{ de } R^m$$

au sens de la norme de Tchebycheff, i.e.;

$$\inf_{x \in R^n} ||r(x)||_\infty = ||r(x^*)||_\infty \qquad (1)$$

où

$$||r(x)||_\infty = \max \{|r_i(x)|;\ 1 \leq i \leq m\}.$$

L'existence d'au moins une solution au problème (1) est assurée par le fait que le sous-espace engendré par les colonnes de la matrice A est de dimension finie.

Dans les paragraphes suivants, on s'intéressera au problème de caractérisation des solutions de (1) ainsi qu'à la condition d'unicité de celle-ci.

2. CARACTERISATION DES SOLUTIONS

Pour plus de facilités, on notera par $\bar{I}$ $(= \bar{I}(x))$ l'ensemble des indices i correspondant aux composantes de r de module égal à $||r||$. De plus, on utilisera le symbole θ_i pour indiquer le signe de r_i.

$$\theta_i = sgn(r_i) \qquad (2)$$

Donc, on peut écrire,

$$r_i = \theta_i \, ||r|| \qquad i \in \bar{I} \quad (3)$$

THEOREME 0.1. : (THEOREME DE CARACTERISATION PRIMAL)

x^* est une solution du problème (1) si et seulement si

$$\min_{i \in \bar{I}(x^*)} r(x^*)\, a_i^t x \leq o \qquad (4)$$

quel que soit x de R^n. Le vecteur a_i^t représente la $i$ème ligne de A.

THEOREME 0.2. : (THEOREME DE CARACTERISATION DUAL)

x^* est une solution du problème (1) si et seulement si il existe un sous-ensemble I de $\bar{I}$ contenant au plus $n+1$ indices et un vecteur λ $(\neq o)$ de R^m tel que :

1. $\lambda_i = o \qquad i \notin I$
2. $A^t \lambda = o$
3. $\lambda_i \, \theta_i \geq o \qquad i \in I$

En pratique, on utilise très souvent le théorème de caractérisation dual plutôt que le théorème de caractérisation primal parce qu'il est plus difficile de prouver l'inégalité (4) que les conditions du théorème 0.2.

COROLLAIRE 0.3. :

Soit x^* une solution de (1) et I tel que :

$$\lambda_i \neq o \qquad i \in I .$$

Alors

$$r_i(\hat{x}) = r_i(x^*) \qquad i \in I$$

pour toute solution $\hat{x}$ du problème (1).

COROLLAIRE 0.4. :

Si la matrice A est de rang t, une solution x^* de (1) existe toujours associée à un ensemble $\bar{I}$ contenant au moins $(t+1)$ indices.

3. UNICITE DE LA SOLUTION

Pour pouvoir établir le théorème d'unicité, on doit définir le concept de condition de Haar; concept qui est fondamental en théorie d'approximation.

Définition :

Une matrice A de $R^{m \times n}$, où $m \geq n$ satisfait à la condition de Haar si toute sous-matrice de dimension n est non-singulière.

THEOREME 0.5. (THEOREME D'UNICITE) :

Si A satisfait à la condition de Haar, alors la solution du problème (1) est unique.

Le sous-ensemble d'indices I déterminé par le théorème 0.2. est souvent appelé sous-ensemble extrémal.

THEOREME 0.6. :

Si A satisfait à la condition de Haar, alors le nombre d'indices se trouvant dans le sous-ensemble extrémal associé à la solution de (1) contient exactement $n+1$ indices.

Pour terminer ce paragraphe, on peut citer une propriété qui sera utilisée par la suite.

PROPRIETE 0.7. :

L'ensemble des solutions du problème (1) est un ensemble convexe.

Après avoir caractérisé les solutions de (1) et donné les conditions d'unicité de ces dernières, il serait intéressant de pouvoir disposer d'un algorithme calculant ces solutions.
C'est ce que nous allons exposer dans le paragraphe suivant.

4. ALGORITHMES DE RESOLUTION DU PROBLEME (1)

Avant de décrire l'algorithme étape par étape, il serait bon de caractériser la solution du sous-problème suivant : on considère :

$$r_i(x) = b_i - a_i^t x \qquad i \in J = \{1,2,\ldots,n+1\} \tag{5}$$

où on doit minimiser

$$\max_{i \in J} |r_i(x)| .$$

Il s'agit en fait du problème (1) où l'on considère $m=n+1$. On note par a_i^t la $i$ème ligne de la matrice A.

La solution x^* du problème (5) est caractérisée par les équations suivantes :

$$b_i - a_i^t x^* = \theta_i^* \xi \qquad i \in J \tag{6}$$

où

$$|\theta_i^*| = 1 \qquad \text{pour tout } i \in J . \tag{7}$$

Soit $\lambda \in R^{n+1}$ tel que

$$1.\ \lambda \neq o \tag{8}$$

$$2.\ \sum_{i \in J} \lambda_i a_i = o \tag{9}$$

$$\text{avec } \sum_{i \in J} |\lambda_i| = 1$$

$$3.\ \lambda^t b \geq o \tag{10}$$

alors

$$\lambda^t \ell = \xi \lambda^t \theta^* \tag{11}$$

et donc

$$|\xi| = \lambda^t \ell / |\lambda^t \theta^*| \tag{12}$$
$$= |r_i(x^*)| \qquad i \in \mathcal{J} .$$

Le membre de droite de l'équation (12) est minimum si l'on choisit

$$\theta_i^* = sgn\ (\lambda_i) \qquad i \in \mathcal{J} . \tag{13}$$

Par le théorème de caractérisation dual 0.2., on peut dire que $\mathcal{J}$ est un ensemble d'indices extrémal.

En résumé, pour obtenir une solution au problème (5), on effectue les étapes suivantes :

(i) Trouver $\lambda \in R^{n+1}$ non nul satisfaisant à l'équation (9)

(ii) $\theta_i^* = sgn\ (\lambda_i) \qquad i \in \mathcal{J}$

(iii) Résoudre l'équation (6) par x^* et ξ

Nous allons maintenant décrire une étape de l'algorithme.

Soit $\mathcal{J}^{(k)}$, un ensemble de $(n+1)$ indices compris entre 1 et m que l'on peut supposer sans perte de généralité égal à $\{1,\ldots,n+1\}$.
Par ce qui vient d'être décrit, on sait calculer la solution de (5) où $\mathcal{J} \equiv \mathcal{J}^{(k)}$. Cette solution est caractérisée par les équations (6), (7),(8),(9),(10),(12) et (13).

Il s'agit maintenant de déterminer si cette solution est également une solution du problème plus général (1). Pour ce faire, il faut calculer :

$$||r(x^{(k)})|| = \max_{1 \leq i \leq m} |r_i(x^{(k)})| = |r_\ell(x^{(k)})|$$

Comme

$$\mathcal{J}^{(k)} \subset \{i : 1 \leq i \leq m\}$$

on déduit que

$$||r(x^{(k)})|| \geq \xi^{(k)}.$$

Si on atteint l'égalité, alors $x^{(k)}$ est la solution du problème (1) et $\mathcal{J}^{(k)}$ est le sous-ensemble d'indices extrémal associé. Sinon, on va construire un nouvel ensemble d'indices $\mathcal{J}^{(k+1)}$ tel que

$$\mathcal{J}^{(k+1)} = \{1, 2, .. j-1, \ell, j+1, \ldots, n+1\}.$$

Pour décider de l'indice j sortant de $\mathcal{J}^{(k)}$, on suit la règle d'échange suivante :

$$sgn\ r_j(x^{(k)}) = sgn\ \lambda_j^{(k+1)}$$

pour tout i dans $\mathcal{J}^{(k+1)}$.

Algorithme :

Initialisations :

. $k = 1$

. Choix d'un ensemble d'indices $\mathcal{J}^{(k)}$
$\mathcal{J}^{(k)} = \{i_1, i_2, \ldots, i_{n+1}\}$
où
$1 \leq i_j \leq m$ pour $1 \leq j \leq n+1$

. Aller à l'étape générale.

Etape générale :

. Résoudre
$\sum_{i \in \mathcal{J}^{(k)}} \lambda_i^{(k)} a_i = o$ et $\sum_{i \in \mathcal{J}^{(k)}} |\lambda_i| = 1$

pour λ_i, $i \in \mathcal{J}^{(k)}$

. Mettre $\theta_i^{(k)} = sgn\ \lambda_i^{(k)}$ pour $i \in \mathcal{J}^{(k)}$

. Résoudre

$$\ell_i - a_i^t x^{(k)} = \theta_i^{(k)} \xi^{(k)} \qquad i \in \mathcal{J}^{(k)}$$

pour $x^{(k)}$ et $\xi^{(k)}$

. Calculer $||r(x^{(k)})|| = ||r_\ell(x^{(k)})||$

. Si $\xi^{(k)} = ||r(x^{(k)})||$, alors $x^{(k)}$ est la solution de (1).

Sinon, on construit

$$\mathcal{J}^{(k+1)} = \{i_1, i_2, \ldots, i_{j-1}, \ell, i_{j+1}, \ldots, i_{n+1}\}$$

où i_j est l'indice sortant de $\mathcal{J}^{(k)}$ de manière à réaliser la règle d'échange :

$$sgn\ r_i(x^{(k)}) = sgn\ \lambda_i^{(k+1)}$$

pour tout i dans $\mathcal{J}^{(k+1)}$.

. Poser $k := k+1$

. Aller à l'étape générale.

Convergence de l'algorithme :

L'algorithme qui vient d'être décrit converge en un nombre fini d'étapes si A satisfait à la condition de Haar.

En effet, il est impossible de retomber sur le même sous-ensemble d'indices $\mathcal{J}$ car on dispose de la propriété suivante :

PROPRIETE 0.8. :

Si A satisfait à la condition de Haar, alors

$$\xi^{(k)} < \xi^{(k+1)}.$$

Donc, comme on ne peut jamais retomber sur le même ensemble d'indices et comme le nombre d'ensemble de $n+1$ indices pris parmi m indices est fini, l'algorithme convergera en un nombre fini d'étapes. Par construction, cet algorithme ne peut converger que vers la solution du problème (1).

Remarque :

Quand la condition de Haar n'est pas satisfaite, l'algorithme peut ne pas converger. En effet, il est possible que

$$\xi^{(k+1)} = \xi^{(k)}$$

Ce qui pose le problème de la dégénérescence de l'algorithme. D'autres algorithmes ont été mis au point pour calculer une solution de (1) lorsque la condition de Haar n'est pas satisfaite. On peut citer entre autres les articles de C.Carasso et P.Laurent [6] et de J.P. Thiran et S.Thiry [16] qui donne un algorithme de détermination de l'approximation stricte au sens de Tchebycheff pour les systèmes d'équations linéaires.

5. LIEN AVEC LA PROGRAMMATION LINEAIRE

Nous allons reformuler le problème (1) sous forme d'un programme linéaire.

Si on pose : $\xi = \max_{1 \leq i \leq m} |r_i|$

alors le problème (1) peut se réécrire de la manière suivante :

$$\min \xi$$

$$\text{s.c.} \quad -\xi \leq r_i \leq \xi \qquad i = 1,2,\ldots,m$$

ou encore

$$\min_{v} \; c_o^t v \qquad \text{(P) primal}$$

$$\text{s.c.} \quad c_j^t v \geq \delta_j \qquad j = 1,\ldots,2m$$

où

$$c_o^t = (1 \; 0 \; \ldots \; 0)$$

$$c_i^t = (1 \; -a_i^t) \qquad i = 1,\ldots,m$$

$$c_{m+i}^t = (1 \; +a_i^t) \qquad i = 1,\ldots,m$$

$$\delta_i = -\ell_i \qquad i = 1,\ldots,m$$

$$\delta_{m+i} = +\ell_i \qquad i = 1,\ldots,m$$

$$v^t = (\xi \; x^t)$$

La formulation duale de (P) donne :

$$\begin{aligned} &\max_{y} \quad \delta^t y \\ \text{s.c.} \quad & y \geqq o \\ & C^t y = C_o \end{aligned} \qquad \text{(P) dual}$$

où

$$y = (\lambda_1 \ldots \lambda_{2m})$$
$$\delta = (\delta_1 \ldots \delta_{2m})$$
$$C^t = (c_1 \ldots c_{2m})$$

L'algorithme d'échange qui a été décrit au paragraphe précédent est un algorithme de type dual. Il suffit de se référer non seulement aux équations (9) et (12) mais aussi à la propriété 0.8. pour en être convaincu.

6. CAS PARTICULIER

Considérons une fonction f, appelée communément fonction objectif, définie sur B un ensemble de m points distincts de R^s $(s \geqq 1 \in N)$. On désire approximer f par un élément de l'espace de dimension finie W. Les éléments de W sont également définis sur B.

Le problème à résoudre peut donc s'écrire de la manière suivante : on doit trouver w^* réalisant,

$$\inf_{w \in W} ||w-f||_\infty = ||f-\hat{w}||_\infty \qquad (14)$$

Ce problème d'approximation de f sur un ensemble de m points suivant la norme de Tchebycheff est, en fait, un cas particulier du problème (1). En effet, supposons que :

$$W = span\ \{w_1, \ldots, w_n\}$$

où

$$dim\ W = n$$

w_i est une fonction définie sur B pour

$i = 1, .., n$.

Quel que soit $w \in W$, on peut lui associer la fonction erreur

$$e = f - w$$

où

$$w = \sum_{i=1}^{n} \alpha_i w_i$$

et

$$||e||_\infty = \max_{x \in B} |e(x)| = \xi .$$

Cette dernière égalité est équivalente au système d'équations suivant :

$$-\xi \leq f(x_i) - \sum_{j=1}^{n} \alpha_j w_j (x_i) \leq \xi$$

pour tout $i=1,..,m$; ou encore

$$\xi - \sum_{j=1}^{n} \alpha_j w_j (x_i) \geq -f(x_i)$$

$$\xi + \sum_{j=1}^{n} \alpha_j w_j (x_i) \geq f(x_i)$$

pour $i = 1,..,m$.

On obtient donc le problème (1) en notant la $i$ème ligne de A par

$$a_i^t = (w_1 (x_i), \ldots, w_n (x_i))$$

$$= u^t (x_i)$$

$$\equiv \text{vecteur caractéristique évalué en } x_i$$

et

$$b_i = f (x_i)$$

pour $i = 1,..,m$.

Le vecteur des inconnues x de R^n est en fait formé des α_j associés à la décomposition de w dans la base de W, i.e. :

$$x^t = (\alpha_1, \ldots, \alpha_n) .$$

Remarques :

1. Le théorème de caractérisation dual 0.2. devient : L'élément w^* de W est meilleure approximation de f si et seulement si il existe k ($\leq n+1$) points extrémaux $x_1,..,x_k$ et des scalaires ε_i de module égal à 1 tels que :

$$1.\ e^*(x_i) = f(x_i) - w^*(x_i)$$

$$= \varepsilon_i \, ||e^*||$$

$$2.\ \sum_{i=1}^{k} \theta_i(\varepsilon_i \, u(x_i)) = o$$

où

$$\theta_i \, \varepsilon_i = \lambda_i \qquad i = 1,..,k$$

$$\theta_i \geq o$$

$u(x_i)$ est le vecteur caractéristique évalué en x_i.

Nous pouvons également adapter la condition de Haar à ce problème d'approximation discret .

Définition :

On dit que W sous-espace d'approximation vérifie la condition de Haar relativement à B si et seulement si, quels que soient $x_1,..,x_n \in B$ distincts deux à deux, les vecteurs caractéristiques $u(x_1),..., u(x_n)$ forment une partie libre.
Cette définition est encore équivalente à la suivante.

Définition

On dit que W sous-espace d'approximation vérifie la condition de Haar relativement à B si et seulement si W a au plus $n-1$ zéros distincts dans B, où w est un élément arbitraire non nul de W.

2. Supposons que f soit une fonction continue sur $B = [\alpha, \beta]$ de R. Alors, si W vérifie la condition de Haar sur B, et $x_1 < x_2 < \ldots < x_{n+1}$, on a $\lambda_i . \lambda_{i+1} < o$ pour $i = 1..n$.

3. En utilisant la remarque 2. et le théorème de caractérisation dual énoncé au début de la première remarque, on trouve le théorème d'équioscillation. Ce théorème indique que l'erreur atteint sa norme avec des signes alternés en au moins $n+1$ points de $[\alpha, \beta]$.

THEOREME 0.9. (THEOREME D'EQUIOSCILLATION) :

Si W vérifie la condition de Haar sur $B = [\alpha, \beta]$, alors, $w^* \in W$ est meilleure approximation de f si et seulement si $e^* = (f - w^*)$ atteint sa norme $||e^*||$ avec des signes alternés en au moins $(n+1)$ points de $[\alpha, \beta]$.

4. Lorsqu'on étudie un problème d'approximation sur $B = [\alpha, \beta]$ pour lequel W vérifie la condition de Haar, la meilleure méthode à utiliser est l'algorithme second de Remes. Voici comment se déroule cet algorithme.

A la *kème* étape, on définit un sous-ensemble $J^{(k)}$ de $(n+1)$ points distincts de B. On résoud le problème d'approximation discret sur ces points, i.e. on résoud le système d'équations :

$$f(x_i) - \sum_{j=1}^{n} \alpha_j \omega_j(x_i) = (-1)^i \xi \qquad (15)$$

pour $i = 1, \ldots, n+1$.

Ensuite, on examine la courbe d'erreur

$$e^*(x) = f(x) - \omega^*(x)$$

où

ω^* est la solution de (15) pour détecter $n+1$ maxima locaux dans B tels que :

- la courbe d'erreur ait des signes alternés sur ces maxima quand on se déplace de la gauche vers la droite.
- un maxima global de $|e^*(x)|$ y soit inclus.

Cet ensemble de points forme l'ensemble $J^{(k+1)}$.

Cet algorithme converge vers la meilleure approximation de f sur B. De plus, Veidinger a montré que si les fonctions $f(x), \omega_1(x), \ldots, \omega_n(x)$ sont deux fois continûment différentiables, alors il a un taux de convergence quadratique.

Dans la suite de ce travail, on présentera non pas une méthode de type dual comme l'algorithme d'échange, mais bien une méthode de type primal. Avant de pouvoir la décrire, les deux chapitres suivants seront consacrés à deux méthodes d'optimisation dont nous aurons besoin pour bien analyser cet algorithme primal.

CHAP I : L'ALGORITHME DU GRADIENT PROJETE

1. INTRODUCTION

Nous allons considérer dans ce chapitre le problème de minimisation d'une fonction continue et différentiable soumise à un ensemble de contraintes linéaires :

$$\min f(x) \qquad \text{(PL)}$$

s.c

$$a_i x = b_i \qquad i \in \mathcal{E}$$

$$a_i x \leq b_i \qquad i \in I$$

où

$f : R^n \to R$ continue et différentiable

$x \in R^n$

$a_i^t \in R^n$ pour tout i $(\mathcal{E} \cup I)$

$b_i \in R$.

Pour traiter les inégalités du problème (PL), on va appliquer une méthode dite de contraintes actives : à une certaine étape k de l'algorithme on va tenir compte des contraintes j telles que :

$$a_j x^{(k)} = b_j \text{ pour } j \in I.$$

Ces contraintes d'inégalités seront considérées à l'étape k comme des égalités; les autres contraintes d'inégalités seront laissées de côté momentanément.
A chaque étape de l'algorithme, on travaillera avec un problème linéaire d'égalités :

$$\min f(x) \qquad \text{(PLE)}$$

s.c $\quad a_i x = b_i \qquad i \in (\mathcal{E} \cup B)$

où

$$\mathcal{B} \subset I$$

La méthode est basée sur le théorème suivant :

THEOREME 1.1 :

Soit x^* un point admissible pour (PL) et soit $\mathcal{B}^* = \{j \in I : a_j x^* = b_j\}$

Si $\{x^*; \mu_i^*, i \in \mathcal{E}; \lambda_j^*, j \in \mathcal{B}^*\}$ est un point de Kuhn-Tucker de

$$\min f(x)$$

$$\text{s.c} \quad \begin{array}{ll} a_i x = b_i & i \in \mathcal{E} \\ a_i x = b_i & i \in \mathcal{B}^* \end{array}$$

et si $\lambda_i^* \geq 0, \; i \in \mathcal{B}^*$

Alors (x^*, μ^*, λ^*) est un point de Kuhn-Tucker de (PL).

On pourra trouver en annexe les conditions d'optimalité d'un problème d'optimisation.

2. ALGORITHME

La méthode des contraintes actives peut être décrite comme suit :

(a) Soit x_1 un point admissible pour (PL) et $\mathcal{B}_1$ l'ensemble des indices actifs en x_1 parmi les contraintes d'inégalités. Posons $k=1$

(b) si x_k est solution de

$$\min f(x) \qquad (PLE_k)$$
$$\text{s.c. } a_i x = b_i \qquad i \in (\mathcal{E} \cup \mathcal{B}_k)$$

alors aller en (c).

Sinon calculer en x_k une direction de descente s_k admissible pour (PLE_k). Aller en (d).

(c) évaluer les multiplicateurs de Lagrange de (PLE_k) et tester si $\lambda_j \geq 0$, $j \in \mathcal{B}_k$.

Si oui → STOP (x_k est solution de (PL))

Sinon retirer de $\mathcal{B}_k$ l'indice du multiplicateur le plus négatif. Calculer en x_k une direction s_k de descente admissible pour le nouveau problème (PLE_k) et aller en (d).

(d) recherche linéaire admissible pour (PL) le long de s_k.

$$x_{k+1} = x_k + \alpha_k s_k$$

(e) calculer α_{max} comme suit :

$$\alpha_{max} = \min \frac{b_j - a_j x_k}{a_j s_k} \quad : \quad j \in \mathcal{B}_k,\ a_j s_k > 0 \qquad \text{- s'il existe un } j \text{ tel que } a_j s_k > 0$$
$$= +\infty \qquad \text{- sinon}$$

Si $\alpha_k = \alpha_{max}$ alors ajouter ℓ à $\mathcal{B}_k$ où ℓ est l'indice donnant le minimum dans l'expression de α_{max}.

(f) $k \leftarrow k+1$ *et aller en* (b)

Remarque :

1. La direction de descente s_k va être déterminée dans le sous-espace des contraintes actives.

Les paragraphes suivants, vont être consacrés à l'explicitation des différentes étapes de l'algorithme.

3. ETUDE DU PROBLEME (PLE)

A chaque itération de l'algorithme on doit vérifier que x_k est solution du problème (PLE_k). On va donc s'intéresser à la résolution de ce problème.
Soit, donc, le problème

$$\begin{aligned} & \min \ f(x) \\ \text{s.c} \quad & A x = b \end{aligned} \qquad \text{(PLE)}$$

où

$f : R^n \to R$ continue et différentiable
$A \in R^{m \times n}$; $b \in R^m$

La matrice A est supposée de rang plein (i.e. rang $(A)=m$).

Si x^* est une solution optimale de (PLE), on peut écrire, par les conditions d'optimalité de Kuhn-Tucker, l'équation suivante :

$$\nabla f(x^*) + A^t \lambda^* = o \qquad (1)$$

où

λ^* est le vecteur des multiplicateurs de Lagrange associé à x^*.

En prémultipliant l'équation (1) par la matrice A, on obtient :

$$A \nabla f(x^*) + (AA^t) \lambda^* = o \qquad (2)$$

D'où, on déduit que :

$$\lambda^* = - (AA^t)^{-1} A \nabla f(x^*). \qquad (3)$$

Pour résoudre le problème (PLE), on va utiliser la méthode d'élimination généralisée. En effet, on va éliminer un certain nombre de variables en utilisant l'égalité

$$Ax = b$$

Soit S une matrice de $R^{n \times m}$ telle que

$$AS = I$$

S est un inverse généralisé à droite de A.
On a alors que $x = S\ell$ est une solution de $Ax = \ell$.
Cette solution n'est pas unique. En effet, il faut considérer le noyau de A.
Toutes les solutions de $Ax = \ell$ sont données par

$$x = S\ell + \delta$$

où

$$\delta \in \{\delta : A\,\delta = o\}.$$

Considérons Z une matrice dont les colonnes forment une base du noyau de A. Comme A est de rang plein, Z sera de dimension $(n, n-m)$. Par définition de Z, on peut écrire :

$$A\,Z = o$$

et

$$\delta = Z\,y \qquad \text{pour un } y \in R^{n-m}$$

Donc, tout point admissible de (PLE) peut s'écrire de la manière suivante :

$$x = S\ell + Zy$$

où

$$y \in R^{n-m}.$$

On va pouvoir travailler avec les variables y et non plus avec les variables x. Le problème réduit est :

$$\min_{y \in R^{n-m}} \quad \psi(y) \equiv f(S\ell + Zy).$$

Immédiatement, on déduit :

$$\nabla \psi(y) = Z^t \nabla f(x).$$

Comme $\nabla f(x^*) + A^t \lambda^* = o$ (1), on obtient en prémultipliant cette dernière équation par S^t :

$$\lambda^* = -S^t \nabla f(x^*). \qquad (4)$$

La direction de la plus forte pente pour le problème réduit est :

$$s_y = -Z^t \nabla f(x)$$

D'où, on peut écrire que la direction correspondante pour le problème (PLE) est :

$$s_x = Z\, sy = -ZZ^t\, \nabla f(x).$$

En effet, soit $\bar{y}$ et $y^* \in R^{n-m}$, on peut définir

$$\bar{x} = Sl + Z\bar{y}$$

et

$$x^* = Sl + Zy^*.$$

La direction de descente s_y peut s'écrire comme suit :

$$s_y = \bar{y} - y^*$$

D'où, on déduit que

$$s_x = \bar{x} - x^* = Z\, s_y .$$

Cette direction s_x est une direction de descente admissible pour (PLE). En effet,

$$s_x^t\, \nabla f(x) = -(\nabla f(x))^t\, Z\, Z^t\, \nabla f(x) < o$$

De plus, s_x est une direction admissible parce que

$$A\, s_x = o.$$

En effet,

$$\begin{aligned} A\, s_x &= -AZZ^t\, \nabla f(x) \\ &= -AZy \qquad \text{où } y \in R^{n-m} \\ &= o. \end{aligned}$$

La dernière égalité est déduite du fait que Z est une base du noyau de A.

Les méthodes diffèrent selon le choix de S et Z. Cependant, un choix particulier est couramment utilisé. On utilise la factorisation orthogonale de A^t à savoir :

où

$$A^t = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R \\ O \end{bmatrix} = Q_1 R$$

Q est une matrice orthogonale.

Comme $A^t = Q_1 R$, on peut prendre

$$S = Q_1 R^{-t}$$
$$Z = Q_2$$

L'équation (4) devient :

$$\lambda^* = -R^{-1} Q_1^t \nabla f(x^*)$$

et λ^* pourra s'obtenir par substitution

$$R \lambda^* = -Q_1^t \nabla f(x^*). \qquad (5)$$

Nous allons terminer ce paragraphe en expliquant pourquoi ce type de méthode est appelée méthode du gradient projeté.

THEOREME 1.2 :

$$ZZ^t = I - A^t (AA^t)^{-1} A$$

Démonstration :

Comme $Z = Q_2$

$$Q_2 Q_2^t = I - Q_1 R (R^t R)^{-1} R^t Q_1^t$$
$$Q_2 Q_2^t = I - Q_1 R R^{-1} R^{-t} R^t Q_1^t$$
$$Q_2 Q_2^t = I - Q_1 Q_1^t$$

Or cette dernière égalité est vraie car Q est une matrice orthogonale. □

Dans le cas où,

$$Z = Q_2 \qquad S = Q_1 R^{-t}$$

la méthode de contraintes actives est appelée méthode du gradient projeté, parce qu'elle projette l'opposé du gradient sur le sous-espace des contraintes actives.

En effet,

$$s_x = -Z\, Z^t \nabla f(x) = -P \nabla f(x)$$

et par le théorème

$$P = I - A^t(AA^t)^{-1} A$$

i.e.

P est la matrice de projection sur le sous-espace des contraintes actives.

On pourra se référer à l'annexe sur les matrices de projection pour plus de détails.

4. DESCRIPTION DES ETAPES DE L'ALGORITHME

Venons-en maintenant à l'explicitation des différentes étapes de l'algorithme .

A l'étape (b)

On prendra comme direction de descente admissible

$$s_k = -Z\, Z^t \nabla f(x_k)$$

A l'étape (c)

Les multiplicateurs de Lagrange sont évalués en résolvant (5) :

$$R\, \lambda_k = -\, Q_1^t \nabla f(x_k).$$

S'ils sont tous positifs ou nuls, on arrête l'algorithme. Sinon on retire de $\mathcal{B}_k$ l'indice correspondant au multiplicateur le plus négatif. Soit, q cet indice. Notons $\hat{Z}$ la nouvelle matrice Z et $\hat{\mathcal{B}}$ le nouveau $\mathcal{B}$.

Montrons que

$$s_k = - \hat{Z}\, \hat{Z}^t \nabla f(x_k) \quad \text{est une direction de descente admissible}$$

i.e. $\quad \nabla f(x_k)^t \hat{s}_k < o$

et
$$a_i \hat{s}_k = o \qquad i \in \mathcal{E}$$
$$a_i \hat{s}_k = o \qquad i \in \hat{B}$$
$$a_q \hat{s}_k < o$$

La dernière inégalité stricte indique qu'on ne désire pas que la qème contrainte redevienne active.

Démonstration

La démonstration se fera pour le choix particulier de S et Z où

$$S = Q_1 R^{-t}$$
$$Z = Q_2 .$$

En arrivant à l'étape (c), on a x_k qui est solution du problème (PLE_k) et donc

$$-P \nabla f(x_k) = o$$

Où
$$P = I - A^t (AA^t)^{-1} A .$$

Les multiplicateurs s'écrivent sous la forme :
Par (3)

$$\begin{bmatrix} \mu \\ \lambda \end{bmatrix} = -(AA^t)\, A \nabla f(x_k)$$

Où
$$\lambda_q < o .$$

Par hypothèse, on a

$$\hat{B} = B \setminus \{q\}$$

et

$$\hat{s}_k = - \hat{P} \nabla f(x_k)$$

où

$$\hat{P} = I - \hat{A}^t (\hat{A}\ \hat{A}^t)^{-1}\ \hat{A}$$

Pour obtenir $\hat{A}$, il suffit de retirer de A la ligne a_q.

Montrons que $\hat{s}_k$ est une direction de descente. Pour ce faire, il suffit de montrer que

$$\hat{P} \nabla f(x_k) \neq o$$

car alors

$$\nabla f(x_k)^t \hat{s}_k = - \nabla f(x_k)^t \hat{P} \nabla f(x_k) < o .$$

La dernière inégalité peut être déduite en utilisant le fait que $\hat{P}$ est une matrice de projection.
Par l'absurde, supposons que

$$\hat{P} \nabla f(x_k) = o .$$

Ce qui peut encore s'écrire :

$$(I - \hat{A}^t (\hat{A}\ \hat{A}^t)^{-1}\ \hat{A}) \nabla f(x_k) = \nabla f(x_k) + \hat{A}^t \begin{bmatrix} \hat{\mu} \\ \hat{\lambda} \end{bmatrix} = o \qquad (6)$$

où

$$\begin{bmatrix} \hat{\mu} \\ \hat{\lambda} \end{bmatrix} = -(\hat{A}\ \hat{A}^t)^{-1}\ \hat{A} \nabla f(x_k)$$

Comme $P \nabla f(x_k) = o$, on déduit que :

$$\nabla f(x_k) + A^t \begin{bmatrix} \mu \\ \lambda \end{bmatrix} = o$$

ou encore

$$\nabla f(x_k) + \hat{A}^t \begin{bmatrix} \mu \\ \lambda \end{bmatrix} + \lambda_q a_q{}^t = o \qquad (7)$$

où

$$\bar{\lambda} = \lambda \setminus \{\lambda_q\},$$

En soustrayant (7) de (6), on a :

$$\hat{A}^t \begin{bmatrix} \hat{\mu}-\mu \\ \hat{\lambda}-\bar{\lambda} \end{bmatrix} - \lambda_q \, a_q{}^t = o$$

Ce qui est impossible puisque A est de rang m et λ_q est différent de o.

Donc, $\hat{s}_k$ est une direction de descente. Il reste encore à démontrer que

$$a_q \, \hat{s}_k < o \, .$$

Multiplions (7) par $a_q \, \hat{P} \neq o$ puisque A est de rang plein, on a :

$$o = a_q \, \hat{P} \, \nabla \ell(x_k) + a_q \, \hat{P} \, \hat{A}^t \begin{bmatrix} \mu \\ \lambda \end{bmatrix} + \lambda_q \, a_q \, \hat{P} \, a_q{}^t$$

Comme $\hat{P} \, \hat{A}^t = o$, on déduit :

$$a_q \, \hat{s}_k = \lambda_q a_q \, \hat{P} \, a_q{}^t < o$$

car $\hat{P}$ est une matrice de projection et $\lambda_q < o$.

□

CHAP II : LES METHODES DE PENALITE

1. INTRODUCTION

Les méthodes de pénalité transforment un problème avec contraintes en un problème sans contrainte. Les contraintes sont insérées dans la fonction objectif via un paramètre de pénalité de manière à pénaliser toute violation de celles-ci.

Le problème se pose de la manière suivante :

$$\begin{aligned} &\min f(x) \\ &\text{s.c } g(x) \leqq o \\ &\quad\ h(x) = o \end{aligned} \qquad (P)$$

où

$f: R^n \to R$ *est la fonction objectif, continue et différentiable*

$x \in R^n$

$g: R^n \to R^m$ *est continue différentiable*

$h: R^n \to R^p$ *est continue différentiable*

2. DEFINITIONS

Pour rappel, on définit l'ensemble admissible S associé au problème (P) comme l'ensemble des éléments x de R^n qui satisfont aux contraintes du problème :

$$S = \{x \in R^n \ ; \ g(x) \leqq o \text{ et } h(x) = o\}.$$

On va remplacer le problème (P) par le problème sans contrainte suivant :

$$\min_{x \in R^n} \quad f(x) + \mu P(x) = \theta(x, \mu)$$

où

$\mu > 0$ *est le paramètre de pénalité*

et $P : R^n \to R$ *est la fonction de pénalité .*

De manière générale, une fonction de pénalité doit être choisie de façon à ajouter une pénalité positive si une contrainte est violée et à être nulle si le point est admissible; ce qui se traduit par :

1. $P(x) \geq 0 \qquad \forall x \in R^n$
2. $P(x) = 0$ *ssi* x *est admissible pour* (P)
3. P *est continue*

La fonction de pénalité $P(x)$ a généralement la forme :

$$P(x) = \sum_{i=1}^{m} \max \{0, g_i(x)\}^q + \sum_{i=1}^{p} |h_i(x)|^q$$

où

$q \in N$.

3. ALGORITHME

* Pas d'initialisation

Soient $\varepsilon > 0$ *précision numérique choisie*

x_0 *point initial*

μ_1 *paramètre de pénalité initial*

$\beta > 1$ *scalaire*

$k = 1$

* Pas principal

1. On commence avec x_{k-1} pour minimiser $f(x) + \mu_k P(x)$

   s.c $x \in R^n$ $\qquad (P_k)$

   On trouve x_k point optimal de (P_k)
   Aller en 2.

2. Si $\mu_k P(x_k) < \varepsilon \rightarrow$ *stop*

   Sinon - on augmente μ

   i.e. $\mu_{k+1} \leftarrow \beta \mu_k$

   - $k \leftarrow k+1$

   Aller en 1.

4. PROPRIETES

Considérons la fonction de pénalité quadratique $(q=2)$:

$$P(x) = \tfrac{1}{2} \sum_{i=1}^{m} (\max \{0, g_i(x)\})^2 + \tfrac{1}{2} \sum_{i=1}^{p} |h_i(x)|^2$$

où le facteur ½ est introduit pour des raisons de commodité ultérieure.

Le but est évidemment d'obtenir la propriété de convergence de la méthode vers un point optimal du problème de départ (P).

Pour ce faire, introduisons un premier lemme qui assure la croissance de ℓ, la décroissance de $\theta(x,\mu)$ et de $P(x)$ en fonction du paramètre μ.

LEMME 2.1 :

Soit $\mu_k \to +\infty$

Alors (1) $\ell(x_k) \leq \ell(x_{k+1})$

(2) $P(x_k) \geq P(x_{k+1})$

(3) $\theta(x_k,\mu_k) \leq \theta(x_{k+1},\mu_{k+1})$

Démonstration :

*(3) En tenant compte que x_k minimise $\theta(x,\mu_k)$ et que $\mu_k \leq \mu_{k+1}$, on obtient les inégalités suivantes :

$$\theta(x_k,\mu_k) = \ell(x_k) + \mu_k P(x_k)$$

$$\leq \ell(x_{k+1}) + \mu_k P(x_{k+1})$$

$$\leq \ell(x_{k+1}) + \mu_{k+1} P(x_{k+1}) = \theta(x_{k+1},\mu_{k+1})$$

*(2) Par définition de $\theta(x,\mu)$ on a :

$$\ell(x_k) + \mu_k P(x_k) \leq \ell(x_{k+1}) + \mu_k P(x_{k+1})$$

et $\ell(x_{k+1}) + \mu_{k+1} P(x_{k+1}) \leq \ell(x_k) + \mu_{k+1} P(x_k)$.

En additionnant ces deux inégalités, on obtient :

$$(\mu_{k+1} - \mu_k) P(x_{k+1}) \leq (\mu_{k+1} - \mu_k) P(x_k)$$

avec $(\mu_{k+1} - \mu_k) \neq 0$; de sorte que $P(x_{k+1}) \leq P(x_k)$.

* (1) Comme

$$\ell(x_k) + \mu_k \; P(x_k) \leq \ell(x_{k+1}) + \mu_k \; P(x_{k+1})$$

on obtient, en utilisant (2) :

$$\ell(x_k) \leq \ell(x_{k+1}) \qquad \square$$

Après avoir établi ce premier lemme, on peut démontrer le théorème de convergence :

THEOREME 2.2 :

Supposons que :

- le problème (P) admet une solution globale admissible.
- la fonction est bornée inférieurement sur S.

Si $\mu_k \to + \infty$, alors tout point d'accumulation de la suite $\{x_k\}$ est un point optimal du problème de départ (P).

Démonstration :

Si ℓ^* est la borne inférieure de ℓ sur S, on peut écrire :

$$\inf_{\{x \in R^n ;\ P(x) = o\}} \theta(x, \mu_k) = \ell^* \qquad \forall\, k \in N$$

D'où,

$$\ell(x_k) \leq \theta(x_k, \mu_k) = \inf_{x \in R^n} \theta(x, \mu_k)$$

$$\leq \inf_{\{x \in R^n ;\ P(x) = o\}} \theta(x, \mu_k) = \ell^*$$

de sorte que :

$$f(x_k) \leq f^* \qquad \forall k . \qquad (1)$$

Soit $\bar{x}$, un point d'accumulation de la suite $\{x_k\}$ i.e. il existe une sous-suite notée $\{x_k\}_{k \in K \subset N}$ convergeant vers $\bar{x}$.

Comme f est continue

$$\lim_{k \in K} f(x_k) = f(\bar{x}) . \qquad (2)$$

Par le lemme 1 (3), on a :

$$\lim_{k \in K} \theta(x_k, \mu_k) \leq f^* . \qquad (3)$$

La suite $\{\theta(x_k, \mu_k)\}_{k \in K}$ est croissante et majorée : elle converge donc vers θ^*.
En utilisant (2) et (3) :

$$\lim_{k \in K} \mu_k P(x_k) = \theta^* - f(\bar{x}) . \qquad (4)$$

Montrons que $\bar{x}$ est admissible.
De (4), on déduit que $P(x_k) \to o$ puisque $\mu_k \to +\infty$ et $\mu_k P(x_k) \to \theta^* - f(\bar{x})$.
Par la continuité de P, on a $P(\bar{x}) = o$ et donc $\bar{x} \in S$.

La suite $\{f(x_k)\}$ est croissante et majorée par (1), elle converge donc vers $f(\bar{x})$ par (2). De sorte que, en tenant compte de $\bar{x} \in S$ et de la définition de f^* :

$$f(\bar{x}) = f^*$$

□

Remarque : Les points $\{x_k\}$ sont situés en général en dehors du domaine admissible. Par conséquent, quand le paramètre de pénalité μ croît, les points générés approchent de la solution optimale par l'extérieur du domaine admissible. Cette technique est souvent appelée méthode de pénalité extérieure.

5. PROBLEMES LIES A L'ALGORITHME

A chaque étape de l'algorithme il faut résoudre un problème de minimisation sans contrainte. Les méthodes les plus efficaces de minimisation utilisent des évaluations du gradient et parfois du hessien de la fonction à minimiser.

Dans les méthodes de pénalité, le hessien de la fonction de pénalité $\theta(x,\mu)$ devient de plus en plus mal conditionné avec l'augmentation du paramètre de pénalité μ.
En effet,

x_k est un minimum local de $\theta(x,\mu_k)$

$$\nabla\,\theta(x_k,\mu_k) = \nabla f(x_k) + \mu_k \sum_{i=1}^{m} \max\{o, g_i(x)\}\nabla g_i(x) + \mu_k \sum_{i=1}^{p} |h_i(x)|\ \nabla\, h_i(x)$$

$$\nabla^2\,\theta(x_k,\mu_k) = \nabla^2 f(x_k) + \sum_{i=1}^{m} \mu_k\ \max\{o, g_i(x)\}\ \nabla^2\ g_i(x) + \sum_{i=1}^{p} \mu_k\ h_i(x)\ \nabla^2 h_i(x) + \sum_{i=1}^{m} \mu_k (\nabla g_i(x))\ (\nabla g_i(x))^t + \sum_{i=1}^{p} \mu_k\ \nabla h_i(x)\ (\nabla h_i(x))^t$$

Les suites $\{\mu_k\ \max\{o, g_i(x)\}\}_k$ et $\{\mu_k\ h_i(x)\}_k$ convergent vers les multiplicateurs de Lagrange associés au problème (P).

Les trois premiers termes de $\nabla^2 \theta(x_k, \mu_k)$ convergent vers le hessien du Lagrangien qui a des valeurs propres finies. Les deux termes restants contiennent le terme μ_k ; cette suite $\{\mu_k\}$ converge vers l'infini et engendre le mauvais conditionnement de la matrice hessienne.

Afin d'éviter ce mauvais conditionnement, on va introduire un autre type de méthodes. Il s'agit des méthodes de pénalité exacte.

6. METHODES DE PENALITE EXACTE

Le but de ces méthodes est de résoudre un problème de minimisation sans contrainte. Ces fonctions à minimiser se présentent sous la forme:

$$\theta(x,\mu) = f(x) + \mu\left(\sum_{j=1}^{m} g_j(x)_+ + \sum_{j=1}^{p} |h_j(x)|\right)$$

où

$$g_j(x)_+ = \max\{0, g_j(x)\}$$

Il est à noter que $\theta(x,\mu)$ est une fonction non différentiable.

Une méthode de pénalité est dite exacte ssi il existe un paramètre μ^* tel que :

$$\min_{x \in R^n} \theta(x,\mu^*) = x^*$$

où

x^* est le point résolvant le problème (P).

Il s'agit d'une définition introduite par EVANS-GOULD and TOLLE [11].

Il serait intéressant de faire le lien entre les solutions locales de la fonction de pénalité et les solutions locales du problème de programmation non linéaire (P).

THEOREME 2.3 :

S'il existe $\mu^* \geq o$ tel que pour tout $\mu \geq \mu^*, \theta(x^*,\mu) \leq \theta(x,\mu)$ pour tout x admissible dans un ensemble y contenant x^*;
Alors, x^* est solution du problème (P) soumis à la condition supplémentaire : $x \in y$.

Démonstration :

On montre d'abord que x^* est admissible pour (P).
Supposons par l'absurde que x^* ne l'est pas, alors $P(x^*) > o$.
Choisissons $\hat{x}$ un point admissible se trouvant dans y et μ tel que

$$\mu > max\{(f(\hat{x})-f(x^*))/P(x^*), \mu^*\}.$$

On obtient alors :

$$f(\hat{x}) = \theta(\hat{x},\mu\ ) \geq \theta(x^*,\mu) = f(x^*) + \mu\ P(x^*) > f(\hat{x})$$

où la dernière inégalité peut être déduite par le choix spécifique de μ.

Pour montrer que x^* est optimal pour (P) , prenons x un autre point admissible dans y et $\mu \geq \mu^*$.

Alors $\quad f(x^*) = \theta(x^*,\mu) \leq \theta(x,\mu) = f(x)$

autrement dit, x^* est optimal pour (P) avec la contrainte supplémentaire que $x \in y$.

THEOREME 2.4 :

Soient $x_o \in S$ et $I = \{i : g_i(x_o) = o\}$
Supposons que la condition suivante est satisfaite :

$\mathcal{G}$: Pour tout $y \in R^n$ non nul tel que $\nabla g_i(x_o)^t\ y \leq o$
pour $i \in I$ et $\nabla h_j(x_o)^t\ y = o$

pour tout $j = 1..p$,
on peut déduire que $\nabla f(x_o)^t y > o$.

Alors, il existe $\mu_o > o$ tel que si $\mu \geq \mu_o$ alors $\theta(x,\mu)$ a un minimum local isolé en x_o.

Remarque :

On peut montrer que sous ces conditions, x_o est un minimum local isolé de f sur S.

Démonstration :

Puisque $g_1,\ldots,g_m$ sont continues, il existe $\delta>o$ tel que $g_i(x) < o$ pour tout $x \in B(x_o,\delta)$ et $i \notin I$. $B(x_o,\delta)$ est une boule ouverte centrée en x_o de rayon δ.
Supposons maintenant pour obtenir une contradiction qu'on dispose d'une suite $\{\mu_n\}$ de scalaires positifs telle que $\mu_n \to +\infty$ quand $n \to +\infty$ et telle que $\theta(x,\mu_n)$ n'ait pas un minimum local isolé en x_o pour tout n. Alors, pour chaque n il y a un point x_n tel que $o < ||x_n - x_o|| < min\{\delta, 1/\mu_n\}$ mais pour lequel $\theta(x_n,\mu_n) \leq \theta(x_o,\mu_n)$.
Notons par $y_n = x_n - x_o$ pour tout n et remarquons que $y_n \to o$ quand $n \to \infty$. Considérons la suite $\{y_n/||y_n||\}$ de vecteurs dans l'ensemble compact $\{Z \in R^n : ||Z|| = 1\}$. De cette suite, on peut extraire une sous-suite convergeante. Soit y_o cette limite. On obtient que $||y_o|| = 1$. Supposons sans perte de généralité que $y_n/||y_n|| \to y_o$ quand $n \to \infty$.

Montrons que $\nabla g_i(x_o)^t y_o \leq o$ pour tout $i \in I$.
Supposons, par l'absurde, que $\nabla g_n(x_o)^t y_o > o$ pour $n \in I$.
Alors, pour tout n

$$\frac{1}{||y_n||} g_n(x_n) = \frac{1}{||y_n||}\left[g_n(x_n) - g_n(x_o)\right]$$

$$= \nabla g_n(x_o)^t \frac{y_n}{||y_n||} + \frac{o(||y_n||)}{||y_n||} .$$

Cette dernière expression converge vers $\nabla g_n(x_o)^t y_o$ quand $n \to \infty$. Donc, pour n suffisamment grand $g_n(x_n) > o$ de telle sorte que $g_n(x_n)_+ = g_n(x_n)$.

Mais alors, pour n suffisamment grand, on peut dire que

$$\frac{1}{\mu_n}\left[\theta(x_o,\mu_n) - \theta(x_n,\mu_n)\right] \leq \frac{1}{\mu_n}\left[f(x_o) - f(x_n)\right] - g_n(x_n)$$

ou

$$\frac{1}{\mu_n}\frac{1}{||y_n||}\left[\theta(x_o,\mu_n) - \theta(x_n,\mu_n)\right] \leq \frac{-1}{\mu_n}\nabla f(x_o)^t \frac{y_n}{||y_n||}$$

$$- \nabla g_n(x_o)^t \frac{y_n}{||y_n||} + \frac{o(||y_n||)}{||y_n||} .$$

Le deuxième membre de l'inégalité converge vers $-\nabla g_n(x_o)^t y_o < o$ quand $n \to \infty$, ce qui implique pour n assez grand, $\theta(x_o,\mu_n) < \theta(x_n,\mu_n)$.

La contradiction permet d'établir que $\nabla g_i(x_o)^t y_o \leq o$ pour tout $i \in I$.

De la même façon, on peut montrer que $\nabla h_j(x_o)^t y_o = o$ pour $j = 1..p$. D'où, par l'hypothèse $\mathcal{G}$, on déduit que $\nabla f(x_o)^t y_o > o$.

Mais alors,

$$\frac{1}{||y_n||}\left[f(x_o) - f(x_n)\right] = -\nabla f(x_o)^t \frac{y_n}{||y_n||} + \frac{o(||y_n||)}{||y_n||}$$

$$\to -\nabla f(x_o)^t y_o \text{ quand } n \to \infty$$

impliquant que pour n suffisamment grand

$$\ell(x_o) < \ell(x_n)$$

et donc $$\theta(x_o, \mu_n) < \theta(x_n, \mu_n)$$

Cette dernière contradiction permet d'établir le théorème.

La condition garantissant le caractère exact de la fonction de pénalité $\theta(x,\mu)$ dépend de la fonction objectif ℓ. Pietrzykowski [15] a démontré une autre condition ne dépendant pas cette fois-ci de la fonction ℓ.

THEOREME 2.5 :

Soit x_o un minimum local de ℓ sur S.
Soient ℓ, g, h de classe C^1 dans un voisinage de x_o et soit les gradients des contraintes actives linéairement indépendants.

Alors il existe $\mu_o > o$ un nombre réel tel que pour tout $\mu \geq \mu_o$ la fonction $\theta(x,\mu)$ a un minimum isolé et local en x_o.

On pourra trouver la démonstration dans l'article de Pietrzykowski [15] .

Montrons par deux exemples les différences entre ces deux conditions.

Exemple 1 : Soient $\ell, g_1, g_2 : R \to R$ définies par $\ell(x) = x$;
$g_1(x) = x$; $g_2(x) = -x$.
Les deux contraintes sont actives en $x_o = o$ qui est par ailleurs le seul point admissible et donc le point optimal. Comme il n'y a pas de vecteur $y \in R$ non nul tel que $y\, g_i'(o) \leq o$ pour $i = 1,2$, l'hypothèse G du théorème 6.2 est trivialement vérifiée.

Par contre les gradients $g_1'(o) = 1$ et $g_2'(o) = -1$ ne sont pas linéairement indépendants. L'hypothèse de Pietrzykowski n'est pas vérifiée.

Exemple 2 : Soient f et $g : R \to R$ définies par $f(x) = -x^3$ et $g(x) = -x$. La contrainte $g(x)$est active au minimum $x_o = o$. Remarquons que $g'(o) = 1$ et donc si $y = -1$, $y\, g'(o) = -1 \leq o$ mais $yf'(o) = o$.
La condition de Pietrzykowski n'implique pas l'hypothèse G.

Pour terminer ce chapitre sur les méthodes de pénalité, signalons que S-P Han et O.L. Mangasarian [13] ont également démontré une condition suffisante d'existence d'un minimum local isolé en x_o. Avant d'énoncer le théorème, introduisons la définition de qualification des contraintes (selon Han-Mangasarian).

Définition :

Soit $I = \{i : g_i(x_o) = o\}$

Les contraintes $g(x) \leq o$ et $h(x) = o$ satisfont à la qualification des contraintes en x_o (selon Han-Mangasarian) si

1. g est différentiable en x_o
2. h est de classe C^1 en x_o
3. $\nabla h_i(x_o)$ $i = 1..p$ sont linéairement indépendants
4. il existe un z de R^n tel que

$$\nabla g_i(x_o)\, z < o \qquad i \in I$$
$$\nabla h_i(x_o)\, z = o \qquad i = 1..p$$

Nous pouvons maintenant énoncer la condition suffisante de Han-Mangasarian.

THEOREME 2.6 :

Soient $\ell, g, h,$ de classe C^1 dans un voisinage de x_o : minimum local strict pour le problème (P) et supposons que la qualification des contraintes soit satisfaite en x_o.

Alors, il existe $\bar{\mu} \geq o$ tel que $\forall\ \mu \geq \bar{\mu}$ x_o est un minimum local de $\theta(x, \mu)$.

C'est cette dernière condition que nous exploiterons dans l'étude de l'algorithme primal.

CHAPITRE III : LA METHODE PRIMALE DE RESOLUTION D'UN SYSTEME D'EQUATIONS LINEAIRES SURDETERMINES AU SENS DE TECHEBYCHEFF.

1. INTRODUCTION

Nous allons maintenant nous attacher à résoudre le programme linéaire développé dans l'introduction, que nous rappelons ici : il s'agit de

$$\min_{v} \; c_o^t \, v \qquad \text{(P) primal}$$

$$\text{s.c.} \quad c_j^t \, v \geq \delta_j \qquad j = 1,\ldots,2m$$

où

$$v^t = (\xi \; x^t)$$

$$c_o^t = (1 \; o \; \ldots \; o)$$

$$c_j^t = (1 \; - a_j^t) \qquad j = 1 \ldots m$$

$$c_{m+j}^t = (1 \; + a_j^t)$$

$$\delta_j = - \ell j \qquad j = 1 \ldots m$$

$$\delta_{m+j} = + \ell j$$

2. PREMIERE METHODE PRIMALE

La première méthode primale que nous allons décrire est un algorithme mis au point par A.K. Cline [7] et développé par R.Bartels, A.Conn et C.Charalambous [2] .

Il s'agit en fait de la méthode du gradient projeté appliquée au problème (P). Rappelons brièvement les différentes étapes:

1. Soit v un point admissible. Déterminer les contraintes actives en v, i.e. :

trouver $I^o = \{j_1, \ldots, j_k\}$ tel que

$$c_{j_i}^t v - \delta_{j_i} = o \qquad i = 1..k. \quad (1)$$

Construire

$$N = (c_{j_1} \ldots c_{j_k})$$

Nous supposons dans cet algorithme que les colonnes de N sont linéairement indépendantes, i.e. : on suppose que l'hypothèse de non-dégénérescence est vérifiée.

2. Calculer $P \nabla(c_o^t v) = P c_o$ où P est la matrice de projection orthogonale sur le sous-espace des contraintes actives.

3. A. Si $P c_o \neq o$, alors la direction de descente admissible d vaut :

$$d = - P c_o. \quad (2)$$

B. Si $P c_o = o$, alors on peut exprimer le gradient de la fonction objectif $(c_o^t v)$ comme combinaison linéaire des contraintes actives :

$$c_o = \sum_{i=1}^{k} \eta_{j_i} c_{j_i} . \quad (3)$$

Par hypothèse de non-dégénérescence, les coefficients η_{j_i} sont déterminés de manière unique.

. si $\eta_{j_{i^*}} < o$ pour un certain j_{i^*} dans I^o, alors

$$I^o := I^o \setminus \{j_{i^*}\} \quad (4)$$

et on construit $\hat{P}$ la matrice de projection orthogonale sur le nouveau sous-espace des contraintes actives. La direction de descente admissible d devient :

$$d = - \hat{P} c_o \quad (5)$$

. si quel que soit $i = 1,..,k$ $\eta_{j_i} \geq o$, alors le point courant est un point de Kuhn-Tucker du problème (P) (cfr. théorème 1.1.).

4. Effectuer une recherche linéaire le long de la direction de descente, i.e.:

$$v := v + \alpha_{min}\, d \qquad (6)$$

où

$$\alpha_{min} = min\{\alpha_j :\ \alpha_j = (\delta_j - c_j^t\, v) \,/\, c_j^t\, d$$

$$\text{et } \alpha_j > o \quad \text{et} \quad j \notin I^o\}$$

De plus, $I^o := I^o \cup \{indice\ j^* :\ \alpha_{min} = \alpha_{j^*}\}$. Aller en 2.

Convergence de l'algorithme :

L'algorithme qui vient d'être rappelé converge en un nombre fini d'étapes.

En effet, la première étape est exécutée une seule fois. La deuxième étape est exécutée une seule fois par itération. L'étape 3 A. n'est répétée qu'un nombre fini de fois, car chaque exécution est associée à une certaine valeur de k $(o \leq k \leq m)$ et cette valeur de k augmente à chaque itération. Chaque exécution de l'étape 3 B. est associée à une certaine matrice N. On ne peut construire qu'un nombre fini de ces matrices. Supposons que cette étape 3 B. soit exécutée deux fois séparément avec la même matrice N. Nous noterons par v_1 et v_2 les deux points associés à ces deux exécutions de l'étape 3 B. Posons

$$S = diag(\varepsilon_{j_1} \ldots \varepsilon_{j_k})$$

où

$$\varepsilon_{j_i} = \begin{cases} -1 & \text{si } j_i \leq m \\ +1 & \text{si } j_i > m \end{cases}$$

et

$$\Delta = diag\,(\delta_{j_1} \ldots \delta_{j_k})$$

Par définition de N, les deux point v_1 et v_2 doivent vérifier les équations suivantes :

$$N^t \quad v_1 - S\,\Delta = o$$

$$N^t \quad v_2 - S\,\Delta = o$$

où

$$c^t_{j_i} \quad v_1 - \varepsilon_{j_i}\,\delta_{j_i} = o \qquad i = 1,..,k$$

$$c^t_{j_i} \quad v_2 - \varepsilon_{j_i}\,\delta_{j_i} = o \qquad i = 1,..,k$$

ce qui est encore équivalent à :

$$-\xi_1 = \varepsilon_{j_i}\,(a^t_{j_i}\,x_1 - \delta_{j_i}) \qquad i = 1,..,k$$

$$-\xi_2 = \varepsilon_{j_i}\,(a^t_{j_i}\,x_2 - \delta_{j_i}) \qquad i = 1,..,k\,.$$

En soustrayant ces deux dernières équations, on déduit que :

$$-(\xi_1 - \xi_2) = \varepsilon_{j_i}\,(a^t_{j_i}(x_1 - x_2)) \qquad i = 1,..,k\,.$$

Cette dernière équation est équivalente à :

$$N^t\,(v_1 - v_2) = o$$

i.e. $(v_1 - v_2)$ est dans le noyau de N^t. Or le gradient de la fonction objectif peut s'écrire comme combinaison linéaire des colonnes de N.
D'où, on déduit que : $c^t_o\,(v_1 - v_2) = o$

$$\text{i.e. } \xi_1 = \xi_2\,.$$

Ce qui veut dire que chaque exécution de l'étape 3 B. avec la même matrice N doit être associée à la même valeur ξ. Or chaque exécution de l'étape 3 B. entraîne une décroissance stricte de la fonction objectif.

D'où, chaque exécution de l'étape 3 B. doit être associée à une matrice N unique et par conséquent, seulement un nombre fini d'itérations de l'étape 3 B. sont possibles.

Comme chaque itérarion de l'algorithme inclut ou bien l'étape 3 A. ou bien l'étape 3 B., l'algorithme comprendra un nombre fini d'itérations. De plus, par construction, l'algorithme se terminera en un point v vérifiant les conditions nécessaires et suffisantes d'optimalité.

Cette première méthode a été améliorée par R.Bartels, A.Conn, et Y.Li [4] . C'est cette nouvelle méthode que nous allons maintenant envisager.

3. SECONDE METHODE PRIMALE

Cette nouvelle approche du problème (P) primal utilise la notion de pénalité exacte développée dans le chapitre II.

Considérons un paramètre $\mu > o$ fixé. On définit la fonction de pénalité ρ comme suit :

$$\rho(v, \mu) = \mu\, c_o^t\, v - \sum_{j=1}^{2m} \min(o, c_j^t\, v - \delta_j) \qquad (7)$$

Remarques :

1. La fonction de pénalité ρ est non différentiable. Ceci pose un problème non négligeable si on veut appliquer un algorithme du type gradient projeté. Cependant, nous verrons que dans notre cas, cette non différentiabilité n'aura aucune influence.

2. Comme nous l'avons vu au deuxième chapitre, les méthodes de pénalité consistent à transformer un problème de minimisation avec contraintes en un problème de minimisation sans contrainte. Pour pouvoir travailler avec cette fonction de pénalité ρ, il

faut s'assurer qu'on a bien un problème de pénalité exacte. Plusieurs conditions suffisantes ont été énoncées (cfr. théorèmes 2.4, 2.5 et 2.6). Vérifions les conditions du théorème 2.6 . Premièrement, les fonctions f et g doivent être de classe c^1, or pour le problème (P) la fonction objectif f et les contraintes g sont linéaires. Deuxièmement, la qualification des contraintes au sens de Han-Mangasarian doit être vérifiée en x_o. Les contraintes sont différentiables en x_o. De plus, on peut facilement trouver un vecteur v de R^{n+1} tel que :

$$\nabla g_i(x_o)\, v < o$$

quel que soit i dans l'ensemble des indices actifs en x_o.
En effet,

$$\nabla g_i(x_o)\, v = - c_i^t \cdot v$$

et pour que cette quantité soit strictement négative, il suffit de prendre v comme suit :

$$v^t = (\xi \quad x^t)$$

où

$$x^t = (o, \ldots, o) \qquad R^n$$

$$\xi > o$$

La seule difficulté pourrait venir du fait que x_o doit être un minimum local <u>strict</u>. En effet, l'ensemble des solutions de (P) forme un ensemble convexe (cfr. propriété 0.7); par conséquent si cette solution n'est pas unique, x_o ne sera pas un minimum local <u>strict</u>.

Les deux autres conditions suffisantes (théorèmes2.4 et 2.5) ne permettent pas non plus de déduire que la fonction définie en(7) détermine un problème de pénalité exacte quelle que soit la matrice A. A notre connaissance, nous ne disposons pas d'autres théorèmes qui pourraient régler ce problème.

A présent, nous pouvons continuer à développer le second algorithme primal.

Quel que soit le point ν de R^{n+1}, l'équation (7) peut être réécrite sous la forme :

$$\begin{aligned} \rho(\nu,\mu) = \mu c_o^t \nu &- \sum_{j \in I^o} \min(o, c_j^t \nu - \delta_j) \\ &- \sum_{j \in I^+} \min(o, c_j^t \nu - \delta_j) \\ &- \sum_{j \in I^-} \min(o, c_j^t \nu - \delta_j) \end{aligned}$$

où les ensembles d'indices sont donnés par :

$$\begin{aligned} I^o &= I^o(\nu) = \{j : c_j^t \nu = \delta_j\} = \{j_1, \ldots, j_k\} \\ I^+ &= I^+(\nu) = \{j : c_j^t \nu > \delta_j\} \\ I^- &= I^-(\nu) = \{j : c_j^t \nu < \delta_j\}. \end{aligned} \tag{8}$$

Quel que soit $d \in R^{n+1}$ et quel que soit $\lambda \geq o$ suffisamment petit, on a :

$$\begin{aligned} \rho(\nu+\lambda d, \mu) &= \rho(\nu,\mu) + \lambda(\mu c_o^t d - \sum_{j \in I^-} c_j^t d \\ &\quad - \sum_{j \in I^o} \min(0, c_j^t d)) \\ &= \rho(\nu,\mu) + \lambda h^t d + \lambda \sum_{j \in I^o} \nabla_j^- c_j^t d \end{aligned} \tag{9}$$

où

$$h = \mu c_o - \sum_{j \in I^-} c_j \tag{10}$$

et

$$\begin{aligned} \nabla_j^- &= o \qquad \text{si} \quad c_j^t d \geq o \qquad j \in I^o \\ &= -1 \qquad \text{si} \quad c_j^t d < o \end{aligned}$$

On définit la matrice

$$N = (c_{j_1} \ \ldots \ c_{j_k}) \tag{11}$$

et

$$P = I - N\,(N^t N)^{-1}\,N^t . \tag{12}$$

La matrice P est en fait le projecteur orthogonal sur le sous-espace noyau de N^t. Deux cas peuvent alors se présenter :

Cas 1. $Ph \neq o$

On définit la direction de descente d par :

$$d = - Ph . \tag{13}$$

On peut voir que d est une direction de descente pour la fonction ρ, car

$$\rho(v + \lambda d, \mu) = \rho(v, \mu) + \lambda h^t d$$

et

$$h^t d < o .$$

Il suffit d'utiliser la définition de P pour déduire cette dernière inégalité stricte.

Cas 2 $Ph = o$

Si on suppose que les colonnes de N sont linéairement indépendantes, h peut s'exprimer de manière unique comme combinaison linéaire des c_{j_i}, $i = 1..k$, i.e. :

$$h = \sum_{i=1}^{k} \eta_{j_i}\ c_{j_i} .$$

L'équation (9) devient :

$$\rho(v + \lambda\, d, \mu) = \rho(v, \mu) + \lambda \sum_{i=1}^{k} (\eta_{j_i} + \nabla^-_{j_i}) c^t_{j_i} d \tag{14}$$

Il faut encore envisager deux possibilités, à savoir :

a) il existe une composante $\eta_{i^*} < o$ pour un certain $i^* \in I^o$. Alors on choisit d tel que :

$$c_j^t\, d = o \qquad j \in I^o \quad j \neq i^* \qquad (15)$$
$$c_{i^*}^t\, d = 1 .$$

Remarquons que (15) implique $\nabla_j^- = o$, $j \in I^o$, l'équation (9) devient :

$$\rho(\nu + \lambda d, \mu) = \rho(\nu, \mu) + \lambda\, h^t\, d$$

De plus,

$$h^t\, d < o$$

Car,

$$h^t\, d = \eta_{i^*}\, c_{i^*}^t\, d = \eta_{i^*} < o .$$

b) Toutes les composantes de η sont positives ou nulles quel que soit i dans I^o. Alors, il n'existe pas de vecteur de direction de descente d pour ρ sans qu'une des contraintes actives ne devienne violée.

Si toutes les composantes de η sont positives ou nulles et si aucune contrainte n'est violée, alors le point ν est optimal pour (P). En effet, le point ν est un point vérifiant les conditions de Kuhn-Tucker.

D'autre part, si au moins une contrainte est violée, alors le paramètre μ sera diminué et on minimisera la fonction ρ à partir du point ν courant.

Si $\lambda > o$ suffisamment petit, les ensembles d'indices changent comme suit :

$$I^o(\nu + \lambda d) = I^o(\nu) \qquad \text{si } d \text{ est donné par (13)}$$
$$= I^o(\nu) - j_{i^*} \qquad \text{si } d \text{ est donné par (15)}$$
$$I^+(\nu + \lambda d) = I^+(\nu) \qquad \text{si } d \text{ est donné par (13)}$$
$$= I^+(\nu) + j_{i^*} \qquad \text{si } d \text{ est donné par (15)}$$
$$I^-(\nu + \lambda d) = I^-(\nu)$$

Quelle que soit la façon dont on détermine d, on peut écrire :

$$\rho(v+\lambda\, d, \mu) = \rho(v, \mu) + \lambda\, h^t d$$

Remarque :

La non-différentiabilité de ρ ne pose pas de problème. En effet, le vecteur défini en (10) est le gradient de la fonction différentiable suivante :

$$\mu\, c_o^t\, v - \sum_{j \in I^+ \cup I^-} \min\,(o, c_j^t v - \delta_j)\,. \qquad (16)$$

Le choix du vecteur de direction de descente peut se faire de deux manières :

1. *d peut être déterminé par (13)* :

Les contraintes actives sont maintenues actives par cette direction de descente d. Par conséquent, les contraintes négligées dans (16) n'apportent pas de contribution supplémentaire au vecteur d.

2. *d peut être déterminé par (15)* :

Toutes les contraintes sauf une sont maintenues actives. La contrainte qui sort de l'ensemble des contraintes actives devient strictement satisfaite. Par conséquent, comme ci-dessus, les contraintes négligées dans (16) n'apportent pas de contribution supplémentaire à d.

Il ne reste plus maintenant qu'à déterminer la longueur du pas λ.

Jusqu'à présent, on a toujours supposé $\lambda \geq o$ suffisamment petit. En fait, on demandait que λ soit tel que :

$$o \leq \lambda \leq \lambda^{(1)}$$

où $\lambda^{(1)}$ est le minimum de

$$\Lambda^{(1)} = \{\lambda : \lambda = (\delta_j - c_j^t \nu) / c_j^t d \quad \text{et } \lambda > o$$

$$\text{et } j \in I^+ \cup I^-\}. \qquad (17)$$

Si

$$\lambda^{(1)} = (\delta_{j^*} - c_{j^*}^t \nu) / c_{j^*}^t d \qquad (18)$$

pour j^* dans I^+, alors une augmentation de λ au-delà de $\lambda^{(1)}$ provoquera une violation de la contrainte j^*. L'indice j^* devra passer de I^+ vers I^-; de plus, le vecteur h devra être modifié :

$$h := h - c_{j^*}. \qquad (19)$$

D'autre part, si :

$$\lambda^{(1)} = (\delta_{j^*} - c_{j^*}^t \nu) / c_{j^*}^t d \qquad (20)$$

pour j^* dans I^-, alors une augmentation de λ au-delà de $\lambda^{(1)}$ provoquera la satisfaction de la contrainte j^*. L'indice j^*devra passer de I^- vers I^+; de plus, le vecteur h sera modifié :

$$h := h + c_{j^*}. \qquad (21)$$

Il serait utile d'envisager une augmentation de λ au-delà de $\lambda^{(1)}$ si :

$$h^t d < o$$

pour le vecteur h mis à jour par (19) ou (21); c'est-à-dire, si d continue à être un vecteur de direction de descente pour :

$$\lambda > \lambda^{(1)}$$

suffisamment petit.

On peut continuer à augmenter λ jusqu'à la valeur $\lambda^{(t)}$ avec λ le minimum de l'ensemble $\Lambda^{(t)}$, où :

$$\Lambda^{(\ell)} = \Lambda^{(\ell-1)} - min\{\Lambda^{(\ell-1)}\}$$

pour $\ell = 2, \ldots, t$.

L'indice t est le premier indice pour lequel d n'est plus une direction de descente après avoir corrigé h comme en (19) ou en (21). On redéfinit v

$$v := v + \lambda^{(t)} d .$$

On note que l'indice j^* doit être transféré dans $I^0(v)$ au nouveau point v, où j^* est l'indice, (18) ou (20), auquel la valeur $\lambda^{(t)}$ correspond.

Remarques :

1. Le premier algorithme primal correspond à l'algorithme décrit ci-dessus si on prend à chaque itération

$$\lambda = \lambda^{(1)}$$

i.e. on s'arrête au premier λ possible.

2. Si on veut choisir un point de départ admissible, il suffit de prendre :

$$\xi = \max_i \{a_i^t x - b_i, -a_i^t x + b_i\}$$

4. IMPLEMENTATION

Pour pouvoir implémenter cet algorithme, il faut encore expliquer plusieurs points.

Comme nous l'avons envisagé dans le premier chapitre concernant l'algorithme du gradient projeté, la matrice des contraintes actives sera factorisée sous la forme :

$$N = Q \begin{bmatrix} R \\ O \end{bmatrix} = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R \\ O \end{bmatrix} = Q_1 R$$

où Q est une matrice orthonormée de dimension $n+1$ et R est une matrice triangulaire supérieure d'ordre k. La matrice Q peut être décomposée en 2 parties : Q_1 et Q_2 où Q_1 est formée des k premières colonnes de la matrice Q et Q_2 des $(n-k+1)$ dernières.

1. Pour évaluer le vecteur Ph, on utilise le théorème 1.2. et on déduit que :

$$P = Q_2 \, Q_2^t \, .$$

Pour obtenir,

$$d = -Ph$$

on pose

$$d = -Q_2 \, Q_2^t \, h \, .$$

On remarque que $d=o$ si et seulement si $Q_2^t \, h = o$; par conséquent, on ne doit pas évaluer d dans ce cas.

2. Pour résoudre $N \, \eta = h$, on pose

$$u = Q_1^t \, h$$

et on résoud l'équation suivante par substitution

$$R \, \eta = u \, .$$

3. Pour résoudre $N^t \, d = e_{j_{i^*}}$, on résoud par substitution

$$R^t \, u = e_{j_{i^*}}$$

et on pose

$$d = Q_1 \, u \, .$$

Nous pouvons à présent donner les grandes lignes de l'algorithme.

er

timum := false

péter

Construire N

Factoriser N : $N = QR \rightarrow N = Q_1 R$.

Si rang(N) < k dégénérescence. STOP

Sinon

calcul de $Q_2^t h$

si $Q_2^t h \neq 0$ alors

. $d = - Q_2 (Q_2^t h)$

. calcul de λ

, $v := v + \lambda d$

sinon

. déterminer η par $N \eta = h$

. s'il existe $\eta_i < 0$ alors

- déterminer d comme solution de $N^t d = e_i$

- mise à jour de I^0 et I^+

- calcul de λ

- $v := v + \lambda d$

sinon

- optimum := true

fin si

fin si

fin si

jusqu'à 1' "optimum"

$\mu := \mu / 8$

squ'à ce que le point soit admissible

n.

t de départ ν et la valeur du paramètre μ.
rtir de là, on détermine les trois ensembles d'indices
I^+, et I^-.

Enfin, il faut remarquer que tous les tests de compa-
on d'un nombre par rapport à zéro se font de la façon sui-
e : x est considéré comme nul si et seulement si :

$$|x| < eps$$

eps est la tolérance que l'on exige. Une discussion à pro-
de la valeur de cette dernière sera faite dans le chapitre
vant consacré aux résultats des tests numériques.

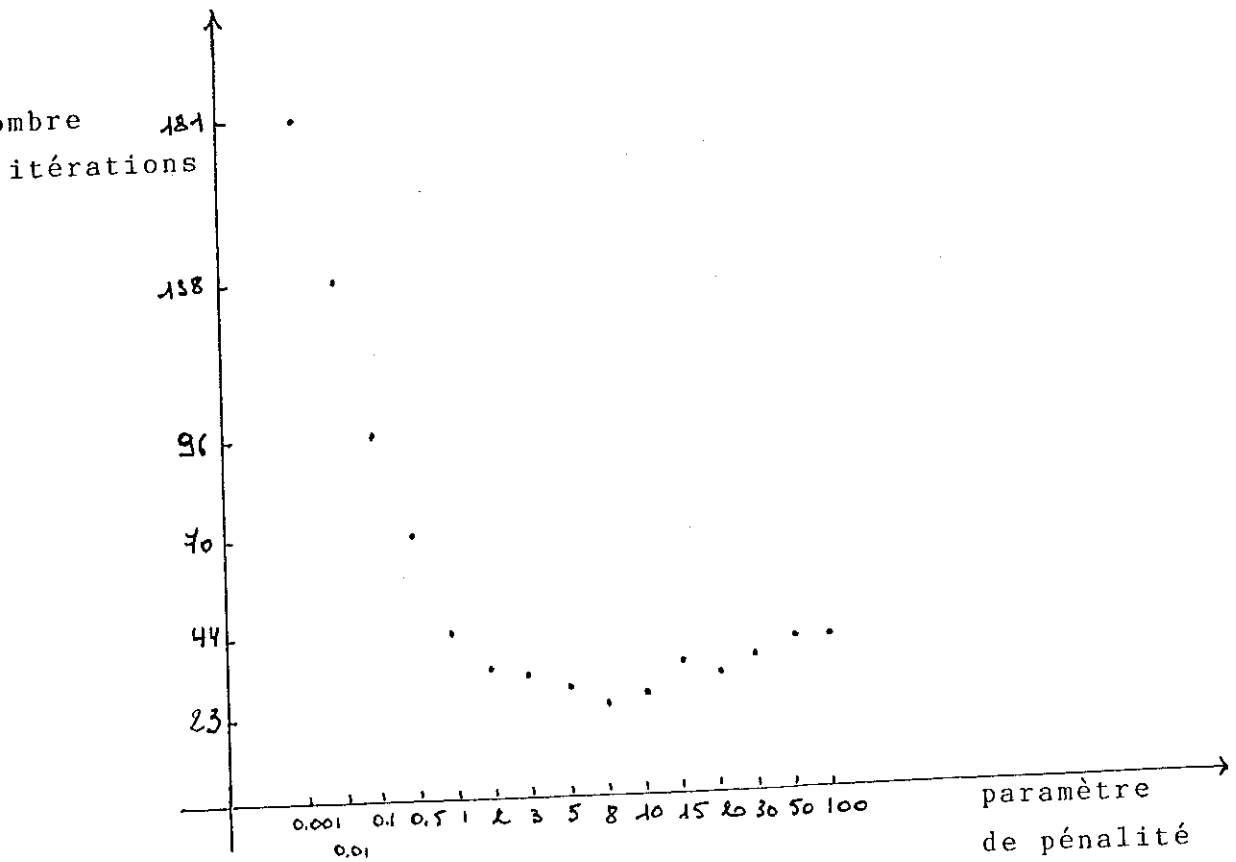
CHOIX DE LA TOLÉRANCE ET DU PARAMÈTRE DE PÉNALITÉ

On a choisi de mettre le paramètre de tolérance *eps* à valeur de 10^{-16} pour tous les tests de l'algorithme, y compris ux concernant la décomposition QR.

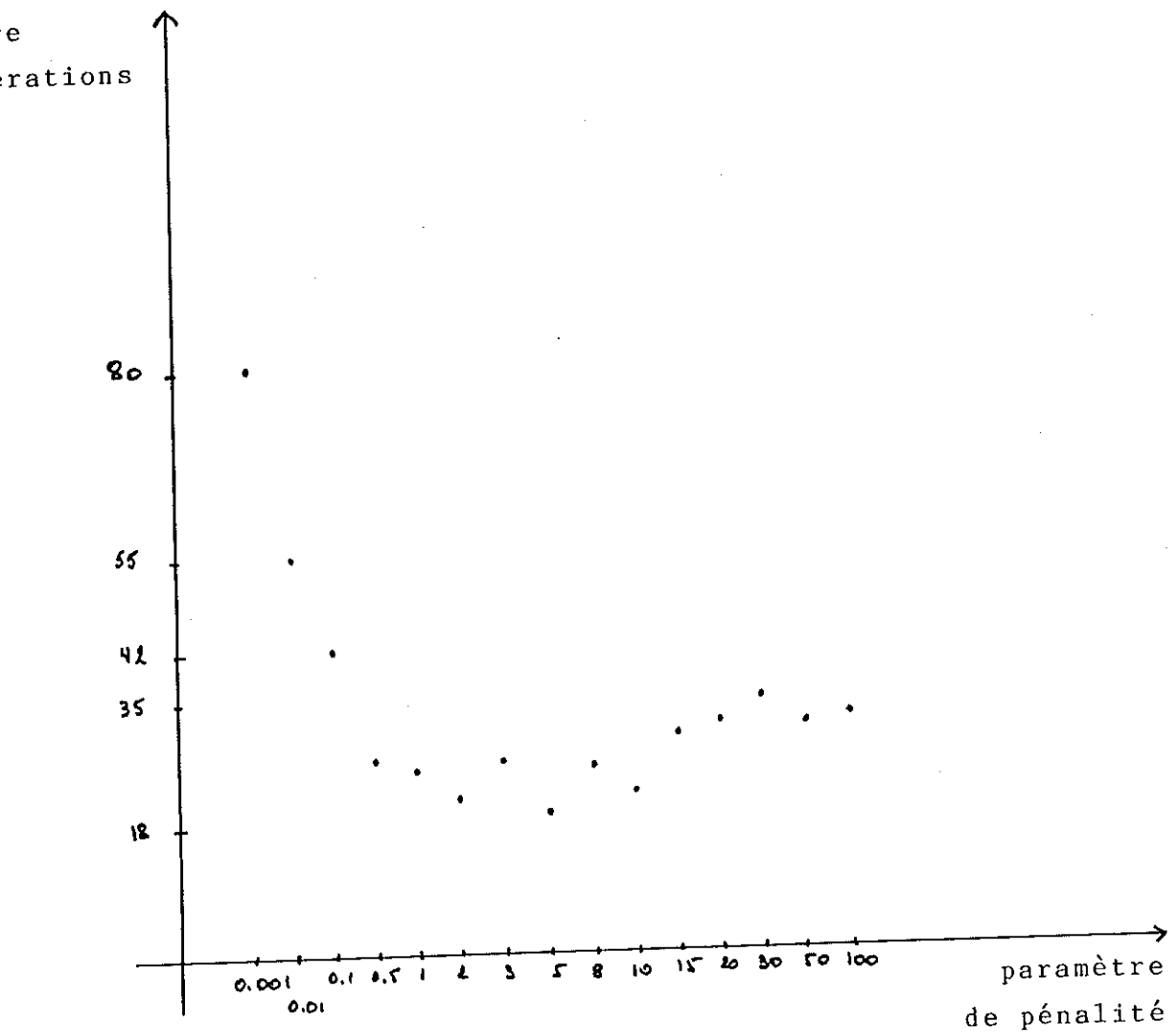
En ce qui concerne le paramètre de pénalité, nous avons supposé fixé strictement positif. Pour décider de la leur à choisir, il a fallu tester quelques problèmes et prendre lle qui sur l'ensemble de ces tests minimise le nombre d'ité- tions. Voici quelques exemples :

emple : Approximer $f(z) = e^z$ évaluée pour z = 0.0 (0.01) 2.0 (i.e.m. = 201) par un polynôme de degré 7.

Point de départ : le vecteur nul.



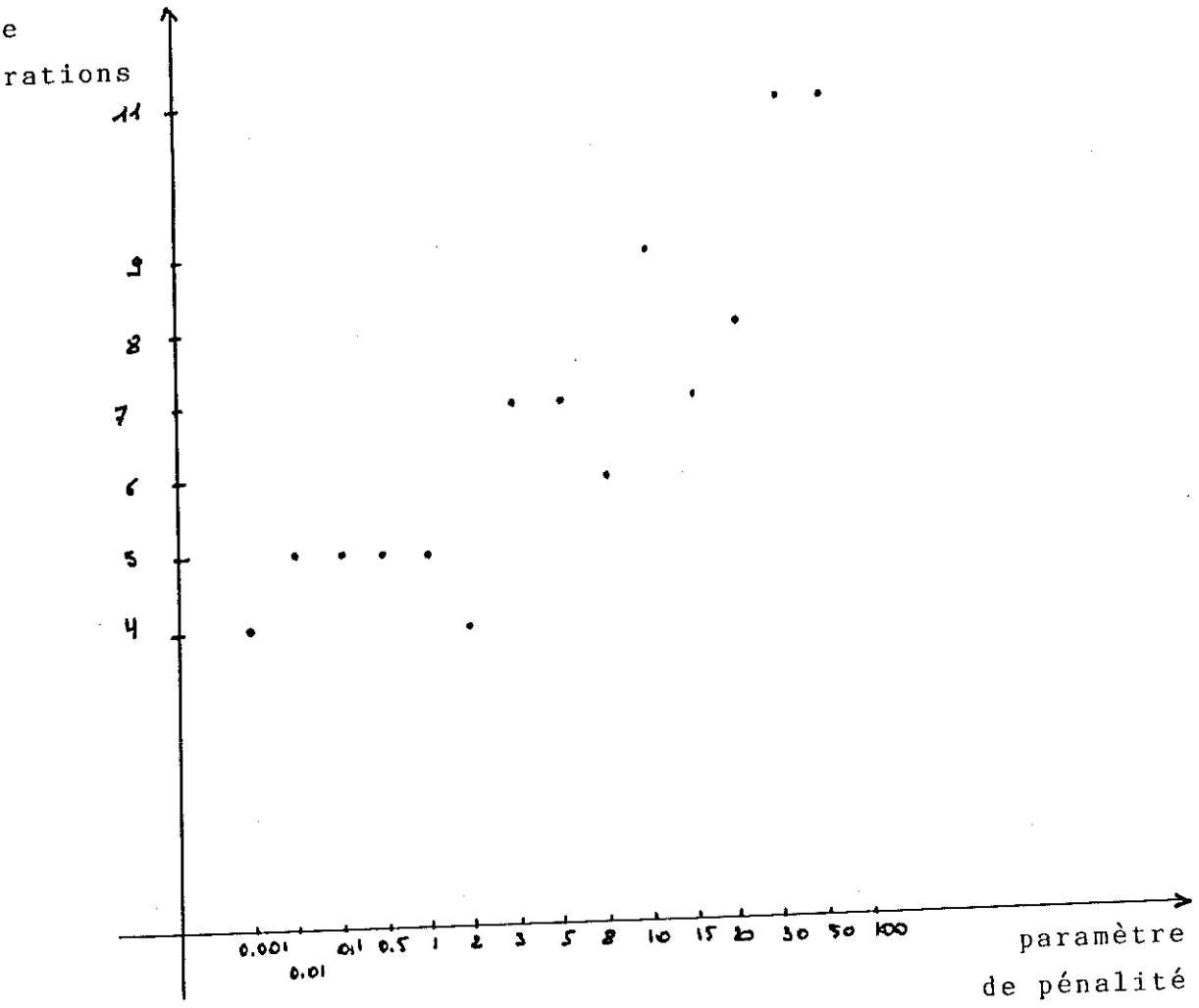
le 2 : Approximer $f(z) = \sin z\, e^{-z}$ évaluée pour z = 0.0 (0.02) 4.0 (i.e m = 201) par un polynôme de degré 7.
Le point de départ admissible est le vecteur nul.



le 3 : Approximer f(z) = sin ($\pi z/2$) évaluée pour z = 0.0 (0.01) 1.0 (i.e m = 101) par un polynôme de degré 2 .

Le point de départ = ($-0.1191\ 10^{-1}$ $0.1824\ 10^{+1}$ 0.8)

= BP Point



ution du paramètre de pénalité. Ce dernier est indiqué

thèses.

	exemple 1	exemple 2	exemple 3
001	181(0)	80(0)	4(0)
01	138(0)	55(0)	5(0)
1	96(0)	42(0)	5(0)
5	70(0)	27(0)	5(0)
	44(0)	25(0)	5(0)
	35(0)	21(0)	4(0)
	32(0)	26(0)	7(1)
	29(0)	18(0)	7(1)
	23(1)	25(1)	6(1)
.	25(1)	22(1)	9(1)
.	34(1)	30(1)	7(1)
.	33(1)	31(1)	8(1)
.	36(1)	35(1)	11(2)
.	40(1)	31(2)	11(2)
00.	41(2)	32(2)	17(2)

rques :

1. L'évolution du nombre d'itérations en fonction du para-
e de pénalité peut être très différent d'un exemple à l'autre.
si, pour le premier exemple, le nombre d'itérations diminue au
et à mesure que le paramètre de pénalité augmente jusqu'à ce qu'il
eigne la valeur 8. Après cette valeur, le nombre d'itérations se
à augmenter. Tandis que pour l'exemple 3, les itérations sont
nombreuses quand le paramètre de pénalité se situe entre les va-
rs 0.001 et 5, avec un nombre d'itérations minimum quand le para-
re vaut 2. Par conséquent, le choix d'une valeur du paramètre
a plus favorable pour certains exemples que pour d'autres.

minaires effectués avec plusieurs fonctions et points de départ
rents, la valeur choisie fut 2.

3. Pour cette valeur, on remarque que sur ces trois exemples,
e diminution du paramètre de pénalité n'a eu lieu. Ce qui si-
e que la solution du problème de pénalité avec le paramètre
si égal à 2 est la même que celle du problème original.

ES POINTS DE DEPART

Plusieurs points de départ ont été expérimentés :

Point $x_o = 0$.

Le point ν de départ de l'algorithme est constitué du point
t de la déviation ξ qui est la norme du vecteur résidu . L'al-
thme débutera avec un point admissible.

Point de Tchebycheff

On considère les n+1 points suivants

$$z_k = \cos((k-1)\pi/n) \qquad k = 1,2,\ldots,n+1 \quad (1)$$

$[-1,+1]$.

Le point de Tchebycheff (noté C point) est la meilleure
roximation au sens de Tchebycheff d'une fonction univariable,
ifiant la condition de Haar, sur la référence déterminée par les
nts z_k ci-dessus.

Les points z_k sont déterminés sur l'intervalle $[-1,+1]$. Il
fit de faire un changement de variable pour les obtenir sur l'in-
valle $[a,b]$ sur lequel on travaille.

, les C points
oblème suivant :

on veut approximer la fonction $f(x) = x + 2$ par la fonction
$a_0 x + a_1 x^2 + a_2 x^4$ sur l'intervalle $[-2,+2]$.

Les différents z_i déterminés par (1) sont:

$$z_1 = 2$$
$$z_2 = 1$$
$$z_3 = -1$$
$$z_4 = -2$$

La meilleure approximation est caractérisée par les équations

$$2 + 2 - a_0 - 2\,a_1 - 4\,a_2 = -\xi \qquad (2)$$
$$1 + 2 - a_0 - a_1 - a_2 = +\xi \qquad (3)$$
$$-1 + 2 - a_0 - a_1 - a_2 = -\xi \qquad (4)$$
$$-2 + 2 - a_0 - 2\,a_1 - 4\,a_2 = +\xi\,. \qquad (5)$$

oustrayant l'équation (3) de l'équation (4), on obtient :

$$-2 = -2\,\xi$$

on déduit que la déviation doit valoir 1. Mais si on soustrait
uation (2) de l'équation (5), ξ doit satisfaire l'équation sui-
e :

$$-4 = +2\,\xi$$

t-à-dire , la déviation doit valoir -2. Ce système est donc im-
sible !! Dans ce cas-ci les C points n'existent pas.

En prenant comme point de départ le vecteur nul, on choisit ésidu de module maximum et on le réduit à zéro. A l'itération ante, on choisit de nouveau le résidu de module maximum et on le it à zéro en maintenant à zéro le résidu annulé à la première é-. On renouvelle cette opération pendant k itérations où k est ang de la matrice A. Enfin, l'itération k+1 consiste à calculer eilleure approximation correspondant à l'équation de résidu maxi- et aux k équations déterminées lors des k premières étapes. On ent ainsi le point de Barrodale et Phillips. Il reste à déter- r la déviation ξ utilisée :

- point admissible : la déviation ξ est la norme du vec- résidu
- point non admissible : ξ est la déviation de référence ciée au point et déterminée automatiquement quand on construit oint de Barrodale et Phillips. De cette manière, l'algorithme al démarre avec k+1 contraintes actives.

DEUX PROBLEMES D'EXPERIMENTATION

PROBLEME A :

Considérons la matrice A et le vecteur ℓ suivants :

$$A = \begin{bmatrix} -1 & 1 & -1 \\ 1 & 0.25 & -0.125 \\ 1 & 0.25 & 0.125 \\ 1 & 1 & 1 \end{bmatrix} \qquad \ell = \begin{bmatrix} 0.25 \\ 0.5 \\ 2. \\ 4 \end{bmatrix}$$

vérifie facilement que A satisfait à la condition de Haar. Voici lques données supplémentaires .

- précision exigée = 10 exposant -10
- facteur de pénalité = 1.
- méthode primale utilisée = 2

onstruit le vecteur h qui vaut :

$$h^t = (\ 1 \quad 0 \quad 0 \quad 0)$$

ation 1 :

. $\rho h \neq 0$. D'où, on met

$$d = -\rho h \quad \text{où} \quad d^t = (\ -0.75 \quad 0.25 \quad 0.25 \quad 0.25)$$

. λ choisi = 7.066...6

. $v = v + \lambda d$ où $v^t = (8.45 \quad -8.233.. \quad 2.0166.. \quad 1.766..)$

. $h^t = (\ 0. \quad -1. \quad -0.25 \quad -0.125)$

ation 2 :

. $\rho h \neq 0$. D'où, on met

$$d = -\rho h \quad \text{où} \quad d^t = (\ -0.5 \quad 0.5 \quad 0. \quad 0.)$$

. Calcul

$$\Lambda = \{\ \lambda : \lambda = c_j^t v - \delta_j \ /\ c_j^t d,\ j \in I^+ \cup I^-\ et\ \lambda<0\ \}$$

$$\Lambda = \{16.6833..;\ 16.9;\ 16.9;\ 17.9583..\}$$

λ . Il faut donc modifier la procédure permettant
ongueur du pas le long de la direction de descente d pour tenir
te de ce phénomène. En effet, il faut considérer ces deux con-
ntes ensemble et par conséquent effectuer la mise à jour de
les deux contraintes à la fois.

ROBLEME B :

Au cours des tests effectués pour fixer la valeur du pa-
tre de pénalité, on a été plusieurs fois confronté au problème
ant : à une certaine itération, la déviation ξ devenait négative;
itération suivante, on ne parvenait pas à trouver une longueur
as le long de la direction de descente strictement positive.
éthode ne pouvait pas converger vers un minimum du problème de
rt.

Il faut remarquer que cela se passait souvent quand le fac-
de pénalité était relativement grand (i.e ≥ 10).

Ces problèmes ne se sont par conséquent plus posés lors de
s les tests numériques qui suivirent puisque la valeur de ce pa-
ètre fut de 2.

ESTS NUMERIQUES

Tous les tests numériques ont été effectués sur système
.

L'algorithme dual qui a servi de comparaison est celui dé-
t dans l'article de Thiran J-P et Thiry S.[16]. Cet algorithme
ermine l'approximation stricte d'un système d'équations. On sait
en résolvant un système d'équations linéaires au sens de Tchebycheff

ain degré de liberté. Déterminer ...
consiste à minimiser les résidus libres. Il faut remarquer que
d l'approximation du problème est unique , cet algorithme dégé-
en un simple algorithme d'échange. Deux types d'échange ont
utilisés : l'échange classique et l'échange optimal. Le lec-
voudra bien se référer à l'article [16] pour une description
précise de l'algorithme.

rques :

1. Je rappelle que les deux méthodes primales implantées sont :
   - la méthode de R.Bartels,A.Conn et C.Charalambous [2]
illée dans le chapitre précédent au paragraphe 2 et notée méthode 1.
   - la méthode de R.Bartels, A.Conn et Y.Li [4] dé-
llée dans le chapitre précédent au parag_raphe 3 et notée méthode 2.

2. Les tableaux qui vont suivre indiquent le nombre d'itéra-
ns nécessaires pour obtenir une solution au problème de résolu-
n d'un système d'équations linéaires surdéterminés au sens de
ebycheff. IL est à noter que pour la seconde méthode primale,
nombre de diminution du paramètre de pénalité est indiqué entre
enthèses.

PROBLEMES ALEATOIRES

Plusieurs problèmes aléatoires ont été générés en utilisant
fonction intrinsèque RAN disponible sur le système VAX.

La matrice A et le vecteur ℓ sont constitués de nombres gé-
és aléatoirement entre -100 et +100.

PROBLEME 1 : Problèmes aléatoires

		Primal				Dual			
		Méthode 1		Méthode 2		Echange classique		Echange optimal	
n	m	X_o=0	BP Points	X_o=0	BP Points	X_o=0	BP Points	X_o=0	BP Point
2	50	2	3	2(0)	4(1)	1	1	1	1
	100	2	3	2(0)	5(1)	1	3	1	3
	150	2	5	2(0)	9(1)	3	4	3	4
	200	2	4	4(1)	4(0)	0	3	0	3
6	50	6	8	7(0)	13(0)	8	4	4	4
	100	8	10	8(0)	10(0)	9	9	6	9
	150	6	14	8(0)	12(0)	8	7	8	7
	200	6	18	8(0)	30(0)	13	13	13	12
10	50	14	16	15(0)	30(0)	13	13	7	13
	100	10	21	10(0)	15(0)	20	9	17	9
	150	12	20	12(0)	25(0)	14	21	14	17
	200	14	21	13(0)	24(0)	23	19	23	21
Moyenne		7	11.9	7.6	13.8	9.4	8.8	8.1	8.6

PROBLEME 2 : Systèmes aléatoires de dimension 200 X 10

N°	Primal					Dual			
	Méthode 1		Méthode 2			Echange classique		Echange opt	
	X_o=0	BP pts	X_o=0	BP pts	BP non admis.	X_o=0	BP pts	X_o=0	BP
1	14	19	10(0)	22(0)	28(0)	14	21	15	21
2	10	18	10(0)	18(0)	24(0)	22	21	15	21
3	15	31	16(0)	26(0)	25(0)	19	22	14	22
4	12	24	11(0)	18(0)	25(0)	20	17	17	17
5	14	21	13(0)	24(0)	31(0)	23	19	23	21
6	12	24	12(0)	18(0)	43(0)	8	25	7	22
7	13	25	13(0)	27(0)	29(0)	13	14	12	13
8	11	24	11(0)	26(0)	15(0)	15	20	18	12
9	13	22	13(0)	26(0)	31(0)	21	18	9	19
10	10	18	10(0)	24(0)	25(0)	22	13	15	14
11	12	26	13(0)	25(0)	16(0)	17	12	10	12
12	16	25	18(0)	32(0)	34(0)	23	19	14	20
13	13	25	15(0)	19(0)	21(0)	15	15	9	1
14	13	23	10(0)	18(0)	21(0)	23	16	12	1
15	13	34	13(0)	27(0)	19(0)	18	21	19	1
16	14	21	14(0)	23(0)	22(0)	20	18	17	2
17	12	23	15(0)	21(0)	27(0)	13	13	10	1
18	14	13	14(0)	16(0)	16(0)	18	14	18	1
19	11	28	11(0)	24(0)	17(0)	23	15	6	1
20	13	19	12(0)	23(0)	30(0)	12	16	7	1
Moy.	12.75	23.15	12.70	22.85	24.95	17.95	18.1	14.1	1

. Le point de Barrodale et Phillips est nettement moins bon
le point $x_0=0$.

. Quand on compare les deux méthodes primales, on observe
l y a très peu de différence entre elles. La deuxième méthode
porte pas d'amélioration.

. L'algorithme primal est légèrement meilleur que l'algorithme
. En effet, si on considère le problème 2, le nombre moyen d'ité-
ons pour la méthode primale est de 12.7 contre 14.1 pour le dual.

ROBLEMES DE DATA-FITTING

Il s'agit en fait de la discrétisation d'un problème d'ap-
ximation sur une partie compacte de R^δ ($\delta \leq 7 \in N$).

A. Problèmes Haar univariables

Ces problèmes sont essentiellement issus de l'article [4] .
faut remarquer que le problème 7 est une approximation discrète
ne fonction discontinue.

PROBLEME 3 : Approximer $f(z) = e^z$ évaluée pour z = 0.0 (0.1) 2.0 (i.e m = 21) par un poly
de degré n-1.

$$p(z) = x_1 + x_2 z + \ldots + x_n z^{n-1}$$

n	Primal							Dual				
	Méthode 1			Méthode 2				Echange classique			Echange op	
	X_o=0	BP	C pts	X_o=0	BP	BP non admis.	C pts	X_o=0	BP	C pts	X_o=0	BP
4	16	1	4	10(0)	1(0)	0(0)	4(0)	6	0	0	6	0
6	24	6	6	12(0)	6(0)	4(0)	6(0)	9	4	0	8	4
8	29	12	8	14(0)	12(0)	9(0)	8(0)	12	7	0	9	6

Problème 4 : Approximer $f(z) = e^z$ évaluée pour z = 0.0 (0.01) 2.0 (i.em = 201) par un polyn
de degré n-1.

n	Primal						Dual				
	Méthode 1			Méthode 2			Echange classique			Echange op	
	X_o=0	BP	C pts	X_o=0	BP	BP non admis.	C pts	X_o=0	BP	C pts	X_o=0	BP
2	25	2	2	6(1)	2(0)	2(1)	2(0)	2	1	1	2	1
4	140	5	4	20(0)	5(0)	3(0)	4(0)	7	2	3	7	2
6	253	16	6	27(0)	11(0)	13(0)	6(0)	14	5	4	13	5
8	354	23	8	35(0)	20(0)	17(0)	8(0)	18	10	5	16	10

PROBLEME 5 : Approximer $f(Z) = \sin(Z)\, e^{-Z}$ évaluée pour Z = 0.0 (0.02) 4.0 (i.e m = 201) par un polynôme de degré n-1.

n	Primal							Dual				
	Méthode 1			Méthode 2				Echange classique			Echange opt	
	X_o=0	BP	C pts	X_o=0	BP	BP non admis.	C pts	X_o=0	BP	C pts	X_o=0	BP
2	13	2	4	4(1)	3(1)	2(1)	3(0)	4	2	2	3	2
4	69	11	5	12(0)	8(0)	8(0)	6(0)	7	6	3	7	6
6	106	17	20	18(0)	14(0)	15(0)	15(0)	16	10	9	14	10
8	178	15	8	45(0)	15(0)	21(0)	9(0)	17	10	6	13	10

PROBLEME 6.1 : Approximer $f(z) = \sqrt{1 + z}$ évaluée pour z = 0.0 (0.01) 1.0 (i.e m = 101) par nôme de degré n-1.

n	Primal							Dual				
	Méthode 1			Méthode 2				Echange classique			Echange op	
	X_o=0	BP	C pts	X_o=0	BP	BP non admis.	C pts	X_o=0	BP	C pts	X_o=0	BP
2	56	1	2	9(1)	1(0)	0(0)	2(0)	2	0	1	2	0
3	80	10	3	5(0)	6(0)	3(0)	3(0)	5	3	2	4	3
4	113	5	4	18(0)	5(0)	3(0)	4(0)	7	3	3	6	3
5	130	13	5	17(0)	12(0)	6(0)	5(0)	8	6	3	8	6
6	170	15	6	24(0)	10(0)	8(0)	7(0)	13	6	4	11	5
7	192	11	8	21(0)	12(0)	7(0)	9(0)	16	6	4	13	6
8	197	14	9	33(0)	14(0)	14(0)	9(0)	17	7	4	14	7

PROBLEME 6.2 : Approximer $f(z) = \sin(\pi z/2)$ évaluée pour $z = 0.0(0.01)\ 1.0$ (i.e m = 101) par un polynôme de degré n-1.

n	Primal							Dual				
	Méthode 1			Méthode 2				Echange classique			Echange opt	
	X_o=0	BP	C pts	X_o=0	BP	BP non admis.	C pts	X_o=0	BP	C pts	X_o=0	BP
2	46	0	2	7(1)	0(0)	0(0)	2(0)	2	0	1	2	0
3	64	4	3	10(0)	4(0)	2(0)	4(0)	5	2	2	5	2
4	85	4	4	12(0)	5(0)	4(0)	4(0)	6	3	3	6	3
5	74	10	5	16(0)	8(0)	6(0)	6(0)	8	4	3	8	4
6	100	10	7	21(0)	9(0)	5(0)	7(0)	12	5	3	11	5
7	152	14	7	26(0)	13(0)	9(0)	7(0)	15	7	3	12	7
8	175	16	8	32(0)	12(0)	7(0)	8(0)	15	5	3	12	5

PROBLEME 6.3 : Approximer f(z) = log (1 + z) évaluée pour z= 0.0 (0.01) 1.0 (i.e m = 101) par un polynôme de degré n-1.

n	Primal							Dual				
	Méthode 1			Méthode 2				Echange classique			Echange opti	
	X_o=0	BP	C pts	X_o=0	BP	BP non admis.	C pts	X_o=0	BP	C pts	X_o=0	BP
2	58	0	2	9(0)	0(0)	0(0)	2(0)	2	0	1	2	0
3	80	8	3	10(0)	6(0)	3(0)	3(0)	5	3	2	5	3
4	113	6	4	18(0)	7(0)	3(0)	5(0)	7	4	3	6	4
5	130	10	6	14(0)	11(0)	8(0)	6(0)	8	7	3	9	7
6	170	17	6	27(0)	16(0)	9(0)	6(0)	13	5	4	11	5
7	193	11	8	24(0)	10(0)	5(0)	8(0)	15	4	4	11	4
8	211	11	10	33(0)	11(0)	10(0)	10(0)	16	8	5	15	8

PROBLEME 7.A : Approximer $f(z) = 1 + z + z^2 + z^3 + z^4 + g(z)$

où $g(z) = 5$ pour $0.94 \leqq z \leqq 1.0$
$= 0$ pour les autres

Cette fonction est évaluée en z = 0.0(0.02) 1.0 (i.e m = 51) et approximée un polynôme de degré n-1.

	Primal							Dual					
	Méthode 1			Méthode 2				Echange classique			Echange opt		
n	X_o=0	BP	C pts	X_o=0	BP	BP non admis.	C pts	X_o=0	BP	C pts	X_o=0	BP	
2	22	0	2	7(1)	0(0)	0(0)	2(0)	3	0	1	2	0	
4	38	6	9	21(0)	6(0)	28(0)	8(0)	8	2	5	5	2	
6	55	8	35	49(0)	7(0)	28(0)	35(0)	11	3	6	8	3	
8	75	18	17	37(0)	16(0)	19(0)	22(0)	19	6	7	10	6	

PROBLEME 7.B : Approximer $f(z) = 1 + z + z^2 + z^3 + z^4 + g(z)$

où $g(z) = 5$ pour $0.5 \leq z \leq 0.8$
$= 0$ pour les autres

Cette fonction est évaluée en z = 0.0 (0.02) 1.0 (i.e m. = 51) et approximée un polynôme de degré n-1.

	Primal							Dual				
	Méthode 1			Méthode 2				Echange classique			Echange opt	
n	$X_o=0$	BP	C pts	$X_o=0$	BP	BP non admis.	C pts	$X_o=0$	BP	C pts	$X_o=0$	BP
2	6	3	3	4(1)	3(0)	4(1)	3(1)	4	3	2	4	3
4	5	6	5	9(0)	6(0)	5(0)	5(0)	8	4	2	6	3
6	rgN < k	10	10	17(0)	9(0)	11(0)	10(0)	21	5	7	11	5
8	38	20	20	rgN < k	24(0)	16(0)	21(0)	31	12	12	14	7

a. Problèmes 3 à 6

1) Point de départ $x_o=0$.

L'amélioration apportée par la deuxième méthode primale très nette. Les résultats peuvent être parfois étonnants. i, pour l'exemple 4 (n=8), on passe de 354 itérations à 35.

Si on compare les résultats des algorithmes primaux avec méthodes duales, on remarque que ces dernières sont beaucoup efficaces.

2) C points

Ce point de départ donne de meilleurs résultats que le nt $x_o=0$. Mais, il faut noter qu'il y a très peu de différence re les deux méthodes primales.

L'algorithme dual testé avec ce point de départ fournit résultats légèrement supérieurs à ceux fournis par les méthodes nales.

Cependant, on remarquera que la notion de C points est e aux problèmes univariables vérifiant la condition de Haar pour quels l'algorithme de Remes est le plus efficace.

3) Point de Barrodale et Phillips

Le point de Barrodale et Phillips non admissible donne bons résultats. Ces résultats sont cependant un peu moins bons ceux obtenus par les méthodes duales.

Pour ce problème, les conclusions sont légèrement diffé-
e. On obtient les meilleurs résultats quand on prend comme point
épart le point de Barrodale et Phillips admissible. Ici encore,
gorithme dual donne de meilleurs résultats.

On remarque que pour le problème 7B avec n=6 et n=8, on
s problèmes de dégénérescence de l'algorithme. Au contraire de
gorithme dual, des problèmes de dégénérescence peuvent avoir lieu
quand la condition de Haar est satisfaite. En fin de ce chapi-
nous analysons un exemple de dégénérescence et une solution
ible pour traiter ce problème.

Pour l'ensemble de ces problèmes, on remarque que le nombre
iminution du paramètre de pénalité est très peu élevé. En fait,
st nul la plupart du temps. Ce qui signifie que pour la valeur
sie du paramètre de pénalité (μ=2) la solution du problème ori-
l est celle du problème de pénalité.

B. Problèmes non Haar (univariables et multivariables)

Les problèmes 8 et 9 sont issus de l'article [4] . D'autres
blèmes ont ensuite été envisagés, notamment ceux décrits dans
rticle de G.A.Watson. Enfin, trois problèmes particuliers (inté-
le elliptique, intégrale complexe et l'approximation sur un "carré")
été testés.

PROBLEME 8.A : Approximer $f_i(z)$ évaluée pour $z = 0.0\ (0.02)\ 1.0\ (1.$ [illegible] $)$ par la fonction suivante :

$$\sum_{j=1}^{4} \left(x_j \, z^{j-1} + x_{j+4} \max\left((z-c_j)^3, 0\right)\right)$$

où

$$c_1=0.1 \ , \ c_2=0.2 \ , \ c_3=0.2 \ , \ c_4=0.7$$

$$f_1(z)=\sqrt{z} \ , \ f_2(z)=\sqrt{1+z} \ , \ f_3(z)= \sin(\pi z/2) \ , \ f_4(z)= \log(1+z$$

	Primal					Dual			
	Méthode 1		Méthode 2			Echange classique		Echange op	
	$X_o=0$	BP pts	$X_o=0$	BP pts	BP non admis.	$X_o=0$	BP pts	$X_o=0$	B
f_1	65	10	29(0)	10(0)	4(0)	19	4	16	
f_2	39	14	21(0)	14(0)	10(0)	18	8	11	
f_3	31	10	18(0)	10(0)	5(0)	21	5	16	
f_4	39	9	26(0)	9(0)	6(0)	18	5	12	

PROBLEME 8.B : Approximer $f_i(z)$ évaluée pour z = 0.0 (0.02) 1.0 (i.e m = 51) par la fonction linéaire par morceaux suivante :

$$g(z) = g_1(z) + g_2(z)$$

$$g_1(z) = c_1 + m_1 z \quad \text{si} \quad z \leq a$$
$$= 0 \quad \text{sinon}$$

$$g_2(z) = \tfrac{1}{2}(m_1 - m_2) + c_1 + m_2 z \quad \text{si} \quad z > a$$
$$= 0 \quad \text{sinon}$$

où

$$a = 0.5 \ , \ f_1(z) = z^2 \ , \ f_2(z) = \sqrt{z} \ , \ f_3(z) = \sin(\pi z/2)$$

$$f_4(z) = \log(1+z) \quad \text{et} \quad x = (c_1, m_1, m_2)$$

	Primal					Dual			
	Méthode 1		Méthode 2			Echange classique		Echange o	
	X_o=0	BP pts	X_o=0	BP pts	BP non admis.	X_o=0	BP pts	X_o=0	BP
f_1	RgN<k	RgN<k	RgN<k	RgN<k	RgN<k	4	1	4	
f_2	11	2	9(0)	2(0)	1(0)	5	0	5	
f_3	15	4	7(1)	4(0)	1(0)	4	2	4	
f_4	11	10	10(1)	7(1)	1(0)	8	2	8	

PROBLEME 9 : Approximer $x + 2$ par $y = a_o + a_1 x^2 + a_2 x^4$ sur l'intervalle $[-2,2]$.

Posons $x = (a_o, a_1, a_2)$

m	Primal					Dual			
	Méthode 1		Méthode 2			Echange classique		Echange op	
	$X_o=0$	BP pts	$X_o=0$	BP pts	BP non admis.	$X_o=0$	BP pts	$X_o=0$	BP
4	2	2	2(0)	2(0)	1(0)	0	0	0	0
10	3	3	2(0)	6(0)	1(0)	4	2	4	2
20	4	3	7(0)	6(0)	1(0)	8	4	8	4
60	4	14	11(0)	19(0)	1(0)	11	7	10	5
100	4	21	25(0)	49(0)	1(0)	15	8	13	6

ntégrale elliptique

On veut approximer

$$F(\chi,\alpha) = \int_0^{\alpha} \frac{ds}{\sqrt{1 - \sin^2\chi \sin^2 s}}$$

la fonction suivante

$$F(x,y) = c_0 + c_1x + c_2y + c_3x^2 + c_4y^2 + c_5xy + c_6x^3 + c_7y^3 + c_8x^2y + c_9y^2x,$$

$$x = (\chi - 35°)/100 \quad \text{et} \quad y = (\alpha - 35°)/100.$$

Intégrale complexe

On veut approximer

$$F(s,t) = \mathrm{Re}\left[\int_{s+it}^{\infty} \frac{\exp(-s-it)}{(s+it)}\, d(s+it)\right]$$

les 100 points définis par s = 0.1 (0.1) 1.0 et t = 0.1 (0.1) 1.0

un polynôme à deux variables

$$c_1 + c_2s + c_3t + c_4st + c_5s^2 + c_6t^2,$$

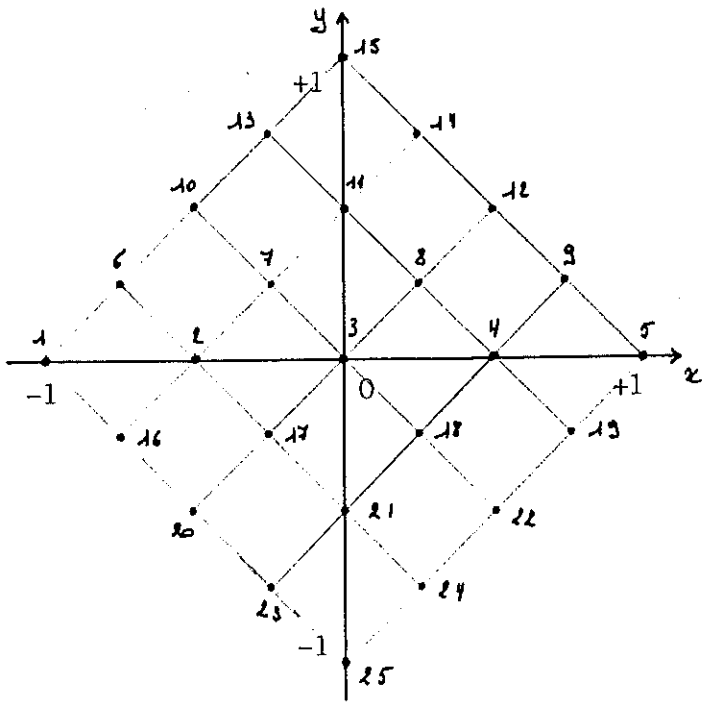
$$x^t = (c_1, c_2, c_3, c_4, c_5, c_6).$$

On veut approximer

$$F(x,y) = x^2 + y^2$$

$$G(x,y) = a + b\,x + c\,y$$

les différents points suivants :



PROBLEME 10

	Primal				Dual			
	Méthode 1		Méthode 2		Echange classique		Echange opt	
	$X_o=0$	BP pts	$X_o=0$	BP pts	$X_o=0$	BP pts	$X_o=0$	BP
Intégrale elliptique	25	12	26(0)	12(0)	17	4	6	
Intégrale complexe	28	6	19(0)	8(0)	16	6	13	
"Carré"	Rg N < k	Rg N < k	Rg N < k	Rg N < k	1	0	1	

On approxime un certain nombre de fonctions à deux variables le domaine suivant:

$$-1 \leq x,y \leq +1 .$$

polynômes approximants sont de la forme

$$\sum_{i=0}^{t} \sum_{i=0}^{t} c_{ij}\, x^i y^j$$

un certain t donné; pour les fonctions symétriques, la symétrie ettra de réduire le nombre de coefficients inconnus.

$f(x,y) = (x + 2y + 4)^{\frac{1}{2}}$

1a --- t = 2 et le nombre de points choisis sur une ligne est 4 . On notera dorénavant ce nombre par nb.

1b --- t = 3 et nb = 5

$f(x,y) = \exp (x^2 + xy)$

2 --- t = 2 et nb = 4

$f(x,y) = \sin (x^2 + y)$

3 --- t = 2 et nb = 4

$f(x,y) = 1/(x + 2y + 4)$

4 --- t = 2 et nb = 4

5a --- t = 2 et nb = 4

5b --- t = 3 et nb = 5

5c --- t = 4 et nb = 6

$f(x,y) = 1/(x + y + 3)$

6a --- t = 2 et nb = 4

6b --- t = 3 et nb = 5

6c --- t = 4 et nb = 6

$f(x,y) = (x + y + 3)^{\frac{1}{2}}$

7a --- t = 2 et nb = 4

7b --- t = 3 et nb = 5

7c --- t = 4 et nb = 6

PROBLEME 11 : Problèmes de G.A. Watson

	Primal				Dual			
	Méthode 1		Méthode 2		Echange classique		Echange op	
	$X_o=0$	BP pts	$X_o=0$	BP pts	$X_o=0$	BP pts	$X_o=0$	B
1a	12	14	15(0)	18(0)	12	2	12	
1b	23	20	32(0)	20(0)	24	8	21	
2	Rg N < k	8	5(0)	11(0)	6	1	10	
3	Rg N < k	Rg N < k	Rg N < k	Rg N < k	10	4	8	
4	8	8	9(0)	11(0)	13	2	11	
5a	Rg N < k	Rg N < k	Rg N < k	Rg N < k	8	0	7	
5b	Rg N < k	Rg N < k	Rg N < k	Rg N < k	10	1	9	
5c	Rg N < k	Rg N < k	Rg N < k	Rg N < k	29	0	29	
6a	Rg N < k	Rg N < k	Rg N < k	Rg N < k	7	1	7	
6b	Rg N < k	Rg N < k	Rg N < k	Rg N < k	11	2	11	
6c	Rg N < k	Rg N < k	Rg N < k	Rg N < k	31	2	26	
7a	Rg N < k	Rg N < k	Rg N < k	Rg N < k	14	1	12	
7b	Rg N < k	Rg N < k	Rg n < k	Rg N < k	12	2	12	
7c	Rg N < k	Rg N < k	Rg N < k	Rg N < k	39	2	37	

Il est assez difficile de tirer quelque conclusion que ce
puisque pratiquement tous les exemples posaient des problèmes
égénérescence. Ce qui nous amène naturellement à analyser cette
ation et les solutions à apporter.

ITUATION DE DEGENERESCENCE

Analysons sur un plus petit exemple le problème de dégéné-
ence qui s'est posé notamment pour les exemples 7A, 7B (n=6 et
) et pour les exemples multivariables.

Considérons la matrice A vérifiant la condition de Haar et
ecteur ℓ suivants :

$$A = \begin{bmatrix} -1. & 1. & -1. \\ 1. & 0.25 & -0.125 \\ 1. & 0.25 & 0.125 \\ 1. & 1. & 1. \end{bmatrix} \qquad \ell = \begin{bmatrix} 0.25 \\ 0.5 \\ 2. \\ 4. \end{bmatrix}$$

- point de départ = (13.75 -10. 0.25 0.)
- précision exigée = 10 exposant -16
- facteur de pénalité = 1
- méthode primale utilisée = 2

lgorithme démarre avec un point de départ admissible.

$I^o = \{8\}$

$I^+ = \{1,2,3,4,5,6,7\}$

I^- est vide

. h^t = (1. 0. 0. 0.)

. $d = -Ph \neq 0$

où d^t = (-0.75 0.25 0.25 0.25)

. λ choisi = 7.066...6

. $v = v + \lambda d$

où v^t = (8.45 -8.233..3 2.0166..6 1.766..6)

. les ensembles d'indices deviennent

$I^0 = \{ 6,8 \}$

$I^+ = \{ 1,2,3,4,5 \}$

$I^- = \{ 7 \}$

eration 2 :

. h^t = (0. -1. -0.25 -0.125)

. $d = -Ph \neq 0$

où d^t = (-0.5 0.5 0. 0.)

. λ choisi = 16.6833..3

. $v = v + \lambda d$

v^t = (0.10833..3 0.10833..3 2.0166..6 1.766)

$I^0 = \{5,6,8\}$

$I^+ = \{1,2,3,4\}$

$I^- = \{7\}$

ration 3 :

. h^t = (0. -1. -0.25 -0.125)

. $d = -Ph \neq 0$

où d^t = (-0.2885 0.1923 0.2884 -0.1923)

. λ choisi = 0.3755..5

En fait, ce λ correspond à 3 contraintes : les contraintes et 4 fournissent toutes les 3 le même λ .

. $v = v + \lambda d$

où v^t = (0. 0.1805 2.125 1.6944)

. les ensembles d'indices sont

$I^0 = \{1,2,4,5,6,8\}$

$I^+ = \{3\}$

$I^- = \{7\}$

Il y a 6 contraintes actives. D'où nécessairement à l'itération suivante les colonnes de la matrice des contraintes actives sont linéairement dépendantes !

Il faut remarquer que, notamment pour les exemples multivariables, le nombre de colonnes linéairement dépendantes peut être supérieur à n+1.

t dans l'article de R.Bartels et A.Conn [3] : on a une
avec un nombre maximum de colonnes linéairement indépendantes.
contraintes qui ne sont pas reprises dans N sont mises aléatoi-
ent dans I^+ et dans I^- (ces contraintes sont malgré tout ac-
es !).

Voici ce que cette technique a donné pour notre exemple :
'itération 3 :

$I^0 = \{1,5,6,8\}$

$I^+ = \{2,3,4\}$

$I^- = \{7\}$

lgorithme a pu converger vers la solution en 6 itérations.

$$v^* = (0.538194 \quad 0.71875 \quad 2.125 \quad 1.6944..)$$

Par manque de temps, on n'a pas su analyser ce que donnait
te technique pour les autres problèmes. Cependant, il serait
ile de le faire au plus vite pour pouvoir travailler avec des
oblèmes multivariables.

Dans ce travail, nous avons présenté un algorithme primal
e résolution de systèmes d'équations linéaires surdéterminés sui-
ant la norme de Tchebycheff mis au point par R. Bartels, A. Conn
t Yi Li [4].

Nous avons comparé cet algorithme primal avec un algo-
ithme d'échange [16] .

En ce qui concerne les problèmes aléatoires, la méthode
rimale semble légèrement supérieure à la méthode duale. Par
ontre, pour les problèmes d'approximation, la méthode duale est
a meilleure.

Dans le cas univariable avec condition de Haar, l'algo-
ithme primal a de bons résultats si le point de départ est le
oint de Barrodale et Phillips non admissible; cependant avec ce
ême point de départ, l'algorithme dual donne de meilleurs ré-
ultats.

Dans la plupart des cas non Haar (univariables et multi-
ariables), il faudrait implémenter une technique de perturbation
our traiter les problèmes de dégénérescence.

Ceci n'a pu être fait par manque d'information et de
emps.

D'OPTIMALITE.

Nous allons envisager les conditions nécessaires et suffi-
es pour qu'un point x^* soit optimal.

Considérons le problème suivant :

$$\min f(x) \qquad \text{(P)}$$

$$\text{s.c.} \quad g_i(x) \leq o \qquad \text{pour} \quad i = 1,..,m$$

$$x \in X$$

Les deux théorèmes suivants vont nous servir pour les dé-
strations ultérieures.

OREME AN.I 1. :

Soient $g_i : R^n \to R$ pour $i = 1,..,m$ et soit X un ensemble
ert non vide de R^n. Supposons que x^* est un point admissible
r (P) et soit $I = \{i : g_i(x^*) = o\}$. De plus, supposons que
t g_i pour $i \in I$ sont différentiables en x^* et que g_i pour $i \notin I$
continue en x^*. Si x^* est une solution optimale locale, alors
$\backslash\ G_o = o$,

$$F_o = \{d : \nabla f(x^*)^t\, d < o\}$$

$$G_o = \{d : \nabla g_i(x^*)^t\, d < o\ ,\ i \in I\}$$

OREME AN.I 2. (théorème de Gordan) :

Soit A une matrice de R^{mxn}. Considérons les deux systèmes,

(1) $A\,x < o\ ,\ x \in R^n$

(2) $A^t\, p = o,\ p \geq o \quad p \neq o \quad p \in R^m$

Les premières conditions d'optimalité que nous allons voir celles de Fritz-John.

REME AN.I 3. (les conditions de Fritz-John) :

Soit X un ensemble ouvert non vide de R^n; soient

$$f : R^n \to R$$

$$g_i : R^n \to R \qquad \text{pour } i = 1,..,m .$$

x^*une solution admissible de (P) et soit $I = \{i : g_i(x^*)=o\}$. lus, supposons que f et g_i pour $i \in I$ sont différentiables en t que g_i pour $i \notin I$ sont continues en x^*.

Si x^* est une solution locale de (P), alors il existe des aires u_o et u_i pour $i \in I$ tels que :

$$u_o \nabla f(x^*) + \sum_{i \in I} u_i \nabla g_i(x^*) = o$$

$$u_o, u_i \geq o \qquad \text{pour } i \in I$$

es u_i ne sont pas tous nuls en même temps.

De plus, si g_i, pour $i \notin I$, est aussi différentiable en x^*, rs les conditions de Fritz-John peuvent être écrites sous la ne équivalente :

$$u_o \nabla f(x^*) + \sum_{i=1}^{m} u_i \nabla g_i(x^*) = o$$

$$u_i g_i(x^*) = o \qquad i=1,..,m$$

$$u_o, u_i \geq o \qquad i=1,..,m$$

onstration :

Comme x^*est une solution locale du problème (P), alors le théorème An.I 1., il n'existe pas de vecteur d tel que

$$\nabla f(x^*)^t d < o$$

$$\nabla g_i(x^*)^t d < o \qquad \text{pour } i \in I .$$

struisons la matrice A dont les lignes sont formées par le gra- nt de f et le gradient des différentes contraintes actives. condition d'optimalité du théorème An. I 1. indique que le systé- :

$$A d < o$$

pas de solution. Par le théorème de Gordan (th.an.I 2.), il ste un vecteur non nul $p \geqq o$ tel que

$$A^t p = o .$$

on note les composantes de p par u_o et u_i pour $i \in I$, on déduit

$$u_o \nabla f(x^*) + \sum_{i \in I} u_i \nabla g_i(x^*) = o$$

$$u_o, u_i \geqq o \qquad \text{pour } i \in I$$

les composantes u_o, u_i ne sont pas toutes nulles en même temps. obtient la forme équivalente de la condition nécessaire d'opti- ité en posant $u_i = o$ pour $i \notin I$.

arques :

1. Les scalaires u_o et u_i pour $i = 1,..,m$ sont appelés les tiplicateurs de Lagrange associés à x^*.

2. La condition

$$u_i g_i(x^*) = o \qquad \text{pour } i = 1,..,m$$

la condition de complémentarité. Le multiplicateur u_i doit

$^*) < o.$

eut être strictement positif si la contrainte est active.

3. Si le multiplicateur de Lagrange u_o est nul, alors les itions de Fritz-John ne tiennent pas compte du gradient de la tion objective. Elles indiquent simplement qu'il existe une inaison linéaire des gradients des contraintes actives non iale et positive qui est égale à zéro. Donc, quand $u_o = o$, conditions de Fritz-John ne sont pas très pratiques pour déiner une solution optimale. Il faudra imposer des conditions que $u_o > o$. C'est ce que nous allons faire grâce aux théorèmes Kuhn-Tucker.

OREME AN.I 4. (Conditions nécessaires de Kuhn-Tucker)

Soit X un ensemble non vide de R^n et soit $f : R^n \to R$ et $: R^n \to R$ pour $i = 1,..,m$. Soit x^* une solution admissible et t $I = \{i : g_i(x^*) = o\}$. Supposons que f et g_i pour $i \in I$ sont férentiables en x^* et que g_i pour $i \notin I$ sont continues en x^* plus, supposons que les gradients des différentes contraintes ives sont linéairement indépendantes.
x^* est une solution locale de (P), alors il existe des scalaires pour $i \in I$ tels que

$$\nabla f(x^*) + \sum_{i \in I} u_i \nabla g_i(x^*) = o$$

$$u_i \geq o \qquad \text{pour} \quad i \in I$$

onstration :

Par le théorème An. I 3., il existe des scalaires u_o et $\hat{u}_i$ r $i \in I$, non tous nuls, tels que

$$u_o,\ \hat{u}_i \geq o \text{ pour } i \in I .$$

note que $u_o > o$, sinon les gradients des contraintes actives ne
t pas linéairement indépendants. Pour obtenir le théorème, il
fit alors de poser $u_i = \hat{u}_i / u_o$.

arques :

1. On peut interpréter géométriquement les conditions de
n-Tucker : on dira que l'opposé du gradient de la fonction ob-
tif au point x^* appartient au cône engendré par les gradients
s contraintes actives.

2. Cette condition nécessaire n'est pas suffisante, il
ut en plus imposer certaines hypothèses de convexité.

EOREME AN.I 5. (Conditions suffisantes de Kuhn-Tucker):

Soit X un ensemble ouvert non vide de R^n et soit
: $R^n \to R$ et soient g_i: $R^n \to R$ pour $i = 1,..,m$.
it x^* une solution admissible et $I = \{i : g_i(x^*) = o\}$.
pposons que ℓ et g_i sont convexes et différentiables. Alors,
x^* satisfait aux conditions de Kuhn-Tucker (i.e il existe des
alaires u_i non négatifs pour $i \in I$ tels que :

$$\nabla \ell(x^*) + \sum_{i \in I} u_i \nabla g_i(x^*) = o).$$

ors, x^* est un minimum global pour le problème (P).

monstration :

Comme ℓ et g_i sont convexes, on a :

$$\ell(y) \geq \ell(x^*) + \nabla\ell(x^*)^t (y-x^*) \qquad (1)$$

ur $i \in I$ et y un point admissible pour (P). En multipliant l'é-
ation (2) par u_i, on déduit :

$$u_i\, g_i(y) \geq u_i\, g_i(x^*) + u_i\, \nabla g_i(x^*)^t\, (y-x^*) \qquad (3)$$

pour $i \in I$.

sommant l'équation (2) et les équations(3) pour les i apparte-
nt à I, on obtient :

$$f(y) + \sum_{i \in I} u_i\, g_i(y) \geq f(x) + \sum_{i \in I} u_i\, g_i(x^*)$$

$$+ (y-x^*)\,(\nabla f(x^*)^t + \sum_{i \in I} u_i\, \nabla g_i(x^*))$$

r les conditions de Kuhn-Tucker, on déduit :

$$f(y) + \sum_{i \in I} u_i\, g_i(y) \geq f(x^*).$$

plus, comme y est un point admissible arbitraire :

$$f(y) \geq f(x^*).$$

ette dernière inégalité nous donne le résultat attendu.

Dans les deux algorithmes, nous avons utilisé les matrices projection. Voici quelques définitions et théorèmes qui permet- t de justifier certains passages théoriques.

inition :

Si V est la somme directe de M et N de telle sorte que quel soit z dans V, on peut écrire z sous la forme :

$$z = x + y$$

$$x \in M$$

$$y \in N .$$

plus, cette décomposition de z est unique. La projection sur M long de N est la transformation $\mathcal{E}$ définie par $\mathcal{E}z = x$.

On remarque que $\mathcal{E}$ est une transformation linéaire.

EOREME AN.II 1. :

Une transformation linéaire $\mathcal{E}$ est une projection si et ulement si :

$$\mathcal{E}^2 = \mathcal{E}$$

monstration :

Si $\mathcal{E}$ est une projection sur M le long de N et si $z=x+y$ ec x dans M et y dans N, alors la décomposition de x est $x+o$, sorte que

$$\mathcal{E}^2 z = \mathcal{E}\mathcal{E}z = \mathcal{E}x = x = \mathcal{E}z .$$

ersement, supposons que $\mathcal{E} \neq 0$. Soit $\mathcal{N}$ l'ensemble
teurs z dans V pour lesquels $\mathcal{E}z=o$; soit $\mathcal{M}$ l'ensemble de tous
vecteurs z pour lesquels $\mathcal{E}z=z$.

On voit facilement que $\mathcal{M}$ et $\mathcal{N}$ sont deux sous-espaces.
faut encore montrer que V est la somme directe de ces deux sous-
paces, ou encore, il faut montrer que $\mathcal{M}$ et $\mathcal{N}$ sont disjoints et
leur réunion engendre V.

Si z est dans $\mathcal{M}$, alors $\mathcal{E}z=z$; si z est dans $\mathcal{N}$, alors $\mathcal{E}z=o$,
où si z est à la fois dans $\mathcal{M}$ et $\mathcal{N}$, alors $z=o$.
ur un point z arbitraire, on a

$$z = \mathcal{E}z + (1-\mathcal{E})\, z .$$

on écrit, $\mathcal{E}\, z = x$ et $(1-\mathcal{E})\, z = y$, alors

$$\mathcal{E}x = \mathcal{E}^2 z = \mathcal{E}z = x$$

$$\mathcal{E}y = \mathcal{E}(1-\mathcal{E})\, z = \mathcal{E}z - \mathcal{E}^2 z = o$$

telle sorte que x est dans $\mathcal{M}$ et y dans $\mathcal{N}$.
ci montre que V est la somme directe de $\mathcal{M}$ et $\mathcal{N}$ et que la projection
r $\mathcal{M}$ le long de $\mathcal{N}$ est $\mathcal{E}$. □

Comme conséquence immédiate de ce théorème on a :

EOREME AN.II 2. :

Si $\mathcal{E}$ est la projection sur $\mathcal{M}$ le long de $\mathcal{N}$, alors $\mathcal{M}$ et $\mathcal{N}$ sont
spectivement les ensembles de toutes les solutions des équations

$$\mathcal{E}z = z \quad \text{et} \quad \mathcal{E}z = o .$$

HEOREME AN.II 3. :

Une transformation linéaire $\mathcal{E}$ est une projection si et eulement si $(1-\mathcal{E})$ est une projection; si $\mathcal{E}$ est une projection sur $\mathcal{M}$ le long de $\mathcal{N}$, alors $(1-\mathcal{E})$ est la projection sur $\mathcal{N}$ le long de $\mathcal{M}$.

Si on munit l'espace V d'un produit scalaire, on peut définir le complément orthogonal d'un sous-espace.

éfinition :

Soit $\mathcal{M}$ un sous-espace de V, on note par $\mathcal{M}^{\perp}$ le complément rthogonal de $\mathcal{M}$. $\mathcal{M}^{\perp}$ est l'ensemble des vecteurs de V orthogonaux tous les vecteurs de $\mathcal{M}$.

HEOREME AN,II 4. : (théorème de projection)

Si $\mathcal{M}$ est un sous-espace de V, alors V est la somme directe e $\mathcal{M}$ et $\mathcal{M}^{\perp}$ et $\mathcal{M}^{\perp\perp} = \mathcal{M}$.

Nous pouvons maintenant définir les projections associées à ette décomposition de V en somme directe.

éfinition :

On appelle projection orthogonale $\mathcal{E}$ sur $\mathcal{M}$, la projection ur $\mathcal{M}$ le long de $\mathcal{M}^{\perp}$.

HEOREME AN.II 5. :

Une transformation linéaire $\mathcal{E}$ est une projection orthogonale si et seulement si

$$\mathcal{E}^2 = \mathcal{E} = \mathcal{E}^*$$

ù $\mathcal{E}^*$ est la transformation linéaire satisfaisant à l'équation uivante :

$$(\mathcal{E}x, y) = (x, \mathcal{E}^* y)$$

our tout x.

1. (x,y) dénote le produit scalaire de deux vecteurs de

Considérons la matrice V de type (n,m) dont les colonnes, tées v_i pour $i = 1,\ldots, m$, sont linéairement indépendantes. On uhaite construire la matrice de projection orthogonale sur le us-espace engendré par les colonnes de cette matrice.

Soit z dans R^n, le vecteur de Vz appartient au sous-pace

$$M = span \{ v_1,\ldots,v_m \}.$$

vecteur sera la projection orthogonale d'un vecteur x de R^n sur si et seulement si $x - Vz$ est orthogonal à chacun des vecteurs $(i = 1,\ldots, m)$, ou encore si et seulement si

$$V^t (x - Vz) = 0,$$

obtient alors que

$$z = (V^t V)^{-1} V^t x$$

uisque les colonnes de V sont linéairement indépendantes. Enfin, déduit que la matrice de projection que l'on cherche est

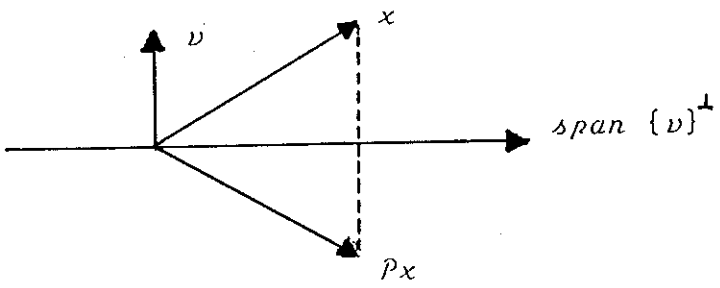
$$P = V (V^t V)^{-1} V^t.$$

Avant de parler de la factorisation QR proprement dite, s allons introduire la notion de *transformation de Householder.*

Considérons un vecteur v de R^n. Une matrice P de type (n,n) la forme

$$P = I - 2\, vv^t / v^t v \qquad (1)$$

appelé *transformation de Householder.* Quand un vecteur x est tiplié par P, il est réfléchi par l'hyperplan engendré par le s-espace orthogonal à v.



matrices de Householder sont symétriques et orthogonales. Elles t très importantes parce qu'elles peuvent annuler plusieurs com- antes d'un vecteur.

En particulier, il est assez facile, étant donné un vecteur on nul de R^n, de construire un vecteur v de telle sorte que

$$Px = \beta e_1 .$$

sait que

$$Px = \left(I - 2 \frac{v\, v^t}{v^t v} \right) x$$

$$= x - 2(v^t x / v^t v)v.$$

on veut que $Px \in span\ \{e_1\}$, ceci entraîne que $v \in span\ \{e_1, x\}$. sons $v = x + \alpha x_1$, ce qui donne

$$v^t v = x^t x + 2\alpha x_1 + \alpha^2$$

ar conséquent

$$Px = \left(1 - 2\frac{x^t x + \alpha x_1}{x^t x + 2\alpha x_1 + \alpha^2}\right) x - 2\frac{v^t x}{v^t v} e_1.$$

annuler le coefficient en x, on doit mettre $\alpha = \pm||x||_2$. Donc, $= x \pm ||x||_2 e_1$ dans (1), alors $Px = \pm||x||_2 e_1$.

Nous pouvons maintenant expliquer le mécanisme de la factotion QR.

Considérons la matrice A de type (m,n) où $m=6$ et $n=5$. On iplie la matrice par une transformation de Householder afin de sformer la première colonne en un multiplie de e_1, i.e

$$P_1 A = \begin{bmatrix} * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \end{bmatrix}$$

P_1 est une transformation de Householder. On va déterminer une rice de Householder de dimension 5x5 telle que

$$\tilde{P}_2 \begin{bmatrix} * \\ * \\ * \\ * \\ * \end{bmatrix} = \begin{bmatrix} * \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$P_2 = diag(I_1, \tilde{P}_2)$. Donc,

$$P_2 P_1 A = \begin{bmatrix} 0 & 0 & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & * & * & * \end{bmatrix}$$

fin de ce processus, on obtient

$$P_n P_{n-1} \cdots P_1 A = R$$

est une matrice triangulaire supérieure. En notant $P_n P_{n-1} \cdots P_1$ Q, on obtient la factorisation QR de la matrice.

Toute matrice peut se factoriser sous cette forme. Il faut r que si la partie de la colonne que l'on traite est nulle, il it simplement de passer aux colonnes suivantes dans la matrice.

```
PROGRAM PENALITE

**********************************************************************
*                                                                    *
*      Il s'agit de l'algorithme implemente a partir de              *
*      l'article de R.Bartels, A.Conn et Y.Li.          OA           *
*                                                                    *
**********************************************************************
```

0. DECLARATIONS

0.1 variables globales

0.1.1. de type caractere

```
NOMIN --> longueur = 80 :nom de fichier des donnees
NOM --> longueur = 80 :nom de fichier des resultats
REP --> longueur = 1 :variable intermediaire pour les reponses

CHARACTER NOMIN*80,NOM*80,REP*1
```

0.1.2. de type entier

```
N --> nombre de colonnes de A
M --> nombre de lignes de A
NMAX --> nombres maximum de lignes de A
STEP --> nombre de pas necessaires pour solutionner
         l'algorithme
DIMMU --> nombre de diminution de mu
CAL --> determine la methode a utiliser :
            1 ... si on suit la methode de R.Bartels, A.Conn et
                  C. Charalambous
            2 ... si on suit la nouvelle methode de R.Bartels,
                  A.Conn et Y. Li

I,J,K --> compteurs
IOS --> sert pour le traitement des erreurs
IRANG --> contient le rang des matrices
JPVT,JPVT1 --> vecteur de permutation des colonnes de longueur
          NMAX
IW --> entier servant dans la decomposition QR
INFO --> parametre de la routine DTRSL de LINPACK
IO(0:NMAX) --> vecteur : indices des contraintes satisfaites
I1(0:NMAX) --> vecteur : indices des contraintes strictement
                         satisfaites
I2(0:NMAX) --> vecteur : indices des contraintes violees

US --> entier servant pour les unites de sortie

IND --> indice de la composante la plus negative

INTEGER N,M,NMAX,STEP,CAL,I,J,K,IOS,IRANG,IW,INFO,MU,IND
PARAMETER (NMAX=405)
```

```
      INTEGER JPVT(NMAX),JPVT1(NMAX),
     1 IU(0:NMAX),I1(0:NMAX),I2(0:NMAX),US
```

0.1.3. de type logique
.....

```
      OPT --> vaut true si l'algorithme est a l'optimum, false sinon
      ZERO --> vaut true si le vecteur est nul,
                     false sinon
      POS --> vaut true si le vecteur est positif, false sinon
      FEAS --> vaut true si le point considere se trouve dans le
                    domaine admissible, false sinon
      TESTN --> vaut true si la composante est negative strictement

      LOGICAL OPT,ZERO,POS,FEAS,TESTN
```

0.1.4. de type double precision
.....

```
      U --> facteur de penalite
      EPS --> precision exigee par l'utilisateur
      AUX --> variable auxiliaire de calcul
      C(0:NMAX,0:NMAX) --> tableau a deux dimensions : stocke les
                           elements de A
      B(NMAX) --> vecteur : contient les elements de b
      V(0:NMAX) --> vecteur : point ou on se trouve a chaque etape
                              de la minimisation et deviation de
                              Tchebycheff
      H(NMAX+1) --> vecteur intermediaire
      NC(NMAX,NMAX) --> matrice formee par les colonnes de la
                        matrice C d'indices se trouvant dans IC
      Q(NMAX,NMAX) --> matrice orthogonale de la factorisation
                       QR de N
      WORK(NMAX) --> vecteur de travail pour la factorisation de N
      VAUX(NMAX+1) --> vecteur auxiliaire
      D(NMAX) --> vecteur donnant la direction de descente
      W(NMAX) --> vecteur intermediaire
      ETAIMAX --> composante la plus petite du vecteur eta

      DOUBLE PRECISION C(0:NMAX-1,0:NMAX),B(NMAX),V(0:NMAX),U,EPS,
     1 AUX,H(NMAX+1),WORK(NMAX),W(NMAX),UN,
     1 VAUX(NMAX+1),D(NMAX),NC(NMAX,NMAX),Q(NMAX,NMAX),ETAIMAX
```

0.3 fonctions intrinseques et externes

```
      DDOT --> fonction de linpack : calcul du produit scalaire
      DCOPY --> fonction de linpack : copie un vecteur dans un autre
      DABS --> fonction intrinseque : calcul de la valeur absolue
      DSIGN --> fonction intrinseque : donne le signe d'une composante

      DOUBLE PRECISION DDOT,DCOPY,DABS,DSIGN
      EXTERNAL DDOT,DCOPY
      INTRINSIC DABS,DSIGN
```

1.PROGRAMME

1.1 Prise des donnees dans le fichier

1.1.1 Les donnees sont-elles dans un fichier ?

```
      WRITE(*,5)
      FORMAT(1X,'--> Avez-vous inscrit vos donnees dans un ',
     1          'fichier.dat ? [O,N] ')
      READ(*,10)REP
      FORMAT(A)
      IF (REP.EQ.'N') THEN
         WRITE(*,15)
         STOP
      ENDIF
      FORMAT(1X,'--> Le programme s''arrete pour l''inscription',
     1          ' des donnees dans un fichier .')
```

1.1.2 Dans cuel fichier sont-elles ?

```
      WRITE(*,20)

      FORMAT(1X,'--> Quel est le nom du fichier de donnees ?')
      READ(*,25)NOMIN
      FORMAT(A80)
      OPEN(UNIT=20,IOSTAT=IOS,FILE=NOMIN,STATUS='OLD',ERR=26)
      IF (IOS.NE.0) THEN
         WRITE(*,27)
         STOP
      ENDIF
      FORMAT(1X,'--> Il y a une erreur dans l''ouverture du',
     1          'fichier de donnees.')

      WRITE(*,*)'NOM FICHIER DES RESULTATS '
      READ(*,25)NOM
      US=15
      OPEN(UNIT=US,STATUS='NEW',IOSTAT=IOS,FILE=NOM,ERR=26)
```

1.1.3 Lecture de m et n

```
      READ(UNIT=20,FMT=*,ERR=70,IOSTAT=IOS)M
      READ(UNIT=20,FMT=*,ERR=70,IOSTAT=IOS)N

      WRITE(US,200)M
      WRITE(*,200)M
      FORMAT(1X,'M = ',I3)
      WRITE(US,205)N
      WRITE(*,205)N
      FORMAT(1X,'N = ',I3)

      IF (IOS.NE.0) THEN

         WRITE(*,*)'--> Erreur dans votre fichier de donnees,IN'
         CLOSE(UNIT=US,IOSTAT=IOS,ERR=58)
         STOP
```

```
ENDIF

1.1.4 Lecture des elements de la matrice A
..........................................


DO 29 J=0,2*M
   C(0,J)=1.
CONTINUE

DO 30 I=1,N
   C(I,0)=0.
CONTINUE

READ(UNIT=20,FMT=*,ERR=70,IOSTAT=IOS)((C(I,J),I=1,N),J=1,M)
DO 31 J=1,M
   DO 32 I=1,N
         C(I,J)=-C(I,J)
   CONTINUE
CONTINUE

IF (IOS.NE.0) THEN

   WRITE(*,*)'--> ERREUR DANS A '
   CLOSE(UNIT=US,IOSTAT=IOS,ERR=58)
   STOP

ENDIF

1.1.5 Lecture de b
..................


READ(UNIT=20,FMT=*,ERR=70,IOSTAT=IOS)(B(I),I=1,M)
DO 33 I=1,M
   B(I)=-B(I)
CONTINUE


1.1.6 Lecture du point de depart
................................


READ(UNIT=20,FMT=*,ERR=70,IOSTAT=IOS)(V(L),L=1,N)
WRITE(US,225)(V(I),I=1,N)
WRITE(*,225)(V(I),I=1,N)
FORMAT(1X,/,1X,'LE POINT DE DEPART',/,:(D25.15))


1.1.7 Lecture de la precision
.............................


READ(UNIT=20,FMT=*,ERR=70,IOSTAT=IOS)EPS

WRITE(US,230)EPS
WRITE(*,230)EPS
FORMAT(1X,/,1X,'LA PRECISION EXIGEE ',D12.4)
```

```
      1.1.8 Lecture du facteur de penalite
      ....................................

      READ(UNIT=20,FMT=*,ERR=70,IOSTAT=IOS)U

      WRITE(US,235)U
      WRITE(*,235)U
      FORMAT(1X,/,1X,'LE FACTEUR DE PENALITE',D12.4)

      1.1.9 Lecture de la deviation de depart
      ......................................

      - On determine la norme du vecteur residu -

      V(0)=0.D+00
      DO 236 I=1,M
         AUX=DDOT(N,V(1),1,C(1,I),1)-B(I)
         IF ((DABS(AUX)-V(0)).GT.EPS) V(0)=DABS(AUX)
      CONTINUE

      WRITE(US,240)V(0)
      WRITE(*,240)V(0)
      FORMAT(1X,/,1X,'LA DEVIATION DE DEPART = ',D24.15)

      1.1.10 Lecture de la methode pour le calcul de la longueur
             du pas
      ..........................................................

      READ(UNIT=20,FMT=*,ERR=70,IOSTAT=IOS)CAL

      WRITE(US,245)CAL
      FORMAT(1X,/,1X,'METHODE UTILISEE = ',I1)


   1.2 Initialisations
   -------------------

      1.2.1 nombre de pas
      ...................

      STEP=0

      WRITE(US,250)STEP
      FORMAT(1X,//,1X,'**********',/,'*STEP= ',
     1          I3,/,1X,'**********',/)

      1.2.2 Determination de IO,I+,I-
      ...............................
```

```
      IO(0)=0
      I1(0)=0
      I2(0)=0

      DO 35 J=1,2*M

         IF (J.LE.M) THEN
            AUX=DDOT(N+1,C(0,J),1,V(0),1)
            AUX=AUX-B(J)
         ELSE
            AUX=V(0)-DDOT(N,C(1,J-M),1,V(1),1)
            AUX=AUX+B(J-M)
         ENDIF

         IF (DABS(AUX).LE.EPS) THEN
            IO(0)=IO(0)+1
            I=IO(0)
            IO(I)=J
         ELSE IF (AUX.GT.EPS) THEN
            I1(0)=I1(0)+1
            I=I1(0)
            I1(I)=J
         ELSE
            I2(0)=I2(0)+1
            I=I2(0)
            I2(I)=J
         ENDIF

      CONTINUE

      WRITE(US,255)IO(0),I1(0),I2(0)
      FORMAT(1X,/,1X,'LE NOMBRE DE CONTRAINTES ACTIVES = ',I3,/,
     1       1X,'LE NOMBRE DE CONTRAINTES STRICT. SATISFAITES = ',
     1       I3,/,1X,'LE NOMBRE DE CONTRAINTES VIOLEES = ',I3,/)
      WRITE(US,260)(IO(I),I=1,IO(0))
      FORMAT(1X,'LE VECTEUR DES C. ACTIVES',/,1X,22(I3,3X))
      WRITE(US,*)(I2(I),I=1,I2(0))
      FORMAT(1X,'LE VECTEUR DES C. VIOLEES',/,1X,22(I3,3X))
```

1.3 Corps du programme

```
      MU=0

      - Repeter jusqu'a ... -

      CONTINUE

         WRITE(US,235)U

        - calcul de h -

        DO 36 I=1,N+1

           H(I)=U*C(I-1,0)

        CONTINUE
```

```
DO 37 K=1,I2(0)

   I=I2(K)
   IF (I.LE.M) THEN

      DO 38 J=1,N+1
         H(J)=H(J)-C(J-1,I)
      CONTINUE

   ELSE

      H(1)=H(1)-C(0,I-M)
      DO 199 J=2,N+1
         H(J)=H(J)+C(J-1,I-M)
      CONTINUE

   ENDIF

CONTINUE

WRITE(US,271)(H(I),I=1,N+1)
FORMAT(1X,/,1X,'LE VECTEUR H VAUT ',/,:(D24.15,4X))

OPT=.FALSE.

- Repeter jusqu'a ... -

CONTINUE

   - remplissage de la matrice N suivant les indices de Io -

   DO 42 J=1,IO(0)

      IF (IO(J).LE.M) THEN
         CALL DCOPY(N+1,C(0,IO(J)),1,NO(1,J),1)
      ELSE
         NO(1,J)=1.
         DO 43 I=2,N+1
            NO(I,J)=-C(I-1,IO(J)-M)
         CONTINUE
      ENDIF

   CONTINUE

   - si aucune contrainte n'est satisfaite en tant qu'egalite -
     on prend comme direction de descente le vecteur e1

   IF (IO(0).EQ.0) THEN

      UN=1.
      D(1)=-DSIGN(UN,H(1))
      DO 100 I=2,N+1
         D(I)=0.
      CONTINUE

      STEP=STEP+1
```

```
      WRITE(US,285)(D(I),I=1,N+1)
      FORMAT(1X,/,1X,'LE VECTEUR DE DIRECTION DE DESCENTE '
     1       ,/, :(D24.15,4X))
      WRITE(US,250)STEP

      CALL LENSTEP(C,V,ID,I1,I2,D,H,EPS,B,NMAX,N,M,CAL,US)

      WRITE(US,290)(V(I),I=0,N)
      FORMAT(1X,/,1X,'LA DEVIATION VAUT : ',D24.15,//,
     1       1X,'LE POINT COURANT VAUT :',/,:(D24.15,4X))
      WRITE(US,255)ID(0),I1(0),I2(0)
      WRITE(US,260)(ID(I),I=1,ID(0))
      WRITE(US,270)(I2(I),I=1,I2(0))
      WRITE(US,271)(H(I),I=1,N+1)

- sinon on calcule la direction de descente suivant
  l'algorithme prevu -

ELSE

      IW=2*(N+1)+2*ID(0)+(NMAX+1)*(N+1)
      CALL DQR(ND,NMAX,N+1,ID(0),EPS,IRANG,Q,IW,WORK,JPVT)

      - si on a un probleme de degenerescence, on arrete -

      IF (IRANG.LT.ID(0)) THEN

         WRITE(US,500)
         FORMAT(1X,//,'--> N EST DE RANG INFERIEUR A K')
         CLOSE(UNIT=US,IOSTAT=IOS,ERR=58)
         STOP

      ELSE

         STEP=STEP+1
         I=ID(0)+1
         ZERO=.TRUE.

         WRITE(US,250)STEP

         - on verifie que au moins une composante de Q2^t h est
           non nulle

         IF ((ZERO).AND.(I.LE.N+1)) THEN

            VAUX(I-ID(0))=DDOT(N+1,Q(1,I),1,H(1),1)
            ZERO=(DABS(VAUX(I-ID(0))).LT.EPS)
            I=I+1
            GOTO 48

         ENDIF

         - si au moins une composante de Q2^t h est non nulle ou
           si le nombre de contraintes actives est inferieur
           a n+1, alors ...

         IF ((.NOT.ZERO).AND.(ID(0).LE.N)) THEN
```

```
                WRITE(US,315)
5               FORMAT(1X,'AU MOINS UNE DES COMPOSANTES DE'
     1                  ' Q2(transposee)H EST NON NULLE ET LE'
     1                  ' NOMBRE DE CONTRAINTES ACTIVES EST'
     1                  ' < N+1',//)

                - calcul des composantes de Q2(t) h qui ne sont pas
                  encore calculees                                  -

                IF (I.LE.N+1) THEN

                   VAUX(I-IO(0))=DDOT(N+1,Q(1,I),1,H(1),1)
                   I=I+1
                   GOTO 81

                ENDIF

                - calcul de la direction de descente : D=Q2(Q2(t) h) -

                DO 49 I=1,N+1

                   D(I)=DDOT(N+1-IO(0),Q(I,IO(0)+1),NMAX,
     1VAUX(1),1)

                   D(I)=-D(I)

                CONTINUE

                WRITE(US,285)(D(I),I=1,N+1)


                - calcul la longueur du pas dans la direction D -

                CALL LENSTEP(C,V,IO,I1,I2,D,H,EPS,B,NMAX,N,M,CAL,
     1US)


                WRITE(US,290)(V(I),I=0,N)
                WRITE(US,255)IO(0),I1(0),I2(0)
                WRITE(US,260)(IO(I),I=1,IO(0))
                WRITE(US,270)(I2(I),I=1,I2(0))
                WRITE(US,271)(H(I),I=1,N+1)

             - si toutes les composantes sont nulles
               ou si le nombre de contraintes actives est superieur
               ou egal a n+1, alors ...

             ELSE

                WRITE(US,320)
                FORMAT(1X,'-->TOUTES LES COMPOSANTES DE ',
     1                  'Q2(transp)h SONT NULLES OU LE NOMBRE '
     1,                 'DE CONTRAINTES ACTIVES EST >= N+1',//)

                - w = Q1(t) h -

                DO 50 I=1,IO(0)
```

```
      W(I)=DDOT(N+1,Q(1,I),1,H(1),1)
   CONTINUE
   - resolution de ND x = w -
   CALL DTRSL(ND,NMAX,IO(0),W,1,INFO)
   - on permute les variables de w suivant JPVT
     w est la solution du systeme triangulaire -
   DO 80 K=1,IO(0)
      JPVT1(K)=JPVT(K)
   CONTINUE
   DO 85 K=1,IO(0)
      IF (K.NE.JPVT1(K)) THEN
         AUX=W(K)
         I=K+1
         IF (K.NE.JPVT1(I)) THEN
            I=I+1
            GOTO 90
         ENDIF
         W(K)=W(I)
         W(I)=AUX
         JPVT1(I)=JPVT1(K)
         JPVT1(K)=K
      ENDIF
   CONTINUE
   - verification du signe des composantes de w -
   POS=.TRUE.
   K=1
   ETAIMAX=0.D+00
   IF (K.LE.IO(0)) THEN
      TESTN=(W(K).LT.(-EPS))
      IF (POS) POS=.NOT.TESTN
      IF (TESTN) THEN
        IF ((DABS(ETAIMAX)-DABS(W(K))).LT.(-EPS))
1       THEN
           IND=K
           ETAIMAX=W(K)
        ENDIF
      ENDIF
      K=K+1
      GOTO 51
   ENDIF
   - si une des composantes de w est negative, alors ..
```

```
                IF (.NOT.POS) THEN

                WRITE(US,340)
                FORMAT(1X,'UNE DES COMPOSANTES DE eta EST ',
1                       'NEGATIVE')

                   - on initialise le vecteur vaux -

                   DO 52 I=1,IO(0)

                      VAUX(I)=0.

                   CONTINUE

                   VAUX(IND)=1.

                   - on permute le vecteur vaux suivant jpvt -

                   IF ((IND).NE.JPVT(IND)) THEN

                      I=1
                      IF ((IND).NE.JPVT(I)) THEN

                         I=I+1
                         GOTO 95

                      ENDIF

                      VAUX(IND)=VAUX(I)
                      VAUX(I)=1.

                   ENDIF

                                          t
                   - resolution de NO  d = vaux -

                   CALL DTRSL(NO,NMAX,IO(0),VAUX,11,INFO)


                   DO 53 I=1,N+1

                      D(I)=DDOT(IO(0),Q(I,1),NMAX,VAUX(1),1)

                   CONTINUE

                   I=IO(IND)
                   CALL LENSTEP(C,V,IO,I1,I2,D,H,EPS,B,
1NMAX,N,M,CAL,US)
                   CALL UPDATE(IC,I1,I)

                   WRITE(US,285)(D(I),I=1,N+1)
                   WRITE(US,290)(V(I),I=0,N)
                   WRITE(US,255)IO(0),I1(0),I2(0)
                   WRITE(US,260)(IO(I),I=1,IO(0))
                   WRITE(US,270)(I2(I),I=1,I2(0))
                   WRITE(US,271)(H(I),I=1,N+1)

                - sinon on est a un point de Kuhn-Tucker
                  si les contraintes sont satisfaites     -
```

```
                ELSE

                    WRITE(US,350)
              FORMAT(1X,'--> Si les contraintes sont satisfaites ,',
1                    'on est a un point de Kuhn-Tucker ')

                    OPT=.TRUE.
                    STEP=STEP-1

                ENDIF

             ENDIF

          ENDIF

       ENDIF
    IF (.NOT.OPT) GOTO 41

    - verification des contraintes -

    FEAS=.TRUE.
    J=1
    I=0
    IF (FEAS.AND.(J.LE.(2*M))) THEN

       IF (J.LE.M) THEN

          AUX=DDOT(N+1,C(0,J),1,V(0),1)
          AUX=AUX-B(J)

       ELSE

          AUX=V(0)-DDOT(N,C(1,J-M),1,V(1),1)
          AUX=AUX+B(J-M)

       ENDIF

       WRITE(US,360)AUX
       FORMAT(1X,D12.4,$)

       IF (I.LT.IO(0)) THEN
          FEAS=((AUX.GE.(-EPS)).OR.(J.EQ.IO(I+1)))
          IF (J.EQ.IO(I+1)) I=I+1
       ELSE
          FEAS=(AUX.GE.(-EPS))
       ENDIF
       J=J+1

       GOTO 54

    ENDIF

    U=U/8.
    MU=MU+1
```

```
 IF (.NOT.FEAS) GOTO 40
```

1.4 Sorti e des resultats finaux

```
WRITE(US,5000)MU-1,STEP
FORMAT(1X,//,1X,'--> Le nombre de diminution de mu : ',I4,
1          //,1X,'--> L''algorithme est alle a son terme en'
1                , I4,' etapes.')
WRITE(*,5000)MU-1,STEP
CLOSE(UNIT=US,IOSTAT=IOS,ERR=58)

END
```

```
            J=J+1
            GOTO 10
         ENDIF
         DO 15 L=I1(0),J,-1
            I1(L+1)=I1(L)
         CONTINUE
         I1(J)=K
         I1(0)=I1(0)+1
         DO 20 L=I,I0(0)-1
            I0(L)=I0(L+1)
         CONTINUE
         I0(0)=I0(0)-1
      ENDIF

END
```

```
      SUBROUTINE LENSTEP(C,V,I0,I1,I2,D,H,EPS,B,NMAX,N,M,MET,US)

*****************************************************************
*                                                               *
*  Cette routine calcule la longueur du pas le long de D        *
*                                                               *
*****************************************************************
```

0.DECLARATIONS

0.1 arguments de la routine

C --> tableau à deux dimensions contenant les elements de A

V --> vecteur contenant le point ou on se trouve dans la minimisation ainsi que la deviation de Tchebyceff

I0,I1,I2 --> tableaux d'indices

D --> direction de descente

H --> vecteur intermediaire

EPS --> precision exigee sur la solution

B --> second membre de $A^t X=B$

NMAX --> nombres maximal de lignes de A

N --> nombre de colonnes de A

M --> nombre de lignes de A

MET --> variable entiere determine la methode a utiliser

```
      INTEGER NMAX,MET,N,M,I0(0:*),I1(0:*),I2(0:*),US
      DOUBLE PRECISION C(0:NMAX-1,0:NMAX),V(0:NMAX),
     1                 D(NMAX+1),H(NMAX+1),EPS,B(NMAX)
```

0.2 variables locales

MINA --> vecteur contenant les differentes longueurs de pas classes par ordre croissant.

MINB --> tableau a deux dimensions : stockera les indices des elements classes par ordre croissant ainsi que leur appartenance a I+ ou a I-

NBRPAS --> indique le nombre de longueur de pas possible

NBRLON --> indique le nombre de contraintes associees a chaque longueur de pas.

TEST --> precise si l'indice est dans I1 ou I2

AUX,LONPAS --> variables auxiliaires

DEUX --> variable boolenne - true si les deux vecteurs ont
ete envisages
- false sinon

DESC --> variable boolenne - true si on a une direction de
descente
- false sinon

L,T,P,I,J,S,K,INDEX --> compteurs

```
      INTEGER L,T,P,I,J,S,K,INDEX,TEST,MINB(2,405),NBRPAS,
     1        NBRLON(405)
      LOGICAL DEUX,DESC
      DOUBLE PRECISION MINA(405),AUX,LONPAS
```

0.3 fonctions externes

DDOT --> routine de linpack : calcul du produit scalaire

```
      DOUBLE PRECISION DDOT
      EXTERNAL DDOT
```

1.PROGRAMME

1.1 mise a jour de la variable test

```
      IF (I1(0).NE.0) THEN

         I=I1(0)
         TEST=1

      ELSE IF (I2(0).NE.0) THEN

         I=I2(0)
         TEST=-1

      ELSE

         WRITE(US,5)
    5    FORMAT(1X,'ERREUR : PAS DE CALCUL DE LONGUEUR DE PAS POSSIBLE',/,
     1        'TOUS LES INDICES SE TROUVENT DANS IO')
         CLOSE(UNIT=LS)
         STOP

      ENDIF
```

1.2 initialisations des variables

```
MINA(1)=1.D+30
MINB(2,1)=0
DEUX=.FALSE.
NBRPAS=0
```

1.3 calcul des elements pour la longueur du pas

- Calcul des lambdas strictement positifs correspondants aux contraintes de I+ et I- . Classement de ces valeurs par ordre croissant.

```
IF (.NOT.DEUX) THEN

   DO 15 T=1,I

      IF (TEST.GT.0) THEN

         K=I1(T)

      ELSE

         K=I2(T)

      ENDIF

      IF (K.LE.M) THEN

         LONPAS=DDOT(N+1,C(0,K),1,D,1)
         AUX=DDOT(N+1,C(0,K),1,V(0),1)
         AUX=B(K)-AUX

      ELSE

         LONPAS=D(1)-DDOT(N,C(1,K-M),1,D(2),1)
         AUX=V(0)-DDOT(N,C(1,K-M),1,V(1),1)
         AUX=-B(K-M)-AUX

      ENDIF

      IF (DABS(LONPAS).GT.EPS) THEN

         LONPAS=1./LONPAS
         LONPAS=LONPAS*AUX

         IF (LONPAS.GT.0.D+00) THEN

            S=1

            IF((LONPAS.GT.MINA(S)).AND.(S.LE.NBRPAS+1))
1           THEN

               S=S+1
               GOTO 20

            ENDIF

              DO 25 P=NBRPAS+1,S,-1
```

```
                  MINA(P+1)=MINA(P)
                  MINB(1,P+1)=MINB(1,P)
                  MINB(2,P+1)=MINB(2,P)

                CONTINUE

                NBRPAS=NBRPAS+1
                MINA(S)=LONPAS
                MINB(1,S)=K
                MINB(2,S)=TEST


            ENDIF

          ENDIF

        CONTINUE

        IF (TEST.EQ.+1) THEN

           TEST=-1
           I=I2(0)

        ELSE

           DEUX=.TRUE.

        ENDIF

      GOTO 10
      ENDIF

      - on va determiner le nombre de lambdas distincts et
        leurs indices dans le tableau mina                  -

      IF (NBRPAS.GT.0) THEN

         I=1
         INDEX=0
         IF (I.LE.NBRPAS) THEN
            J=I
            IF ((DABS((MINA(I)-MINA(I+1))/MINA(I)).LE.EPS)
     1                        .AND.(I.LT.NBRPAS)) THEN
               I=I+1
               GOTO 26
            ENDIF
            INDEX=INDEX+1
            NBRLON(INDEX)=I-J+1
            I=I+1
            GOTO 21
         ENDIF
       ENDIF


      1.4 calcul de la longueur du pas suivant la methode demandee
      ------------------------------------------------------------


      IF (NBRPAS.EQ.0) THEN
```

```
      WRITE(US,*)'IL N''Y A PAS DE LONGUEUR DE PAS POSSIBLE '
      CLOSE(UNIT=US)
      STOP
ELSE
 - Premiere metnode primale -
 IF (MET.EQ.1) THEN
    J=NBRLON(1)
    DO 35 I=1,J
     IF (MINB(2,I).GT.0) THEN
       CALL UPDATE(I1,IO,MINB(1,I))
     ELSE
       CALL UPDATE(I2,IO,MINB(1,I))
       IF (MINB(1,1).LE.M) THEN
          DO 50 S=1,N+1
             H(S)=H(S)+C(S-1,MINB(1,I))
          CONTINUE
       ELSE
          H(1)=H(1)+C(0,MINB(1,I)-M)
          DO 55 S=2,N+1
             H(S)=H(S)-C(S-1,MINB(1,I)-M)
          CONTINUE
       ENDIF
     ENDIF
     CONTINUE
    AUX=MINA(1)
    DO 30 T=1,N+1
       V(T-1)=V(T-1)+(AUX*D(T))
    CONTINUE
    WRITE(US,31)AUX
    FORMAT(1X,/,1X,'LA LONGUEUR DU PAS = ',D23.15)
 - Seconde methode primale -
 ELSE
    T=1
    DESC=.TRUE.
    INDEX=1
    IF (DESC.AND.(T.LE.NBRPAS)) THEN
```

```
      IF (T.NE.1) DESC=(DDOT(N+1,H,1,D,1).LT.(-EPS))

      IF (DESC) THEN

         J=NBRLON(INDEX)
         DO 40 I=T,T+J-1

          IF (MINB(2,I).LT.0) THEN

            CALL UPDATE(I2,I1,MINB(1,I))
            L=1

          ELSE

            CALL UPDATE(I1,I2,MINB(1,I))
            L=-1

          ENDIF

          - mise a jour de H -

          IF (MINB(1,I).LE.M) THEN

            DO 300 S=1,N+1
               H(S)=H(S)+L*C(S-1,MINB(1,I))
            CONTINUE

          ELSE

            H(1)=H(1)+L*C(0,MINB(1,I)-M)
            DO 310 S=2,N+1
               H(S)=H(S)-L*C(S-1,MINB(1,I)-M)
            CONTINUE

          ENDIF
          CONTINUE

          T=T+NBRLON(INDEX)
          INDEX=INDEX+1

       ENDIF

       GOTO 39

     ENDIF

- si on sait descendre avec le lambda precedent ... -

IF (.NOT.DESC.OR.(T.GT.NBRPAS)) THEN

     AUX=MINA(T-NBRLON(INDEX-1))
     DO 45 S=1,N+1
        V(S-1)=V(S-1)+AUX*D(S)
     CONTINUE
     WRITE(US,31)AUX
     DO 51 I=T-NBRLON(INDEX-1),T-1
      IF (MINB(2,I).LT.0) THEN

       CALL UPDATE(I1,I2,MINB(1,I))
```

```
        ELSE

          CALL UPDATE(I2,IO,MINB(1,I))
          IF (MINB(1,I).LE.M) THEN

            DO 350 S=1,N+1
               H(S)=H(S)+C(S-1,MINB(1,I))
            CONTINUE

          ELSE

            H(1)=H(1)+C(0,MINB(1,I)-M)
            DO 355 S=2,N+1
               H(S)=H(S)-C(S-1,MINB(1,I)-M)
            CONTINUE

          ENDIF

        ENDIF
      CONTINUE
      ENDIF

    ENDIF

 ENDIF

END
```

```
**********************************
NE DE CALCUL DE Q ET DU RANG DE X
**********************************

 ROUTINE CALCULE LA FACTORISATION QR D'UNE MATRICE X A PARTIR
 ROUTINE PRINCIPALE "QRMAIN"(LINPACK).

BLES D'ENTREE  : X=MATRICE A FACTORISER DE DIMENSIONS REELLES (N,P)
                 LDX="LEADING" DIMENSION DE X
                 N,P=DIMENSIONS REELLES DE X
                 TOL=TOLERANCE DEFINISSANT L'ASSIMILATION D'UN ELEMENT
                     DE R A ZERO;SI R(I,I)<=TOL*R(1,1),ALORS R(I,I)=0.
                 IW=2*N+2*P+LDX*N=DIMENSION DU VECTEUR DE TRAVAIL WORK
                 IN=INDICE D'ORIENTATION D'IMPRESSION DES RESULTATS

BLES DE SORTIE : IRANG=RANG DE X
                 Q=MATRICE DE DIMENSIONS REELLES (N,N) DE LA FACTORI-
                     SATION QR DE X

BLES INTERNES :  JPVT=VECTEUR DE TRAVAIL DE DIMENSION P RELATIF A DQRDC
                     (VOIR LINPACK)
                 WORK=VECTEUR DE TRAVAIL DE DIMENSION IW DANS LEQUEL
                      SONT STOCKEES TOUTES LES VARIABLES INTERNES DE
                      QRMAIN;LEUR LOCALISATION EST LA SUIVANTE :
 COMPOSANTES 1 ---> N : TRAV(N)= VECTEUR DE TRAVAIL POUR LA CONSTRUCTION
                        DE Q DEFINI DANS QRMAIN
             N+1 ---> N+P : QRAUX(P)= VECTEUR DEFINI DANS LINPACK
             N+P+1 ---> N+2*P : WORK(P)= VECTEUR DEFINI DANS LINPACK
             N+2*P+1 ---> 2*N+2*P : U(N)=VECTEUR DEFINI DANS QRMAIN
             2*N+2*P+1 ---> 2*N+2*P+LDX*N : H(LDX,N)=MATRICE DEFINIE
                                            DANS QRMAIN.

 SUBROUTINE DQR(X,LDX,N,P,TOL,IRANG,Q,IW,WORK,JPVT)

 INTEGER P,JPVT(P),LDX,N
 DOUBLE PRECISION X(LDX,P),Q(LDX,N),WORK(IW),TOL

 I1=1
 I2=N+1
 I3=N+P+1
 I4=N+2*P+1
 I5=2*N+2*P+1

 CALL QRMAIN(X,LDX,N,P,TOL,IRANG,Q,JPVT,WORK(I1),WORK(I2)
 1,WORK(I3),WORK(I4),WORK(I5))

 RETURN
 END


**************************************************
NE PRINCIPALE DE CALCUL DE Q ET DU RANG DE X
**************************************************

 ROUTINE CALCULE LA DECOMPOSITION QR D'UNE MATRICE X DE DIMENSION
E (N,P),EN FOURNIT LE RANG AINSI QUE Q,R,LES REFLECTEURS ET LEURS
ITS SUCCESSIFS.LA MATRICE R ET LES VECTEURS PERMETTANT LA CONSTRUC-
DES REFLECTEURS SONT FOURNIT PAR LA ROUTINE DQRDC (LINPACK).

BLES D'ENTREE  : X,LDX,N,P,TOL = VARIABLES DECRITES DANS DQR
```

```
BLES DE SORTIE : IRANG,Q = VARIABLES DECRITES DANS DQR

BLES INTERNES  : RAUX,JPVT,WORK=VECTEUR RELATIF A "DQRDC" (VOIR LINPACK)
                 U=VECTEUR DE DIMENSION REELLE N A PARTIR DUQUEL EST
                   CONSTRUIT LE REFLECTEUR
                 H=MATRICE DE DIMENSION REELLE (N,N) CORESPONDANT AU
                   REFLECTEUR H(U)
                 TRAV=VECTEUR DE TRAVAIL UTILISE LORS DU PRODUIT DES
                   REFLECTEURS


 SUBROUTINE QRMAIN(X,LDX,N,P,TOL,IRANG,Q,JPVT,TRAV,QRAUX,WORK
 1,U,H)

 INTEGER P,JPVT(P)
 DOUBLE PRECISION X(LDX,P),U(N),Q(LDX,N),QRAUX(P),WORK(P),TRAV(N),
 1A,TOL,H(LDX,N)


ITIALISATION DE JPVT :
----------------------

  DO 10 I=1,P
    JPVT(I)=0
    CONTINUE

COMPOSITION QR DE X :
---------------------

  CALL DQRDC(X,LDX,N,P,QRAUX,JPVT,WORK,1)
==========================================
  WRITE(IN,300)
  FORMAT(///' MATRICE X APRES DECOMPOSITION QR :')
  WRITE(IN,301)
  FORMAT(' ********************************')
  DO 310 I=1,N
    WRITE(IN,201)(X(I,J),J=1,P)
    CONTINUE
  WRITE(IN,320)
  FORMAT(//,' VECTEUR QRAUX :')
  WRITE(IN,201)(QRAUX(I),I=1,P)
  WRITE(IN,330)
  FORMAT(//' VECTEUR JPVT :')
  WRITE(IN,340)(JPVT(I),I=1,P)
  FORMAT(10I3)
  FORMAT(/X,10(D12.4),X)
==========================================

TECTION DU RANG DE X :
----------------------

  IRANG=0
  M=MINO(N,P)
  DO 20 K=1,M
    IF(DABS(X(K,K)).LE.TOL*DABS(X(1,1)))GO TO 30
    IRANG=K
    CONTINUE
```

```
    CONTINUE
=============================================
  WRITE(IN,21)
  FORMAT(//' LE RANG DE LA MATRICE X =')
  WRITE(IN,22)IRANG
  FORMAT('+',27X,I2)
=============================================

ONSTRUCTION DE Q :
------------------

 1 INITIALISATION DU PREMIER VECTEUR U ET DE H1=H(U):
   -------------------------------------------------

  U(1)=QRAUX(1)
  DO 40 I=2,N
     U(I)=X(I,1)
     CONTINUE
  CALL REFLEC(LDX,N,U,Q)
==============================
  FORMAT('+',15X,I2)
  WRITE(IN,1510)
  FORMAT(////' REFLECTEUR NO.1 :')
  WRITE(IN,1511)
  FORMAT(' ****************'//)

  WRITE(IN,1512)
  FORMAT(' VECTEUR :')
  WRITE(IN,1513)
  FORMAT(' ----------')
  WRITE(IN,201)(U(I),I=1,N)

  WRITE(IN,1514)
  FORMAT(//' MATRICE :')
  WRITE(IN,1513)
  DO 1520 I=1,N
     WRITE(IN,201)(Q(I,J),J=1,N)
     CONTINUE
==============================


 2 BOUCLE SUR I =1,IRANG-1 :
   -------------------------

  II=IRANG-1
  DO 50 I=1,II

  4.2.1. INITIALISATION DE U (VECTEUR DEFINISSANT LE REFLECTEUR
         NO. I+1) ET DE NN (NBRE DE COLONNES DE H(U)) :
         ---------------------------------------------

     NN=N-I
     U(1)=QRAUX(I+1)
     DO 90 J=2,NN
        U(J)=X(I+J,I+1)
        CONTINUE
==============================
  WRITE(IN,1550)
  FORMAT(//' REFLECTEUR NO.')
  IND=I+1
```

```
      WRITE(IN,503)IND
      WRITE(IN,1511)

      WRITE(IN,1512)
      WRITE(IN,1513)
      WRITE(IN,201)(U(IM),IM=1,NN)
=================================
```

4.2.2. CONSTRUCTION DE H :

```
      CALL REFLEC(LDX,NN,U,H)

=================================
      WRITE(IN,1514)
      WRITE(IN,1513)
      DO 1560 IH=1,NN
        WRITE(IN,201)(H(IH,J),J=1,NN)
        CONTINUE
=================================
```

4.2.3. PRODUIT Q=Q(I+1,...,N)H(U):

```
        DO 100 J=1,N
          DO 110 K=I+1,N
            A=0.
            DO 120 L=I+1,N
              A=A+Q(J,L)*H(L-I,K-I)
              CONTINUE
            TRAV(K)=A
            CONTINUE
          DO 130 IK=I+1,N
            Q(J,IK)=TRAV(IK)
            CONTINUE
        CONTINUE

=================================================
      WRITE(IN,1570)
      FORMAT(////' PRODUIT DES REFLECTEURS H1...H')
      WRITE(IN,1571)IND
      FORMAT('+',31X,I2)
      WRITE(IN,1572)
      FORMAT(' ********************************'//)

      DO 1580 I1=1,N
        WRITE(IN,201)(Q(I1,J),J=1,N)
        CONTINUE
=================================================

       CONTINUE
      RETURN
      END
```

```
**********************************
NE DE CONSTRUCTION D'UN REFLECTEUR
**********************************

  SUBROUTINE REFLEC(LDX,N,U,H)
  DOUBLE PRECISION U(N),H(LDX,N),MULT

BLES D'ENTREE :  U=VECTEUR DE DIMENSION REELLE N A PARTIR DUQUEL
                   SERA CONSTRUIT LE REFLECTEUR
                 N=DIMENSION DU VECTEUR U
                 LDX="LEADING" DIMENSION DU REFLECTEUR

BLES DE SORTIE : H=REFLECTEUR CONSTRUIT A PARTIR DE U,DE DIMENSION
                   REELLE (N,N).

MPLISSAGE DU TRIANGLE SUPERIEUR :
--------------------------------

MPLISSAGE DE LA PREMIERE LIGNE :

  H(1,1)=1.-U(1)
  DO 10 I=2,N
    H(1,I)=-U(I)
    CONTINUE

MPLISSAGE DE LA LIGNE NO.J :

  DO 20 J=2,N
    MULT=U(J)/U(1)
    H(J,J)=1.-MULT*U(J)
    DO 30 I=J+1,N
      H(J,I)=-MULT*U(I)
      CONTINUE
    CONTINUE

MPLISSAGE DU TRIANGLE INFERIEUR :
--------------------------------

  DO 40 I=2,N
    DO 50 J=1,I-1
      H(I,J)=H(J,I)
      CONTINUE
    CONTINUE
  RETURN
  END
```

BIBLIOGRAPHIE

1 AVRIEL M., "Non linear programming, analysis and method", Prentice-Hall, Englewood Cliffs, New-Jersey, 1976.

2 BARTELS R.B., CONN A. et CHARALAMBOUS C., "On cline's direct method for solving overdetermined linear systems in the ℓ_∞ sense", SIAM J. Numer. Anal., Vol. 15, N° 2, April 1978, pp. 255-270.

3 BARTELS R.H. et CONN A.R., "Linealy constrained discrete ℓ_1 Problems", ACM Transaction on Mathematical software, Vol. 6, N° 4, December 1980, pp. 594-608.

4 BARTELS R.H., CONN A.R. et LI YIUNG, "Primal methods are better than dual methods for solving overdetermined linear systems in the ℓ_∞ sense ?", Research Report CS-87-12, University of Waterloo, Computer Science department, Februrary 1987.

5 BAZARAA Mokhtar S. et SHETTY C.M., "Non linear programming, theory and algorithms", Ed. John Wiley and sons, USA, 1979.

6 CARRASSO C. et LAURENT P.J., "un algorithme de minimisation en chaîne en optimisation convexe.", SIAM J. Control Optimization 16, 1978, pp. 209-235.

7 CLINE A.K., "A descent method for the uniform solution to overdetermined systems of linear equations", SIAM J Numer. Anal., Vol 13, N° 3, June 1976, pp. 293-309.

September 1973, pp. 760-789.

9 CONN A.R., "Linear programming via a nondifferentiable penalty function", SIAM J. Numer. Anal., Vol 13, N° 1, March 1976, pp. 145-154.

10 DURIS C.S. et TEMPLE M.G., "A finite step algorithm for determining the strict Chebyshev solution to Ax = b", SIAM J. Numer. Anal., Vol. 10, N° 4, September 1973, pp. 690-699.

11 EVANS J-P., GOULD F.J. et TOLLE J.W., "Exact penalty functions in nonlinear programming", Mathematical programming 4, 1973, pp. 72-97.

12 HALMOS Paul R., "Finite-dimensional vector spaces", D. VAN NOSTRAND Co. inc., Princeton New Jersey, 1958.

13 HAN S.P. et MANGASARIAN O.L., "Exact penalty functions in nonlinear programming", Mathematical programming 17, 1979, pp. 251-269.

14 JOE B. et BARTELS R., "An exact penalty method for constrained, discrete, linear l_∞ data fitting", SIAM J. Sci. Stat. Comp., Vol 4, N° 1, March 1983, pp. 69-84.

15 PIETRZYKOWSKI T., "An exact potential method for constrained maxima", SIAM J. Numer. Anal., Vol 6, N° 2, June 1969, pp. 299-304.

general systems of linear equations", Numer. Math. 51, 1987, pp. 701-725.

17 WATSON, PHILIPSON et OATES, "Numerical analysis - The mathematics of computing 2", E. Arnold, London, 1974.

18 WATSON G.A., "A multiple exchange algorithm for multivariate Chebyshev approximation", SIAM J. Numer. Anal., Vol 12, N° 1, March 1975, pp. 46-52.

19 WATSON G.A., "Approximation theory and numerical methods", Ed. John Wiley and Sons, USA, 1980.

