Études adaptatives et comparatives de certains algorithmes en optimisation : implémentations effectives et applications

Adnan Yassine

THESE

Présentée à

l'Université Joseph Fourier - Grenoble I

pour obtenir le grade de
Docteur de l'Université Joseph Fourier
"Mathématiques Appliquées"

par

Adnan YASSINE

ETUDES ADAPTATIVES ET COMPARATIVES DE CERTAINS ALGORITHMES EN OPTIMISATION. IMPLEMENTATIONS EFFECTIVES ET APPLICATIONS.

Thèse soutenue le 4 Juillet 1989 devant la Commission d'Examen

P.J. LAURENT	Président
R. JANIN	
VAN HIEN NGUYEN	Rapporteurs
PHAM DINH TAO	
F. ROBERT	Examinateurs
R. CLERMONT	
G. DROUET D'AUBIGNY	Invités

THESE PREPAREE AU SEIN DU LABORATOIRE TIM 3

UNIVERSITE Joseph FOURIER (GRENOBLE I)

Président de l'Université :
M. PAYAN Jean Jacques

Année Universitaire 1987 - 1988

MEMBRES DU CORPS ENSEIGNANT DE SCIENCES ET DE GEOGRAPHIE

PROFESSEURS DE 1ère Classe

ARNAUD Paul	Chimie Organique
ARVIEU ROBERT	Physique Nucléaire I.S.N.
AUBERT Guy	Physique C.N.R.S
AURIAULT Jean-Louis	Mécanique
AYANT Yves	Physique Approfondie
BARBIER Marie-Jeanne	Electrochimie
BARJON Robert	Physique Nucléaire ISN
BARNOUD Fernand	Biochimie Macromoléculaire Végétale
BARRA Jean-René	Statistiques-Mathématiques Appliquées
BECKER Pierre	Physique
BEGUIN Claude	Chimie Organique
BELORISKY Elie	Physique
BENZAKEN Claude	Mathématiques Pures
BERARD Pierre	Mathématiques Pures
BERNARD Alain	Mathématiques Pures
BERTRANDIAS Françoise	Mathématiques Pures
BERTRANDIAS Jean-Paul	Mathématiques Pures
BILLET Jean	Géographie
BOELHER Jean-Paul	Mécanique
BONNIER Jane Marie	Chimie Générale
BOUCHEZ Robert	Physique Nucléaire ISN
BRAVARD Yves	Géographie
CARLIER Georges	Biologie Végétale
CAUQUIS Georges	Chimie Organique
CHARDON Michel	Géographie
CHIBON Pierre	Biologie Animale
COHEN ADDAD Jean-Pierre	Physique
COLIN DE VERDIERE Yves	Mathématiques Pures
CYROT Michel	Physique du Solide
DEBELMAS Jacques	Géologie Générale
DEGRANGE Charles	Zoologie
DEMAILLY Jean-Pierre	Mathématiques Pures
DENEUVILLE Alain	Physique
DEPORTES Charles	Chimie Minérale
DOLIQUE Jean-Michel	Physique des Plasmas
DOUCE Roland	Physiologie Végétale
DUCROS Pierre	Cristallographie
FONTAINE Jean-Marc	Mathématiques Pures
GAGNAIRE Didier	Chimie Physique
GERMAIN Jean-Pierre	Mécanique,
GIRAUD Pierre	Géologie
HICTER Pierre	Chimie
IDELMAN Simon	Physiologie Animale
JANIN Bernard	Géographie
JOLY Jean-René	Mathématiques Pures
KAHANE André, détaché	Physique
KAHANE Josette	Physique
KRAKOWIAK Sacha	Mathématiques Appliquées

LAJZEROWICZ Jeanine	Physique
LAJZEROWICZ Joseph	Physique
LAURENT Pierre-Jean	Mathématiques Appliquées
LEBRETON Alain	Mathématiques Appliquées
DE LEIRIS Joël	Biologie
LHOMME Jean	Chimie
LLIBOUTRY Louis	Géophysique
LOISEAUX Jean-Marie	Sciences Nucléaires I.S.N.
LUNA Domingo	Mathématiques Pures
MACHE Régis	Physiologie Végétale
MASCLE Georges	Géologie
MAYNARD Roger	Physique du Solide
OMONT Alain	Astrophysique
OZENDA Paul	Botanique (Biologie Végétale)
PAYAN Jean-Jacques	Mathématiques Pures
PEBAY-PEYROULA Jean-Claude	Physique
PERRIER Guy	Géophysique
PIERRARD Jean-Marie	Mécanique
PIERRE Jean-Louis	Chimie Organique
RENARD Michel	Thermodynamique
RINAUDO Marguerite	Chimie CERMAV
ROSSI André	Biologie
SAXOD Raymond	Biologie Animale
SENGEL Philippe	Biologie Animale
SERGERAERT Francis	Mathématiques Pures
SOUCHIER Bernard	Biologie
SOUTIF Michel	Physique
STUTZ Pierre	Mécanique
TRILLING Laurent	Mathématiques Appliquées
VALENTIN Jacques	Physique Nucléaire I.S.N.
VAN CUTSEM Bernard	Mathématiques Appliquées
VIALON Pierre	Géologie

PROFESSEURS de 2ème Classe

ADIBA Michel	Mathématiques Pures
ANTOINE Pierre	Géologie
ARMAND Gilbert	Géographie
BARET Paul	Chimie
BLANCHI J.Pierre	STAPS
BLUM Jacques	Mathématiques Appliquées
BOITET Christian	Mathématiques Appliquées
BORNAREL Jean	Physique
BRUANDET J.François	Physique
BRUGAL Gérard	Biologie
BRUN Gilbert	Biologie
CASTAING Bernard	Physique
CERFF Rudiger	Biologie
CHIARAMELLA Yves	Mathématiques Appliquées
COURT Jean	Chimie
DUFRESNOY Alain	Mathématiques Pures
GASPARD François	Physique
GAUTRON René	Chimie
GENIES Eugène	Chimie
GIDON Maurice	Géologie
GIGNOUX Claude	Sciences Nucléaires
GILLARD Roland	Mathématiques Pures
GIORNI Alain	Sciences Nucléaires
GONZALEZ SPRINBERG Gérardo	Mathématiques Pures
GUIGO Maryse	Géographie
GUMUCHAIN Hervé	Géographie
GUITTON Jacques	Chimie

HACQUES Gérard	Mathématiques Appliquées
HERBIN Jacky	Géographie
HERAULT Jeanny	Physique
JARDON Pierre	Chimie
JOSELEAU Jean-Paul	Biochimie
KERCKHOVE Claude	Géologie
LONGEQUEUE Nicole	Sciences Nucléaires I.S.N.
LUCAS Robert	Physique
MANDARON Paul	Biologie
MARTINEZ Francis	Mathématiques Appliquées
NEMOZ Alain	Thermodymamique CNRS - CRTBT
OUDET Bruno	Mathématiques Appliquées
PECHER Arnaud	Géologie
PELMONT Jean	Biochimie
PERRIN Claude	Sciences Nucléaires I.S.N.
PFISTER Jean-Claude	Physique du Solide
PIBOULE Michel	Géologie
RAYNAUD Hervé	Mathématiques Appliquées
RICHARD Jean-Marc	Physique
RIEDTMANN Christine	Mathématiques Pures
ROBERT Gilles	Mathématiques Pures
ROBERT Jean-Bernard	Chimie Physique
SARROT-REYNAULD Jean	Géologie
SAYETAT Françoise	Physique
SERVE Denis	Chimie
STOECKEL Frédéric	Physique
SCHOLL Pierre-Claude	Mathématiques Appliquées
SUBRA Robert	Chimie
VALLADE Marcel	Physique
VIDAL Michel	Chimie Organique
VIVIAN Robert	Géographie
VOTTERO Philippe	Chimie

MEMBRES DU CORPS ENSEIGNANT DE L' IUT 1

PROFESSEURS de 1ère Classe

BUISSON Roger	Physique IUT 1
DODU Jacques	Mécanique Appliquée IUT 1
NEGRE Robert	Génie Civil IUT 1
NOUGARET Marcel	Automatique IUT 1
PERARD Jacques	EEA. IUT 1

PROFESSEURS de 2ème classe

BOUTHINON Michel	EEA. IUT 1
CHAMBON René	Génie Mécanique IUT 1
CHEHIKIAN Alain	EEA. IUT 1
CHENAVAS Jean	Physique IUT 1
CHOUTEAU Gérard	Physique IUT 1
CONTE René	Physique IUT 1
GOSSE Jean-Pierre	EEA.IUT 1
GROS Yves	Physique IUT 1
KUHN Gérard, (Détaché)	Physique IUT 1
MAZUER Jean	Physique IUT 1
MICHOULIER Jean	Physique IUT 1
MONLLOR Christian	EEA.IUT 1
PEFFEN René	Métallurgie IUT 1
PERRAUD Robert	Chimie IUT 1
PIERRE Gérard	Chimie IUT 1
TERRIEZ Jean-Michel	Génie Mécanique IUT 1
TOUZAIN Philippe	Chimie IUT 1
VINCENDON Marc	Chimie IUT 1

PROFESSEURS DE PHARMACIE

AGNIUS-DELORD Claudine	Physique	Faculté La Tronche
ALARY Josette	Chimie Analytique	Faculté La Tronche
BERIEL Hélène	Physiologie et Pharmacologie	Faculté La Tronche
CUSSAC Max	Chimie Therapeutique	Faculté La Tronche
DEMENGE Pierre	Pharmacodynamie	Faculté La Tronche
FAVIER Alain	Biochimie	C.H.R.G.
JEANNIN Charles	Pharmacie Galénique	Faculté Meylan
LATURAZE Jean	Biochimie	Faculté La Tronche
LUU DUC Cuong	Chimie Générale	Faculté La Tronche
MARIOTTE Anne-Marie	Pharmacognosie	Faculté La Tronche
MARZIN Daniel	Toxicologie	Faculté Meylan
RENAUDET Jacqueline	Bactériologie	Faculté La Tronche
ROCHAT Jacques	Hygiéne et Hydrologie	Faculté La Tronche
SEIGLE-MURANDI Françoise	Botanique et Cryptogamie	Faculté Meylan
VERAIN Alice	Pharmacie Galénique	Faculté Meylan

MEMBRES DU CORPS ENSEIGNANT DE MEDECINE

PROFESSEURS CLASSE EXEPTIONNELLE ET 1ère CLASSE

AMBLARD Pierre	Dermatologie	C.H.R.G.
AMBROISE-THOMAS Pierre	Parasitologie	C.H.R.G.
BEAUDOING André	Pédiatrie-Puericulture	C.H.R.G.
BEZEZ Henri	Orthopédie-Traumatologie	Hopital SUD
BONNET Jean-Louis	Ophtalmologie	C.H.R.G.
BOUCHET Yves	Anatomie	Faculté La Merci
	Chirurgie Générale et Digestive	C.H.R.G.
BUTEL Jean	Orthopédie-Traumatologie	C.H.R.G.
CHAMBAZ Edmond	Biochimie	C.H.R.G.
CHAMPETIER Jean	Anatomie-Topographique et Appliquée	C.H.R.G.
CHARACHON Robert	O.R.L.	C.H.R.G.
COLOMB Maurice	Immunologie	Hopital sud
COUDERC Pierre	Anatomie-Pathologique	C.H.R.G.
DELORMAS Pierre	Pneumophtisiologie	C.H.R.G.
DENIS Bernard	Cardiologie	C.H.R.G.
GAVEND Michel	Pharmacologie	Faculté La Merci
HOLLARD Daniel	Hématologie	C.H.R.G.
LATREILLE René	Chirurgie Thoracique et Cardiovasculaire	C.H.R.G.
LE NOC Pierre	Bactériologie-Virologie	C.H.R.G.
MALINAS Yves	Gynécologie et Obstétrique	C.H.R.G.
MALLION Jean-Michel	Médecine du Travail	C.H.R.G.
MICOUD Max	Clinique Médicale et Maladies Infectieuses	C.H.R.G.
MOURIQUAND Claude	Histologie	Faculté La Merci
PARAMELLE Bernard	Pneumologie	C.H.R.G.
PERRET Jean	Neurologie	C.H.R.G.
RACHAIL Michel	Hépato-Gastro-Entérologie	C.H.R.G.
DE ROUGEMONT Jacques	Neurochirurgie	C.H.R.G.
SARRAZIN Roger	Clinique Chirurgicale	C.H.R.G.
STIEGLITZ Paul	Anestésiologie	C.H.R.G.
TANCHE Maurice	Physiologie	Faculté La Merci
VIGNAIS Pierre	Biochimie	Faculté La Merci

PROFESSEURS 2ème CLASSE

BACHELOT Yvan	Endocrinologie	C.H.R.G.
BARGE Michel	Neurochirurgie	C.H.R.G.
BENABID Alim Louis	Biophysique	Faculté La Merci
BENSA Jean-Claude	Immunologie	Hopital Sud
BERNARD Pierre	Gynécologie-Obstétrique	C.H.R.G.
BESSARD Germain	Pharmacologie	ABIDJAN
BOLLA Michel	Radiothérapie	C.H.R.G.
BOST Michel	Pédiatrie	C.H.R.G.
BOUCHARLAT Jacques	Psychiatrie Adultes	Hopital Sud
BRAMBILLA Christian	Pneumologie	C.H.R.G.
CHIROSSEL Jean-Paul	Anatomie-Neurochirurgie	C.H.R.G.
COMET Michel	Biophysique	Faculté La Merci
CONTAMIN Charles	Chirurgie Thoracique et Cardiovasculaire	C.H.R.G.
CORDONNIER Daniel	Néphrologie	C.H.R.G.
COULOMB Max	Radiologie	C.H.R.G.
CROUZET Guy	Radiologie	C.H.R.G.
DEBRU Jean-Luc	Médecine Interne et Toxicologie	C.H.R.G.
DEMONGEOT Jacques	Biostatistiques et Informatique Médicale	Faculté La Merci
DUPRE Alain	Chirurgie Générale	C.H.R.G.
DYON Jean-François	Chirurgie Infantile	C.H.R.G.
ETERRADOSSI Jacqueline	Physiologie	Faculté La Merci
FAURE Claude	Anatomie et Organogénèse	C.H.R.G.
FAURE Gilbert	Urologie	C.H.R.G.
FOURNET Jacques	Hépato-Gastro-Entérologie	C.H.R.G.
FRANCO Alain	Médecine Interne	C.H.R.G.
GIRARDET Pierre	Anesthésiologie	C.H.R.G.
GUIDICELLI Henri	Chirurgie Générale et Vasculaire	C.H.R.G.
GUIGNIER Michel	Thérapeutique et Réanimation Médicale	C.H.R.G.
HADJIAN Arthur	Biochimie	Faculté La Merci
HALIMI Serge	Endocrinologie et Maladies Métaboliques	C.H.R.G.
HOSTEIN Jean	Hépato-Gastro-Entérologie	C.H.R.G.
HUGONOT Robert	Médecine Interne	C.H.R.G.
JALBERT Pierre	Histologie-Cytogénétique	C.H.R.G.
JUNIEN-LAVILLAUROY Claude	O.R.L.	C.H.R.G.
KOLODIE Lucien	Hématologie Biologique	C.H.R.G.
LETOUBLON Christian	Chirurgie Générale	C.H.R.G.
MACHECOURT Jacques	Cardiologie et Maladies Vasculaires	C.H.R.G.
MAGNIN Robert	Hygiène	C.H.R.G.
MASSOT Christian	Médecine Interne	C.H.R.G.
MOUILLON Michel	Ophtalmologie	C.H.R.G.
PELLAT Jacques	Neurologie	C.H.R.G.
PHELIP Xavier	Rhumatologie	C.H.R.G.
RACINET Claude	Gynécologie-Obstétrique	Hopital Sud
RAMBAUD Pierre	Pédiatrie	C.H.R.G.
RAPHAEL Bernard	Stomatologie	C.H.R.G.
SCHAERER René	Cancérologie	C.H.R.G.
SEIGNEURIN Jean-Marie	Bactériologie-Virologie	Faculté La Merci
SELE Bernard	Cytogénétique	Faculté La Merci
SOTTO Jean-Jacques	Hématologie	C.H.R.G.
STOEBNER Pierre	Anatomie Pathologique	C.H.R.G.
VROUSOS Constantin	Radiothérapie	C.H.R.G.

بسم الله الرحمن الرحيم

A mes parents,
A mes frères et sœurs,
A ma famille,
A tous ceux qui me sont chers,

Je dédie ce travail

الى أمي الحنون ... نبع الحنان ومصدر الأمان ، من ضحت بزهرة شبابها لتمهد لنا الطريق وقبلت الصعاب لنصل الى هذا .

الى أبي الغالي ... الذي رباني فاحسن تربيتي ، وطعم عقلي بالعلم والإيمان وغذاني بالحلال .

الى كل أخواتي وأخوتي الاحبه ، الذين كان وجودهم مبعث قوة لي و حافزاً لتقدمي وإستمراري وحبهم سراً في نجاحي .

الى كل من ساهم في تربيتنا وكان مصدر سعادة لنا في هذه الحياة

أهدي هذا العمل المتواضع . جعله الله ذخراً لنا

عدنان

Cette thèse a été préparée au sein du Laboratoire TIM3 de l'Institut d'Informatique et de Mathématiques Appliquées de Grenoble.

Mes sincères remerciements s'adressent à mon professeur, Monsieur Pham Dinh Tao, Directeur de l'Equipe Modélisation & Optimisation au Laboratoire TIM3, pour sa disponibilité, ses encouragements, ses conseils scientifiques et humains, son aide constante m'a permis de mener à bien ce travail dans les meilleures conditions. Il a su me faire profiter avec beaucoup de modestie de sa profonde connaissance en optimisation, j'avoue que cela a été pour moi un émerveillement et enrichissement. Je lui témoigne toute ma gratitude.

Je suis très sensible à l'honneur que me fait Monsieur le professeur Pierre-Jean Laurent, professeur à l'Université Joseph Fourier - Grenoble I, en acceptant de présider ce jury. Il a grandement contribué au developpement de l'analyse et l'optimisation convexes sur le plan national et international. Nous sommes tous, les membres de l'Equipe Optimisation & Modélisation, ses élèves et fiers de perpétuer l'héritage intellectuel qu'il nous a laissé. Qu'il permette de lui témoigner notre sincère et respectueuse reconnaissance.

Mes vifs remerciements vont également à :

Monsieur François Robert, professeur à l'I.N.P.G pour ses enseignements et l'intérêt qu'il porte à une partie de ce travail et pour son amabilité d'avoir bien voulu participer au jury.

Messieurs les professeurs Nguyen Van Hien, Directeur du département de Mathématiques à l'Université de Namur (Belgique) et Robert Janin, de l'Université de Poitiers, pour l'attention toute particulière qu'ils ont accordée à ce travail et qui ont accépté d'en être les rapporteurs extérieurs.

Messieurs Clermont Robert, Directeur de Recherches à l'Institut de Mécanique de Grenoble, et Gérard Drouet D'Aubigny, Professeur à l'Université de Grenoble II, pour les discussions et les nombreuses collaborations fructueuses que nous avons pu avoir ensemble, et pour avoir accepté de faire partie du Jury.

Les collaborations avec S. Wang de l'Equipe de Calcul Parallèle (Labo TIM3) et Marie Emmanuelle de la Lande de l'Equipe Rhéologie des Fluides Industriels (Institut de Mécanique de Grenoble) sur la résolution numérique de quelques problèmes importants de Mathématiques Appliquées ont été très bénéfiques pour nous tous. Qu'ils soient remerciés pour leur amabilité et leur dévouement.

J'adresse ma reconnaissance à ma chère Karima. Dans les moments les plus difficiles, elle m'a offert son amour, sa sympathie et son soutien moral qui m'a été d'une grande vertu.

Je remercie également la "Famille", mes collègues de l'Equipe Modélisation & Optimisation avec lesquels j'ai eu des discussions et échanges de vues intéressants, sans oublier les années de bonheur, d'amitié et d'harmonie dans le travail.

Je tiens enfin à exprimer toute ma sympathie à tous mes grands amis qui ont rendu mon séjour en France très agréable, en particulier N. Hakim pour sa gentillesse et sa serviabilité.

Je remercie toute l'équipe du service de reprographie pour l'exellente qualité de leur travail.

Résumé :

Cette thèse est consacrée aux études adaptatives et comparatives de certains algorithmes en optimisation, à leurs implémentations effectives et leurs applications concrètes à la résolution de différents problèmes importants de Mathématiques Appliquées :
1) L'algorithme S.G.G.P pour la résolution d'un programme linéaire général.
2) La méthode de pivotage de Lemke, La méthode du gradient conjugué conditionnel, La méthode de l'inverse partiel pour la résolution des programmes quadratiques convexes.
3) Les méthodes d'approximation extérieure et les méthodes de coupes planes et les méthodes de région de confiance pour l'optimisation non convexe (minimisation d'une fonction concave sur un convexe, problèmes de moindres carrés non linéaires).
Les modélisations mathématiques adéquates et les expérimentations numériques comparatives ont été réalisées sur les problèmes suivants:
1) Les problèmes de régression en analyse de données.
2) Minimisation d'une fonction concave sur un convexe, problème non linéaire complémentaire.
3) Les réseaux neuronaux
4) L'écoulement bi et tridimensionnels à géométrie simplement connexe.
5) Les problèmes d'analyse multidimensionnelle des données de dissimilarité (MDS).
6) Minimisation d'une forme quadratique hermitienne sur la sphère euclidienne unité de C^n.

Mots-clés :

programmation linéaire, algorithme S.G.G.P, programmation linéaire et non linéaire complémentaire, programmation quadratique, méthode de pivotage de Lemke, gradient conjugué conditionnel, inverse partiel, régression isotone, régression concave, minimisation d'une fonction concave, méthodes de région de confiance, réseaux neuronaux, écoulements de fluides incompressibles, codage multidimensionnel.

Abstract :

This thesis studies comparative and adaptive algorithms for optimization, their implementation and their applications to the solution of various problems in Applied Mathematics :
1) The S.G.G.P algorithm to the solution of the general linear programming.
2) Lemke's pivoting method, the conditional conjugate method and the inverse partial method to the solution of the convex quadratic programs.
3) Outer approximation methods, cutting plane methods and trust region methods for non convex optimization (minimizing a concave function over a compact convex set, non linear least squares problems)
Adaquate mathematical modelling and comparative numerical experimentations have been realized in the following problems :
1) Regression problems in Data Analysis
2) Minimizing a concave function over a compact convex set, non linear complementary problem.
3) Neural networks
4) Analysis of plane and axisymmetric flows of incompressible fluids.
5) The multidimensional scaling problems of dissimilarity data. (Multidimensional Dissimilarity Scaling).
6) Minimizing of a hermitian quadratic form on the unit euclidean sphere on $\mathbf{C}^n$.

Key words :

Linear programming, S.G.G.P algorithm, quadratic programming, Linear and non linear complementary problem, Lemke's pivoting method, Conditional conjugate gradient, inverse partial, isotone regression, concave regression, Minimizing of concave function, Trust region method, Neural networks, axisymmetric flows of incompressible fluid, Multidimensional coding.

TABLE DES MATIERES

INTRODUCTION GENERALE

Cette thèse, composée de six chapitres, est spécialement consacrée aux études adaptatives et comparatives de certains algorithmes en optimisation, de leurs implémentations effectives et leurs applications concrètes à la résolution de différents problèmes importants de Mathématiques Appliquées.

Dans le premier chapitre, nous présentons un algorithme appelé SGGP (Simplexe Généralisé, Gradient Projeté) que Pham Dinh Tao a introduit récemment pour la résolution du programme linéaire général : Max{cx : Ax=b, Bx≤a}. Cet algorithme, plutôt situé dans le cadre de l'analyse et de l'optimisation convexes, est fondamentalement basé sur les conditions d'optimalité et sur la caractérisation d'un sommet de $K=\{x\in \mathbb{R}^n : Ax=b, Bx\leq a\}$ comme étant un point de K qui admet n contraintes actives linéairement indépendantes. Cet algorithme n'introduit pas les variables d'écart, on travaille directement dans l'espace des variables x.

La cohérence et l'unité de l'algorithme SGGP résident (comme dans l'algorithme du simplexe) dans le fait suivant: la phase I utilise la phase II. Deux nouvelles techniques de construction de la phase I (procédé primal et procédé dual) sont proposées.

Le procédé primal consiste en l'application de la phase II à une suite finie de programmes linéaires (dans le même espace primal $\mathbb{R}^n$) dont on connaît un sommet, solution réalisable. Ces programmes linéaires seront définis successivement en tenant compte des contraintes restant violées de K au sommet solution du programme linéaire précédent. On arrive ainsi à obtenir un sommet de K ou la conclusion de la vacuité de K.

Le procédé dual, par contre ne résoud qu'un seul programme linéaire. Son principe est très simple: on calcule un point défini par n contraintes linéairement indépendantes de K, (en général par la méthode d'élimination de Gauss par exemple) puis on définit le vecteur coût d (à partir des contraintes explicitées) de manière que le problème primal auxiliaire suivant soit dual réalisable (i.e. un sommet solution réalisable de son dual est explicité):

$$(PA)\begin{cases} Ax = b \\ Bx \leq a \\ dx = z(max) \end{cases}$$

L'application de la phase II de l'algorithme SGGP à ce problème dual auxiliaire conduit à un sommet de K (de surcroît solution du problème primal auxiliaire) ou à la conclusion de la vacuité de K.

Ces deux procédés éclairent sous un nouveau jour la recherche des techniques appropriées de détermination d'un sommet d'un polyèdre convexe.

L'application de SGGP au programme linéaire mis sous forme standard par rapport à une base réalisable nous redonne l'algorithme révisé du simplexe. Plus exactement, la description de l'algorithme du simplexe est la forme simplifiée de SGGP dans laquelle la résolution des systèmes linéaires (intervenant dans le calcul de la direction de déplacement, le pas de déplacement et la vérification des conditions d'optimalité) se fait via la technique de décomposition de ces systèmes (utilisant la structure creuse de la matrice représentant les contraintes de non négativité de la variable x). De même l'algorithme dual du simplexe est exactement l'algorithme SGGP appliqué au problème dual dont le primal est dual réalisable.

D'autre part la présentation de cet algorithme est particulièrement claire et constructive dans les programmes linéaires avec bornes. Enfin ce qui est fondamental est que l'algorithme SGGP n'est rien d'autre que l'algorithme du gradient projeté appliqué au programme linéaire général lorsque le point de départ est un sommet du polyèdre K. Ce point de vue original permet de situer l'algorithme du simplexe dans le cadre général de l'optimisation convexe.

Certains problèmes d'application qui étaient à l'origine de ce travail sont présentés ainsi que la description simplifiée de la phase I dans les programmes linéaires usuels.

Des expérimentations numériques comparatives sont exposées dans le dernier paragraphe. Ces résultats montrent que SGGP est plus rapide que l'algorithme révisé du simplexe : le rapport du temps total de calcul (phase I et phase II) varie entre 1 et 3.5. La rapidité est encore accrue pour la phase I dont le rapport varie entre 3 et 8.Ces résultats sont encourageants, ils prouvent l'utilité de l'algorithme SGGP.

Une étude comparative complète de ces algorithmes est en cours, dans laquelle le procédé de la "forme produit de l'inverse" sera utilisé pour optimiser les calculs dans les résolutions des systèmes linéaires.

Le chapitre II est consacré au problèmes linéaires et non linéaire complémentaires. Nous exposons, dans le premier temps, les domaines d'applications de ce type de problèmes. Puis, nous présentons la méthode de pivotage de Lemke, considérée comme une adaptation de la méthode du simplexe à la résolution de problèmes linéaires et non linéaires complémentaires (PLC).

Nous présenterons ensuite différents autres algorithmes de résolution du **PLC** (méthodes de Newton et méthodes itératives) et les propriétés de convergence pour chacun de ces algorithmes.

A la fin de ce chapitre, nous étudierons la méthode de Quasi-Newton (Newton inexacte) adaptée par Pang et la méthode de l'inverse partiel dévéloppée par Spingarn en 1983 pour la résolution de PNC.

L'algorithme de pivotage de Lemke et la méthode de l'inverse partiel ont été implémentés sur compatible PC (Olivetti_M24) et BULL DPS 8/70 M. sous MULTICS. Les expérimentations numériques relatives à la résolution des problèmes de régressions isotone et concave sont présentées dans le dernier chapitre.

Dans le chapitre III, nous présenterons le problème de la programmation quadratique dans un cadre théorique général. Nous exposons la méthode du gradient conjugué destinée à la minimisation d'une forme quadratique convexe dans tout l'espace (sans contrainte), elle est ensuite adaptée, via les conditions d'optimalité, à la programmation quadratique, i.e. la minimisation d'une forme quadratique convexe sur un polyèdre convexe.

Comme dans les algorithmes de type gradient projeté, les gros efforts dans la méthode du gradient conjugué conditionnel sont dûs au calcul des projections. La dualité permet de remédier à cet inconvénient. En effet le problème dual d'un programme quadratique est encore un programme quadratique qui possède un avantage de n'avoir que des contraintes de nonnégativité : la projection y sera explicite. Par contre la forme quadratique associée fait apparaître la matrice C^{-1} (C étant la matrice correspondant à la forme quadratique du problème primal).

La finitude de ces méthodes (convergence en un nombre fini d'itérations) : n (au plus) pour le cas sans contraintes, et k(n,p), fonction de la dimension de l'espace et du nombre des facettes du polyèdre convexe, pour le cas des contraintes linéaires) est particulièrement appréciée en pratique.

A la fin de ce chapitre, nous donnerons comme application, le problème de la régression isotone et celui de la régression concave.

Nous décrirons, ensuite, la transformation du problème de programmation quadratique en un problème linéaire complémentaire à partir des conditions d'optimalité de Kuhn-Tucker, et cela, en vue de l'application des algorithmes propres au programme linéaire complémentaire (à sa résolution) (cf. Chapitre II).

Une modélisation simple permet de prouver qu'il s'agit des programmes quadratiques. L'implémentation de l'algorithme du gradient conjugué conditionnel a été réalisée sur compatible PC (Olivetti_M24) et BULL DPS 8/70 M. sous MULTICS.

Une étude comparative de la performance de l'algorithme du gradient conjugué conditionnel (appliqué à la résolution de ces deux problèmes) est présentée dans le chapitre VI.

Pour terminer, il est intéressant de noter le rôle important de la dualité dans ces applications :
Pour résoudre le problème primal, les méthodes pivotage de Lemke et de l'inverse partiel travaillent directement dans le problème dual.

Quant à l'algorithme du gradient conjugué conditionnel, les experimentations numériques montrent que pour résoudre le problème primal, il est beaucoup plus rapide de l'appliquer au problème dual.

Dans le chapitre IV nous étudierons certaines méthodes qui permettent d'obtenir les solutions globales du problème d'optimisation suivant :

$$(P) : \begin{cases} \min\ f(x) \\ x \in S = \{x / g_i(x) \leq 0 ; i=1, \ldots, m\} \end{cases}$$

où f est une fonction concave définie sur $\mathbb{R}^n$ et g_i (i = 1, ..., m) sont des fonctions convexes sur $\mathbb{R}^n$.

Il s'agit des problèmes d'optimisation non convexe dont la difficulté majeure réside dans le fait qu'il n'existe pas de caractérisation complète de solution (i.e. sous forme de condition nécessaire et suffisante) comme en optimisation convexe.

Nous présentons deux méthodes dues respectivement à H. Tuy et K. Hofman pour résoudre le problème (P).

(i) La première méthode (méthode de partitionnement conique) basée sur l'introduction du procédé de subdivision de S et sur le calcul d'un certain minorant de f(x) dans chaque sous-domaine. Ceci, jusqu'à ce que l'on trouve une solution globale.

(ii) La deuxième utilise le principe de l'approximation extérieure de l'ensemble admissible S et la technique de coupes planes.

Ensuite, on fera une comparaison entre ces deux méthodes.

Dans la classification de problème d'optimisation non convexe faite par Pham Dinh Tao & El Bernoussi Souad [20,23], le problème (P) est un cas particulier du problème suivant :

$$(P^*) \qquad \min\{f(x) : x \in C,\ g(x) \geq 0\} \qquad (f, g, C \text{ convexes}).$$

L'étude des algorithmes pour la résolution du problème (P*), en dehors de l'aspect unificateur, permet d'obtenir un algorithme original (avec ses différentes versions). La description de cet algorithme est présentée dans le paragraphe § 4 .

Nous avons implémenté ces algorithmes sur PC (Olivetti_380) en langage PASCAL et les expérimentations numériques comparatives sont données à la fin de ce chapitre.

Le chapitre V est consacré à l'étude du problème de minimisation sans contrainte :

$$(P) : \begin{cases} \min\ f(x) \\ x \in \mathbb{R}^n \end{cases}$$

où f est une fonction de $\mathbb{R}^n$ dans $\mathbb{R}$, bornée inférieurement et deux fois continûment différentiable.

Nous désignons par g_k le gradient de f en x_k et par H_k le Hessien de f en x_k ou une matrice symétrique qui approxime le Hessien de f en x_k (quasi-Hessien).

En général, le problème (P) est traité par des variantes de la méthode de Newton qui consiste à trouver un zéro du gradient de la fonction f.
Ces variantes recouvrent aussi bien les méthodes utilisant l'approximation d'ordre 1 de f que celles qui utilisent les approximations de la matrice Hessienne de f.

Le but principal de ce chapitre est de décrire et d'analyser une technique pour la résolution de ce problème. L'approche que nous allons présenter est bien connue (Moré 1983, Sorensen 1981). Elle est communément appelée "méthode de région de confiance" dans laquelle le pas déterminant un nouvel itéré x_{k+1} est obtenu par la minimisation d'une approximation quadratique locale de la fonction objective sur une région restreinte (qui est la boule euclidienne centrée en x_k).

Le diamètre de cette région est agrandi ou réduit selon la qualité d'approximation du modèle quadratique local. Il est possible de contrôler l'itération de manière à forcer la convergence de la méthode sous certaines conditions raisonnables portant sur la fonction objective. La méthode de région de confiance génère une suite d'itérés en résolvant à chaque pas un problème quadratique de la forme:

$$(P_k) \quad : \quad \min \{q_k(d) : \|d\| \leq \delta_k\}.$$

où q_k désigne l'approximation quadratique de la variation de f en x_k définie par: $q_k(d) = \langle g_k, d\rangle + 1/2 \langle H_k d, d\rangle$

Le nombre strictement positif δ_k est appelé rayon de confiance. Dans les algorithmes de région de confiance, le calcul du gradient de la fonction objective est requis. L'utilisation des informations du premier ordre nous amène en général à des points stationnaires du premier ordre.
En y incorporant des informations du second ordre ces algorithmes pourraient satisfaire les conditions nécessaires du second ordre pour le problème (P) : dans ce cas la méthode de région de confiance peut être regardée comme une méthode de Newton modifiée appliquée à la recherche d'un zéro du gradient de la fonction objective.
La méthode "générale" de région de confiance peut être décrite comme suit:

En calculant la direction d_k, solution de (P_k), nous pouvons facilement contrôler la qualité de l'approximation locale $q_k(d)$ et par suite prendre la décision appropriée:

* Si l'approximation est satisfaisante, alors la solution de (P_k) établit un nouveau itéré et le rayon de confiance est augmenté.

* Dans le cas contraire, l'itéré est inchangé, en plus le rayon de confiance est diminué jusqu'à ce que q_k établisse une approximation satisfaisante à l'intérieur de la région de confiance (qui a nécessairement lieu pour de petits rayons car le gradient est supposé connu exactement et le terme linéaire en g_k devient dominant si $||g_k||\neq 0$).

Naturellement le schéma général peut être implémenté de différentes manières. La qualité de l'approximation est en général examinée à travers la qualité suivante appelée "coefficient de qualité":

$$r_k = \frac{f(x_k) - f(x_k+d_k)}{q_k(0) - q_k(d_k)}$$

Le numérateur représente l'actuelle réduction de la fonction f quand on se déplace de x_k à x_{k+1}, le dénominateur représente la réduction relative à l'approximation quadratique.

Ainsi dans un algorithme de région de confiance, les principaux efforts de calcul, à part les évaluations de la fonction requises, sont centrés dans la résolution de (P_k) pour déterminer le pas de déplacement.

Les algorithmes de région de confiance diffèrent dans leurs stratégies de résoudre approximativement (P_k).

Pour les besoins d'ordre pratique, nous avons implémenté diverses versions de la méthode de region de confiance sur différents matériels (PC, VAX, Station APOLLO DN 4000) en langage Pascal et Fortran.

A la fin de ce chapitre on expose les résultats de convergence de cette méthode.

Dans le chapitre VI, nous présentons les expérimentations numériques comparatives à travers les applications concrètes :

Dans un premier temps on expose les résultats numériques de la méthode de région de confiance appliquée sur les problèmes suivants :
Les réseaux neuronaux, l'écoulement de fluides incompressibles dans les conduites planes ou axisymétriques, les problèmes d'analyse multidimensionnelle des données de dissimilarité (MDS) et la minimisation d'une forme quadratique hermitienne sur la sphère euclidienne unité de C^n.

A la fin de ce chapitre, nous exposons les résultats numériques de différentes méthodes étudiées dans les chapitres II, III, IV (méthode de pivotage de Lemke, méthode du gradient conjugué et méthode de l'inverse partiel) pour la résolution de problèmes de régression isotone et concave.

CHAPITRE I

ALGORITHME S.G.G.P POUR LA RESOLUTION DU PROGRAMME LINEAIRE GENERAL IMPLEMENTATION.EXPERIMENTATIONS NUMERIQUES COMPARATIVES APPLICATIONS

1. Introduction

Nous présentons dans ce chapitre un algorithme appelé SGGP (Simplexe Généralisé, Gradient Projeté) que Pham Dinh Tao a introduit récemment pour la résolution du programme linéaire général : Max{cx : Ax=b, Bx≤a}. Cet algorithme, plutôt situé dans le cadre de l'analyse et de l'optimisation convexes, est fondamentalement basé sur les conditions d'optimalité et sur la caractérisation d'un sommet de K={x∈ $\mathbb{R}^n$: Ax=b, Bx≤a} comme étant un point de K qui admet n contraintes actives linéairement indépendantes. Cet algorithme n'introduit pas les variables d'écart, on travaille directement dans l'espace des variables x.

La cohérence et l'unité de l'algorithme SGGP résident (comme dans l'algorithme du simplexe) dans le fait suivant: la phase I utilise la phase II. Nous présentons ici deux nouvelles techniques de construction de la phase I (procédé primal et procédé dual).

Le procédé primal consiste en l'application de la phase II à une suite finie de programmes linéaires (dans le même espace primal $\mathbb{R}^n$) dont on connaît un sommet, solution réalisable. Ces programmes linéaires seront définis successivement en tenant compte des contraintes restant violées de K au sommet solution du programme linéaire précédent. On arrive ainsi à obtenir un sommet de K ou la conclusion de la vacuité de K.

Le procédé dual, par contre ne résoud qu'un seul programme linéaire. Son principe est très simple: on calcule un point défini par n contraintes linéairement indépendantes de K, (en général par la méthode d'élimination de Gauss par exemple) puis on définit le vecteur coût d (à partir des contraintes explicitées) de manière que le problème primal auxiliaire suivant soit dual réalisable (i.e. un sommet solution réalisable de son dual est explicité):

$$(PA)\begin{cases} Ax = b \\ Bx \leq a \\ dx = z(max) \end{cases}$$

L'application de la phase II de l'algorithme SGGP à ce problème dual auxiliaire conduit à un sommet de K (de surcroît solution du problème primal auxiliaire) ou à la conclusion de la vacuité de K.

Ces deux procédés éclairent sous un nouveau jour la recherche des techniques appropriées de détermination d'un sommet d'un polyèdre convexe. L'application de SGGP au programme linéaire mis sous forme standard par rapport à une base réalisable nous redonne l'algorithme révisé du simplexe. Plus exactement, la description de l'algorithme du simplexe est la forme simplifiée de SGGP dans laquelle la résolution des systèmes linéaires (intervenant dans le calcul de la direction de déplacement, le pas de déplacement et la vérification des conditions d'optimalité) se fait via la technique de décomposition de ces systèmes (utilisant la structure creuse de la matrice représentant les contraintes de non négativité de la variable x).

De même l'algorithme dual du simplexe est exactement l'algorithme SGGP appliqué au problème dual dont le primal est dual réalisable.

D'autre part la présentation de cet algorithme est particulièrement claire et constructive dans les programmes linéaires avec bornes. Enfin ce qui est fondamental est que l'algorithme SGGP n'est rien d'autre que l'algorithme du gradient projeté appliqué au programme linéaire général lorsque le point de départ est un sommet du polyèdre K. Ce point de vue original permet de situer l'algorithme du simplexe dans le cadre général de l'optimisation convexe.

Certains problèmes d'application qui étaient à l'origine de ce travail sont présentés ainsi que la description simplifiée de la phase I dans les programmes linéaires usuels.

Des expérimentations numériques comparatives sont exposées dans le dernier paragraphe. Ces résultats montrent que SGGP est plus rapide que l'algorithme révisé du simplexe : le rapport du temps total de calcul (phase I et phase II) varie entre 1 et 3.5. La rapidité est encore accrue pour la phase I dont le rapport varie entre 3 et 8.

Ces résultats sont encourageants, ils prouvent l'utilité et la performance de l'algorithme SGGP.

Une étude comparative complète de ces algorithmes est en cours, dans laquelle le procédé de la "forme produit de l'inverse" sera utilisé pour optimiser les calculs dans les résolutions des systèmes linéaires.

2. Notations. Définitions et Propriétés fondamentales de la programmation linéaire.

Ce paragraphe contient des notations élémentaires et des résultats fondamentaux de la programmation linéaire qui sont nécessaires à notre travail. Soit A une matrice de type mxn. Soient I et J deux parties de {1,...,m} et de {1,...,n} respectivement. On note A_I la matrice formée par les lignes $A_{ra(i)}$, i=1,..., |I| (|I| désigne le cardinal de I) rangées dans cet ordre, ra étant l'application injective de {1,...,|I|} dans {1,...,m} qui correspond d'une façon biunivoque à I : I={ra(1),...,ra(|I|)}. De même A^J désigne la matrice formée par les colonnes $A^{ra(j)}$, j=1,..., |J|, rangées dans cet ordre. Le complément de I dans {1,...,m} est noté I*.

Considérons maintenant le polyèdre K défini par : $K=\{x\in \mathbb{R}^n : Ax=b, Bx\leq a\}$ où A est une matrice de type mxn, de rang m et B est une matrice de type pxn. On dit que x est un sommet de K si x est la solution d'un système linéaire :

$$Ax = b \quad (1)$$
$$B_I x = a_I \quad (2)$$

telle que

$$B_{I*}x \leq a_{I*} \quad (3)$$

où I est un sous ensemble de {1,...,p}, |I| =n-m, tel que les lignes de A et celles de B_I sont linéairement indépendantes. L'ensemble des sommets de K sera noté V(K). Il s'ensuit que la non vacuité de V(K) implique que la matrice $\begin{bmatrix} A \\ B \end{bmatrix}$ (formée par les lignes de A et B) est de rang n. Une contrainte $B_i x \leq a_i$ est dite active en $x \in K$ si $B_i x = a_i$. Pour tout $x \in K$, on note $\mathcal{K}(x)=\{i=1,...,p : B_i x = a_i\}$.

Un sommet x de K est dit non dégénéré si $|\mathcal{K}(x)| = n-m$, dans le cas contraire x est dit dégénéré. Un sommet x de K est non dégénéré si et seulement si il existe une partie I unique telle que x est la solution de (1), (2), (3). Deux sommets x et x' de K sont dits adjacents s'ils sont les solutions de (1) et (2) avec I et I' tels que $|I \cap I'| = n-(m+1)$.

Conditions d'optimalité d'un programme linéaire général sous forme canonique.

Considérons maintenant le programme linéaire général (P) sous sa forme canonique

$$(P) \quad \begin{cases} Ax = b \\ Bx \leq a \\ cx = z(max) \end{cases}$$

Nous rappelons le résultat bien connu suivant **[Luenberger, Rockafellar, Sakarovitch].**

Théorème 2 [Pham Dinh Tao].
1) $x \in K$ est une solution de (P) si et seulement si il existe un m-vecteur ligne λ et un p-vecteur ligne $\mu \geq 0$ tels que:

$$c = \lambda A + \mu B \qquad (4)$$
$$\mu(Bx-a) = 0 \qquad (5)$$

2) En particulier soit $x \in V(K)$ défini par (1), (2) et (3) et soit (λ, μ^I) la solution du système linéaire:

$$(\lambda, \mu^I)\begin{bmatrix} A \\ B_I \end{bmatrix} = c \qquad (6)$$

Alors x est une solution de (P) si $\mu^I \geq 0$. Cette condition suffisante est aussi nécessaire si $x \in V(K)$ est non dégénéré. •

3. Algorithme pour la résolution du problème (P).

Nous commençons par indiquer comment calculer un sommet u de K adjacent à un sommet v donné.

3.1. Détermination d'un sommet adjacent

Le processus est très simple: Soit $v \in V(K)$ déterminé par (1), (2) et (3). Soit $i \in \{1,...,|I|\}$ fixé, on pose $I(i) = I\setminus\{ra(i)\}$. Considérons le système linéaire suivant:

$$Ax = 0 \qquad (7)$$
$$B_{I(i)}x = 0 \qquad (8)$$
$$B_{ra(i)}x = -1 \qquad (9)$$

qui admet une solution unique $d^{(i)}$.
Soit $v^{(i)} = v + \lambda_i d^{(i)}$. On a $Av^{(i)} = b$, $B_{I(i)}v^{(i)} = a_{I(i)}$ et $B_{ra(i)}v^{(i)} = a_{ra(i)} - \lambda_i$. Par suite $B_{ra(i)}v^{(i)} \leq a_{ra(i)}$ si et seulement si $\lambda_i \geq 0$.
Voyons les autres conditions portant sur λ_i pour assurer que $v^{(i)} \in K$.
Il est clair que $v^{(i)} \in K$ si et seulement si $\lambda_i \geq 0$ vérifie: $B_{I*}v^{(i)} = B_{I*}v + \lambda_i B_{I*}d^{(i)} \leq a_{I*}$ Puisque $B_{I*}v \leq a_{I*}$ on a les propriétés suivantes:
i) Si $B_{I*}d^{(i)} \leq 0$ alors $v^{(i)} = v + \lambda_i d^{(i)} \in K$ pour tout $\lambda_i \geq 0$. Dans ce cas il n'y a pas de sommet adjacent à v dans la direction $d^{(i)}$: $\{v+\lambda_i d^{(i)} : \lambda_i \geq 0\}$ est un rayon extrémal de K issu de v.
ii) Sinon l'ensemble $C(I,i)=\{k \in I^*: B_k d^{(i)} > 0\}$ est non vide et alors $v^{(i)}=v+\lambda_i d^{(i)} \in K$ si et seulement si:

$$\lambda_i \leq \min\left\{ \frac{a_k - B_k v}{B_k d^{(i)}} : k \in C(I,i) \right\} = \lambda_i^* \quad (10)$$

Dans ce cas $v^{(i)}=v+\lambda^*_i d^{(i)}$ est un sommet de K adjacent à v. En effet pour tout $k \in C(I,i)$, les lignes de A et celles de $B_{I(i)}$ sont linéairement indépendantes. Par suite si $k \in C'(I,i)=\{k \in C(I,i) : (a_k - B_k v)/B_k d^{(i)} = \lambda^*_i\}$ alors:

$$B_J v^{(i)} = a_J$$
et $$B_{J^*} v^{(i)} \leq a_{J^*}$$
où $J = I(i) \cup \{k\}$, $ra(i) = k$.

Autrement dit $v^{(i)} \in V(K)$ qui est défini par (1), (2) et (3) dans lesquels I est remplacé par J. De plus si $\lambda^*_i > 0$ alors $v^{(i)}$ est différent de v et on a : $\mathcal{K}(v^{(i)}) = I(i) \cup C'(I,i)$. Il s'ensuit qu'au cas où $\lambda^*_i > 0$, $v^{(i)}$ est un sommet non dégénéré si et seulement si $|C'(I,i)|=1$.

Remarquons enfin que si v est non dégénéré alors $\lambda^*_i>0$ et que dans ce cas il y a exactement |n-m| sommets adjacents à v que l'on peut tous déterminer.

3.2. Description de la phase II de l'algorithme SGGP pour la résolution de (P).

La phase II de SGGP suppose connu un sommet de K qui sera fourni par la phase I présentée dans §5. Sa description est très simple:

Il suffit d'indiquer, lorsqu'un sommet courant v de K n'est pas une solution de (P) (ou plutôt ne vérifie pas la condition suffisante 2) du Théorème 2.), comment passer à un sommet adjacent afin d'augmenter la valeur de la fonction objective cx. Soit $v\in V(K)$ défini par (1), (2) et (3).

On résoud (6) pour obtenir μ^I. Si $\mu^I\geq 0$ alors v est une solution de (P). Sinon soit $i\in\{1,\ldots,|I|\}$ tel que $\mu^{ra(i)}<0$ et soit $d^{(i)}$ la solution du système linéaire (7),(8) et (9). Tenant compte de (6), (7), (8) et (9) on a:

$$cd^{(i)} = (\lambda A + \mu^I B_I)d^{(i)} = \mu^{ra(i)} B_{ra(i)}d^{(i)} = -\mu^{ra(i)}.$$

Par suite le déplacement de v à $v+\lambda_i d^{(i)}$ avec $\lambda_i\geq 0$ augmente la valeur de cx de $-\lambda_i \mu^{ra(i)}$. Comme dans §3.1. on va distinguer deux cas :

i) Si $B_I\cdot d^{(i)} \leq 0$ alors $v+\lambda_i d^{(i)} \in V(K)$ pour tout $\lambda_i\geq 0$. Dans ce cas on a:

$$\lim_{\lambda_i\to+\infty} \{cx : x = v + \lambda_i d^{(i)}\} = +\infty$$

et le problème (P) n'admet pas de solution finie.

ii) Sinon $C(I,i) = \{k\in\{1,..n\} : B_k d^{(i)} >0\} \neq \emptyset$ et $v^{(i)}=v+\lambda^*_i d^{(i)}\in K$ avec λ^*_i défini par (10), est un sommet de K adjacent à v qui sera le prochain sommet courant dans l'algorithme SGGP.

La convergence de l'algorithme SGGP résulte de la stricte augmentation de la fonction objective (au moins en l'absence de la dégénérescence) et de la finitude de V(K).

Théorème 3 [Pham Dinh Tao, 1989].

Si aucun sommet généré par cet algorithme n'est dégénéré alors après un nombre fini d'itérations il permet d'obtenir une solution de (P) ou de déclarer que (P) n'a pas de solution.•

4. Relation avec l'algorithme du simplexe et l'algorithme dual du simplexe.

4.1. Lien avec l'algorithme du simplexe.

Considérons maintenant le programme linéaire (P') sous forme standard

$$(P') \begin{cases} Ax = b\ ,\ x \geq 0 \\ cx = z(max) \end{cases}$$

où A est une matrice de type mxn de rang m. (P') est un cas particulier de (P) où p=n, a=0 et $B=-U_n$, (U_n étant la matrice identité d'ordre n).

On va montrer que l'application de l'algorithme SGGP au problème (P') permet de retrouver l'algorithme du simplexe. Soit $K'=\{x \in \mathbb{R}^n : Ax=b,\ x \geq 0\}$. Voyons tout d'abord comment introduire les bases réalisables de (P') et caractériser les sommets de K' à l'aide de ces bases.
Comme dans §2., $x \in V(K')$ si et seulement si x est la solution de (1), (2) et (3) avec ici $B=-U_n$.
On a donc:

$$x_I = 0 \qquad (11)$$

Et le système linéaire (1) devient :

$$Ax = A^I x_I + A^{I*} x_{I*} = b$$

$$A^{I*} x_{I*} = b \qquad (12)$$

On va maintenant montrer le résultat suivant:

Lemme

Soit A une matrice de type mxn, de rang m, et $U=U_n$ la matrice identité d'ordre n. Soit I un sous ensemble de {1,...,n} tel que |I|=n-m. Une condition nécessaire et suffisante pour que la matrice $\begin{bmatrix} A \\ U_I \end{bmatrix}$ soit non singulière est que la sous matrice A^{I*} le soit. •

Preuve

Considérons la relation suivante: $\lambda A + \mu U_I = 0 \qquad (13)$
avec λ un m-vecteur ligne et un n-m vecteur ligne. Puisque $(U_I)^{I*}=0$, (13) est équivalent à $\lambda A^{I*} = 0$ et $\lambda A^I + \mu (U_I)^I = \lambda A^I + \mu = 0 \qquad (14)$
Si A^{I*} est non singulière alors $\lambda A^{I*} = 0$ implique $\lambda=0$ et par suite $\mu=0$; autrement dit la non singularité de la matrice $\begin{bmatrix} A \\ U_I \end{bmatrix}$.

Inversement supposons que cette dernière matrice est non singulière. Si A^{I^*} est singulière alors il existe un m-vecteur $\lambda \neq 0$ tel que $\lambda A^{I^*}=0$. Puisque l'équation $\lambda A^I + \mu = 0$ admet une solution (unique) en μ, on obtient (14) avec (λ,μ) non tous nuls, contradiction avec l'hypothèse.

On retrouve ainsi le théorème bien connu (sur la caractérisation des sommets de K') de la programmation linéaire qui est la base de l'algorithme du simplexe:

Théorème 4.1 **[Pham Dinh Tao, 1989].**

Soit $K'=\{x \in \mathbb{R}^n : Ax=b, x \geq 0\}$ avec A une matrice de type mxn, de rang m. Alors $x \in V(K')$ si et seulement si il existe $J \subset \{1,...,n\}$; $|J|=m$, tel que:
i) A^J est non singulière et $(A^J)^{-1}b \geq 0$
ii) $x_J = (A^J)^{-1}b$ et $x_{J^*} = 0$. •

Une telle partie J est usuellement appelée, en programmation linéaire, base réalisable de (P') et le sommet x défini par i) et ii) solution réalisable de base J. On remarque que J et J* ne sont autres que I* et I respectivement dans (11) et (12). Dès lors l'équation (6) devient:

$$\begin{cases} c^I = \lambda A^I - \mu^I \\ c^{I^*} = \lambda A^{I^*} \end{cases}$$

car $(\mu^I B_I)^I = -\mu^{I^*}$ et $(\mu^I B_I)^{I^*} = 0$. On en déduit que $-\mu^{J^*}=\mu^I = c^{J^*}-c^J(A^J)^{-1}A^{J^*}$. Tenant compte du Théorème 4.1., l'application du Théorème 2. nous donne le résultat classique suivant:

Théorème 4.2 **[Pham Dinh Tao, 1989].**

Soit $x \in V(K')$ défini par i) et ii) du Théorème 4.1. Si $c^{J^*}-c^J(A^J)^{-1}A^{J^*} \leq 0$ alors x est une solution de (P'). Cette condition suffisante est aussi nécessaire si x est un sommet non dégénéré de K'.•

Nous allons maintenant décrire un pas de l'algorithme SGGP appliqué au problème (P') à partir d'un sommet x de K' défini par (11) et (12) ou d'une façon équivalente par i) et ii) du Théorème 4.1. (J=I*, J*=I) (Cf. §3.2). Si $c^{J^*}-c^J(A^J)^{-1}A^{J^*} \leq 0$ alors x est une solution de (P'). Sinon soit $s \in J^*$ tel que $c^s-c^J(A^J)^{-1}A^s>0$ et soit $d^{(s)}$ la solution du système linéaire

$$Ad = 0 \qquad (13)$$
$$U_{J^*\setminus\{s\}}d = 0 \qquad (14)$$
$$d_s = 1 \qquad (15)$$

Ce qui implique que

$$A^J d_J = -A^s \qquad (16)$$

et on se déplace de x dans la direction d^s. Autrement dit on considère $x^{(s)}=x+\lambda_s d^{(s)}$, $\lambda_s \geq 0$, qui est défini par:

$$x^{(s)}_{J^*\setminus\{s\}} = 0,\ x_s^{(s)} = \lambda_s \qquad (17)$$

$$x^{(s)}_J = (A^J)^{-1}b - \lambda_s (A^J)^{-1}A^s \qquad (18)$$

Procédons comme dans §3.2 avec $I=J^*$, $J=I^*$ et $B=-U_n$, on a:

$B_I.d^{(s)} = -U_I.d = -d_{I^*} = -d_J = (A^J)^{-1}A^s$. Par suite deux cas sont à distinguer:

i) Si $(A^J)^{-1}A^s \leq 0$ alors $x^{(s)} = x + \lambda_s d^{(s)} \in K'$ pour tout $\lambda_s \geq 0$.
Dans ce cas $\{x+\lambda_s d^{(s)} : \lambda_s \geq 0\}$ est un rayon extrémal de K' issu de x sur lequel la fonction objective cx augmente indéfiniment avec λ_s : le problème (P') n'admet pas de solution finie.

ii) Sinon $C(J,s) = \{k \in J^* : ((A^J)^{-1}A^s)_k > 0\} \neq \emptyset$ et $x^{(s)}=x+\lambda^*_s d^{(s)}$ avec $\lambda^*_s = \min\{((A^J)^{-1}b)_k / ((A^J)^{-1}A^s)_k : k \in C(J,s)\}$, est le prochain sommet courant. On a alors en notant ra l'application de {1,...,m} dans {1,...,n} qui définit le sous ensemble J, ($J=\{ra(1),...,ra(m)\}$) :
1) Pour tout $k \in C'(J,s) = \{k \in C(J,s) : ((A^J)^{-1}b)_k / ((A^J)^{-1}A^s)_k = \lambda^*_s\}$; $J'=\{J \cup \{s\}\}\setminus ra(k)$ est une base réalisable de (P') dont la solution de base est $x^{(s)}=x+\lambda^*_s d^{(s)}$.
2) $\lambda^*_s > 0$ si x est non dégénéré.
3) Si $\lambda^*_s > 0$ alors $x^{(s)}$ est différent de x. Dans ce cas $x^{(s)}=x+\lambda^*_s d^{(s)}$ est non dégénéré si et seulement si $|C'(J,s)|=1$.
4) En se déplaçant de x à $x^{(s)}=x+\lambda^*_s d^{(s)}$ la fonction objective cx augmente de $(c^s - c^J(A^J)^{-1}A^s)\lambda^*_s$. On peut donc conclure que l'algorithme du simplexe peut être considéré comme un cas particulier de l'algorithme SGGP qui est appliqué au problème linéaire (P').

Remarques importantes

Les systèmes linéaires intervenant dans le calcul de la direction de déplacement (13), (14), (15), le calcul du pas de déplacement et la vérification des conditions d'optimalité deviennent plus simples dans la programmation linéaire sous forme standard. C'est sous cette forme simplifiée de SGGP que l'on retrouve l'algorithme du simplexe.

Remarquer que cette simplification est dûe à la technique de décomposition des systèmes linéaires précédents. En effet, soit le système linéaire Mu = g (M étant une matrice carrée d'ordre n), et considérons la décomposition suivante:

$$M = \begin{array}{|c|c|}\hline M_1 & M_2 \\ \hline M_3 & M_4 \\ \hline \end{array} \quad , \quad u = \begin{array}{|c|}\hline u_1 \\ \hline u_2 \\ \hline \end{array} \quad , g = \begin{array}{|c|}\hline g_1 \\ \hline g_2 \\ \hline \end{array}$$

(M_1 étant une matrice carrée d'ordre n_1, M_2 une matrice carrée d'ordre n_2, $n_1+n_2=n$; u_1, g_1 sont n_1-vecteurs colonnes et u_2, g_2 des n_2-vecteurs colonnes), dans laquelle les systèmes linéaires $Mu = g$ s'écrit:

$$M_1 u_1 + M_2 u_2 = g_1$$
$$M_3 u_1 + M_4 u_2 = g_2$$

Si M_2 ou M_3 est nulle alors la résolution de $Mu = g$ se décompose en double résolutions de deux systèmes linéaires de dimension n_1 et n_2 (réduction dimensionnelle). Par exemple si $M_2 = 0$ alors on résoud en premier lieu le système $M_1 u_1 = g_1$ et puis le système $M_4 u_2 = g_2 - M_3 u_1$ dans lequel on remplace u_1 par sa valeur obtenue.

C'est la structure creuse de la matrice des contraintes de non négativité de la variable x dans le programme linéaire sous forme standard, qui permet l'utilisation de cette technique de décomposition. En effet les système linéaires formulés dans l'algorithme SGGP sont de ces formes:

$$Mu = g \quad \text{ou} \quad M^T u = g$$

où $M = \begin{array}{|c|}\hline A \\ \hline -U_{J*} \\ \hline \end{array}$

La décomposition "optimale" utilisée dans l'algorithme du simplexe est:

$$M_1 = A^J \ , \ M_2 = A^{J*} \ , \ M_3 = -U^J_{J*} \ , \ M_4 = -U^{J*}_{J*}$$

Vu que $M_2 = 0$ et $- M_4$ est la matrice identité, la résolution du système $Mu = g$ (on procède de façon analogue avec le système $M^T u = g$) se ramène à celle du système linéaire réduit suivant:

$$A^J u_J = g_J + A^{J*} g_{J*}$$

Cette technique de décomposition sera d'une utilisation constante dans l'implémentation de différentes versions de l'algorithme SGGP (notamment dans la phase I de SGGP appliqué aux programmes linéaires sous forme canonique et standard, cf. §6.3 et §6.4).

4.2. Lien avec l'algorithme dual du simplexe.

Considérons un programme linéaire sous forme standard:

$$(P') \begin{cases} Ax = b\ ,\ x \geq 0 \\ cx = z(max) \end{cases}$$

On suppose que le rang de la matrice A, de type mxn, est égal à m et que le problème (P) est dual réalisable, c'est à dire que l'on dispose d'un ensemble $J \subset \{1,...,n\}$, $|J|=m$ tel que:

i) A^J est non singulière.

ii) $c^{J*} - \pi A^{J*} \leq 0$ où $\pi = c^J (A^J)^{-1}$

L'ensemble J est une base du système linéaire Ax=b qui n'est pas nécessairement réalisable pour (P). Le problème dual (D) s'écrit :

$$(D) \begin{cases} yA \geq c \\ yb = w(min) \end{cases}$$

Il est clair que π est une solution réalisable de (D), et plus précisément un sommet du polyèdre convexe des solutions réalisables de (D). L'algorithme SGGP décrit dans §3 travaille avec les variables vecteurs colonne, pour y être conforme et pour ne pas compliquer les raisonnements, on va associer à (D) le problème équivalent suivant:

$$(D^T) \begin{cases} Bu \leq a \\ du = \xi(max) \end{cases}$$

où $B = A^T$, $a = -c^T$, $d = b^T$. Les variables y et u sont liées par $u = -y^T$.

Le problème (D^T) est un programme linéaire général dont on connaît un sommet du polyèdre des solutions réalisables qui est défini par:

$$B_J u = a_J \quad (19)$$

$$B_{J*} u \leq a_{J*} \quad (20)$$

Décrivons alors un pas de l'algorithme SGGP appliqué au problème (D^T) à partir du sommet défini par (19) et (20). Si $d(B_J)^{-1} = ((A^J)^{-1}b)^T \geq 0$ alors $u = (B_J)^{-1} a_J$ est un sommet solution de (D^T) et $x = (x_J = (A^J)^{-1}b,\ x_{J*} = 0)$ un sommet solution de (P) en vertu du Théorème 4.1.

Dans le cas contraire l'ensemble $\{i=1,...,m : (d(B_J)^{-1})_i = ((A^J)^{-1}b))_i < 0\}$ est non vide. On choisit un élément r de cet ensemble et on calcule la direction de déplacement $d^{(r)}$, solution du système: $(J = \{ra(1),...,ra(m)\})$

$$B_{J\setminus\{ra(r)\}} d = 0 \quad (21)$$

$$B_{ra(r)} d = -1 \quad (22)$$

qui est équivalent à

$$B_J d = -e_r \quad (23)$$

(e_k désigne le $k^{ème}$ vecteur de la base canonique de $\mathbb{R}^m$).

(Remarquer que $B_{J*} = (A^{J*})^T$, $(B_J)^{-1} = ((A^J)^{-1})^T$. Si $B_{J*}\, d^{(r)} = -B_{J*}(B_J)^{-1}e_r = -[e_r^T(A^J)^{-1}A^{J*}]^T \leq 0$ (i.e. $((A^J)^{-1}A^j)_r \geq 0$, pour tout $j \in J^*$) alors il n'y a pas de sommet (du polyèdre de solutions réalisables de (D^T)) adjacent à u dans cette direction $d^{(r)}$:
$\{u+\lambda d^{(r)} : \lambda \geq 0\}$ est un rayon extrémal dudit polyèdre le long duquel la fonction objective de (D^T) augmente indéfiniment : (D^T) n'admet pas de solution finie, il en est de même pour (D) et par suite le problème (P) n'admet pas de solution réalisable.
On peut aussi prouver cela en observant que la $r^{ème}$ équation de

$$x_J + (A^J)^{-1}A^{J*}\, x_{J*} = (A^J)^{-1}b$$

n'est pas vérifiée pour $x=(x_J,x_{J*}) \geq 0$ si $((A^J)^{-1}b)_r < 0$ et $((A^J)^{-1}A^j)_r \geq 0$ pour tout $j \in J^*$. Dans le cas contraire, soit $s \in J^*$, $((A^J)^{-1}A^s)_r < 0$ tel que:

$$\frac{a_s - B_s(B_J)^{-1}a_J}{-B_s(B_J)^{-1}e_r} = \text{Min}\left\{ \frac{a_k - B_k(B_J)^{-1}a_J}{-B_k(B_J)^{-1}e_r} : k \in J^*,\ ((A^J)^{-1}A^k)_r < 0 \right\}$$

or d'une façon équivalente:

$$\frac{c^s - c^J(A^J)^{-1}A^s}{((A^J)^{-1}A^s)_r} = \text{Min}\left\{ \frac{c^k - c^J(A^J)^{-1}A^k}{((A^J)^{-1}A^k)_r} : k \in J^*,\ ((A^J)^{-1}A^k)_r < 0 \right\}$$

Et on définit $J'=J\cup\{s\}\setminus\{ra(r)\}$ qui en remplaçant J dans les équations (19) et (20), détermine le sommet courant suivant dans l'algorithme SGGP appliqué au problème (D^T).
Finalement en revenant au problème (D) avec les variables y (qui sont des vecteurs lignes) la description précédente représente exactement l'algorithme dual du simplexe pour la résolution des problèmes (P) et (D) lorsque (P) est dual réalisable.
Remarquer qu'il est erroné de dire que l'algorithme dual du simplexe (pour le programme linéaire sous forme standard (P')) est l'algorithme du simplexe appliqué au problème dual (D) car (D) n'est pas sous forme standard : contrainte inégalité $yA \geq c$ au lieu d'égalité et la variable y est arbitraire.

4.3. Programme linéaire avec contraintes de bornes.

Ce sont des problèmes de ce type

$$(P)\begin{cases} Ax = b \\ a_I \leq x_I \leq b_I \\ cx = z(\max) \end{cases}$$

où I est une partie {1,...,n}. Les constantes a_i peuvent être égales à $-\infty$ et les constantes b_i peuvent être égales à $+\infty$.Des méthodes issues de l'algorithme du simplexe sont adaptées à la résolution de ce type de problème. Mais très souvent leur présentation (plus compliquées que l'algorithme du simplexe et l'algorithme dual du simplexe) ne sont pas assez naturelles pour faciliter leur compréhension en profondeur.

Le problème (P) est un cas particulier du programme linéaire général auquel l'application de l'algorithme SGGP permet de retrouver les algorithmes connus **[Luenberger, Sakarovitch]** et surtout de remédier aux inconvénients cités ci-dessus.

4.4. Lien avec la méthode du gradient projeté.

Donnons tout d'abord une description succinte de la méthode du gradient projeté dû à Rosen **[Luenberger]** pour la résolution du problème d'optimisation suivant:

$$(Q)\begin{cases} \text{Min } f(x) \\ Ax = b \\ Bx \le a \end{cases}$$

où A et B sont les matrices vérifiant les mêmes hypothèses que dans §2. et f(x) est une fonction différentiable.

Cette méthode consiste en la construction d'une suite (x^k) de solutions admissibles de la manière suivante: On suppose la régularité des contraintes en tout point x^k, i.e. les lignes de A et les lignes de $B_{\mathcal{K}(x^k)}$ sont linéairement indépendantes. Le passage de x^k à x^{k+1} est défini par :

i) La direction de déplacement x^k qui est la projection orthogonale de $-\nabla f(x^k)$ sur le sous espace tangent $V_k = \{d : Ad=0, B_{\mathcal{K}(x^k)}d=0\}$ de dimension $n-(m+|\mathcal{K}(x^k)|)$.

ii) Si $d^k \neq 0$, on calcule α^k et β^k solutions respectivement de

$$\text{Max}\{\alpha : x^k+\alpha d^k \in K\},\ K=\{x\in \mathbb{R}^n : Ax=b,\ Bx\le a\}$$
$$\text{Min}\{f(x^k+\beta d^k) : 0\le\beta\le\alpha^k\}$$

On pose $x^{k+1}= x^k+\beta^k d^k$ et réitérer le processus.

iii) Si $d^k=0$ alors le système $\begin{bmatrix} A \\ B_{\mathcal{K}(x^k)} \end{bmatrix}^T \gamma = M_k^T \gamma = -\nabla f(x^k)$ admet une solution unique, plus précisement : $\gamma=-(M_k M_k^T)^{-1} M_k \nabla f(x^k)$.

a) Si $\gamma_j \geq 0$ pour tout j correspondant aux contraintes définies par $B_{\mathcal{K}(x^k)}$ alors x^k vérifie les conditions de Kuhn-Tucker relatives au problème (Q) et par suite est une solution de (Q) si f est convexe.

b) Sinon on choisit un $\gamma_k < 0$ qui correspond à une contrainte active en définie par une ligne de $B_{\mathcal{K}(x^k)}$. Soit $\mathcal{K}'(x^k) = \mathcal{K}(x^k) \backslash \{\text{indice de cette ligne}\}$. Retourner à i) avec $\mathcal{K}'(x^k)$ à la place de $\mathcal{K}(x^k)$. Soit maintenant $f(x)=cx$ et appliquons la méthode du gradient projeté avec un vecteur initial $x^0 \in V(K)$, il est clair que $V_0=\{0\}$ et par suite $d^0=0$. L'étape iii) correspond exactement à la résolution du système linéaire (6) car M_k est une matrice carrée non singulière. Dans les itérations suivantes le calcul de d^k, $k \geq 1$, est immédiat car dim $V_k = 1$, V_k est exactement une droite parallèle à une arête de K issue de x^k. De plus il est clair que $\alpha^k = \beta^k$. Finalement si $f(x)=cx$ et si l'on part d'un $x^0 \in V(K)$, la méthode du gradient projeté donne exactement l'algorithme SGGP.

5. Initialisation de l'algorithme SGGP (Phase I) Détermination d'un sommet de K par l'algorithme SGGP.

Ce chapitre est consacré à l'initialisation de l'algorithme SGGP pour la résolution du programme linéaire général (cf. §3.)

$$(P) \begin{cases} Ax = b \\ Bx \leq a \\ cx = z(\max) \end{cases}$$

i.e., a la détermination d'un sommet de $K=\{x \in \mathbb{R}^n : Ax=b, Bx \leq a\}$, A étant une matrice de type mxn et B une matrice de type pxn. Et comme dans l'algorithme du simplexe, la phase I de SGGP n'est autre que sa phase II appliquée à des programmes linéaires convenablement définis.

Nous allons présenter ci-dessous deux nouveaux procédés d'initialisation. On peut supposer, sans perte de généralité, que le rang de A est égal à m. Si tel n'est pas le cas, il suffirait d'appliquer la méthode d'élimination de Gauss au système Ax=b pour enlever les lignes redondantes lorsque les équations du système Ax=b sont compatibles ou arriver à la conclusion de la vacuité de K dans le cas contraire.

5. Initialisation de l'algorithme SGGP (Phase I) Détermination d'un sommet de K par l'algorithme SGGP.

Ce chapitre est consacré à l'initialisation de l'algorithme SGGP pour la résolution du programme linéaire général (cf. §3.)

$$(P)\begin{cases} Ax = b \\ Bx \leq a \\ cx = z(max) \end{cases}$$

i.e., a la détermination d'un sommet de $K=\{x\in \mathbb{R}^n : Ax=b, Bx\leq a\}$, A étant une matrice de type mxn et B une matrice de type pxn. Et comme dans l'algorithme du simplexe, la phase I de SGGP n'est autre que sa phase II appliquée à des programmes linéaires convenablement définis.

Nous allons présenter ci-dessous deux nouveaux procédés d'initialisation. On peut supposer, sans perte de généralité, que le rang de A est égal à m. Si tel n'est pas le cas, il suffirait d'appliquer la méthode d'élimination de Gauss au système Ax=b pour enlever les lignes redondantes lorsque les équations du système Ax=b sont compatibles ou arriver à la conclusion de la vacuité de K dans le cas contraire.

5.1. Procédé primal

On recherche un ensemble $I_0=\{ra(1),...,ra(|I_0|)\}$ de $\{1,...,p\}$ tel que $|I_0|=n-m$ et que le rang de $\begin{bmatrix} A \\ B_{I_0} \end{bmatrix}$ soit égal à n (Cf. §3.1).

On pourrait, pour cela, utiliser la méthode d'élimination de Gauss opérant sur les lignes de A et de B. Si un tel ensemble I_0 n'existe pas, $V(K)=\emptyset$.

Le principe de ce procédé est très simple: On résoud le système linéaire

$$\begin{bmatrix} A \\ B_{I_0} \end{bmatrix} x = \begin{bmatrix} b \\ a_{I_0} \end{bmatrix}$$

pour obtenir la solution $x^{(0)}$ qui est un sommet du polyèdre $K(I_0)=\{x : Ax=b, B_{I_0}x \leq a_{I_0}, B_{\mathfrak{I}(x^{(0)})}x \leq a_{\mathfrak{I}(x^{(0)})}\}$ où:

$$\mathfrak{I}(x^{(0)}) = \{i\in I_0^* : B_i x^{(0)} \leq a_i\}$$

Si $\mathfrak{I}^*(x^{(0)}) = I_0^* \setminus \mathfrak{I}(x^{(0)}) = \emptyset$ alors $x^{(0)}\in V(K)$.

Dans le cas contraire on va construire une suite finie de programmes linéaires auxquels la phase II de l'algorithme SGGP peut être applicable. On obtiendra ainsi une suite finie (x^k),(de sommets solutions desdits programmes linéaires), le passage de x^k à x^{k+1} augmente d'au moins une unité le nombre des contraintes de K qui sont vérifiées. Ainsi après la

résolution d'un nombre fini de programmes linéaires, ce procédé permet d'obtenir un sommet de K ou de conclure que K est vide.

Voici donc la description du procédé primal:

0) On résoud le système linéaire

$$\begin{bmatrix} A \\ B_{I_0} \end{bmatrix} x = \begin{bmatrix} b \\ a_{I_0} \end{bmatrix}$$

pour obtenir la solution $x^{(0)}$.

1) Si $\mathfrak{J}^*(x^{(0)}) = \{i \in I_0^* : B_i x^{(0)} > a_i\} = \emptyset$ alors $x^{(0)} \in V(K)$.

2) Sinon on choisit $i_0 \in \mathfrak{J}^*(x^{(0)})$, par exemple i_0 défini par:

$B_{i_0}x^{(0)} - a_{i_0} = \text{Max}\{B_i x^{(0)} - a_i : i \in \mathfrak{J}^*(x^{(0)})\}$. Soit:

$$(P(I_0)) \begin{cases} Ax = b \\ B_{I_0} x \leq a_{I_0} \\ B_{\mathfrak{J}(x^{(0)})} x \leq a_{\mathfrak{J}(x^{(0)})} \\ -B_{i_0} x = z(\max) \end{cases}$$

On a $x^{(0)} \in V(K(I_0))$ par suite on peut appliquer la phase II de SGGP à la résolution de $(P(I_0))$. On résoud le système linéaire

$$(\lambda, \mu^{I_0}) \begin{bmatrix} A \\ B_{I_0} \end{bmatrix} = -B_{i_0}$$

2.1) Si $\mu^{I_0} \geq 0$ alors $x^{(0)}$ est une solution de $(P(I_0))$ (Cf. §2.). Dans ce cas on a

$$-B_{i_0} x < -a_{i_0} \quad , \quad \forall\, x \in K(I_0)$$

Autrement dit $K = \emptyset$ car $K \subset K(I_0)$.

2.2) Sinon on choisit $j_0 \in \{1,\ldots,|I_0|\}$ tel que $\mu^{ra(j_0)} < 0$, par exemple :

$$\mu^{ra(j_0)} = \min\{\mu^{ra(i)} : i \in \{1,\ldots,|I_0|\}\}$$

On calcule $d^{(j_0)}$ solution du système linéaire (Cf. §3.2)

$$\begin{cases} Ad = 0 \\ B_{I_0(j_0)} d = 0 \\ B_{ra(j_0)} d = -1 \end{cases}$$

2.2.1) Si $\mathfrak{J}(x^{(0)})$ est vide ou si $B_{\mathfrak{J}(x^{(0)})} d^{(j_0)} \leq 0$ alors $x^{(0)} + \lambda d^{(j_0)} \in K(I_0)$ pour tout $\lambda \geq 0$: l'ensemble $\{ x^{(0)} + \lambda d^{(j_0)} : \lambda \geq 0 \}$ est un rayon extrémal de $K(I_0)$ le long duquel la fonction objective $-B_{i_0}x$ augmente indéfiniment.Dans ce cas on calcule λ_{j_0} tel que: $-B_{i_0}(x^{(0)} + \lambda_{j_0} d^{(j_0)}) = -a_{i_0}$

Soit $\lambda_{j_0} = \dfrac{a_{i_0} - B_{i_0}(x^{(0)})}{B_{i_0} d^{(j_0)}} > 0$ (24)

(Rappelons ici que $B_{i_0} d^{(j_0)} = \mu^{ra(j_0)} < 0$ (Cf. §3.2.) et $B_{i_0} x^{(0)} > a_{i_0}$).

On pose $I_1 = [I_0 \setminus \{ra(j_0)\}] \cup \{i_0\}$, $ra(j_0) = i_0$, $x^{(1)} = x^{(0)} + \lambda_{j_0} d^{(j_0)}$.

La contrainte $-B_{i_0} x \leq -a_{i_0}$ qui est violée par $x^{(0)}$ devient active en $x^{(1)}$.

On remplace I_0 par I_1 et $x^{(0)}$ par $x^{(1)}$, on retourne à l'étape 1).

2.2.2) Sinon, on calcule $\lambda^*_{j_0} = \text{Min}\{\dfrac{a_i - B_i x^{(0)}}{B_i d^{(j_0)}} : i \in \mathfrak{I}(x^{(0)}), B_i d^{(j_0)} > 0\}$

On choisit $i_1 \in \mathfrak{I}(x^{(0)})$ tel que $B_{i_1} d^{(j_0)} > 0$ et $\dfrac{a_{i_1} - B_{i_1}(x^{(0)})}{B_{i_1} d^{(j_0)}} = \lambda^*_{j_0}$.

On pose $I_1 = [I_0 \setminus \{ra(j_0)\}] \cup \{i_1\}$, $ra(j_0) = i_1$, $x^{(1)} = x^{(0)} + \lambda^*_{j_0} d^{(j_0)}$.

On remplace I_0 par I_1 et $x^{(0)}$ par $x^{(1)}$, on retourne à l'étape 1).

5.2. Procédé dual

Ce deuxième procédé est basé sur l'application de notre algorithme à un problème dual dont le primal est construit de façon que les conditions suivantes soient vérifiées:

i) Le problème primal est dual réalisable

ii) Tout sommet solution du problème primal est un sommet de K.

La définition de la duale-réalisabilité d'un programme linéaire général sera donnée dans §6.5.

Comme dans le procédé primal, on détermine une matrice B_I telle que le rang de $\begin{bmatrix} A \\ B_I \end{bmatrix}$ soit égal à n.

Soit $d = (\lambda, \mu^I)\begin{bmatrix} A \\ B_I \end{bmatrix} = \lambda A + \mu^I B_I$, où λ est un m-vecteur ligne quelconque et μ un p-vecteur ligne tel que $\mu^I > 0$ et $\mu^{I^*} = 0$.

Considérons les programmes linéaires auxiliaires duaux suivants:

$$(PA) \begin{cases} Ax = b \\ Bx \leq a \\ dx = z(\max) \end{cases} \qquad (DA) \begin{cases} yA + zB = d,\ z \geq 0 \\ yb + za = w(\min) \end{cases}$$

Il est clair que (λ,μ) est un sommet du polyèdre convexe des solutions réalisables de (DA). Autrement dit (PA) est dual réalisable et l'application de notre algorithme (phase II) au problème dual (DA) va fournir un sommet de K, solution de (PA) ou bien conduire à la conclusion de la vacuité de K (ce qui correspond à la non existence de solution de (DA)).

6. Applications

Ce travail est motivé par la résolution d'un certain nombre de problèmes dont les principaux sont présentés ci-dessous.

6.1. Programme linéaire général

La résolution de (P) défini dans §2.

$$(P)\quad \begin{cases} Ax = b \\ Bx \leq a \\ cx = z(max) \end{cases}$$

dans l'espace des variables x elles-mêmes (sans être obligé de se ramener dans $\mathbb{R}^n_+$ par translation ce qui n'est possible que si pour tout $i=1,\ldots,n$, $\text{Min}\{x_i : x\in K\}$ existe, ni de décomposer $x=x^+-x^-$, $x^+\geq 0$, $x^-\geq 0$ pour devoir travailler dans $\mathbb{R}^{2n}_+$, ce qui augmente maladroitement la complexité du problème, d'autant plus que l'application $x\rightarrow (x^+,x^-)$ n'échange pas les sommets entre eux) peut se faire grâce à l'algorithme SGGP dans les deux cas suivants:

a) On connaît un sommet x^0 de K.

b) On connaît un point x^0 de K, $x^0\notin V(K)$

Dans ce cas on peut démarrer avec la méthode du gradient projeté et obtenir :

- Soit une solution de (P) en x^k sans rencontrer auparavant aucun sommet de K, (i.e. $x^l\notin V(K)$, $\forall\ l \leq k-1$)
- Soit un sommet x^k de K et puis continuer la méthode du gradient projeté qui coïncide avec l'algorithme SGGP à partir de cet x^k.

Les situations décrites ci-dessus figurent dans les méthodes d'optimisation globales (pour la résolution des problèmes d'optimisation non convexe) utilisant le principe d'approximation extérieure et la technique des coupes planes **[Pham Dinh Tao, El Bernoussi Souad]**.

6.2. Détermination de l'ensemble des sommets $V(S^k)$ d'une suite emboîtée de polyèdres convexes bornés.

L'élaboration des algorithmes d'optimisation globale pose le problème de calcul de $V(S^k)$ où (S^k) est une suite emboîtée de polyèdres convexes bornés (non nécessairement contenus dans $\mathbb{R}^n_+$) qui représente les approximations extérieures : S^{k+1} est défini à partir de S^k en y ajoutant une contrainte linéaire supplémentaire : $S^{k+1} = \{x \in S^k : a_k x \leq \alpha_k\}$

La connaissance de $V(S^k)$ permet de déterminer $V(S^{k+1})$ de la manière suivante **[Pham Dinh Tao, El Bernoussi Souad]**: $v \in V(S^{k+1})$ si et seulement si

a) soit $v \in V(S^k)$ et $a_k v \leq \alpha_k$

b) soit $v \in [v_1, v_2] \cap H_k$ où $H_k = \{x : c_k x = a_k\}$, $v_1 \in V^+(S^k) = \{x \in V(S^k) : a_k x > \alpha_k\}$ et $v_2 \in V^-(S^k) = \{x \in V(S^k) : a_k x < \alpha_k\}$

La détermination d'un sommet $v \in V(S^{k+1})$ du cas b) passe par la détermination (au cas de non dégénérescence) de n sommets adjacents de v_1 (ou de v_2) déja calculé qui peut être faite par l'algorithme SGGP (voir §3.1.).

6.3. Initialisation de l'algorithme SGGP au cas où $K=\{x \in \mathbb{R}^n : Ax=b, x \geq 0\}$ (A étant une matrice de type mxn)

On va décrire ci-dessous les versions simplifiées de deux procédés présentés dans §5.

6.3.1. Procédé primal

En utilisant par exemple la méthode d'élimination de Gauss on peut facilement calculer le rang m de A, enlever les contraintes redondantes (lorsqu'il y a compatibilité des équations de Ax=b) et en extraire une sous matrice de type m'xm' non singulière. Le polyèdre convexe K serait vide si lesdites équations sont incompatibles.

Pour simplifier les notations, on suppose que le rang de A est égal à m et on dispose d'une sous matrice A^{J_0}, non singulière ($|J_0|=m$). Tenant compte des résultats de §4.1. ($B=-U_n$, $a=0$), $I_0 \subset \{1,\ldots,p\}$, $|I_0|=n-m$, tel que le rang de $\begin{bmatrix} A \\ B_{I_0} \end{bmatrix}$ soit égal à n revient à chercher $J_0=I_0^*$ tel que A^{J_0} soit non singulière.

La détermination de A^{J_0} pourrait être faite en utilisant la méthode d'élimination de Gauss. Le procédé primal pour la détermination d'un sommet de K peut être ainsi simplifié:

0) On résoud $A^{J_0} x_{J_0} = b$ pour obtenir $(x^{(0)} = x^{(0)}_{J_0} = (A^{J_0})^{-1}b,\ x_{J_0^*} = 0)$.

On note : $J_0=\{ra(1),...,ra(m)\}$; $J_0^*=\{ra^*(1),...,ra^*(n-m)\}$

$\mathcal{H}(x^{(0)})=\{i=1,...,m : ((A^{J_0})^{-1}b)_i \geq 0\}$ et $\mathcal{H}^*(x^{(0)})$ = le complémentaire de $\mathcal{H}(x^{(0)})$ dans $\{1,...,m\}$. On a : $J_0 \backslash ra(\mathcal{H}^*(x^{(0)})) = ra(\mathcal{H}(x^{(0)})) = \mathcal{J}(x^{(0)})$

1) Si $\mathcal{H}^*(x^{(0)})$ est vide alors $x^{(0)} \in V(K)$.

2) Sinon on choisit $i_0 \in \mathcal{H}^*(x^{(0)})$, par exemple:

$$((A^{J_0})^{-1}b)_{i_0} = Min\{((A^{J_0})^{-1}b)_i : i=1,...,m\}$$

et on résoud le problème $(P(J_0))$ par la phase II de l'algorithme SGGP:

$$(P(J_0)) \begin{cases} Ax = b \\ x_{J_0^*} \geq 0 \\ x_{J_0 \backslash ra(\mathcal{H}^*(x^{(0)}))} = x_{ra(\mathcal{H}(x^{(0)})} \geq 0 \\ x_{ra(i_0)} = z(max) \end{cases}$$

2.1) Si $((A^{J_0})^{-1}A^{*}_{J_0})_{i_0} \geq 0$ alors $x^{(0)}$ est une solution de $(P(J_0))$ et $K=\emptyset$.

2.2) Sinon, on choisit $j_0 \in \{1,...,n-m\}$ tel que $((A^{J_0})^{-1}A^{ra^*(j_0)})_{i_0} < 0$. Par exemple:

$$((A^{J_0})^{-1}A^{ra^*(j_0)})_{i_0} = Min\{((A^{J_0})^{-1}A^{ra^*(j)})_{i_0} : j=1,...,n-m\}.$$

2.2.1) Si $\mathcal{H}(x^{(0)})$ est vide ou $((A^{J_0})^{-1}A^{ra^*(j_0)})_{\mathcal{H}(x^{(0)})} \leq 0$ alors on calcule λ_{j_0} tel que:

$$\lambda_{j_0} = \frac{((A^{J_0})^{-1}b)_{i_0}}{((A^{J_0})^{-1}A^{ra^*(j_0)})_{i_0}} > 0 \qquad (25)$$

On pose $J_1=[J_0\backslash\{ra(i_0)\}]\cup\{ra^*(j_0)\}$; $ra(i_0)= ra^*(j_0)$. On calcule $x^{(1)}$ défini par :

$$\begin{cases} x^{(1)}_{J_0} = x^{(0)}_{J_0} - \lambda_{j_0}(A^{J_0})^{-1}A^{ra^*(j_0)} \\ x^{(1)}_{J_0^*\backslash\{ra^*(j_0)\}} = 0;\ x^{(1)}_{ra^*(j_0)} = \lambda_{j_0} \end{cases}$$

(On a : $x^{(1)}_{ra(i_0)} = x^{(0)}_{ra(i_0)} - \lambda_{j_0}((A^{J_0})^{-1}A^{ra^*(j_0)})_{i_0} = 0$)

On remplace I_0 par I_1 et $x^{(0)}$ par $x^{(1)}$, on retourne à l'étape 1).

2.2.2) Sinon on calcule

$$\lambda^*_{j_0}=\text{Min}\{\frac{((A^{J_0})^{-1}b)_i}{((A^{J_0})^{-1}A^{ra^*(j_0)})_i} : i\in \mathcal{H}(x^{(0)}),\ ((A^{J_0})^{-1}A^{ra^*(j_0)})_i > 0\} \quad (26)$$

On choisit $i_1\in \mathcal{H}(x^{(0)})$ tel que $((A^{J_0})^{-1}A^{ra^*(j_0)})_{i_1} > 0$ et $\dfrac{((A^{J_0})^{-1}b)_{i_1}}{((A^{J_0})^{-1}A^{ra^*(j_0)})_{i_1}} = \lambda^*_{j_0}$

On pose $J_1=[J_0\backslash\{ra(i_1)\}]\cup\{ra^*(j_0)\}$; $ra(i_1) = ra^*(j_0)$. On calcule

$$\begin{cases} x^{(1)}_{J_0} = x^{(0)}_{J_0} - \lambda^*_{j_0}(A^{J_0})^{-1}A^{ra^*(j_0)} \\ x^{(1)}_{J_0\backslash\{ra^*(j_0)\}} = 0;\ x^{(1)}_{ra^*(j_0)} = \lambda^*_{j_0} \end{cases}$$

On remplace I_0 par I_1 et $x^{(0)}$ par $x^{(1)}$, et on retourne à l'étape 1).

6.3.2. Procédé dual

On procède comme §6.3.1. pour obtenir un ensemble J_0, $|J_0|=m$, tel que A^{J_0} non singulière .

Soit $d=(d^{J_0}, d^{J_0*})$ un n-vecteur ligne défini par:
$d^{J_0} = y_0A^{J_0}$; $d^{J_0*} < y_0A^{J_0*}$

où y_0 est un m-vecteur ligne quelconque. D'après les résultats de §5.2., le programme linéaire auxiliaire (PA) suivant

$$(PA)\begin{cases} Ax = b \\ x \geq 0 \\ dx = z(max) \end{cases}$$

est dual réalisable. Son dual (DA) défini par

$$(DA)\begin{cases} yA \geq d \\ yb = w(min) \end{cases}$$

admet un sommet solution réalisable $y_0= (A^{J_0})^{-1}d^{J_0}$. L'application de la phase II de SGGP au problème dual auxiliaire (DA) va fournir un sommet de K, solution de (PA) ou bien conduire à la conclusion de la vacuité de K (ce qui correspond à la non existence de solution de (DA)).

6.4. Initialisation de l'algorithme SGGP au cas $K=\{x\in \mathbb{R}^n : Ax\le b, x\ge 0\}$

C'est un cas particulier de §5. où il n'y a pas de contrainte d'égalité, on pose:

$$B=\begin{bmatrix} A \\ -U_n \end{bmatrix}\ ,\quad a=\begin{bmatrix} b \\ 0 \end{bmatrix}$$

où U_n est la matrice d'identité d'ordre n.

6.4.1. Procédé primal

On obtient ainsi la version simplifiée du procédé primal pour la recherche d'un sommet de K.

0) On pose $I_0=\{m+1,\dots,m+n\} = \{ra(1),\dots,ra(n)\}$ et on résoud le système linéaire

$$B_{I_0}x = 0$$

dont la solution évidente est $x^{(0)}=0$.

On pose $\mathfrak{J}(x^{(0)})=\{i\in I_0^* : B_ix^{(0)}\le b_i\}$ et $\mathfrak{J}^*(x^{(0)}) = I_0^* \setminus \mathfrak{J}(x^{(0)})$

1) Si $\mathfrak{J}^*(x^{(0)})$ est vide alors $x^{(0)}\in V(K)$.

2) Sinon on choisit $i_0\in \mathfrak{J}^*(x^{(0)})$, par exemple i_0 tel que:

$$B_{i_0}x^{(0)} - b_{i_0} = \mathrm{Min}\{B_ix^{(0)} - b_i : i\in \mathfrak{J}^*(x^{(0)})\}$$

et on résoud le problème (P_0):

$$(P_0)\quad \begin{cases} B_{I_0}x \le a_{I_0} \\ B_{\mathfrak{J}(x^{(0)})}x \le a_{\mathfrak{J}(x^{(0)})} \\ -B_{i_0}x = z(\max) \end{cases}$$

On calcule la solution λ_0 du système linéaire: $\lambda B_{I_0} = - B_{i_0}$

2.1) Si $\lambda_0\ge 0$ alors $x^{(0)}$ est une solution de (P_0) et $K=\emptyset$.

2.2) Sinon, on choisit j_0 tel que $\lambda_0^{j_0} < 0$, par exemple:

$$\lambda_0^{j_0} = \mathrm{Min}\{\lambda_0^{j} : j=1,\dots,n\}$$

On calcule la solution $d^{(j_0)}$ du système linéaire:

$$\begin{cases} B_{I_0(j_0)}d = 0 \\ B_{ra(j_0)}d = -1 \end{cases}$$

$(I_0(j_0) = I_0\setminus\{ra(j_0)\})$

2.2.1) Si $\mathfrak{I}(x^{(0)})$ est vide ou si $B_{\mathfrak{I}(x^{(0)})}d^{(j_0)} \leq 0$ alors on calcule

$$\lambda_{j_0} = \frac{b_{i_0} - B_{i_0}x^{(0)}}{B_{i_0}d^{(j_0)}} > 0 \qquad (27)$$

On pose $I_1=[I_0\backslash\{ra(j_0)\}]\cup\{i_0\}$; $ra(j_0)= i_0$. On calcule $x^{(1)}$ défini par :

$$x^{(1)} = x^{(0)} + \lambda_{j_0} d^{(j_0)}$$

On remplace I_0 par I_1 et $x^{(0)}$ par $x^{(1)}$, on retourne à l'étape 1).

2.2.2) Sinon on calcule

$$\lambda^{*}_{j_0} = \text{Min}\{\frac{b_i - B_i x^{(0)}}{B_i d^{(j_0)}} : i \in \mathfrak{I}(x^{(0)}),\ B_i d^{(j_0)} > 0\}$$

On choisit $i_1 \in \mathfrak{I}(x^{(0)})$ tel que $B_{i_1}d^{(j_0)} > 0$ et $\frac{b_{i_1} - B_{i_1}x^{(0)}}{B_{i_1}d^{(j_0)}} = \lambda^{*}_{j_0}$

On pose $I_1=[I_0\backslash\{ra(j_0)\}]\cup\{i_1\}$; $ra(j_0) = i_1$.

$$x^{(1)} = x^{(0)} + \lambda_{j_0} d^{(j_0)}$$

On remplace I_0 par I_1 et $x^{(0)}$ par $x^{(1)}$, et on retourne à l'étape 1).

6.4.2. Procédé dual

Le choix de I_0 dans 0) de §6.4.1 et les résultats de §5.2. permettent d'affirmer que pour tout n-vecteur ligne $d \leq 0$, le programme linéaire (PA) suivant

$$(PA)\begin{cases} Ax \leq b \\ x \geq 0 \\ dx = z(max) \end{cases}$$

est dual réalisable. Son dual (DA) défini par

$$(DA)\begin{cases} yA \geq d\ ,\ y \geq 0 \\ yb = w(min) \end{cases}$$

admet y=0 comme un sommet solution réalisable. Dès lors l'application de la phase II de SGGP au problème dual (DA) permet d'obtenir un sommet de K, solution de (PA) ou bien conduire à la conclusion de la vacuité de K (ce qui correspond à la non existence de solution de (DA)).

6.5. Problème linéaire général dual réalisable

Reprenons le programme linéaire général sous forme canonique (P) défini précédemment:

$$(P)\quad \begin{cases} Ax = b \\ Bx \le a \\ cx = z(max) \end{cases}$$

Son problème dual (D) s'écrit

$$(D)\quad \begin{cases} yA + zB = c,\ z \ge 0 \\ yb + za = w(min) \end{cases}$$

On dit que (P) est dual réalisable si l'on dispose d'un ensemble $I \subset \{1,\dots,p\}$, $|I|=n-m$, tel que:

i) La matrice $\begin{bmatrix} A \\ B_I \end{bmatrix}$ soit non singulière

ii) La solution (λ,μ) du système linéaire:

$$\begin{cases} (\lambda,\mu)\begin{bmatrix} A \\ B \end{bmatrix} = c \\ \mu^{I^*} = 0 \end{cases} \qquad (28)$$

vérifie $\mu^{I} \ge 0$.

Cette définition généralise celle plus connue dans les cas usuels où $K=\{x\in \mathbb{R}^n : Ax=b,\ x\ge 0\}$ ou $K=\{x\in \mathbb{R}^n : Ax\le b,\ x\ge 0\}$ **[Luenberger, Sakarovitch]**. Lorsque (P) est dual réalisable, la solution (λ,μ) du (28) est un sommet du polyèdre convexe de solutions réalisables de (D). L'application de l'algorithme SGGP du problème (D) en partant de cette solution (λ,μ) permet de résoudre simultanément les problèmes duaux (P) et (D).

6.6. Programmes linéaires avec variables bornées. Programmes linéaires avec contraintes linéaires additionnelles en post optimisation.

Les bornes inférieures et supérieures des variables dans un programme linéaire général peuvent être traitées de différentes façons:

1) Les considérer comme des contraintes "ordinaires" dans un programme linéaire général que l'on résoud par SGGP.

2) Les considérer comme des contraintes linéaires additionnelles en post optimisation: On résoud en premier lieu le programme linéaire sans tenir compte des contraintes de bornes. Si la solution obtenue vérifie les bornes, elle est une solution optimale. Dans le cas contraire on y ajoute une à une (à commencer par une contrainte de borne la plus violée) ou toutes en même temps pour obtenir des programmes linéaires que l'on résoud par SGGP. L'utilisation des contraintes linéaires additionnelles en post optimisation est particulièrement efficace dans:

i) Les programmes linéaires de formes (standard et canonique) présentées dans §6.3 et §6.4) auxquels des contraintes de bornes sont ajoutées [Sakarovitch, Luenberger].

ii) Optimisation convexe : dans les méthodes d'approximation extérieure dans lequelles on approxime les fonctions objectives et on linéarise les contraintes [Luenberger, Pchenitchny & Daniline].

iii) L'optimisation combinatoire et l'optimisation non convexe où l'usage des contraintes linéaires additionnelles en post optimisation est essentiel [Sakarovitch, Pham Dinh Tao, Rosen].

7. Expérimentations numériques comparatives.

Trois versions du logiciel de SGGP ont été implémentées sur compatibles PC. Ecrites en PASCAL, elles correspondent au programme linéaire général (cf. § 3), au programme linéaire sous forme standard (cf. § 6.3) et au programme linéaire sous forme canonique (cf. § 6.4).

Ces trois logiciels utilisent le procédé primal dans la phase d'initialisation de SGGP (Phase I).

Dans ces trois programmes comme dans celui de l'algorithme du simplexe (sous forme révisé) que nous avons implémenté, la méthode utilisée pour résoudre les systèmes linéaires est l'algorithme d'élimination de Gauss couplé avec avec la technique de décomposition (cf. Remarque du §4.1).

Des expérimentations numériques ont été faites sur des programmes linéaires sous forme standard et sous forme canonique.

Nous avons voulu obtenir des indications concernant :

1) Le temps de calcul de la phase I de SGGP. Son rapport avec celui de la phase I de l'algorithme révisé du simplexe.

2) Le temps de calcul de la phase II de SGGP. Son rapport avec celui de la phase II de l'algorithme révisé du simplexe.

Il est bien connu que si A' (non singulière) est différente de A (non singulière) par une colonne ou une ligne alors l'inverse de A' s'exprime explicitement en fonction de A^{-1} et la colonne ou la ligne en question [Gastinel, Sakarovitch].

Ce procédé (appelé la "forme produit de l'inverse" dans la littérature de la programmation linéaire) couplé avec la forme révisé du simplexe peut être utilisé pour optimiser les calculs dans les résolutions des systèmes linéaires.

Nous avons l'intention de programmer les différentes versions de SGGP utilisant ledit procédé et d'en faire une étude comparative plus détaillée.

Il en ressort qu'à travers les exemples traités, la phase I de SGGP est beaucoup plus rapide que celle de l'algorithme révisé du simplexe : le rapport varie entre 3 et 8.

Quant à la phase II, le temps de calcul est le même pour les deux algorithmes (appliqués au programme linéaire sous forme standard) si l'on part d'un même sommet, car dans ce cas l'algorithme SGGP coïncide avec l'algorithme du simplexe.

Les exemples de § 7.1 montrent que la phase II de l'algorithme révisé du simplexe est moins coûteuse, cela n'a rien d'anormal : les sommets fournis par la phase I de ces deux algorithmes ne sont pas les mêmes. Il semble que les sommets obtenus par la phase I de l'algorithme révisé du simplexe sont meilleurs (i.e. qui optimise le coût de la phase II).

Ceci dit, le temps total de calcul (pour les deux phases) est plus faible pour l'algorithme SGGP : le rapport varie entre 1 et 3,5).

Dans le cas des programmes linéaires sous forme canonique les exemples de § 7.2 offrent les mêmes conclusions.

Sans prétendre d'être exhaustifs, les expérimentations numériques obtenus sont encourageantes.

L'utilité et l'efficacité de l'algorithme SGGP dans un très large domaine d'applications (dont nous indiquerons quelques lignes générales dans § 6.) sont des arguments incontestables pour son appréciation et son importance.

Dans les tableaux ci-dessous on compare les résultats numériques obtenus en appliquant l'algorithme S.G.G.P (resp. l'algorithme révisé du simplexe) sur des programmes linéaires sous forme standard " **Tableau 1.1**" (resp. **Tableau 1.2**) et sous forme canonique " **Tableau 2.1**" (resp. **Tableau 2.2**) .

7.1. Problèmes sous forme standard

Exemple 1.1

$$\min\ 2x_1 - x_2 - 3x_3 + 5x_4 - 2x_5 + 4x_7 + x_8 + 2x_9 - x_{10} + x_{11} - x_{12} + 2x_{14}$$

$$\begin{aligned}
x_1 + x_2 + x_{15} &= 8\\
2x_2 + x_3 + x_{16} &= 4\\
3x_3 - x_4 + x_{17} &= 6\\
2x_4 + 4x_5 + x_{18} &= 2\\
6x_5 + 2x_6 + x_{19} &= 5\\
-x_6 + 2x_7 + x_{20} &= 1\\
4x_7 - x_8 + x_{21} &= 2\\
3x_9 + x_{10} + x_{23} &= 6\\
3x_9 - x_{10} + x_{11} + x_{23} &= 3\\
x_{10} + x_{11} + 2x_{12} + x_{24} &= 9\\
x_{12} + 2x_{13} + x_{14} + x_{25} &= 4\\
x_i \geq 0\ ,\ i = 1,\ldots..,25 &
\end{aligned}$$

La solution optimale est :

$x^* = (0, 1, 2, 0, 0.5, 1, 0, 0, 0, 1.5, 0, 3.75, 0.125, 0, 7, 0, 0, 0, 0, 2, 2, 0, 4.5, 0, 0)$

La valeur de la fonction objective est : -13.25

Exemple 1.2

$$\min\ -18x_1 + 7x_2 - 12x_3 - 5x_4 - 8x_6$$

$$\begin{aligned}
2x_1 - 6x_2 + 2x_3 + 7x_4 + 3x_5 + 8x_6 + x_7 &= 1\\
-3x_1 - x_2 + 4x_3 - 3x_4 + x_5 + 2x_6 + x_8 &= -2\\
8x_1 - 3x_2 + 5x_3 - 2x_4 + 2x_6 + x_9 &= 4\\
4x_1 + 8x_3 + 7x_4 - x_5 + 3x_6 + x_{10} &= 1\\
5x_1 + 2x_2 - 3x_3 + 6x_4 - 2x_5 - x_6 + x_{11} &= 5\\
x_i \geq 0\ ,\ i = 1,\ldots..,11 &
\end{aligned}$$

La solution optimale est : $x^* = (2, 4, 0, 0, 7, 0, 0, 1, 0, 0, 1)$

La valeur de la fonction objective est : -8

Exemple 1.3

$$\min\ -x_1 - x_2 - x_3 - x_4 - x_5 - x_6$$

$$\begin{aligned}
-x_1 - x_2 + x_3 &= -6 \\
-x_1 + x_2 + x_4 &= 0 \\
x_2 + x_5 &= 5 \\
x_1 + x_2 + x_6 &= 12 \\
x_1 - x_2 &= 6 \\
x_2 &= 1 \\
x_i \geq 0\ ,\ i &= 1,\ldots\ldots,6
\end{aligned}$$

La solution optimale est : $x^* = (7, 1, 2, 6, 4, 4)$

La valeur de la fonction objective est : -24

S.G.G.P	Valeur de la fonction objective	Temps de calcul de la PHASE 1 (s)	Temps de calcul en secondes
Exemple 1.1	-13.25	5.33	10.16
Exemple 1.2	- 8	0.17	0.44
Exemple 1.3	-24	0.15	0.22

Tableau 1.1
L'algorithme S.G.G.P (cas de la forme standard)

SIMPLEXE	Valeur de la fonction objective	Temps de calcul de la PHASE 1 (s)	Temps de calcul en secondes
Exemple 1.1	-13.25	12.37	16.8
Exemple 1.2	- 8	0.44	0.54
Exemple 1.3	-24	0.40	0.48

Tableau 1.2
La méthode du simplexe (cas de la forme standard)

Résultats détaillés de l'algorithme révisé du simplexe pour le cas de la forme standard (PC, Olivetti_380) :

Exemple 1.1 :

La solution de la phase I :
(7.16667, 0.83333, 2.33333, 1.00000, 0.00000, 2.50000, 0.50000, 0.00000, 0.66667, 4.00000, 5.00000, 0.00000, 2.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 2.50000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000).

La solution optimale de la phase II :
(0.00000, 1.00000, 2.00000, 0.00000, 0.50000, 1.00000, 0.00000, 0.00000, 0.00000, 1.50000, 0.00000, 3.75000, 0.12500, 0.00000, 7.00000, 0.00000, 0.00000, 0.00000, 0.00000, 2.00000, 2.00000, 0.00000, 4.50000, 0.00000, 0.00000).

La valeur de la fonction objective = -13.25
Temps de calcul de la phase I = 12.37
Temps de calcul global = 16.80

Exemple 1.2 :

La solution de la phase I :
(2.57143, 5.50000, 0.00000, 0.03571, 9.53571,0.00000, 0.00000, 1.78571, 0.00000, 0.00000, 0.00000).

La solution optimale de la phase II :
(2.00000, 4.00000, 0.00000, 0.00000, 7.00000, 0.00000, 0.00000, 1.00000, 0.00000, 0.00000, 1.00000).

La valeur de la fonction objective = -8.00
Temps de calcul de la phase I = 0.44
Temps de calcul global = 0.56

Exemple 1.3 :

La solution de la phase I : (7.0000, 1.0000, 2.0000, 6.0000, 4.0000, 4.0000).
La solution de la phase II : (7.0000, 1.0000, 2.0000, 6.0000, 4.0000, 4.0000).
La valeur de la fonction objective = -24.00
Temps de calcul de la phase I = 0.40
Temps de calcul global = 0.48

Résultats détaillés de l'algorithme SGGP pour le cas de la forme standard (PC, Olivetti_380) :

Exemple 1.1 :

La solution de la phase I :
(7.16667, 0.83333, 2.33333, 1.00000, 0.00000, 2.50000, 0.50000, 0.00000, 0.66667, 4.00000, 5.00000, 0.00000, 2.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 2.50000, 0.00000, 0.00000, 0.00000, 0.00000,0.00000).

La solution optimale de la phase II :
(0.00000, 1.00000, 2.00000, 0.00000, 0.50000, 1.00000, 0.00000, 0.00000, 0.00000, 1.50000, 0.00000, 3.75000, 0.12500, 0.00000, 7.00000, 0.00000, 0.00000, 0.00000, 0.00000, 2.00000, 2.00000, 0.00000, 4.50000, 0.00000, 0.00000).

La valeur de la fonction objective = -13.25
Temps de calcul de la phase I = 5.33
Temps de calcul en secondes = 10.16

Exemple 1.2 :

La solution de la phase I :
(0.25000, 1.87500, 0.00000, 0.00000, 0.00000, 0.00000, 11.75000, 0.62500, 7.62500, 0.00000, 0.00000).

La solution optimale de la phase II :
(2.00000, 4.00000, 0.00000, 0.00000, 7.00000, 0.00000, 0.00000, 1.00000, 0.00000, 0.00000, 1.00000).

La valeur de la fonction objective = -8.00
Temps de calcul de la phase I = 0.17
Temps de calcul global = 0.44

Exemple 1.3 :

La solution de la phase I :
(7.00000, 1.00000, 2.00000, 6.00000, 4.00000, 4.00000).

La solution optimale de la phase II :
(7.00000, 1.00000, 2.00000, 6.00000, 4.00000, 4.00000).

La valeur de la fonction objective = -24.00
Temps de calcul de la phase I = 0.15
Temps de calcul global = 0.22

2. Problèmes sous forme canonique

Problème 2.1 (HITAC, IBM)

$$\text{Min } z = c^t.x$$
$$Ax \geq b$$
$$x \geq 0$$

où A est une matrice de type (m,n), $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$.

$m = 11$, $n = 17$

c = (25,30,350,150,40,20,100,40,60,100,17,20,20,12,75,900,20)

b = (2300,75,38,660,1300,10,1900,1.2,1.2,63,400)

et la matrice A est donnée par :

A =

351	270	260	451	156	59	721	58	130	118	77	51	40	24	28	311	40
5.2	8	17.5	12	12.7	2.9	0.6	6	17.5	20	1.9	1.3	1.2	1.6	3	34.2	0.8
0.8	1.5	20.5	44.3	11.2	3.3	81.6	3.5	6	3.5	0.1	0.2	0.2	0.2	0.4	0.7	0.3
6	11	6	9	65	100	10	120	80	12	5	35	40	45	98	470	14
2	480	90	90	90	36	780	5	100	90	12	57	10	15	25	600	4
0.4	1	1.3	1.2	2.6	0.1	0.1	1.4	3	0.7	0.5	0.5	0.5	0.4	3.3	23	0.2
0	0	25	0	800	120	2400	0	60	40	0	1300	6	33	2600	10000	40
0.09	0.1	0.04	0.4	0.1	0.04	0.01	0.02	0.02	0.15	0.1	1.06	0.03	0.08	0.12	0.21	0.09
0.03	0.03	0.11	0.1	0.4	0.15	0.03	0.02	0.15	0.2	0.03	0.04	0.02	0.05	0.3	1	0.02
0	0	0	0	0	2	0	0	1	1	15	7	10	50	100	20	50
0	0	0	0	10	0	0	0	530	0	0	0	0	0	0	0	0

La solution optimale est :

x^* = (3.45, 1.77, 0, 0, 0.75, 2.07, 0.14, 0, 0.74, 0, 0, 0.34, 0, 6.46, 0, 0, 0)

La valeur de la fonction objective est : 354.03042

Problème 2.2

$$\text{Max} \quad 139x_1 + 88x_2 + 133x_3 + 137x_4 + 165x_5$$

$$\begin{aligned}
1.4x_1 + 1.8x_2 + 1.5x_3 + 1.4x_4 + 1.4x_5 &\leq 420 \\
9.8x_1 + 2.4x_2 + 1.4x_3 + 1.4x_4 + 1.4x_5 &\leq 415 \\
4.0x_1 + 0.4x_2 + 1.4x_3 + 1.4x_4 + 1.4x_5 &\leq 355 \\
2.8x_1 + 0.6x_2 + 1.3x_3 + 1.4x_4 + 1.5x_5 + 5.5x_6 &\leq 345 \\
2.2x_1 + 0.4x_2 + 1.3x_3 + 1.5x_4 + 1.2x_5 + 5.5x_7 &\leq 160
\end{aligned}$$

$$
\begin{aligned}
&2.2x_1 + 0.4x_2 + 1.3x_3 + 1.3x_4 + 1.2x_5 + 5.5x_8 && \leq 95\\
&2.2x_1 + 0.6x_2 + 1.3x_3 + 1.3x_4 + 1.2x_5 + 5.5x_9 && \leq 380\\
&2.6x_1 + 5.8x_2 + 1.5x_3 + 1.5x_4 + 1.2x_5 + 5.5x_{10} && \leq 395\\
&0.6x_1 + 4.0x_2 + 1.3x_3 + 5.5x_{11} && \leq 270\\
&0.6x_1 + 1.2x_2 + 1.3x_3 + 1.3x_4 + 2.6x_5 && \leq 230\\
&0.6x_1 + 1.8x_2 + 1.2x_3 + 1.2x_4 + 1.2x_5 && \leq 310\\
&0.6x_1 + 1.8x_2 + 1.5x_3 + 1.4x_4 + 1.4x_5 && \leq 420\\
&16x_1 + 12x_4 + 35x_5 + 50x_6 + 50x_7 && \leq 5200\\
&20x_1 + 36x_4 + 50x_5 + 50x_8 + 50x_9 && \leq 5200\\
&16x_1 + 12x_4 + 35x_5 + 50x_{10} + 50x_{11} && \leq 3600\\
&0.1x_2 + 0.9x_3 + 0.8x_4 + 2.3x_5 - x_6 - x_7 - x_8 - x_9 - x_{10} - x_{11} && = 0
\end{aligned}
$$

$x_i \geq 0$, $i = 1, \ldots, 11$

La solution optimale est :
$x^* = (0, 54.8, 0, 20.3572, 38.8458, 40.9728, 11.078, 0, 49.8254, 0, 9.235)$
La valeur de la fonction objective est : 14021.03787

Problème 2.3

$$\text{Max} \quad x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10}$$

$$
\begin{aligned}
&x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 5x_6 + 4x_7 + 3x_8 + 2x_9 + x_{10} && \leq 10000\\
&6x_1 + 7x_2 + 8x_3 + 9x_4 + 10x_5 + 5x_6 + 2x_7 + 8x_8 + 3x_9 + x_{10} && \leq 10000\\
&11x_1 + 12x_2 + 13x_3 + 14x_4 + 15x_5 + 6x_6 + 7x_7 + 80x_8 + 90x_9 + 10x_{10} && \leq 10000\\
&x_1 + 10x_2 + 20x_3 + 30x_4 + 40x_5 + 50x_6 + 60x_7 + 80x_8 + 90x_9 + 10x_{10} && \leq 10000\\
&3x_1 + 9x_2 + 27x_3 + 60x_4 + 45x_5 + 60x_6 + 75x_7 + 8x_8 + 9x_9 + 46x_{10} && \leq 10000
\end{aligned}
$$

$x_i \geq 0$, $i = 1, \ldots, 10$

La solution optimale est : $x^* = (841.12, 0, 0, 0, 0, 124.61, 0, 0, 0, 0)$
La valeur de la fonction objective est : 965.732

Exemple 2.4 (Klee-Minty)

$$\text{Max} \quad 0.0256x_1 + 0.064x_2 + 0.16x_3 + 0.4x_4 + x_5$$

$$
\begin{aligned}
&3x_1 && \leq 1\\
&0.8x_1 + x_2 && \leq 1\\
&0.32x_1 + 0.8x_2 + x_3 && \leq 1\\
&0.128x_1 + 0.32x_2 + 0.8x_3 + x_4 && \leq 1\\
&0.0512x_1 + 0.128x_2 + 0.32x_3 + 0.8x_4 + x_5 && \leq 1
\end{aligned}
$$

$x_i \geq 0$, $i = 1, \ldots, 5$

La solution optimale est : $x^* = (0, 0, 0, 0, 1)$
La valeur de la fonction objective est : 1

Problème 2.5

$$\text{Max} \quad 139x_1 + 88x_2 + 133x_3 + 137x_4 + 165x_5$$

$$\begin{aligned}
&1.4x_1 - 1.8x_2 + 1.5x_3 + 1.4x_4 + 1.4x_5 && \leq 420\\
&9.8x_1 + 2.4x_2 - 1.4x_3 + 1.4x_4 - 1.4x_5 && \leq -15\\
&4.0x_1 + 0.4x_2 + 1.4x_3 - 1.4x_4 + 1.4x_5 && \leq 355\\
&-2.8x_1 + 0.6x_2 + 1.3x_3 + 1.4x_4 + 1.5x_5 - 5.5x_6 && \leq 345\\
&2.2x_1 + 0.4x_2 - 1.3x_3 + 1.5x_4 + 1.2x_5 + 5.5x_7 && \leq -16\\
&2.2x_1 + 0.4x_2 - 1.3x_3 - 1.3x_4 - 1.2x_5 - 5.5x_8 && \leq 95\\
&2.2x_1 + 0.6x_2 - 1.3x_3 + 1.3x_4 + 1.2x_5 + 5.5x_9 && \leq -38\\
&2.6x_1 + 5.8x_2 + 1.5x_3 + 1.5x_4 - 1.2x_5 + 5.5x_{10} && \leq 395\\
&0.6x_1 - 4.0x_2 + 1.3x_3 + 5.5x_{11} && \leq 270\\
&0.6x_1 - 1.2x_2 - 1.3x_3 - 1.3x_4 - 2.6x_5 && \leq -230\\
&0.6x_1 + 1.8x_2 + 1.2x_3 + 1.2x_4 + 1.2x_5 && \leq 310\\
&0.6x_1 + 1.8x_2 + 1.5x_3 + 1.4x_4 + 1.4x_5 && \leq 420\\
&16x_1 + 12x_4 + 35x_5 + 50x_6 + 50x_7 && \leq 5200\\
&x_i \geq 0 \ , \ i = 1, \ldots\ldots , 11
\end{aligned}$$

La solution optimale est :
$x^* = (24.637, 0, 156.224, 31.418, 58.371, 44.858, 2.857, 0, 0, 21.731, 9.477)$
La valeur de la fonction objective est : 38138.12381

Problème 2.6

$$\text{Max} \quad x_1 + x_2 + x_3$$

$$\begin{aligned}
&-x_1 + 2x_2 && \leq 4\\
&x_1 - 3x_2 - x_3 && \leq -3\\
&x_1 + x_2 && \leq 9\\
&-x_1 - 0.5x_2 + x_3 && \leq -2\\
&-2x_2 && \leq -5\\
&x_i \geq 0 \ , \ i = 1, 2, 3.
\end{aligned}$$

La solution optimale est : $x^* = (6.5, 2.5, 5.75)$
La valeur de la fonction objective est : 14.75

S.G.G.P	Valeur de la fonction objective	Temps de calcul de la PHASE 1 (s)	Temps de calcul en secondes
Exemple 2.1	354.0304	0.60	5.87
Exemple 2.2	14021.0309	0.00	6.36
Exemple 2.3	965.732	0.00	0.22
Exemple 2.4	1	0.00	0.1
Exemple 2.5	38138.123281	0.60	2.36
Exemple 2.6	14.75	0.11	0.22

Tableau 2.1
L'algorithme S.G.G.P (cas de la forme canonique)

SIMPLEXE	Valeur de la fonction objective	Temps de calcul de la PHASE 1 (s)	Temps de calcul en secondes
Exemple 2.1	354.0304	4.76	6.38
Exemple 2.2	14021.03	0.15	6.53
Exemple 2.3	965.732	0.07	0.29
Exemple 2.4	1	0.05	0.21
Exemple 2.5	38138.123281	7.20	7.47
Exemple 2.6	14.75	0.38	0.55

Tableau 2.2
La méthode du simplexe (cas de la forme canonique)

Résultats détaillés de l'algorithme révisé du simplexe pour le cas de la forme canonique (PC, Olivetti_380) :

Exemple 2.1 :

La solution de la phase I :
(3.50155, 1.69228, 0.00000, 0.00075, 0.25893, 2.89234, 0.16157, 0.00000, 0.74983, 0.42294, 0.00000, 0.09234, 0.00000, 4.78539, 0.00000, 0.06181, 0.00000).

La solution de la phase II :
(3.45097, 1.77091, 0.00000, 0.00000, 0.75577, 2.07180, 0.14065, 0.00000, 0.74046, 0.00000, 0.00000, 0.34738, 0.00000, 6.46019, 0.00000, 0.00000, 0.00000).

La valeur optimale de la fonction objective = 354.0304
Le temps de calcul de la phase I = 4.76
Le temps global de calcul = 6.38

Exemple 2.2 :

La solution de la phase I :
(0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000).

La solution de la phase II :
(0.00000, 54.80158, 0.00000, 20.35732, 38.84580, 40.97277, 11.07792, 0.00000, 49.82540, 0.00000, 9.23521).

La valeur optimale de la fonction objective = 14021.0309
Le temps de calcul de la phase I = 0.15
Le temps global de calcul = 6.53

Exemple 2.3 :

La solution de la phase I :
(0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000).

La solution de la phase II :
(841.12150, 0.00000, 0.00000, 0.00000, 0.00000, 124.61059, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000).

La valeur optimale de la fonction objective = 965.732
Le temps de calcul de la phase I = 0.07
Le temps global de calcul = 0.29

Exemple 2.4 :

La solution de la phase I : (0.00000, 0.00000, 0.00000, 0.00000, 0.00000).
La solution de la phase II : (0.00000, 0.00000, 0.00000, 0.00000, 1.00000).
La valeur optimale de la fonction objective = 1.00000
Le temps de calcul de la phase I = 0.05
Le temps global de calcul = 0.21

Exemple 2.5 :

La solution de la phase I :
(24.63768, 0.00000, 156.22484, 31.41822, 58.37143, 44.85805, 2.85752, 0.00000, 0.00000, 21.73148, 9.47729).

La solution de la phase II :
(24.63768, 0.00000, 156.22484, 31.41822, 58.37143, 44.85805, 2.85752, 0.00000, 0.00000, 21.73148, 9.47729).

La valeur optimale de la fonction objective = 38138.12381
Le temps de calcul de la phase I = 7.20000
Le temps global de calcul = 7.47000

Exemple 2.6 :

La solution de la phase I : (4.66667, 4.33333, 4.83333).
La solution de la phase II : (6.50000, 2.50000, 5.75000).
La valeur optimale de la fonction objective = 14.75
Le temps de calcul de la phase I = 0.38
Le temps global de calcul = 0.55

Résultats détaillés de l'algorithme SGGP pour le cas de la forme canonique (PC, Olivetti_380) :

Exemple 2.1 :
La solution de la phase I :
(0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.07771, 0.00000, 4.66202, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 5.26658, 0.00000).
La solution de la phase II :
(3.45097, 1.77091, 0.00000, 0.00000, 0.75577, 2.07180, 0.14065, 0.00000, 0.74046, 0.00000, 0.00000, 0.34738, 0.00000, 6.46019, 0.00000, 0.00000, 0.00000).
La valeur optimale de la fonction objective = 354.0304
Le temps de calcul de la phase I = 0.60
Le temps global de calcul = 5.87

Exemple 2.2 :

La solution de la phase I :
(0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000).
La solution de la phase II :
(0.00000, 54.80158, 0.00000, 20.35732, 38.84580, 40.97277, 11.07792, 0.00000, 49.82540, 0.00000, 9.23521).
La valeur optimale de la fonction objective = 14021.0309
Le temps de calcul de la phase I = 0.00
Le temps global de calcul = 6.36

Exemple 2.3 :

La solution de la phase I :
(0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000).
La solution de la phase II :
(841.12150, 0.00000, 0.00000, 0.00000, 0.00000, 124.61059, 0.00000, 0.00000, 0.00000, 0.00000).

La valeur optimale de la fonction objective = 965.732
Le temps de calcul de la phase I = 0.00
Le temps global de calcul = 0.22

Exemple 2.4 :

La solution de la phase I : (0.00000, 0.00000, 0.00000, 0.00000, 0.00000).
La solution de la phase II : (0.00000, 0.00000, 0.00000, 0.00000, 1.00000).
La valeur optimale de la fonction objective = 1.00000
Le temps de calcul de la phase I = 0.00
Le temps global de calcul = 0.10

Exemple 2.5 :

La solution de la phase I :
(0.00000, 0.00000, 176.92300, 0.00000, 0.00000, 104.00000, 0.00000, 0.00000, 0.00000, 0.00000, 0.00000).
La solution optimale de la phase II :
(24.63768, 0.00000, 156.22484, 31.41822, 58.37143, 44.85805, 2.85752, 0.00000, 0.00000, 21.73148, 9.47729).

La valeur de la fonction objective = 38138.12381
Temps de calcul de la phase I = 0.60000
Temps global de calcul en secondes = 2.36000

Exemple 2.6 :

La solution de la phase I : (4.66667, 4.33333, 0.00000).
La solution de la phase II : (6.50000, 2.50000, 5.75000).
La valeur optimale de la fonction objective = 14.75
Le temps de calcul de la phase I = 0.11
Le temps global de calcul = 0.22

Références

Luenberger, D. G. (1972), Introduction to linear and nonlinear programming. Addison-Welsey.

Pham Dinh Tao, El Bernoussi Souad (1989), Numerical algorithms for solving a class of global nonconvex optimization problems. International Series of Numerical Mathematics: "New Algorithms in Optimization and Their Industrial Use", Birkhauser Verlag.

Pham Dinh Tao, Jamal Chaarani, El Bernoussi Souad (1989), Global numerical algorithms for solving a class of nonconvex optimization problems. To appear in Zeitschrift für Operations Research (ZOR).

Rockafellar, R.T. (1970), Convex analysis, Princeton Univ. Press (1970).

Sakarovitch, M. (1984), Programmation linéaire, Hermann.

Sakarovitch, M. (1984), Optimisation combinatoire (Méthodes mathématiques et algorithmiques), Programmation discrète. Hermann.

Pchenitchny & Daniline (1977), Méthodes Numériques dans les problèmes d'extrémum. Edition Mir.

Rosen, J. B. and Pardalos, P. M. (1986), Methods for global concave minimization : a bibliographic survey. SIAM Review vol. 28, n° 3.

Gastinel, N. (1966), Analyse Numérique Linéaire. Hermann, Paris.

Pham Dinh Tao (1989), Un algorithme pour la résolution du programme linéaire général. A paraître.

CHAPITRE II

PROGRAMMES LINEAIRES ET NON LINEAIRES COMPLEMENTAIRES THEORIE, ALGORITHMES ET APPLICATIONS

1. Introduction.

Les problèmes de complémentarité constituent un chapitre important des mathématiques modernes. Il a attiré, durant les dernières décennies, l'intérêt de plusieurs auteurs. La littérature sur ce problème est déjà impressionnante.

L'étude de la complémentarité a commencé par les travaux de R. W. Cottle publiés dans la période 1964-1966 [55,56]. Nous avons remarqué que le même problème se retrouve dans des contextes différents (sans avoir le nom de complémentarité), voir [52,54].

Pour expliquer l'origine du terme "complémentarité" porté par les problèmes considérés dans cette Thèse, il faut analyser la forme la plus simple de ce problème.

Soit $\mathbb{R}^n$ l'espace vectoriel réel de dimension n sur lequel on considère l'ordre naturel défini par le cône convexe $\mathbb{R}^n_+$, c'est-à-dire,

$x \geq y \Leftrightarrow x-y \in \mathbb{R}^n_+$.

Si $f : \mathbb{R}^n_+ \mapsto \mathbb{R}^n$, est une fonction donnée, alors le problème de complémentarité général associé à f et $\mathbb{R}^n_+$ est donné par:

$$(1) : \left[\begin{array}{l} \text{Trouver } x^* \in \mathbb{R}^n \text{ tel que,} \\ x^* \geq 0 \; ; \; f(x^*) \geq 0 \; ; \; \langle x^*, f(x^*) \rangle = 0 \end{array} \right.$$

Si f est une fonction affine, c'est-à-dire $f(x) = Mx+q$, avec $M \in \mathbb{R}^{nxn}$ et $q \in \mathbb{R}^n$ on obtient le problème linéaire complémentaire noté par la suite **PLC**, sinon, on dit que c'est un problème non linéaire complémentaire et on le désigne par **PNC**.

On dit qu'une solution x^* est non-dégénérée si le vecteur $(x^*, f(x^*))$ (qui a 2n composantes) a au moins n composantes non nulles.
On observe que si x^* est une solution non-dégénérée du problème (1) et si on note $y^* = f(x^*)$ alors les deux ensembles,
$A = \{i : x^*_i > 0\}$; $B = \{i : y^*_i > 0\}$ sont complémentaires non vides.
Cette observation a inspiré le nom de "complémentarité".

La complémentarité a attiré de plus en plus de chercheurs, spécialement après les travaux de: Cottle, Dantzig, Lemke, Karamardian, Mangasarian, Moré, Kojima, Eaves, Tamir, Tolle et récemment Pang.
L'intérêt grandissant sur la complémentarité s'explique par ses domaines d'applications [2,14,17,19,20,21,22,26,30,33].

Ce chapitre veut exposer, dans un premier temps, les domaines d'applications de problèmes de complémentarité et ses relations avec d'autres problèmes mathématiques importants (en particulier la programmation linéaire, la programmation quadratique, les inéquations variationnelles et d'autres problèmes d'optimisation non convexes).

Ensuite, nous ferons un rappel concernant le problème de programmation linéaire et sa méthode de résolution connue sous le nom de méthode du simplexe découverte par **G.Dantzig** en 1947 sur laquelle est basée la méthode développée par **Lemke** en 1969 pour résoudre le problème linéaire complémentaire.

Nous présenterons ensuite différents autres algorithmes de résolution du **PLC** (méthodes de Newton et méthodes itératives) et les propriétés de convergence pour chacun de ces algorithmes.

A la fin de ce chapitre, nous étudierons la méthode de Quasi-Newton (Newton inexacte) adaptée par Pang et la méthode de l'inverse partiel développée par Spingarn en 1983 pour la résolution de PNC.

Mes contributions propres sont centrées sur les points suivants :

1) Etude adaptative de ces méthodes aux problèmes d'applications (en particulier le problème de régression isotone et de régression concave en Analyse des Données) (cf. Chap. III, §5.).

2) Implémentation effective de l'algorithme de Lemke. Expérimentations numériques comparatives.

Les résultats numériques seront présentés dans le dernier chapitre de cette Thèse (Chap. VI).

2. Domaines d'applications.

2.1. Programmation Linéaire.

Soient $c = (c_1, c_2, ..., c_n) \in \mathbb{R}^n$ et $b = (b_1, b_2, ..., b_m) \in \mathbb{R}^m$ et A une matrice de type mxn. On considère le programme linéaire primal:

$$(P.L.P.) : \quad \min \{ \langle c,x \rangle ; \ x \in D_1 \}$$
$$D_1 = \{ x \in \mathbb{R}^n ; \ x \geq 0, \ Ax \leq b \}$$

dont le dual s'écrit :

$$(P.L.D.) : \quad \max \{ \langle y,b \rangle ; \ y \in D_2 \}$$
$$D_2 = \{ y \in \mathbb{R}^m ; \ y \geq 0, \ A^t y \geq c \}$$

Le résultat suivant de la programmation linéaire [8]:

$(x,y) \in D_1 x D_2$ est un couple de solutions de (P.L.P) et de (P.L.D) respectivement si et seulement si $\langle c,x_0 \rangle = \langle y_0,b \rangle$ conduit à associer au couple desdits programmes le programme linéaire complémentaire:

$$(P.L.C.) : \left[\begin{array}{l} \text{Trouver } z \in \mathbb{R}^{n+m} \text{ tel que} \\ z \geq 0 \ , \ w = Mz + q \geq 0 \ , \ \langle z,w \rangle = 0 \end{array} \right.$$

où $u = 0.x + A^t y - c$, $v = -Ax + 0.y + b$

$$w = \begin{bmatrix} u \\ v \end{bmatrix}, z = \begin{bmatrix} x \\ y \end{bmatrix}, M = \begin{bmatrix} 0 & A^t \\ -A & 0 \end{bmatrix} \text{ et } q = \begin{bmatrix} -c \\ b \end{bmatrix}$$

($u \in \mathbb{R}^n$, $v \in \mathbb{R}^m$ et M est une matrice carrée d'ordre m+n).

On peut ainsi résoudre (P.L.C) pour obtenir un couple de solutions des programmes duaux et vice-versa.

Remarques:

1) L'idée fondamentale de la complémentarité en programmation linéaire est que la complémentarité transforme un problème d'optimisation en une équation.
2) La transformation d'un programme linéaire en un (PLC) est pour nous d'une grande importance car on résout le problème primal et le problème dual en même temps. ◊

2.2. Programmation quadratique.

On considère maintenant le problème de programmation quadratique suivant:

$$(1): \left[\begin{array}{l} \min \{ f(x) ; x \in D \} \\ \text{où } D = \{ x \in \mathbb{R}^n / x \geq 0 , Ax \leq b \} \\ f(x) = \frac{1}{2} \langle x, Qx \rangle + \langle c, x \rangle ; \end{array}\right.$$

où $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, A est une matrice de type mxn et Q est une matrice carrée symétrique d'ordre n.

Les conditions nécessaires de Kuhn-Tucker pour le programme (1) (qui sont aussi suffisantes au cas où Q est en plus semi-définie positive) sont:

$$(2): \left[\begin{array}{l} \exists\, u \in \mathbb{R}^n, v \in \mathbb{R}^m, \lambda \in \mathbb{R}^m \text{ tels que :} \\ Qx + c + A^t\lambda - u = 0 \\ Ax + v = b \\ u, v, x, \lambda \geq 0 \\ \langle u, x \rangle = 0 \text{ et } \langle v, \lambda \rangle = 0 \end{array}\right]$$

Autrement dit :

$$(3): \left[\begin{array}{l} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} c \\ b \end{bmatrix} + \begin{bmatrix} Q & A^t \\ -A & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} \\ \begin{bmatrix} u \\ v \end{bmatrix} \in \mathbb{R}_+^{n+m} ; \begin{bmatrix} x \\ \lambda \end{bmatrix} \in \mathbb{R}_+^{n+m} ; \left\langle \begin{bmatrix} u \\ v \end{bmatrix}, \begin{bmatrix} x \\ \lambda \end{bmatrix} \right\rangle = 0 \end{array}\right]$$

Si on note : $w = \begin{bmatrix} u \\ v \end{bmatrix}$; $z = \begin{bmatrix} x \\ \lambda \end{bmatrix}$; $q = \begin{bmatrix} c \\ b \end{bmatrix}$; $M = \begin{bmatrix} Q & A^t \\ -A & 0 \end{bmatrix}$.

alors les conditions Kuhn-Tucker (2) sont équivalentes au problème de complémentarité linéaire:

$$(4): \left[\begin{array}{l} \text{Trouver } z \in \mathbb{R}^{n+m} \text{ tel que :} \\ z \geq 0 \ ; w = Mz + q \geq 0 \ ; \langle z, w \rangle = 0. \end{array}\right]$$

Remarque:
Avec la contrainte de positivité on a un problème linéaire complémentaire dont la matrice M de dimension plus grande mais on n'a pas besoin de calculer Q^{-1} (comme dans le cas (5) ci-dessous). ◊

2.3. Programmation linéaire, programmation quadratique et complémentarité.

Considérons dans cette partie, le programme linéaire:

$$(1): \begin{cases} \min \{\langle p,x\rangle : x\in \mathbb{R}^n \text{ et } Ax \le b\} \\ \text{où } p\in \mathbb{R}^n,\ b\in \mathbb{R}^m,\ A \text{ est une matrice de type mxn.} \end{cases}$$

Nous supposons que la matrice A n'a pas de ligne nulle et considérons une perturbation quadratique du problème (1) de la forme suivante ($\varepsilon > 0$) :

$$(2): \begin{cases} \min \{\langle p,x\rangle + \frac{\varepsilon}{2}\langle x,x\rangle : x\in D\} \\ \text{où } D = \{x\in \mathbb{R}^n : Ax \le b\} \end{cases}$$

On pourrait utiliser ainsi l'algorithme de Lemke via cette perturbation quadratique pour le traitement de la dégénérescence dans la programmation linéaire. En 1979 Mangasarian et Meyer ont établi le théorème suivant [53]:

Théorème 2 .

Le programme (2) a toujours une solution unique $x_0(\varepsilon)$ (pour $\varepsilon>0$). Si le programme (1) admet une solution alors il existe $\alpha >0$ tel que $x_0(\varepsilon)$ ne dépend pas de $\varepsilon \in [0,\alpha]$, et par conséquent elle est une solution du programme (1). ◊

Nous considérons maintenant le cas général de (2).

$$(3): \begin{cases} \min \{\frac{1}{2}\langle x,Qx\rangle + \langle p,x\rangle : x\in D\} \\ \text{où } D = \{x\in \mathbb{R}^n : Ax \le b\} \\ p\in \mathbb{R}^n,\ b\in \mathbb{R}^m \\ Q \text{ est une matrice carrée d'ordre n et A est une matrice de type mxn} \end{cases}$$

dont le problème dual s'écrit:

$$(4): \begin{cases} \max\ g(u) \\ u \in \mathbb{R}^m_+ \\ \text{où } g(u) = \min \{\frac{1}{2}\langle x,Qx\rangle + \langle p,x\rangle + \langle u,Ax-b\rangle : x\in \mathbb{R}^n\} \end{cases}$$

Un calcul simple montre que:

$$-g(u) = \langle u, AQ^{-1}A^t u\rangle/2 + \langle AQ^{-1}p + b, u\rangle + \langle p, Q^{-1}p\rangle$$

Par suite le problème dual (4) est équivalent à:

$$(5): \quad \min\{\frac{1}{2}\langle u, AQ^{-1}A^t u\rangle + \langle b+AQ^{-1}p, u\rangle : u \in \mathbb{R}^m_+\}$$

Comme la matrice $AQ^{-1}A^t$ est semi-définie positive, résoudre (5) revient à résoudre le problème de complémentairité suivant:

$$(P.L.C)\left[\begin{array}{l} \text{Trouver } u \in \mathbb{R}^m \text{ tel que :} \\ u \geq 0\,,\ v = AQ^{-1}A^t u + (AQ^{-1}p + b) \in \mathbb{R}^m_+\,,\ \langle u, v\rangle = 0 \end{array}\right.$$

Pour tous les détails concernant cette partie, voir [53].

2.4. Inéquations variationnelles et complémentarité.

Les inéquations variationnelles sont imposées par le calcul des variations quand la fonction à minimiser est définie sur un ensemble convexe de contraintes.Dans ce cas, l'équation classique d'Euler est remplacée par un système d'inéquations.
Pour préciser cette idée: soit $f: \mathbb{R}^n \mapsto \mathbb{R}$ une fonction de classe C^1 et $K \subset \mathbb{R}^n$ un ensemble convexe fermé. S'il existe $x_0 \in K$ tel que:
(1) : $f(x_0) = \min\{f(x);\ x \in K\}$, alors x_0 vérifie :

$$(2): \left[\begin{array}{l} x_0 \in K \\ \langle f'(x_0), x-x_0\rangle \geq 0\ ;\ \forall\, x \in K\,. \end{array}\right]$$

Généralement, une solution du problème (2) n'est pas une solution du problème (1) que seulement si f est une fonction convexe.
Une généralisation naturelle du problème (2) est le problème suivant:

Soit $f: \mathbb{R}^n \mapsto \mathbb{R}^n$ est une application continue,

$$(3): \left[\begin{array}{l} \text{Trouver } x_0 \in \mathbb{R}^n \text{ tel que :} \\ x_0 \in K \text{ et } \langle f(x_0), x-x_0\rangle \geq 0\ ;\ \forall\, x \in K. \end{array}\right]$$

Lorsque K est un cône convexe, le problème (3) est équivalent au problème de complémentarité suivant:

$$(4) : \left[\begin{array}{l} \text{Trouver } x_0 \in K \text{ tel que :} \\ f(x_0) \in K^* \quad \text{et} \quad < x_0 , f(x_0) > = 0 \end{array} \right]$$

où K^* est le cône polaire de K : $K^* = \{y \in \mathbb{R}^n : \langle x,y \rangle \leq 0, \forall\, x \in K\}$.

Les inéquations variationnelles ont été étudiées principalement par: Stampacchia, Browder, Brézis, Rockafellar, Lions, Mosco etc .

Les principaux domaines d'application des inéquations variationnelles sont [51] :

- l'étude des phénomènes physiques à frontière libre
- l'étude des phénomènes subsoniques
- l'étude des phénomènes de filtration dans des milieux poreux
- l'étude des problèmes de contrôle stochastique
- l'étude de certains phénomènes en élasticité
- les équations différentielles d'évolution.

2.5. La théorie du min-max (point selle).

Soit $D \subset \mathbb{R}^n \times \mathbb{R}^m$ un sous-ensemble ouvert tel que $\mathbb{R}^n_+ \times \mathbb{R}^m_+ \subset D$ et $f: D \mapsto \mathbb{R}$ une fonction différentiable.

On considère le problème:

$$(1) : \left[\begin{array}{l} \text{Trouver } z_* \in \mathbb{R}^n_+ \ , \ y_* \in \mathbb{R}^m_+ \ \text{ tel que ,} \\ f(z , y_*) \leq f(z_* , y_*) \leq f(z_* , y) \ ; \ \forall\, (z,y) \in \mathbb{R}^n_+ \times \mathbb{R}^m_+ \end{array} \right]$$

Si (z_*, y_*) est une solution du problème (1), alors on dit que (z_*, y_*) est un point selle ou un point de min-max pour la fonction f sur $\mathbb{R}^n_+ \times \mathbb{R}^m_+$.
On considère la fonction F de composantes, $F_1, F_2, F_3, \ldots\ldots, F_n, \ldots\ldots, F_{n+m}$ définie par :

$$\begin{bmatrix} F_1 \\ F_2 \\ \cdot \\ \cdot \\ \cdot \\ F_n \end{bmatrix} (z,y) = - \begin{bmatrix} \frac{\partial f}{\partial z_1}(z,y) \\ \cdot \\ \cdot \\ \cdot \\ \frac{\partial f}{\partial z_n}(z,y) \end{bmatrix} \quad \text{et} \quad \begin{bmatrix} F_{n+1} \\ F_{n+2} \\ \cdot \\ \cdot \\ \cdot \\ F_{n+m} \end{bmatrix} (z,y) = \begin{bmatrix} \frac{\partial f}{\partial y_1}(z,y) \\ \cdot \\ \cdot \\ \cdot \\ \frac{\partial f}{\partial y_m}(z,y) \end{bmatrix} .$$

Si (z_*, y_*) est un point de min-max positif pour f, alors il est aussi une solution du problème de complémentarité suivant:

$$(\text{P.C.N.L.}) \; : \left[\begin{array}{l} \text{Trouver} \quad X = \begin{bmatrix} z \\ y \end{bmatrix} \in \mathbb{R}_+^n \times \mathbb{R}_+^m \quad \text{tel que ,} \\ \\ F(X) \in \mathbb{R}_+^n \times \mathbb{R}_+^m \quad \text{et} \quad \langle X , F(X) \rangle = 0 \end{array} \right]$$

Remarque:
Si f est concave en z et convexe en y, alors les points selle de f et les solutions du problème de complémentarité non-linéaire (P.C.N.L.) coïncident. ◊

2.6. La théorie du temps d'arrêt optimal

On considère une chaine de Markov (X, E, P) où : $X = \{X_n / n \in \mathbb{N}\}$, E est l'ensemble des états et P est la matrice de transition. On suppose donner la fonction bornée f: $X \mapsto \mathbb{R}$.

Avec la matrice P et la fonction f on observe longtemps le processus X.
Si le processus est arrêté, et il se trouve dans l'état j, il résulte un paiement f(j) et le jeu s'arrête. Si le processus n'est pas stoppé il n' y a pas de paiement.
Pour $i \in E$ soit $V(i) = \sup \{ E_i [f(x_T)] / T \in \mathbb{N} \}$ où E_i est la valeur attendue si le processus a démarré de l'état $i \in E$. La fonction V s'appelle la valeur du jeu.
Le problème est de calculer V et de trouver T_0 tel que, $V(i) = E_i [f(x_{T0})]$.
Si E est fini, un théorème, prouvé par Dynkin et Yushkevich affirme que la solution de ce problème est donnée par le problème de programmation suivant:

$$(1) \; : \left[\begin{array}{l} \min \sum_{i \in E} V(i) \\ \text{sous les contraintes :} \\ V(i) \geq \sum_{j \in E} P(i,j).V(j) \\ V(i) \geq g(i) = \max \{ 0 , f(i) \}. \end{array} \right]$$

Si on note Z = V - g, M = I - P et q = Mg, alors le problème (1) est équivalent au problème:

$$(2) \; : \left[\begin{array}{l} \min \{ \langle e , z \rangle \; / \; z \in D \} \\ \text{où : } D = \{ Z / Z \geq 0 , q + MZ \geq 0 \} \text{ et } e = (1,1, \ldots , 1). \end{array} \right]$$

Le problème (2) est un problème linéaire complémentaire. Ce type de problème a imposé l'étude des problèmes de complémentarité dégénérés.

2.7. Complémentarité et mécaniques.

De nombreux problèmes de mécanique peuvent être transformés en problèmes de complémentarité. Citons par exemple [51]:
- la mécanique structurale
- la mécanique des fluides et les problèmes à frontière libre
- la théorie des plaques minces

2.8. Minimisation d'une fonction concave sur un ensemble convexe.

Soit f: $\mathbb{R}^n \mapsto \mathbb{R}^n$ une fonction continue

On considère le problème de complémentarité suivant :

(P) : Trouver $x \in \mathbb{R}^n$: $x \geq 0$, $f(x) \geq 0$, $\langle x, f(x) \rangle = 0$.

Si on considère la fonction F: $\mathbb{R}^n \mapsto \mathbb{R}$ telle que:

$$F(x) = \sum_{i=1}^{n} \min(x_i , f_i(x))$$

alors le problème (P) est équivalent à un problème d'optimisation non convexe suivant:

(P') : $0 = \min \{F(x) : x \in D\}$, où $D = \{x \in \mathbb{R}^n : x \geq 0, f(x) \geq 0\}$

Dans le cas où les f_i (i=1,...,n) sont d.c., (i.e. différence de deux fonctions convexes) on dit que $f = (f_i)$ est d.c. vectoriellement.

La classe des fonctions d.c. (qui a une structure d'espace vectoriel) est stable par rapport à l'enveloppe supérieure et l'enveloppe inférieure finies [58,59]. On démontre que toute fonction f: $\mathbb{R}^n \mapsto \mathbb{R}^n$ deux fois continûment différentiable est d.c. vectoriellement.

En utilisant les techniques propres à l'optimisation non convexe [57,58], on montre que le problème (P') avec f(x) d.c. vectoriellement est équivalent à un programme convexe avec une contrainte additionnelle anti-convexe.

Le cas où les f_i (i=1,...,n) sont concaves (en particulier linéaires affines), est étudié dans le chapitre IV.

Le problème (P') est alors la minimisation d'une fonction concave (en particulier concave polyédrale) sur un convexe (en particulier sur un polyédre convexe).

2.9. Théorie des jeux [voir 51].

3. PROBLEME LINEAIRE COMPLEMENTAIRE:

NOTATIONS:
P.C désigne le problème complémentaire.
P.L.C. désigne le problème linéaire complémentaire .
P.N.C. désigne le problème non linéaire complémentaire .
P.L . désigne le problème linéaire .

On rappelle que le PLC consiste à trouver un vecteur $x \in \mathbb{R}^n$ tel que:

$$x \geq 0 \; ; \quad Mx + q \geq 0; \quad x^t(Mx + q) = 0 . \qquad (1).$$

où : $M \in \mathbb{R}^{n \times n}$ une matrice carrée d'ordre n et $q \in \mathbb{R}^n$ quelconque.

3.1. Définitions:

Soient $M \in \mathbb{R}^{n \times n}$ et $x \in \mathbb{R}^n$.

1. On dit que M est copositive si et seulement si:

$$\forall\, x \geq 0 \text{ on a } x^t.M.x \geq 0 \qquad (2).$$

2. On dit que M est strictement copositive si et seulement si

$$\forall\, x \geq 0 , x \neq 0 \text{ alors } x^t.M.x > 0 \qquad (3).$$

3. On dit que M est copositive plus si et seulement si elle vérifie les deux conditions suivantes:

(I) M est copositive

(II) $x \geq 0$ et $x^t.M.x = 0 \Rightarrow (M+M^t).x = 0$. (4).

Dans le cas où M est symétrique (ii) $\Leftrightarrow$ ($x \geq 0$ et $x^t.M.x = 0 \Rightarrow M.x = 0$).

4. Soit $x \in \mathbb{R}^n$ un vecteur quelconque .

$$(x)_+ = \max (0 , x) \qquad \text{i.e.,} \qquad (x)_{+i} = \begin{cases} x_i & \text{si } x_i > 0 \\ 0 & \text{sinon} \end{cases}$$

On rappelle les propriétés bien connues suivantes :

* $x = (x)_+ - (-x)_+$
* $(x+y)_+ \leq (x)_+ + (y)_+$.
* $(x)_+ - (y)_+ \leq (x-y)_+$.
* $x \leq y \Rightarrow (x)_+ \leq (y)_+$.
* $|x| = (x)_+ + (-x)_+$.

3.2. METHODE DE PIVOTAGE DE LEMKE:

3.2.1. PROGRAMMATION LINEAIRE:

FORME STANDARD D'UN PROGRAMME LINEAIRE:

Un problème de programmation linéaire consiste à minimiser (ou maximiser) une fonction linéaire sous contraintes linéaires; il s'agit donc d'un programme mathématique de la forme:

$$(P_1) \quad \begin{cases} \min z = \langle c,x \rangle \\ A.x = b \\ x \geq 0 \end{cases}$$

où
n = nombre de variables.
m = nombre de contraintes ; $m \leq n$.
A est une matrice réelle mxn(dite matrice des contraintes) de rang m, $c=(c_1,c_2,.....,c_n)$ est le vecteur-ligne des coûts, $b=(b_1,b_2,.....,b_m)^t$ le m-vecteur des seconds membres et $z = \langle c,x \rangle = \sum c_i.x_i$ est la fonction à minimiser (fonction objective).

BASES, BASES REALISABLES, SOLUTIONS DE BASE:

DEFINITION ([1],[8]):
On appelle base toute sous-matrice carrée régulière (mxm) extraite de A (il en existe au moins une, puisque rang(A)=m).

Soit B une base, en permutant les colonnes de A, on peut toujours mettre A sous la forme A=[B,N] où N est la sous-matrice formée par les colonnes de A qui ne sont pas dans la base; de même on peut partitionner x en $[x_B,x_N]^t$ et c en $[c_B,c_N]^t$.

Toute solution de (P_1) vérifie A.x=b et par suite:

$$B.x_B+N.x_N=b. \qquad (1)$$

On appelle solution de base (associée à la base B), une solution particulière de (1) obtenue en faisant $x_N=0$.
x_B est alors déterminé de façon unique par la résolution du système:

$$B.x_B=b \quad \text{soit} \quad x_B= B^{-1}.b.$$

Une solution de base est dite réalisable si $x_B \geq 0$, autrement dit si:

$$B^{-1}.b \geq 0.$$

Une base correspondante à une solution de base réalisable est appelée base réalisable.
Une solution de base est dite dégénérée si au moins une des composantes du vecteur $x_B= B^{-1}.b$ est nulle.

ALGORITHME DU SIMPLEXE ([1], [8])

L'algorithme du simplexe, découvert en 1947 par G.Dantzig, est le plus connu des algorithmes de résolution d'un programme linéaire. Sa célébrité est due à son efficacité constatée dans de nombreuses applications concrètes. Il fut utilisé pour résoudre des programmes linéaires qui trouvaient leur origine dans des secteurs très variés de la vie économique. Notons d'ailleurs que cette efficacité pratique est mal expliquée aujourd'hui encore.

L'algorithme du simplexe se déroule généralement en deux phases:
La phase (I) de l'algorithme du simplexe consiste à chercher si le système d'équations et d'inéquations définissant les contraintes de (P_1)

$$x \geq 0; \quad A.x=b$$

a une solution réalisable ou non.
Si ce système n'a aucune solution réalisable, l'algorithme est définitivement terminé, sinon la phase (I) consiste à trouver une solution réalisable de ce système.
La phase (II) de l'algorithme du simplexe consiste à résoudre le programme linéaire (P_1) à partir de la solution de base réalisable fournie par la phase (I).

FINITUDE DE L'ALGORITHME DU SIMPLEXE:

On dira que le programme linéaire est non dégénéré si, à chaque itération, les variables de base de la solution de base obtenue à l'entrée sont strictement positives. Ceci implique, qu'à chaque nouvelle itération, le second membre b vérifie b>0.

THEOREME ([8]):

Supposons que le programme linéaire reste non dégénéré à chaque itération du simplexe, alors, l'algorithme du simplexe converge en un nombre fini d'itérations. ◊

3.2.2. POSITION DU PROBLEME PLC:

Le problème linéaire complémentaire peut s'écrire sous cette forme :
Trouver $w \in \mathbb{R}^m$, $z \in \mathbb{R}^m$ tels que:

$$w - Mz = q \qquad (1.1)$$

$$(P_2) \quad w_i \geq 0,\ z_i \geq 0 \quad \text{pour } i=1.....m \qquad (1.2)$$

$$w_i.z_i = 0 \quad \text{pour } i=1.....m \qquad (1.3)$$

où M est une matrice carrée d'ordre m et $q \in \mathbb{R}^m$.

REMARQUES:

- (w_i, z_i) est un couple de composantes complémentaires.
- La composante w_i (resp. z_i) est dite variable de base si $w_i \geq 0$ (resp. $z_i \geq 0$).

- Si w_i (resp z_i) est une variable hors base (non basique) alors forcément $w_i = 0$ (resp. $z_i = 0$).

Pour faciliter la compréhension de ce texte, il est nécessaire d'introduire deux nouvelles définitions.

DEFINITIONS :

Une solution (w^*,z^*) du système **(1.1)**,**(1.2)** et **(1.3)** est dite **solution de base réalisable complémentaire** si:

1) (w^*,z^*) est une solution de base réalisable de **(1.1)** et **(1.2)**.

2) une composante du couple (w_i,z_i) est une variable de base pour $0 \leq i \leq m$.

Si q est non-négatif, alors on a immédiatement la solution du problème $(\mathbf{P_2})$ en prenant $w=q$ et $z=0$.

Dans le cas où il existe un i tel que $q_i<0$, on introduit un nouveau vecteur colonne **1** dont toutes les composantes sont égales à 1, et une variable artificielle z_0 pour définir un nouveau système. On le définit par:

	$w-Mz-\mathbf{1}z_0 = q$		**(1.4)**
$(\mathbf{P'_2})$	$w_i \geq 0, z_i \geq 0, z_0 \geq 0$	pour i=1.....m	**(1.5)**
	$w_i.z_i = 0$	pour i=1.....m	**(1.6)**

où $z_0 = \max\{-q_i ;\ 1 \leq i \leq m\}$, $z=0$ et $w=q+\mathbf{1}z_0$.

On obtient ainsi une solution du système **(1.4)**, **(1.5)** et **(1.6)** ci-dessus.

DEFINITION :

Une solution du système **(1.4)**, **(1.5)** et **(1.6)** est dite **solution de base réalisable presque complémentaire** si:

1) (w,z,z_0) est une solution de base réalisable de **(1.4)** et **(1.5)**.

2) Ni w_s ni z_s est une variable de base pour un certain $s \in \{1.....m\}$.

3) z_0 est une variable de base, et on a soit w_i soit z_i qui est une variable de base pour tout $i \in \{1.....m\}$ tel que $i \neq s$.

En faisant entrer w_s ou z_s dans la base, on obtient une solution de base réalisable adjacente presque complémentaire. Nous constatons donc que chaque solution de base réalisable presque complémentaire admet deux solutions adjacentes, l'une en entrant w_s dans la base et l'autre en entrant z_s dans la base.

Le plus ancien algorithme proposé pour la résolution du problème PLC fut la Méthode de Pivotage Principal proposé par Cottle et Dantzig **[2]**, suivi par la suite d'un algorithme dû à Lemke **[4]** qui utilise des pivots complémentaires.

DESCRIPTION DE L'ALGORITHME :

Initialisation:

Si $q \geq 0$, stop; $(w,z)=(q,0)$ est une solution de base réalisable complémentaire.

Sinon, (i.e., il existe $i \in \{1....m\}$ tel que $q_i<0$), nous présentons les problèmes (**1.4**) et (**1.5**) sous forme d'un tableau (pareil au tableau du simplexe). Soit $q_s=\min\{q_i : 1 \leq i \leq m\}$, on met à jour le tableau par le pivotage de la ligne s et la colonne z_0, w_s quitte la base et z_0 rentre en base, ainsi z_0 et w_i pour $i=1....m$ avec $i \neq s$ sont non-négatifs. Posons $y_s=z_s$ et passons à l'étape principale.

Etape principale:

Cette étape se compose de quatre phases:

1) Soit d_s la colonne qui correspond à la variable y_s dans le tableau courant. Si $d_s \leq 0$, on va en **4**. Sinon, on détermine un indice r tel que:

$$\frac{q_r^*}{d_{s_r}}=\min\{\frac{q_i^*}{d_{s_i}} \quad ; \quad d_{s_i}>0 \; ; \; 1 \leq i \leq m\}$$

Le vecteur q^* définit la colonne du second membre (noté dans le tableau par s.m) après la mise à jour du tableau.

Si la variable de base à la ligne r est z_0, faire **3)** sinon faire **2.**

2) La variable de base à la ligne r est, soit w_k, soit z_k pour un certain $k \neq s$. La variable y_s rentre en base et le tableau se définira par le pivotage de la ligne r et de la colonne de y_s. Si la variable, qui a quitté la base, est w_k (resp. z_k), on pose $y_s=z_k$(resp w_k) et on retourne en **1**.

3) On fait le pivotage entre la colonne y_s (c'est a dire z_s) et la ligne z_0. Ainsi la variable z_s rentre en base, et z_0 quitte la base. On obtient une solution de base réalisable complémentaire.

4) On s'arrête avec un rayon de termination. Le rayon $\mathbf{R}=\{(w,z,z_0)+\lambda d; \quad \lambda \geq 0\}$ est déterminé tel que tout point de **R** satisfait au système (**1.4**), (**1.5**) et (**1.6**) .

(w,z,z_0) est une solution de base réalisable presque complémentaire et d est une direction extrémale dont les composantes sont déterminées ainsi:

1	pour la ligne correspondante à y_s
$-d_s$	pour les lignes correspondantes aux variables de base
0	ailleurs.

L'algorithme se termine ainsi.

EXEMPLE (CONVERGENCE VERS UNE SOLUTION REALISABLE)

On veut déterminer une solution du problème linéaire complémentaire PLC défini par:

$$M=\begin{bmatrix} 0 & 0 & -1 & -1 \\ 0 & 0 & 1 & -2 \\ 1 & -1 & 2 & -2 \\ 1 & 2 & -2 & 4 \end{bmatrix} \qquad q=\begin{bmatrix} 2 \\ 2 \\ -2 \\ -6 \end{bmatrix}$$

Etape d'initialisation : On introduit la variable auxiliaire z_0 et on construit le tableau suivant:

	w_1	w_2	w_3	w_4	z_1	z_2	z_3	z_4	z_0	s.m
w_1	1	0	0	0	0	0	1	1	-1	2
w_2	0	1	0	0	0	0	-1	2	-1	2
w_3	0	0	1	0	-1	1	-2	2	-1	-2
w_4	0	0	0	1	-1	-2	2	-4	(-1)	-6

On voit que min $\{q_i : 1 \le i \le 4\} = q_4$. On entoure donc l'élément qui se trouve à l'intersection de la colonne z_0 et de la quatrième ligne dans laquelle figurait la variable w_4. Cet élément s'appelle le pivot. On fait donc le pivotage entre la ligne 4 et la colonne z_0. On passe à l'itération 1 avec $y_s=z_4$.

Itération 1:

	w_1	w_2	w_3	w_4	z_1	z_2	z_3	z_4	z_0	s.m
w_1	1	0	0	-1	1	2	-1	5	0	8
w_2	0	1	0	-1	1	2	-3	6	0	8
w_3	0	0	1	-1	0	3	-4	(6)	0	4
z_0	0	0	0	-1	1	2	-2	4	1	6

$y_s=z_4$ rentre dans la base. En calculant le quotient défini dans la phase 1 de l'étape principale de l'algorithme, son minimum est atteint pour l'indice 3. Ainsi w_3 quitte la base et on pose $y_s=z_3$. On fait le pivotage entre la ligne w_3 et la colonne z_4. On passe à l'itération 2.

Itération 2:

	w_1	w_2	w_3	w_4	z_1	z_2	z_3	z_4	z_0	s.m
w_1	1	0	-5/6	-1/6	1	-1/2	(7/3)	0	0	14/3
w_2	0	1	-1	0	1	-1	1	0	0	4
z_4	0	0	1/6	-1/6	0	1/2	-2/3	1	0	2/3
z_0	0	0	-2/3	-1/3	1	0	2/3	0	1	10/3

$y_s = z_3$ rentre dans la base. En calculant le quotient défini dans la phase 1 de l'étape principale de l'algorithme, son minimum est atteint pour l'indice 1 ainsi w_1 quitte la base et z_3 y rentre et on pose $y_s=z_1$ et on passe à:

Itération 3

	w_1	w_2	w_3	w_4	z_1	z_2	z_3	z_4	z_0	s.m
z_3	3/7	0	-5/14	-1/14	3/7	-3/14	1	0	0	2
w_2	-3/7	1	-9/14	-1/14	4/7	-11/14	0	0	0	2
z_4	2/7	0	-1/14	-3/14	2/7	5/14	0	1	0	2
z_0	-2/7	0	-3/7	-2/7	(5/7)	1/7	0	0	1	2

$y_s = z_1$ rentre dans la base. En calculant le quotient défini dans la phase 1 de l'étape principale de l'algorithme, son minimum est atteint pour l'indice correspondant à la ligne z_0 ainsi z_0 quitte la base et z_1 y rentre. On obtient ainsi une solution de base réalisable complémentaire donnée par:

	w_1	w_2	w_3	w_4	z_1	z_2	z_3	z_4	z_0	s.m
z_3	3/5	0	-1/10	1/10	0	-3/10	1	0	-3/5	4/5
w_2	-1/5	1	-3/10	3/10	0	-9/10	0	0	-4/5	2/5
z_4	2/5	0	1/10	-1/10	0	3/10	0	1	-2/5	6/5
z_1	-2/5	0	-3/5	-2/5	1	1/5	0	0	7/5	14/5

En résumé, l'algorithme de pivotage complémentaire produit le point
$(w_1, w_2, w_3, w_4, z_1, z_2, z_3, z_4)=(0, 2/5, 0, 0, 14/5, 0, 4/5, 6/5)$

3.2.3. CONDITIONS DE CONVERGENCE D'UNE SOLUTION REALISABLE A BASE COMPLEMENTAIRE .

LEMMES DE CONVERGENCE:

Lemme 1 [1]:

On suppose que chaque solution presque réalisable, à base complémentaire du système formé par (4),(5),(6), est non dégénérée (i.e. chaque variable de base est strictement positive). Alors, aucun point généré par cet algorithme de pivotage ne se répète, de plus, cet algorithme s'arrête au bout d'un nombre fini d'étapes . ◊

Lemme 2:

On suppose que chaque solution presque réalisable, à base complémentaire du système définie par (4),(5) et (6) est non dégénérée, et on suppose que l'algorithme de pivotage complémentaire se termine avec un rayon de termination :

$\mathbf{R = \{ (w^*, z^*, z_0^*) + p(w^\wedge, z^\wedge, z_0^\wedge);\quad p \geq 0 \}}$.

avec, (w^*,z^*,z_0^*) une solution presque réalisable à base complémentaire et $(w^\wedge,z^\wedge,z_0^\wedge)$ une direction extremale . Alors:

1. $(w^\wedge,z^\wedge,z_0^\wedge) \neq (0,0,0)$ $w^\wedge,z^\wedge \geq 0$ et $z_0^\wedge \geq 0$.
2. $w^\wedge - M.z^\wedge - \mathbf{1}.z_0^\wedge = 0$.
3. $w^{*t}.z^* = w^{*t}.z^\wedge = w^{\wedge t}.z^* = w^{\wedge t}.z^\wedge = 0$.
4. $z^\wedge \neq 0$.
5. $z^{\wedge t}.M.z^\wedge = -\mathbf{1}^t.z^\wedge.z_0^\wedge \leq 0$.

Preuve:

1. et **2.** sont évidents d'après la définition de la direction extrême.
Puisque chaque point de **R** vérifie l'équation (6), alors:

$$(w^* + p.w^\wedge)^t.(z^* + p.z^\wedge) = 0 \quad \forall p \geq 0$$

et puisque $w^*,w^\wedge,z^*,z^\wedge$ sont ≥ 0, alors, on obtient:

$w^{*t}.z = w^{*t}.z^\wedge = w^{\wedge t}.z^* = w^{\wedge t}.z^\wedge = 0$. D'ou (**3.**).

On suppose par l'absurde que $z^\wedge=0$, alors $z_0^\wedge=0$ car sinon puisque $z_0^\wedge \geq 0$ alors, $z_0^\wedge = 0$. Or $w^\wedge - M.z^\wedge - \mathbf{1}.z_0^\wedge = 0 \Rightarrow w^\wedge = 0 \Rightarrow (w^\wedge,z^\wedge,z_0^\wedge) = (0,0,0)$ contradiction avec **1;** on constate que $z_0^\wedge > 0$.

Puisqu'on a supposé que $z^\wedge=0$ alors $w^\wedge-\mathbf{1}.z_0^\wedge=0$, par suite $w^\wedge=\mathbf{1}.z_0^\wedge > 0$.

Or $w^{\wedge t}.z^* = 0 \Rightarrow z_0^\wedge.\mathbf{1}^t.z^* = 0 \Rightarrow \mathbf{1}^t.z^* = 0 \Rightarrow z^* = 0 \Rightarrow z^*$ est non basique $\Rightarrow z_0^*$ est basique. Puisqu' on a p éléments dans la base, (p-1) éléments basiques sont des composantes de w^* et le $p^{ième}$ élément est z_0^*.

Ceci implique que (w^*,z^*,z_0^*) est la première solution, ce qui est impossible d'après le Lemme 1: aucune solution ne se répète et il est impossible d'obtenir un rayon de termination dans la première solution. Donc on constate que $z^\wedge \neq 0$ d'ou (4.).

$w^\wedge - M.z^\wedge - 1.z_0^\wedge = 0 \Rightarrow z^{\wedge t}.(w^\wedge - M.z^\wedge - 1.z_0^\wedge) = 0 \Rightarrow$

$z^{\wedge t}.w^\wedge - z^{\wedge t}.M.z^\wedge - z^{\wedge t}.1.z_0^\wedge = 0$ or $z^{\wedge t}.w^\wedge = 0$ d'où:

$z^{\wedge t}.M.z^\wedge = -z^{\wedge t}.1.z_0^\wedge \leq 0$ car $z^\wedge \geq 0$ et $z_0^\wedge \geq 0$ donc $z^{\wedge t}.M.z^\wedge \leq 0$. ◊

Théorème de Farkas [1] :

Soit A une matrice (m,n) et c un n-vecteur .

Si l'un de ces deux systèmes suivants admet une solution, alors, l'autre n'admet aucune solution .

Système 1: $A.x \leq 0$ et $c^t.x > 0$ avec x un n-vecteur.

Système 2: $A^t.y = c$ et $y \geq 0$ avec y un m-vecteur. ◊

THEOREME DE CONVERGENCE

Théorème [1]:

On suppose que chaque solution presque réalisable, à base complémentaire du système défini par (4),(5) et (6), est non dégénérée; et on suppose que la matrice M est copositive-plus . Alors, l'algorithme s'arrête au bout d'un nombre fini d'étapes .

Si le système défini par (1) et (2) est consistent, alors, l'algorithme s'arrête avec une solution réalisable à base complémentaire du système formé par (1),(2),(3) (i.e. du problème initial (P_2)); sinon, si le système défini par (1) et (2) est inconsistent, alors on obtient un rayon de termination .◊

Corollaire:

Si la matrice M admet des éléments positifs et si les éléments diagonaux sont strictement positifs, alors l'algorithme s'arrête avec une solution réalisable à base complémentaire .

(On prend z assez grand; $w = M.z + q \geq 0$ avec $z \geq 0$) . ◊

3.3. LA METHODE DE NEWTON POUR RESOUDRE UN PLC.

3.3.1. Notations :

Soit M une Matrice quelconque . On dit que M est :

Z-Matrice ($M \in \mathbf{Z}$) si $M_{ij} \leq 0 \quad \forall i \neq j$.

P-Matrice ($M \in \mathbf{P}$) si tous les mineurs principaux sont ≥ 0.

K-Matrice ($M \in \mathbf{K}$) si elle vérifie une de ces trois conditions équivalentes :

(i) $M \in \mathbf{Z}$ et $M^{-1} \geq 0$.

(ii) $M \in \mathbf{Z}$ et tous les mineurs principaux sont ≥ 0 .

(iii) $M \in \mathbf{Z}$ et il existe un vecteur $r \geq 0$; $r^t.M > 0$.

Remarque 1 : $M \in \mathbf{K} \Leftrightarrow M \in \mathbf{P} \cap \mathbf{Z}$.

Si $a \in \mathbb{R}^n$, alors A = Diag(a) est une matrice diagonale dans $\mathbb{R}^{nxn}$ telle que : $A_{ii} = a_i$, i = 1, ..., n. Inversement, si $A \in \mathbb{R}^{nxn}$ alors a = diag(A) est un vecteur dans $\mathbb{R}^n$ tel que : $a_i = A_{ii}$, i = 1, ..., n.

Soient a et b deux vecteurs dans $\mathbb{R}^n$,

$$x = \min(a,b) \Leftrightarrow \begin{cases} x_i = a_i & \text{si } a_i \leq b_i \\ x_i = b_i & \text{sinon} \end{cases}$$

3.3.2. Fondement de la méthode

Définition 1 :

On dit que la Matrice M est de classe $\mathfrak{C}$ ($M \in \mathfrak{C}$) si et seulement si:

Il existe deux **Z-Matrices** X et Y, et deux vecteurs r,s ≥ 0 tels que :

$$M.X = Y \qquad (1.1)$$

$$r^t.X + s^t.Y > 0 \qquad (1.2)$$

cette classe $\mathfrak{C}$ est découverte par Mangasarian **[19,20]**.

Théorème 1 :

Si $M \in \mathfrak{C}$ alors P.L.C (1) est équivalent au programme linéaire suivant :

$$\min \{p^t.z / z \geq 0,\ M.z+q \geq 0\} \qquad (2')$$

où $p = r + M^t.s$.

(2') admet comme programme dual :

$$\max \{ -q^t.y / y \geq 0 ;\ p - M^t.y \geq 0 \} \qquad (3')$$

Lemme :
Soient z une solution du P.L.primal (2') et y une solution du P.L.dual (3'). Si $(I-M^t).y + p > 0$ alors z est solution du P.L.C (1).

Preuve : $y^t.(M.z + q) + z^t.(p-M^t.y) = y^t.q + z^t.p = 0$.
Or $y \geq 0$, $M.z+q \geq 0$, $z \geq 0$, $p-M^t.y \geq 0$. D'où : $y^t(Mz+q) = 0$ et $z^t.(p-M^t.y) = 0$
i.e., $y_i.(M.z + q)_i = 0$ et $z_i.(p-M^t.y)_i = 0 \quad \forall\, i = 1,...,n$.
Or $(I-M^t).y + p > 0 \Rightarrow y_i + (p-M^t.y)_i > 0 \ \forall\, i =1,..,n \Rightarrow$
$y_i > 0$ ou $(P-M^t.y)_i > 0 \ \forall\, i =1,..,n \Rightarrow (M.z + q)_i = 0$ ou $z_i = 0 \ \forall\, i =1,..,n$
et par suite z résout (1) . ◊

Démonstration du théorème 1 :
$r^t.X + s^t.Y > 0 \Rightarrow r^t.X + s^t.M.X > 0 \Rightarrow (r+M^t.s)^t.X > 0 \Rightarrow p^t.X > 0 \Rightarrow$
$p^t.X + y^t.(-M.X + Y) > 0 \Rightarrow (p^t-y^t.M).X + y^t.Y > 0.$
On pose : X = D - U et Y = D - V, où D est une matrice diagonale positive ; U et V sont deux matrices non négatives (≥0). On obtient :
$(p^t-y^t.M).(D - U) + y^t.(D - V) > 0 \Rightarrow (p^t-y^t.M + y^t).D - [(p^t-y^t.M).U + y^t.V] > 0$
$\Rightarrow (p^t-y^t.M + y^t).D > 0 \Rightarrow [(I-M^t).y + p]^t.D > 0$.
Or D > 0 d'où : $(I-M^t).y + p > 0$, tel que, d'après le Lemme précédent, z résolve le P.L.C (1) .◊

Théorème 2 [19,20] :
Si $S(q,M) = \{ z \geq 0$ et $M.z+q \geq 0 \}$ est non vide alors dans chacun des cas suivants le P.L.C (1) admet une solution qui est celle du P.L. (2'):

1) $M=Z_2.Z_1^{-1}$ avec $Z_1 \in \mathbf{K}$ et $Z_2 \in \mathbf{Z}$. $(p=r \geq 0 ; s=0; r^t.Z_1>0)$
2) $M=Z_2.Z_1^{-1}$ avec $Z_1 \in \mathbf{Z}$ et $Z_2 \in \mathbf{K}$. $(p=M^t.s ; s\geq 0; s^t.Z_2>0)$
3) $M \in \mathbf{Z}$. ($Z_1 = I$, $Z_2 = M$, r=e, et on applique 1)) .
4) $M^{-1} \in \mathbf{Z}$. ($Z_1 = M^{-1}$, $Z_2 = I$, s=e, $p=M^te$ et on applique 2)) . ◊

Remarque 2 :
Si $M \in \mathbf{K}$ alors il est évident que $S(q,M) \neq \emptyset$.[car si on prend a=Max(0,q), alors $z= M^{-1}(a-q) \geq 0 \in S(q,M)$] . ◊

Lemme 1 [19,20]:
Si $S(q,M) = \{z \in \mathbb{R}^n / \ z \geq 0$ et $M.z+q \geq 0\}$ est non vide alors on obtient le résultat suivant :
z* est le plus petit élément de S(q,M) ⇔ z* est une solution du P.L.(2'). ◊

Théorème 3 [38] :
Si $M \in \mathbf{Z}$ et $S(q,M) = \{z \in \mathbb{R}^n / z \geq 0 \text{ et } M.z+q \geq 0\}$ est non vide alors:
S(q,M) admet un plus petit élément qui résout le P.L.C.(1). ◊

Théorème 4 [35,39] :
Si M est une matrice carrée, symétrique, semi-définie positive et si $M \in \mathfrak{C}$ telle que $S(q,M) \neq \emptyset$, alors le problème de la programmation quadratique :

$$\min \{1/2\langle z,Mz\rangle + \langle q,z\rangle : z \geq 0\}$$

admet une solution qui est une solution du P.L.(2'). (avec $p = r+M^t.s$). ◊

Théorème 5 [30] :
Si $M \in \mathfrak{C}$, alors le P.L.C.(1) est équivalent à résoudre le problème suivant : trouver $x \in \mathbb{R}^n$ tel que :

$$X.x \geq 0 \ ; \ Y.x+q \geq 0 \ ; \ (Y.x+q)^t.X.x = 0 \qquad (4')$$

et, de plus, si x résout (4'), alors $z=X.x$ résout (1). ◊

Théorème 6 [30] :
Soit $S(q,X,Y) = \{x \in \mathbb{R}^n / X.x \geq 0, Y.x+q \geq 0\}$ est non vide et admet un plus petit élément x* ; alors $z^*=X.x^*$ résout le P.L.C.(1). ◊

Lemme 2 [23] : Si $A \in \mathbb{R}^{n \times n}$ et $D \in \mathbb{R}^{n \times n}$ une matrice diagonale à éléments diagonaux positifs, alors les propositions suivantes sont équivalentes :

1) $A \in \mathbf{P}$
2) $(I-\lambda).D + \lambda A \in \mathbf{P}$ avec $0 \leq \lambda \leq I$ (I= matrice d'identité de $\mathbb{R}^{n \times n}$).
3) $(I-\lambda).D + \lambda A$ est inversible . ◊

Remarque 3 :
Si on associe au problème (4') la fonction F de $\mathbb{R}^n \mapsto \mathbb{R}^n$ définie par : $F(x) = \min(X.x, Y.x+q)$, alors x* résout (4') $\Leftrightarrow F(x^*)=0$.
Ainsi ce problème (4') se ramène à trouver un zéro de F (qui est concave polyèdrale).

Définitions [23]:
1- Soit $X \in \mathbf{Z}$. On dit que la décomposition $X = U-V$ est une décomposition **K**-régulière si : $U \in \mathbf{K}$ et $V \geq 0$.
2- Deux matrices X et Y sont dites **K**-régulières communes si:

$$(I-\lambda).X + \lambda.Y \in \mathbf{K} \quad \forall\ 0 \leq \lambda \leq I$$

3- Soient : $X=U-V$; $U \in \mathbf{K}$ et $V \geq 0$.

$Y=H-G$; $H \in \mathbf{K}$ et $G \geq 0$. deux décompositions **K**-régulières.

On dit que ces deux décompositions sont **K**-régulières communes si et seulement si : U et H sont deux matrices **K**-régulières communes .

Lemme 3 : Si $M \in \mathbb{P}$ et si X et Y sont les deux matrices qui vérifient : $M.X=Y$, $r^tX + s^tY > 0$ où r et s sont deux vecteurs nonnégatifs, alors les deux propriétés suivantes sont équivalentes:

1) M est **P-Matrice** .
2) X et Y sont **K**-régulières communes .

Preuve : voir Pang **[34,Théorème1]** et Lemme 2. ◊

Lemme 4.1:

Soit $X \in \mathbf{Z}$, et $0<d \in \mathbb{R}^n$ tel que $d \geq \text{diag}(X)$ et soit $U=\text{Diag}(d)$, alors : $X=U-V$ est une décomposition **K**-régulière . ◊

Lemme 4.2 :

Soient $X=U-V$ et $Y=H-G$ deux décompositions **K**-régulières de deux **Z-Matrices** X et Y . U est une matrice diagonale à éléments positifs, alors ces deux décompositions sont **K**-régulières communes. ◊

Lemme 4.3 [23] :

Soient $X=U-V$ et $Y=H-G$ deux décompositions **K**-régulières de deux **Z-Matrices** X et Y.S'il existe une matrice diagonale $D \geq 0$ telle que $U \geq D.H$, alors ces deux décompositions sont **K**-régulières communes . ◊

Sur - Gradient de F :

Une matrice $D \in \mathbb{R}^{n \times n}$ est appelée sur-gradient d'une fonction concave f si : $f(y) \leq f(x)+D.(y-x) \quad \forall y \in \mathbb{R}^n$.

Lemme 5.1 :

D une matrice sur-gradient de F $\Leftrightarrow$ $D=(I-\lambda).X + \lambda.Y$, où $\lambda \in \mathbb{R}^{n \times n}$ une matrice diagonale avec :

$$\lambda_{ii} = \begin{cases} 0 & \text{Si } F_i(x) < X_i\, x \\ \mu : 0 \leq \mu \leq 1 & \text{Si } F_i(x) = X_i\, x = Y_i\, x + q_i \\ 1 & \text{ailleurs.} \end{cases} \qquad i=1,.....,n.$$

Preuve : voir Eaves **[37, Lemma 4.3]**. ◊

Soit $x \mapsto D(x)$ une application sur-gradient i.e., pour chaque $x \in \mathbb{R}^n$.
On associe l'unique sur-gradient : $D(x) = (I-\lambda_X).X + \lambda_X.Y$

et soit : $T: \mathbb{R}^n \mapsto \mathbb{R}^n$

$$x \mapsto T(x) = (I-\lambda_X).U + \lambda_X.H .$$

On a $T(x) \in \mathbf{K}$ $\forall x \in \mathbb{R}^n$, et en plus : $[T(x)]^{-1}.D(x) \leq I$ et $D(x).[T(x)]^{-1} \leq I$.

Algorithme de base [30]:

Soient X et Y deux **Z-Matrices** qui vérifient la condition (1.2), alors il existe un réarrangement de partitionnement de X et Y :

$$P^t.X.P = \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix} \quad \text{et} \quad P^t.Y.P = \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix}$$

où P est une matrice de permutation telle que :

$$\begin{bmatrix} X_{11} & X_{12} \\ Y_{21} & Y_{22} \end{bmatrix} \in \mathbf{K}$$

$$\text{Soit} \quad P^t.q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} \quad \text{et} \quad x = \begin{bmatrix} X_{11} & X_{12} \\ Y_{21} & Y_{22} \end{bmatrix}^{-1} . \begin{bmatrix} 0 \\ q_2 \end{bmatrix}$$

alors, x est la borne inférieure de S(q,X,Y) et $F(x) \leq 0$. ◊

Théorème 7 [23,Théorème 6.1] :
Soient X et Y les deux **Z-Matrices** vérifiant la condition (1.2) et X=U-V, Y=H-G deux décompositions **K**-régulières communes.
On suppose que S(q,X,Y) est non vide, alors, l'itération :

$$x^0=x \; ; \; x^{k+1} = x^k - [T(x^k)]^{-1}.F(x^k) \qquad k=0,1,..... \qquad (6.1)$$

est une généralisation d'une suite monotone $\{x^k\}$ qui converge vers une solution x^* du problème (4) . De plus, si y^* est une autre solution du (4), alors, $x^* \leq y^*$. ◊

Théorème 8 [23,Théorème 6.2] :
Si les matrices X et Y sont **K**-régulières communes, alors, il existe une matrice diagonale λ; $0 \leq \lambda \leq I$ et $(I-\lambda).X + \lambda.Y \in \mathbf{K}$
et soit : $D(x) = (I-\lambda_x).X + \lambda_x.Y$. (6.2) tel que:

$$(\lambda_x)_{ii} = \begin{cases} 1 & \text{Si } F_i(x) < X_i\, x \\ 0 & \text{Sinon} \end{cases} \qquad (6.3)$$

alors l'itération :

$$x^0 = 0 \; ; \; x^{k+1} = x^k - D(x^k)]^{-1}.F(x^k) \qquad k=0,1,..... \qquad (6.4)$$

généralise une suite croissante monotone finie $\{x^k\}$ qui converge vers une solution unique x^* du problème (4) . ◊

Remarque :
Si $M \in \mathbf{K}$, alors l'itération (6.4) avec M=Y et X=I est équivalente à l'algorithme de **CHANDRASEKARIAN [24] :**

Itération 0 : $x^0 = 0$; $k = 0$;

Itération (k+1) : $k = 0,1,2,.....$
1. Calculer $d^{k+1} = F(x^k)$.
Si $d^{k+1} = 0$ on termine avec une solution x^k .
Sinon:
(i) Résolvons $D(x^k).y = d^{k+1}$.
(ii) $x^{k+1} \longleftarrow x^k - y$.
(iii) $k \longleftarrow k+1$.
Retournons à 1.
(Si $M \in \mathbf{Z-P}$ le problème n'admet pas toujours de solutions car $D(x^k)$ peut être non inversible et par suite $D(x^k).y = d^{k+1}$ n'admet pas de solutions).

3.4. Méthodes itératives pour la résolution du P.L.C :

3.4.1. Introduction:
On résout le problème PLC (1) par des méthodes itératives en utilisant:

- la fonction quadratique $f(x) = 1/2x^t.M.x + q^t.x$
- une suite $\{x^i\}$ générée par l'algorithme 2.1 (ci-dessous).

sous certaines conditions, sur la matrice M, on montre que chaque valeur d'adhérence de cette suite $\{x^i\}$ résout le P.L.C (1).

3.4.2. Partie A :

On suppose dans cette partie que la matrice M est symétrique .

Algorithme 2.1 : Soit $x^0 \geq 0$ un vecteur arbitrairement choisi . On définit la suite $\{x^i\}$ par la formule de recurrence suivante :

$$x^{k+1} = \lambda.[x^k - \omega.E^k.(M.x^k + q + K^k(x^{k+1}-x^k))]_+ + (1- \lambda).x^k \quad (5) .$$

où : λ,ω sont deux réels positifs (les paramètres de relaxation) tels que :
$\omega>0$ et $0\leq \lambda \leq 1$ (on prend $0\leq \lambda \leq 1$ pour que $x^i\geq 0\ \forall i$)
$\{E^i\}$ et $\{K^i\}$ sont deux suites de matrices bornées telles que :
E^i est une matrice diagonale >0 (dont les termes sont strictement positifs) et K^i est une matrice triangulaire inférieure ou supérieure qui vérifie la propriété suivante :

$$\exists\ \gamma\geq 0\ ;\ y^t.[(\lambda.\omega.E^i)^{-1}+ K^i -M/2].y \geq \gamma.\|y\|^2 \quad \forall y\in \mathbb{R}^n\ ;\ y\geq 0 . \qquad (6').$$

Lemme 2.1:

Soient $\omega>0$ et E une matrice diagonale >0 de terme général e_{ij}, alors :

$$x \text{ résout le P.L.C} \Leftrightarrow [\,x - \omega.E(M.x + q)]_+ - x = 0.$$

Preuve :

Condition nécessaire:

x résout le P.L.C $\Rightarrow x\geq 0\ ;\ M.x + q \geq 0\ ;\ x^t.(M.x + q) = 0$

$\Rightarrow x_i\geq 0\ ;\ (M.x + q)_i \geq 0$ et $x_i.(M.x + q)_i = 0 \quad \forall\, i =1,..,n$

$\Rightarrow [x_i \geq 0$ et $(M.x + q)_i = 0]$ ou $[x_i = 0$ et $(M.x + q)_i \geq 0] \quad \forall\, i =1,..,n$

Si $[x_i \geq 0$ et $(M.x + q)_i = 0]$ alors $[\,x_i - \omega.e_{ii}(M.x + q)_i]_+ - x_i = (x_i)_+ - x_i = x_i - x_i = 0,$

sinon, $[\,x_i - \omega.\, e_{ii}\,(M.x + q)_i]_+ - x_i = [\,-\omega.\, e_{ii}\,(M.x + q)_i]_+ = 0$

donc dans tous les cas $[\,x - \omega.E(M.x + q)]_+ - x = 0$.

Condition suffisante:

$x = [x -\omega.E(M.x + q)]_+ \Rightarrow x \geq 0$. Ainsi $M.x + q \geq 0$ car sinon il existe au moins un indice i tel que $(M.x + q)_i < 0$ et par suite :

$0 = [\,x_i - \omega.e_{ii}\,(M.x + q)_i]_+ - x_i = -\omega.e_{ii}\,(M.x + q)_i > 0$ ce qui est absurde .

On considère deux cas :

1ère cas : Si $[\,x_i - \omega.\, e_{ii}\,(M.x + q)_i] \leq 0$ alors $[\,x_i - \omega.\, e_{ii}\,(M.x + q)_i]_+ = 0$ d'où $x_i=0$

2ème cas : Si $[x_i - \omega . e_{ii} (M.x + q)_i] > 0$ alors :

$[x_i - \omega . e_{ii} (M.x+q)_i]_+ = x_i - \omega . e_{ii} (M.x+q)_i \Rightarrow - \omega . e_{ii} (M.x+q)_i = 0 \Rightarrow (M.x+q)_i = 0.$

Donc dans tous les cas on a : $x^t.(M.x + q) = 0$ d'où x est une solution du P.L.C (1). ◊

Lemme 2.2 [22] :

Soient E une matrice diagonale >0 et $x \in \mathbb{R}^n$ alors :

$[(x)_+ - x]^t.E^{-1}.((x)_+ - y) \leq 0 \quad \forall y \geq 0 ; y \in \mathbb{R}^n$. ◊

Théorème 2.1 :

Chaque valeur d'adhérence de la suite $\{x^i\}$ générée par l'algorithme 2.1 résout le P.L.C (1). ◊

Preuve :

En utilisant la fonction quadratique $f(x) = 1/2x^t.M.x + q^t.x$, la formule de récurrence et (6'), on obtient :

$f(x^i) - f(x^{i+1}) \geq \gamma . \| x^i - x^{i+1} \| \geq 0$, alors la fonction f est décroissante .Soit x* une valeur d'adhérence de la suite $\{x^i\}$ alors il existe une sous-suite $\{x^{i_k}\}$ qui converge vers x*. On pose $\lim E^{i_k} = E$, $\lim K^{i_k} = K$.

La fonction f est bornée. Sinon, il existerait deux indices i_k et k* tels que :

$f(x^{i_k}) > f(x^*) \geq f(x^{k^*}) \geq f(x^{i_k})$ ce qui est absurde.

Donc f est une fonction bornée et décroissante, par suite $\{f(x^{i_k})\}$ converge et comme f est continue alors $\lim f(x^{i_k}) = f(x^*)$ et on a : $0 = \lim [f(x^{i_k}) - f(x^{i_k+1})] \geq \gamma . \lim \|x^{i_k+1} - x^{i_k}\|$.

D'où : $\lim x^{i_k+1} = \lim x^{i_k} = x^*$ et par suite $x^* = \lambda.[x^* - w.E.(Mx^*+q)]_+ + (1-\lambda)x^*$.

On obtient : $[x^* - w.E.(M.x^* + q)]_+ - x^* = 0$ tel que d'aprés le lemme 2.1 x* résoud le P.L.C.

Lemme 2.3 [22] :

Soit M une matrice symétrique, copositive et la suite $\{x^i\}$ non bornée ; $x^i \geq 0$ et $f(x^i) \leq c \quad \forall i = 1, \ldots$ où $f(x) = 1/2x^t.M.x + q^t.x$ et c = constante alors :

A) $\exists$ une sous suite $\{x^{i_k}\}$ tel que la suite $\left\{ y^{i_k} = \dfrac{x^{i_k}}{\|x^{i_k}\|} \right\}$ converge vers y^* qui vérifie :

$y^* \geq \neq 0$; $y^{*t}.M.y^* = 0$; $q^t.y^* \leq 0$.

B) Si M est copositive plus, alors : $y^* \geq \neq 0$; $M.y^* = 0$; $q^t.y^* \leq 0$.

C) Si M est copositive plus, alors: $M.x > 0$ et $Mx+q > 0$ n'ont pas de solutions.

Preuve:

A) Soit $\{ x^{i_k} \}$ est la sous suite de $\{ x^i \}$ tel que : $x^{i_k} >\neq 0$ et Lim $\|x^{i_k}\| = +\infty$.

On pose $y^{i_k} = \frac{x^{i_k}}{\|x^{i_k}\|}$ alors $y^{i_k} >\neq 0$ ($y^{i_k} \in \{ y \in \mathbb{R}^n / \|y\| = 1 \}$) et parsuite $y^* >\neq 0$.

$$\frac{c}{\|x^{i_k}\|^2} \geq \frac{f(x^{i_k})}{\|x^{i_k}\|^2} = \frac{1}{2} (y^{i_k})^t.M.y^{i_k} + \frac{q^t.y^{i_k}}{\|x^{i_k}\|} \Rightarrow 0 \geq \frac{1}{2} y^{*t}.M.y^* \geq 0 \Rightarrow y^{*t}.M.y^* = 0$$

$$\text{De même : } \frac{c}{\|x^{i_k}\|} \geq \frac{f(x^{i_k})}{\|x^{i_k}\|} = \frac{1}{2}\|x^{i_k}\|.(y^{i_k})^t.M.y^{i_k} + q^t.y^{i_k} \geq q^t.y^{i_k} \Rightarrow 0 \geq q^t.y^* .$$

B) Si M est copositive plus alors d'après la définition on a :
$y^* >\neq 0$; $M.y^* = 0$; $q^t.y^* <= 0$.

C) Si M est copositive plus alors :

Si $M.x > 0 \Rightarrow 0 = x^t.(-M.y^*) < x^t.(-M.y^*) + y^{*t}.(M.x) = y^{*t}.(-M.x + M.x) = 0$, ce qui est absurde .

Si $M.x+q > 0 \Rightarrow 0 \geq q^t.y^* = q^t.y^* + x^t.(M.y^*) = (M.x+q)^t.y^* > 0$, ce qui est contraire à l'hypothèse. ◊

Théorème 2.2 [22] :

Soit M une matrice symétrique vérifiant l'une de ces deux propriétés :

1. M est strictement copositive .
2. M est copositive plus et vérifie l'une de ces deux conditions :
   $M.x > 0$ ou $M.x + q > 0$

alors, la suite $\{x^i\}$, générée par l'algorithme 2.1 est bornée et admet une valeur d'adhérence x^* qui résout le P.L.C (1). ◊

Corollaire 2.1 :

Si M est une matrice symétrique à éléments ≥ 0 et éléments diagonaux >0 alors la suite $\{x^i\}$ générée par l'algorithme 2.1, est bornée et admet une valeur d'adhérence x^* qui résout le problème PLC (1). ◊

Théorème 2.3 [36] :

Si M est une matrice symétrique, copositive plus et lim $\|x^{i+1} - x^i\| = 0$ telle que l'ensemble S une valeur d'adhérence de $\{x^i\}$ est fermé, non vide, et ne peut

pas s'écrire sous la forme d'une réunion de deux ensembles fermés, non vides, alors la suite $\{x^i\}$ converge vers une solution de P.L.C (1). ◊

Théorème 2.4 [22] :
Si M est une matrice symétrique, copositive plus, alors la suite $\{x^i\}$, générée par l'algorithme 2.1., converge vers une solution x* de P.L.C (1). ◊

Corollaire 2.2 :
Si M est une matrice symétrique, définie positive alors la suite $\{x^i\}$ générée par l'algorithme2.1converge vers x* solution du (P.L.C). ◊

3.4.3. Partie B :

Dans cette partie, la matrice M est non symétrique. On pose:
$E^i = E$ une matrice diagonale > 0, $K^i = K$ une matrice triangulaire strictement inférieure ou strictement supérieure telle que : $[2(\lambda.\omega.E)^{-1} - M]$ soit définie positive .

Algorithme 3.1 :
Soit $z^0 \geq 0$ un vecteur arbitrairement choisi . On définit la suite $\{z^i\}$ par :
$z^{k+1} = [z^k - \omega.E.(M.z^k + q + K(z^{k+1}-z^k))]_+$.

Propriétés de convergence [22] :

Lemme 3.1 : Soient z^{k+1} et z^k les termes de la suite $\{ z^k \}$ donnés par l'algorithme 4.1, alors : $| z^{k+1} - z^k | \leq (I -\omega.E.|K|)^{-1}.|I-w.E.(M-K)|.| z^k - z^{k-1}|$.

Preuve : $z^{k+1} - z^k =$
$[z^k - \omega.E.(M.z^k + q + K(z^{k+1}-z^k))]_+ - [z^{k-1}-\omega.E.(M.z^{k-1} + q + K(z^k-z^{k-1}))]_+$
$\leq [(z^k -z^{k-1}) - \omega.E.M. (z^k -z^{k-1}) - w.E.K.(z^{k+1} -z^k) + \omega.E.K.(z^k -z^{k-1})]_+$
$\leq [(I-\omega.E.(M-K)). (z^k -z^{k-1}) -\omega.E.K. (z^{k+1} -z^k)]_+$.
De la même facon : $z^k - z^{k+1} \leq [(I-\omega.E.(M-K)). (z^{k-1} -z^k) -\omega.E.K. (z^k -z^{k+1})]_+$.
D'où : $| z^{k+1} - z^k | \leq | I-\omega.E.(M-K) |. | z^k - z^{k-1}| + \omega.E.K. | z^{k+1} - z^k |$ et par suite :
$| z^{k+1} - z^k | \leq (I -\omega.E.|K|)^{-1}.|I-\omega.E.(M-K)|.| z^k - z^{k-1}|$.
En posant : $G = (I -\omega.E.|K|)^{-1}.|I-\omega.E.(M-K)|$, on obtient :
$| z^{k+1} - z^k | \leq G.| z^k - z^{k-1}|$. ◊

Théorème 3.1 [22] :
Soient ω, E, K, M vérifiant $\rho(G) < 1$ ($\rho(G)$ est le rayon spectral de G), alors la suite $\{z^i\}$ générée par l'algorithme 4.1, converge vers z* solution du P.L.C (1). ◊

4. PROBLEME NON LINEAIRE COMPLEMENTAIRE.

4.1. METHODE DE QUASI-NEWTON POUR RESOUDRE LE PNC

4.1.1. Introduction :

On considère le problème non linéaire complémentaire (**PNC**) :
Chercher $x \in \mathbb{R}^n$ tel que :

$$x \geq 0 \ ; \ f(x) \geq 0 \ ; \ x^t.f(x) = 0 \qquad (1)$$

où $f : \mathbb{R}^n \mapsto \mathbb{R}^n$ est une fonction continûment différentiable dans $\mathbb{R}^n$.

La méthode de Quasi-Newton génére une suite $\{x^k\}$ définie comme suit: $x^0 \geq 0$ quelconque, si on suppose que l'itéré x^k calculé, alors x^{k+1} est la solution du problème linéaire complémentaire (**PLC**) suivant :

$$x \geq 0; \ w = f(x^k) + \nabla f(x^k)(x-x^k) \geq 0; \ x^t.w = 0. \qquad (2_k)$$

Sous certaines conditions concernant la suite $\{x^k\}$, on montre que celle-ci converge vers une limite x^* solution du problème (1) .

Un caractéristique très important de la méthode de Newton est sa convergence superlinéaire et quadratique.
On rappelle que la convergence est dite superlinéaire si :

$$\lim_{k \to +\infty} \frac{\| x^{k+1}-x^* \|}{\| x^k-x^* \|} = 0.$$

La convergence est dite quadratique s'il existe un scalaire $c>0$ tel que :

$$\| x^{k+1}-x^* \| < c. \| x^k-x^* \|^2 \quad \text{pour k assez grand} \qquad (3)$$

A chaque itération on a deux calculs importants :
(i) L'évaluation de $\nabla f(x^k)$ qui est une matrice carrée d'ordre n.
(ii) La résolution de sous-problème linéaire complémentaire (2_k) .
Pour cela, on approxime $\nabla f(x^k)$ par une matrice A_k semi-définie positive et on résout le **PLC** suivant :

$$x \geq 0 \ ; w = f(x^k) + A_k(x-x^k) \geq 0 \ ; x^t w = 0 \qquad (4_k)$$

4.1.2. Mesure d'erreurs :

On considère le **PLC** en général : trouver $x \in \mathbb{R}^n$ tel que :

$$x \geq 0 \ ; w = M.x + q \geq 0 \ ; x^t.w = 0 \qquad (5)$$

où M est une matrice carrée d'ordre n et q un n-vecteur donné.

Il est clair que : x résout (5) $\Leftrightarrow h(x) = \min(x, Mx+q) = 0$.

$$\text{Ici} \qquad h_i(x) = \begin{cases} x_i & \text{Si} \quad x_i < (Mx+q)_i \\ (Mx+q)_i & \text{Sinon} \end{cases}$$

Définition :

Soit x* une solution de (1). On dit que x* est une solution régulière s'il existe un voisinage V* de x* et un scalaire $\delta>0$ tel que pour tout vecteur y où $\|y\|<\delta$, il existe un vecteur unique $x(y) \in V^*$ qui résout le **PLC** perturbé :
$x \geq 0 ; w = f(x^*)-y + \nabla f(x^*)(x-x^*) \geq 0 ; x^t.w = 0$. De plus x(y) est lipchizienne en y, i.e.,

$\forall y,y' : \|y\| < \delta$ et $\|y'\| < \delta \Rightarrow \exists L>0 ; \|x(y) - x(y')\| < L \|y - y'\|$ (6).

Lemme 1 :

On suppose que le **PLC** (5) admet une solution régulière x*. Alors, il existe un voisinage V de x* et un scalaire $\lambda>0$ tels que : $\forall x \in V$ on a:

$$\|x-x^*\| \leq \lambda \|h(x)\| \qquad (7)$$

Preuve :

Soient δ, V* et L déclarés dans la définition d'une solution régulière; h(x) est clairement continue et il existe un voisinage V de x* tel que pour chaque x dans V on a :

(a) $x-h(x) \in V^*$

(b) $\|(I-M).h(x)\| < \delta$.

Soit $x \in V$; $u = h(x)$ et $v = x-u$ alors le vecteur $v \in V^*$; d'après (a) et d'après la définition de la fonction, h(x) on a : $\min(v+u, M.(v+u)+q)=h(x) = u$
ce qui implique que $\min (v,q-(I-M).u + M.v) = 0$.
De plus, v est la solution du problème linéaire complémentaire perturbé :

$$v \geq 0 ; \; w = q-(I-M).u + M.v \geq 0; \; v^t.w = 0 \qquad (8)$$

D'après (b) et en utilisant (6) on déduit que : $\|v-x^*\| \leq L.\|(I-M).u\|$.
Si on prend $\lambda = L.\|I-M\|$, alors l'inéquation (7) est vérifiée . ◊

Lemme 2 :

On suppose que le **PLC** (5) admet une solution x* pour laquelle il existe $\alpha>0$ tel que: $(v-x^*)^t.M.(v-x^*) \geq \alpha \|v-x^*\|^2 . \; \forall v \geq 0$ (9)
alors, il existe $\lambda>0$ tel que l'inéquation (7) est vraie pour tout x.

Preuve :

Soit $x \in \mathbb{R}^n$, $u = h(x)$ et $v = x-u$. D'après le Lemme 1 on constate :
Si v est une solution de (8) et si on pose que $w^* = M.x^*+q$, alors :

$(x^*-v)^t(w^*-w) = (x^*-v)^t[(I-M)u + M(x^*-v)]$ et par complémentarité $(x^*-v)^t(w^*-w) \leq 0$.

D'après (9) et l'inégalité de Cauchy-Schwarz on a :

$\|x^*-v\|.\|(I-M).u\| \geq -(x^*-v)^t.(I-M).u \geq (v-x^*)^t.M.(v-x^*) \geq \alpha \|x^*-v\|^2$

Si $x^*=v$ alors $x^*=x-u$ et par suite $\|x^*-x\| = \|u\|$ et (7) est vérifiée avec $\lambda=1$.

Si $x^*\neq v$ alors $\|(I-M).u\| \geq \alpha\|x^*-v\|$ et par suite (7) est vérifiée avec $\lambda=1+\|I-M\|/\alpha$, ainsi le Lemme est établi.

4.1.3. Méthodes de Quasi-Newton :

Soit x^k l'élément courant à la $k^{ième}$ itération et A_k la matrice d'approximation de $\nabla f(x^k)$. On génère x^{k+1} par :

$$h_k(x^{k+1}) \leq n_k.\|\min(x^k,f(x^k))\| \qquad (10.a)$$

où $n_k>0$ et $h_k(x) = \min(x,f(x^k)) + A_k.(x-x^k)$ (10.b)

Il est évident que x^k est la solution du **PNC** (1) si et seulement si $\min(x^k,f(x^k))=0$.

Lemme 3 :

Soit f est une fonction lipschitzienne continue. Il existe $\mu>0$ tel que :

$\|\min(x,f(x))-\min(y,f(y))\| \leq \mu.\|x-y\| \quad \forall x,y$ (11).

Preuve :

Soit $\alpha>0$ tel que : $\|f(x)-f(y)\| \leq \alpha.\|x-y\| \quad \forall x,y$.

On suppose que $\min(x_i,f_i(x)) \geq \min(y_i,f_i(y))$. Si $y_i \geq f_i(y)$ alors :

$|\min(x_i,f_i(x))-\min(y_i,f_i(y))| \leq f_i(x)-f_i(y) \leq \|f(x)-f(y)\| \leq \alpha\|x-y\|$.

Si $y_i < f_i(y)$ alors : $|\min(x_i,f_i(x))-\min(y_i,f_i(y))| \leq x_i-y_i \leq \|x-y\|$.

En supposant $\mu = n^{1/2}.\max(\alpha, 1)$ la condition (11) est vérifiée .◊

Remarque :

D'après le Lemme (3) ; si f est continûment différentiable dans un voisinage V de x^* alors il existe $\mu>0$ tel que (11) est vraie $\forall x,y \in V$. ◊

Lemme 4 :

Soient x^* une solution régulière du **PNC** (1) et f continûment différentiable dans un voisinage de x^* alors : il existe $\delta>0$, deux voisinages V1 et V2 de x^* et une constante $L>0$ tels que: $\forall x\in V1$ et $\|y\| < \delta$, il existe un vecteur unique $z(x,y) \in V2$ tel que : $z(x,y)$ est la solution du **PLC** perturbé :

$z \geq 0$; $w = f(x) - y + \nabla f(x)(z-x) \geq 0$; $w^t.z = 0$. De plus:

Si $\| y\| < \delta$ et $\| y'\| < \delta$ on a : $\| z(x,y) - z(x,y') \| \leq L. \| y-y'\|$. ◊

Théorème 1 [50] :

Soient x^* est une solution régulière du **PNC** (1) et f une fonction continûment différentiable dans un voisinage V de x^*.
Soient $\mu>0$ tel que (11) est vérifiée $\forall x,y \in V$ et $L>0$ une constante donnée par le Lemme 4.
Soit $n_k.[1 + Max(1,L). \| I-\nabla f(x^k) \|] \leq n. \min(1,1/\mu) \quad \forall k$ (12.a)
et ceci $\forall n<1$ alors :

A) Il existe un voisinage de x^* tel que si on choisit x^0 dans ce voisinage, la suite $\{x^k\}$ définie par la méthode Quasi-Newton vérifiant la condition (10) (avec $A_k = \nabla f(x^k)$) converge vers x^* .

B) Si en plus $\lim_{k \to +\infty} n_k = 0$ (12.b) .
alors la convergence est superlinéaire .

C) Si ∇f est lipschitzienne continue dans un voisinage de x^* et $\forall$ k assez grand on a : $n_k \leq \gamma.\| \min(x^k, f(x^k))\|$ pour $\gamma >0$ (12.c)
la convergence est quadratique . ◊

Lemme 5 :

Soient f et x^* déclarés dans le Lemme 4. Il existe δ_1 et $\delta_2>0$ et deux voisinages V_1 et V_2 de x^* et une constante $L>0$ tels que :
$\forall x \in V_1$; $\| A-\nabla f(x^*) \| < \delta_1$ et $\| y\| < \delta_2$, il existe un vecteur unique $z(x,A,y) \in V_2$ tel que : $z \geq 0$; $w = f(x) - y + A.(z-x) \geq 0$; $w^t.z = 0$. De plus si $\| y\| < \delta_1$ et $\| y'\| < \delta_2$ alors : $\| z(x,A,y) - z(x,A,y') \| \leq L. \| y-y'\|$. ◊

Théorème 2 [50] :

Soient f, x^*, μ déclarés dans le Théorème 1 . Soient L, δ_1 les scalaires positifs donnés par le Lemme 5 . On suppose que la suite $\{ x^k \}$ vérifie (10) et $\| A_k-\nabla f(x^k) \| < n^* < \delta$ pour tout k et $n_k[1+Max(1,L).\| I-A_k\|] \leq n^\circ.\min(1,1/\mu)$ pour tout k, avec n° et n^* sont deux scalaires vérifiant $n^\circ + L.n^* < 1$ alors :
A) Il existe un voisinage de x^* tel que pour tout x^0 choisit dans ce voisinage la suite $\{ x^k \}$ est bien définie et converge vers x^* .
B) Si de plus (12.b) est vérifiée et :

$$\lim_{k \to +\infty} \frac{\|(A_k - \nabla f(x^*))(x^k - x^*)\|}{\| x^k - x^* \|} = 0$$

alors la convergence est superlinéaire . ◊

Lemme 6 :

Soit la condition suivante : $h_k(x^{k+1}) \leq \delta_k . \| x^{k+1}-x^* \|$ (16).

On suppose que les Lemmes 1 et 2 et la condition (16) sont vérifiées, alors :

$$\| x^{k+1}-y^{k+1} \| \leq \delta'_k . \| x^{k+1}-x^k \| \quad (16')$$

où y^{k+1} est la solution exacte du sous **PLC** (2_k) .

Preuve : la même démonstration du théorème 1.

Théorème 3 [50] :

Soit x^* une solution du **PNC** (1). On suppose que f est continûment différentiable dans un voisinage V de x^* et que $\nabla f(x^*)$ est définie positive. Soit $\rho>0$ la plus petite valeur propre de la partie symétrique de $\nabla f(x^*)$ et soit $\rho_k = \rho - \| \nabla f(x^*) - \nabla f(x^k) \|$.

On suppose que :

$$\delta'_k = \delta_k (1 + \| I - \nabla f(x^k) \| / \rho_k) \leq \delta < 1/2 \quad \forall k \quad (16'') \text{ alors:}$$

A) Il existe un voisinage de x^* tel que pour tout x^0 choisi dans ce voisinage la suite $\{x^k\}$, produite par la méthode Quasi-Newton et vérifiant (16) converge vers x^* .

B) Si $\lim_{k \to +\infty} \delta_k = 0$ alors la convergence est superlinéaire .

C) Si en plus ∇f est Lipschitzienne continue dans un voisinage de x^* et si k est assez grand, on obtient $\delta_k \leq \beta . \| x^{k+1}-x^k \|$ pour $\beta>0$, alors la convergence est quadratique . ◊

5. Méthode de l'inverse partiel pour résoudre un P.C :

5.1. Introduction :

a) Une multi-application T: $\mathbb{R}^n \mapsto \mathbb{R}^n$ est dite monotone si :

$\langle x-x', y-y' \rangle \geq 0$ pour $y \in T(x)$, $y' \in T(x')$.

Le graphe de T est l'ensemble $Gr(T) = \{(x,y) \in \mathbb{R}^n \times \mathbb{R}^n ; y \in T(x)\}$. Si le graphe de T n'est pas contenu dans aucun autre graphe d'un opérateur monotone alors T est dit maximal .

Les opérateurs monotones maximaux possèdent la propriété intéréssante suivante :

pour tout $x \in \mathbb{R}^n$, il existe un unique $P(x) \in \mathbb{R}^n$ tel que $x-P(x) \in T(P(x))$ **MINTY**.

P: $\mathbb{R}^n \mapsto \mathbb{R}^n$ est dite application proximale relative à T .

Le problème fondamental des opérateurs maximaux monotones est:

" Trouver $x \in \mathbb{R}^n$ tel que $0 \in T(x)$ " (1.1).

Il est facile de voir que : $x=P(x)=(I+T)^{-1}(x)$ si et seulement si $0 \in T(x)$.

L'algorithme proximal se base sur cette propriété pour la résolution du problème (1.1). Pour un x^0 arbitraire cet algorithme génère une suite (x^k) de $\mathbb{R}^n$ selon le processus itératif suivant :

$$x^{k+1}=P(x^k)=(I+T)^{-1}(x^k) \text{ pour } k=0,1,2,.....$$

Théorème 1(Rockaffellar [44]) :

La suite (x^k) converge vers le zéro de T, s'il en existe un. Si T n'admet pas de zéro, alors $\|x^k\|^2 \rightarrow +\infty$.◊

b) Soient A,B deux sous-espaces complémentaires de $\mathbb{R}^n$ (i.e., $A^{\perp}=B$ et $B^{\perp}=A$).

Spingarn [47] a introduit la notion d'inverse partiel d'un opérateur T associé à A, dénoté par $T_A: \mathbb{R}^n \mapsto \mathbb{R}^n$ dont le graphe est:

$$Gr(T_A)=\{(x_A+y_B,\ y_A+x_B) : y \in T(x)\}.$$

Proposition 1 (Spingarn [47]) :

T_A est monotone (maximal) si et seulement si T est monotone (maximal).

L'importance des zéros de T_A réside dans le fait que:

pour tout $x \in A$ et $y \in B$, $0 \in T_A(x+y)$ si et seulement si $y \in T(x)$.

Ainsi $0 \in T_A(z)$ implique que $x=z_A$ et $y=z_B$ sont les solutions du problème:

"Trouver $x \in A$ et $y \in B$ tel que $y \in T(x)$" (1.2).

L'opérateur inverse partiel ainsi défini représente un outil efficace pour la résolution du problème (1.2).

La méthode de l'inverse partiel introduite par **Spingarn** n'est autre que l'application de l'algorithme proximal pour la résolution du problème:

"Trouver $z \in \mathbb{R}^n$ tel que $0 \in T_A(z)$" (1.3).

5.2. Présentation de la méthode de l'inverse partiel:

Nous présentons la méthode de l'inverse partiel via l'algorithme proximal appliqué à l'opérateur inverse partiel T_A.

Algorithme (ou méthode de l'inverse partiel):

étape 1: $x^0 \in A$, $y^0 \in B$ choisis arbitrairement. On initialise k=0.

étape 2: trouver $x'^k \in \mathbb{R}^n, y'^k \in \mathbb{R}^n$ tel que : $x^k+y^k=x'^k+y'^k$ et $y'^k \in T(x'^k)$.

étape 3: $x^{k+1}=(x'^k)_A$ et $y^{k+1}=(y'^k)_B$.

k=k+1 et on retourne à l'**étape2** avec ces nouvelles valeurs .

Le Théorème de convergence de cet algorithme (voir[47]) est le suivant :

Théorème :

Soient (x^k) et (y^k) deux suites générées par l'algorithme précédent, on a soit

(i) $x^k \rightarrow x^*$ et $y^k \rightarrow y^*$ faiblement, pour x^* et y^* solutions de (1.2)

(ii) $\|x^k+y^k\|^2 \rightarrow +\infty$ et (1.2) n'admet pas de solution.◊

Appliquons la méthode de l'inverse partiel à deux cas particuliers très importants de problèmes de complémentarité:

1ère cas) T=∂F le sous-différentiel d'une fonction F convexe propre s.c.i .

Si F une fonction univoque convexe propre semi continue inférieurement, alors ∂F (le sous- différentiel de F) est maximal monotone .

Soit F une fonction convexe propre s.c.i., on considère le P.C :

Trouver $x \in \mathbb{R}^n$ tel que : $x \geq 0$, $y \geq 0$; $y \in \partial F(x)$, $\langle x,y \rangle = 0$. **(1)**

Proposition : Soit $\chi_{\geq}$ est la fonction indicatrice de $(\mathbb{R}^n_+)$ i.e., :

$$\chi_{\geq}(x)=\begin{cases} 0 & \text{Si } x\geq 0 \\ +\infty & \text{Sinon} \end{cases} \quad ; \text{Alors : } \quad \partial\chi_{\geq}(x)=\begin{cases} \{y\leq 0; \langle x,y\rangle=0\} & \text{Si } x\geq 0 \\ \emptyset & \text{Sinon} \end{cases}$$

où $\partial\chi_{\geq}$ désigne le sous gradient de la fonction $\chi_{\geq}$.◊

Le problème (1) est équivalent à résoudre le problème suivant :

$$\text{Trouver } x\in\mathbb{R}^n;\ 0\in(\partial F + \partial\chi_{\geq})(x)$$

Algorithme de résolution :

On considère $A=\{(x,x)\in\mathbb{R}^n\times\mathbb{R}^n\}$ et $B=\{(y_1,y_2)\in\mathbb{R}^n\times\mathbb{R}^n\ ;\ y_1+y_2=0\}$, soit F une fonction convexe propre s.c.i, i.e., ∂F est un opérateur maximal monotone. On pose : $T = \partial F\times\partial\chi_{\geq}$, on cherche $x\in\mathbb{R}^n$ tel que $0\in T(x)$.

étape 1 : $x\geq 0$, $y\in\mathbb{R}^n$ arbitrairement choisis.

étape 2 : Trouver $x'_1,x'_2,y'_1,y'_2\in\mathbb{R}^n$ tel que :

$x+y = x'_1+y'_1$ avec $y'_1 \in \partial F(x'_1)$

$x-y = x'_2+y'_2$ avec $y'_2 \in \partial\chi_{\geq}(x'_2)$

étape 3 : $x = (x'_1+x'_2)/2$.

$y = y'_1 -(y'_1+y'_2)/2 = (y'_1-y'_2)/2$.

et retournons à l'étape 2) avec des nouvelles valeurs de x et y.

Remarque :

L'étape 2) consiste à trouver x'_1,x'_2 tel que :

$x'_1 = (I + \partial F)^{-1}(x+y)$ (a)

$x'_2 = (I +\partial\chi_{\geq})^{-1}(x-y) = \pi(x-y)$ (b)

où $\pi(x)$ est la projection de x sur l'orthant positif .

Le système (b) est facile à résoudre car nous savons que :
Si $x=(x_1, x_2,..., x_n)\in\mathbb{R}^n$ alors $\pi(x)=(y_1, y_2,..., y_n)\in\mathbb{R}^n$ tel que :

$$y_i = \begin{cases} x_i & \text{si } x_i \geq 0 \\ 0 & \text{sinon} \end{cases}$$

Il reste donc le calcul de l'application proximal ∂F, qui sauf cas particuliers, devrait se ramener à la résolution du problème d'optimisation convexe suivant :

$$\text{Inf}\{F(x) +1/2\ \|y-x\|^2\ : x\in\mathbb{R}^n\}.$$

2ème cas) problème linéaire complémentaire:

Soit $T: \mathbb{R}^n\mapsto\mathbb{R}^n$ un opérateur univoque défini par : $T(x) = Mx+q$
où M est une matrice carrée d'ordre n et $q\in\mathbb{R}^n$.

Le problème (P.L.C) peut s'écrire sous cette forme :

Trouver $x \geq 0$ tel que : $T(x) \geq 0$ et $\langle x, T(x) \rangle = 0$.

Pour pouvoir appliquer l'algorithme de l'inverse partiel il faut que T soit maximal monotone. Pour cela il suffit de supposer que la matrice M est symétrique semi-définie positive (**Rockafellar** [41,44]).

L'algorithme de l'inverse partiel appliqué au problème linéaire complémentaire se résume ainsi :

étape 1: $x^0, y^0 \in \mathbb{R}^n$, choisis arbitrairement. On initialise k=0.

étape 2: déterminer $u^k \in \mathbb{R}^n$ solution du système linéaire :
$(I + M)u^k = x^k + y^k - q$.

étape 3: $x^{k+1} = 1/2.[\, u^k + \pi (x^k - y^k)]$.
$y^{k+1} = 1/2.[\, y^k - u^k + x^k + \pi (y^k - x^k)]$.
k=k+1 et on retourne à l'**étape2** avec ces nouvelles valeurs .

On remarque que la mise en oeuvre de cet algorithme est très simple puisque l'étape 2 est simplement la résolution d'un système linéaire dont la matrice est symétrique définie positive .

REFERENCES

[1] M.S.BAZARAA and C.M.SHETTY (1972)
Non Linear Programming.

[2] R.W.COTTLE and G.B.DANTZIG (1968)
Complementary Pivot Theory of Mathematical Programming.
Am. Math. Soc. Providence, pp. 115-138

[3] C.E.LEMKE (1965)
Bimatrix Equilibrium points and Mathematical Programming.
Management Science 4, pp. 681-689.

[4] C.E.LEMKE (1969)
Recent Results on Complementarity Problems. In J.B.Rosen, O.L.Mangasarian and K.Ritter, eds., Academic Press New York, Non Linear Programming, pp. 349-384.

[5] O.L.MANGASARIAN (1969)
Non Linear Programming. Mc Graw-Hill, New York.

[6] O.L.MANGASARIAN (1985)
Simple Computable Bounds for Solutions of Linear Complementarity Problems and Linear Programs. Mathematical Programming 25, pp. 1-12.

[7] M.MINOUX (1983)
Programmation Mathématique : théorie et algorithmes. Dunod, Paris.

[8] M.SAKAROVITCH (1982)
Linear Programming. Springer-Verlag, New York.

[9] J.A.TOMLIN (1978)
Robust Implementation of LEMKE's Method for the Linear Complementary Problem. Mathematical Programming Study, 7, pp. 55-60.

[10] A.YASSINE (1985)
Méthode de Pivotage de LEMKE et ses applications.
Rapport de D.E.A., IMAG, Grenoble.

[11] M. A. BOUGHAZI (1987)
Contribution à l'étude des algorithmes d'optimisation en analyse de données.
Thèse de Doctorat, Université Scientifique, Technologique et Medicale de Grenoble.

[12] J . M. ORTEGA [1972]
Numerical Analysis, A Second Course, Academic Press, New York, New York.

[13] R.W.COTTLE, G.H.GOLUB and R.S. SACHER [1973]
On the Solution of Large Structured Linear Complementarity Problems,III,Stanford University, Stanford,California, Operations Research Report No. 73-8.

[14] A.TAMIR [1973]
The Complementarity Problem of Mathematical Programming, Case estern Reserve University, Cleveland, Ohio, Ph.D .Thesis.

[15] U. ECKHARDT [1974]
Quadratic Programming by Successive Overrelaxation, Kernforschungsanlage Jülich, Technical Report No. Jül-1064-MA.

[16] B. MARTINET and A. AUSLENDER [1974]
Methodes de décomposition pour la Minimisation d'une Fonction sur un espace Produit, SIAM Journal on Control, Vol. 12, pp. 635-642.

[17] J. J. MORE [1974]
Classes of Functions and Feasibility Conditions in Nonlinear Complementarity Problems, Mathematical Programming, Vol. 6, pp. 327-338.
[18] K. G. MURTY [1972]
On the Number of Solutions to the Complementarity Problem and Spanning Properties of Complementary Cones, Linear Algebra and Its Applications, Vol. 5, pp. 65-108.
[19] O. L. MANGASARIAN [1976]
Solution of Linear Complementary Problems by Linear Programming, Numerical Analysis Dundee 1975, Edited by G. W. Watson, Springer-Verlag, Berlin, Germany, pp. 166-175.
[20] O. L. MANGASARIAN [1976]
Linear Complementarity Problems solvable by a single Linear Program, Mathematical Programming, Vol. 10, pp.263-270.
[21] O. L. MANGASARIAN [1976]
Characterization of Linear Complementarity Problems as Linear Programs, University of Wisconsin, Madison, Wisconin, Computer Sciences Report No. 271.
[22] O. L. MANGASARIAN [1977]
Solution of Symmetric Linear Complementarity Problems by Iterative Methods, Journal of Optimization Theory and Applications, Vol. 22, pp. 465-485.
[23] MUHAMED. AGANAGIC [1984]
Newton's Method for Linear Complementarity Problems. Mathematical Programming 28, pp. 349 - 362. North-Holland.
[24] R. CHANDRASEKARIAN [1970]
A special case of the complementarity pivot problem . Opsearch 7, pp. 263 - 268.
[25] E. CINLAR [1975]
Introduction to stochastic processes (Prentice-Hall,Englewood cliffs, NJ,1975).
[26] R.W. COTTLE [1976]
Complementarity and variationnal problems. Symposia Mathematica 19, pp. 177-208.
[27] R.W. COTTLE [1977]
Numerical methods for Complementarity Problems in engineering and applied science. Technical report SOL 77-24, Departement of Operations Research, Stanford University, October 1977.
[28] R.W. COTTLE and G.B. DANTZIG [1968]
Complementarity pivot theory of Mathematical Programming. Linear Algebra and Its Applications 1 (1968) 103 - 125.
[29] R.W. COTTLE, G.H. GOLUB and R.S. SACHER [1974]
On the solution of large, structured linear complementarity problems: III.Technical report 74-7, Departement of Operations Research, Stanford University (1974).
[30] R.W. COTTLE and J.S. PANG [1978]
On solving linear complementarity problems as Linear programs. Mathematical Programming Study 7 (1978) pp. 88 - 107.
[31] R.W. COTTLE and R.S. SACHER [1977]
On the solution of large, structured linear complementarity problems: The Tridiagonal case.Journal of Applied Mathematics and Optimization 3 (1977) . pp. 321 - 340.
[32] O. L. MANGASARIAN [1976]
Equivalence of the complementarity to a system of nonlinear equations. SIAM J. Appl. Math. Vol 31, pp. 96 - 119.

[33] J.S. PANG [1979]
On a class of least-element complementarity problems. Mathematical Programming 16 pp.111 - 126.
[34] J.S. PANG [1979]
Hidden Z-Matrices with positive principal minors . Linear Algebra and Its Applications vol 23, pp. 201 - 215.
[35] C. W. CRYER [1971]
The Solution of A Quadratic Programming Problem Using Systematic Overrelaxation, SIAM Journal on Control, Vol.9, pp. 385-392.
[36] A. M. OSTROWSKI [1966]
Solution of equations and systems of equations. Second Edition, Academic press. New york, New york.
[37] C. EAVENS [1974]
Solving piecewise linear convex equations. Mathematical Programming Study 1, pp 96-119.
[38] A. TAMIR [1974]
Minimality and complementarity properties associed with Z-functions. Mathematical Programming 7. pp 17-31.
[39] O. L. MANGASARIAN [1969]
Nonlinear Programming. Academic Press, New York, pp 349-384.
[40] P. ALART (1985)
Contribution à la résolution numérique des inclusions différentielles.
Thèse de 3ème cycle. Université des Sciences et Techniques du Languedoc. Académie de Montpellier.
[41] R. T. ROCKAFFELAR (1970)
Monotone Operators Associated with Saddle Fonctions and Minimax Problems. American Math Society Providence, pp 241-250.
[42] R. T. ROCKAFFELAR (1970)
On the Maximality of Sums of Nonlinear Monotone Operators.
Trans Amer Math Soc 149, pp 75-88.
[43] R. T. ROCKAFFELAR (1972)
Convex Analysis.
Princeton Univercity Press, Princeton.
[44] R. T. ROCKAFFELAR (1976]
Monotone Operators and Proximal Point Algorithm.
S.I.A.M J Control Optim 14, pp 877-898.
[45] R. T. ROCKAFFELAR (1976)
Augmented Lagrangians and the Proximal Point Algorithm in Convex Programming. Math O.R 1, pp 97-116.
[46] R. T. ROCKAFFELAR (1979)
La théorie des sous-gradient et ses applications en optimisation.
Les presses de l'Université de Montréal.
[47] J. E. SPINGARN (1980)
Fixed and Variable Constraints in Sensitivity Analysis.
SIAM Journal on Control and Optimization 18, pp 297-310.
[48] J. E. SPINGARN (1983)
Partiel Inverse of Monotone Operators.
App. Mth. Optim 10, pp 247-265.

[49] J. E. SPINGARN (1986)
Applications of the Method of Partiel Inverse to Convex Programming : Decomposition. Mathematical Programming 32, pp 199-223.
[50] Jong-Shi PANG [1986]
Inexact Newton Methods for the Nonlinear Complementarity Problem. Mathematical Programming 36, pp 54-71, North-Holland.
[51] G. ISAC [1985]
Problème de complémentarité (en dimension infinie). Publication du département de Mathématiqueset Informatique. Université de Limoges.
[52] P. DU VAL [1940]
The Unloading Problem for Plane Curves. Amer. J. Math. 62, p. 307-311.
[53] O.L. MANGASARIAN and R.R. MEYER [1979]
Nonlinear perturbation of linear programs. SIAM. J. Control and Optimization.
[54] A.W. INGLETON [1966]
A problem in linéar inéqualities. Proc. London Math. Soc. 16, p. 519-536.
[55] R.W. COTTLE [1964]
Note on fundamental theorem in quadratic programming. Siam J. of Appl. Math. Vol 12, p. 663-665.
[56] R.W. COTTLE [1966]
Nonlinear programs with positively bounded Jacobians. Siam J. of Appl. Math. Vol 14, p. 147-158.
[57] R. BENACER & PHAM DINH TAO [1985]
Two general algorithms for solving linear programming with an additional reverse convex constraint. submitted to Mathematical Programming Study.
[58] R. BENACER [1986]
Contribution à l'étude des algorithmes de l'optimisation non convexe et non différentiable. Thèse d'Université. Grenoble.
[59] R. ALLAIA [1984]
Contribution à l'analyse et l'optimisation de différence de fonctions convexes. Thèse 3ème cycle. Toulouse.

CHAPITRE III

ALGORITHME DU GRADIENT CONJUGUE CONDITIONNEL POUR LA PROGRAMMATION QUADRATIQUE IMPLEMENTATION EFFECTIVE ET EXPERIMENTATIONS NUMERIQUES

1. INTRODUCTION

Dans ce chapitre nous présenterons le problème de la programmation quadratique dans un cadre théorique général. Après avoir passé en revue la méthode des directions conjuguées, destinée à résoudre ce type de problème, nous nous intéresserons plus particulièrement à une de ses variantes, connue sous le nom de méthode de gradient conjugué ([7],[14],[15]). Initialement destinée à la minimisation d'une forme quadratique convexe dans tout l'espace (sans contrainte), elle est ensuite adaptée, via les conditions d'optimalité, à la programmation quadratique, i.e. la minimisation d'une forme quadratique convexe sur un polyèdre convexe.

Comme dans les algorithmes de type gradient projeté, les gros efforts dans la méthode du gradient conjugué conditionnel sont dûs au calcul des projections. La dualité permet de remédier à cet inconvénient. En effet le problème dual d'un programme quadratique est encore un programme quadratique qui possède un avantage de n'avoir que des contraintes de nonnégativité : la projection y sera explicite. Par contre la forme quadratique associée fait apparaître la matrice C^{-1} (C étant la matrice correspondant à la forme quadratique du problème primal).

La finitude de ces méthodes (convergence en un nombre fini d'itérations) : n (au plus) pour le cas sans contraintes, et k(n,p), fonction de la dimension de l'espace et du nombre des facettes du polyèdre convexe, pour le cas des contraintes linéaires) est particulièrement appréciée en pratique.

Nous réprésenterons, ensuite, la transformation du problème de programmation quadratique en un problème linéaire complémentaire à partir des conditions d'optimalité de Kuhn-Tucker, et cela, en vue de l'application des algorithmes propres au programme linéaire complémentaire (à sa résolution) (cf. Chapitre II).

Pour terminer, nous donnerons comme application, le problème de la régression isotone et celui de la régression concave rencontrés en Analyse des Données, (dont les résultats numériques et l'étude comparative seront présentés dans le dernier chapitre consacré aux applications).

2. PROBLEME DE PROGRAMMATION QUADRATIQUE CONVEXE

On appelle problème de programmation quadratique, la recherche du minimum d'une fonction quadratique avec des contraintes linéaires :

$$(P): \begin{cases} \min\ f(x) = 1/2\langle x,Cx\rangle + \langle d,x\rangle & (1.1) \\ \text{sous les contraintes linéaires :} \\ Ax \le b & (1.2) \end{cases}$$

où
n = nombre des variables.
m = nombre des contraintes.
C = matrice carrée réelle de type n×n, symétrique.
A = matrice réelle de type m×n (matrice des contraintes) définie par :
$A = [a_1, a_2, \ldots, a_m]^t$ avec $a_i \in \mathbb{R}^n$ pour $1 \le i \le m$, et $x \in \mathbb{R}^n, d \in \mathbb{R}^n, b \in \mathbb{R}^m$.

(P) est un programme quadratique convexe si f(x) est convexe, i.e. si C est semi définie positive.
Avec la programmation linéaire, la programmation quadratique joue un rôle fondamental dans la théorie de l'optimisation [2, 12, 20, 21]. Ce paragraphe est consacré à l'étude de la programmation quadratique convexe, de la dualité et de la méthode du gradient conjugué conditionnel pour sa résolution.

2.1. Conditions nécessaires et suffisantes d'optimalité :

Soit S l'ensemble des solutions réalisables du problème (1.1) - (1.2) c'est-à-dire : $S=\{x \in \mathbb{R}^n; \langle a_i,x\rangle \le b_i,\ i=1,\ldots,m\}$. Posons $I=\{1,2,\ldots,m\}$.

Le théorème suivant est fondamental et donne une condition nécessaire et suffisante d'optimalité globale pour le problème (1.1) sous les contraintes de type (1.2).

Théorème 1 (Kuhn-Tucker [2, 20]) :

Une condition nécessaire et suffisante pour que $x^* \in S$ soit solution du problème (P) est qu'il existe des réels positifs v_i, pour i=1,...,m tel que :

$$Cx^* + d + \sum_{i=1}^{m} v_i a_i = 0$$

$$v_i[\langle a_i,x\rangle - b_i] = 0 \quad \forall\ i \in I\ .\ \Diamond$$

Définition 1 :
Les nombres $v_i \ge 0$ (i=1,.....,m) figurant dans le théorème précédent s'appellent les multiplicateurs de Lagrange (associés à la contrainte linéaire $Ax \le b$).

2.2. Méthodes des directions conjuguées :

Principe général :

Il s'agit des méthodes itératives qui, appliquées à une fonction quadratique de n variables, conduisent à l'optimum en n étapes au plus.
Considérons la fonction quadratique : $f(x) = 1/2\langle x,Cx\rangle + \langle d,x\rangle$

Le principe des méthodes de directions conjuguées consiste, à partir d'un point, à minimiser f(x), successivement suivant n directions linéairement indépendantes $d_0, d_1, \ldots, d_{n-1}$, possédant la propriété d'être mutuellement conjuguées par rapport à la forme quadratique f(x). Rappelons que cette propriété s'exprime par :

$$\left.\begin{array}{l}\forall i \text{ tel que } 1\le i\le n-1\\ \forall j \text{ tel que } 1\le j\le n-1\\ i\neq j\end{array}\right\} \Rightarrow d_i.C.d_j = 0 \ .$$

Une méthode de directions conjuguées converge de façon finie en, au plus n étapes dans le cas de la recherche d'un minimum d'une fonction quadratique sans contraintes.
Comme il existe un certain nombre de degrés de liberté pour le choix des directions $(d_0, d_1, \ldots, d_{n-1})$, on peut trouver de nombreux algorithmes basés sur le principe général des directions conjuguées. On présente plus particulièrement la méthode du gradient conjugué conditionnel.

2.3. Méthode du gradient conjugué [2, 5, 6, 12, 20] :

La méthode du gradient conjugué a été proposée à l'origine par Hestenes [5] pour résoudre un système d'équations linéaires; et la première application en optimisation non linéaire est due à Fletcher et Reeves [6].

Le problème considéré consiste à minimiser, sur tout l'espace $\mathbb{R}^n$:

$$f(x) = 1/2\langle x,Cx\rangle + \langle d,x\rangle \qquad (3.1)$$

L'objective de cette méthode est de construire progressivement des directions $d_0,d_1,\ldots,d_{n-1}$ mutuellement conjuguées par rapport à C.
A chaque étape k, la direction d_k est obtenue par combinaison linéaire du gradient $-\nabla f(x^k)$ en x^k et de d_{k-1}.

Les directions précédentes $d_0,d_1,\ldots,d_{k-1}$ et les coefficients de la combinaison linéaire sont choisis de telle sorte que d_k soit conjuguée par rapport à toutes les directions précédentes.

On décrit brièvement l'algorithme général du gradient conjugué pour les fonctions quadratiques :

0) On prend $x^0 \in \mathbb{R}^n$ un point de départ arbitrairement choisi ;
$d_0 = -g_0 = -\nabla f(x^0) = -Cx^0 - d$; $k = 0$.
Si $g_0 = \nabla f(x^0) = 0$, on s'arrête : x^0 est une solution optimale du problème.

1) i) Si $<d_k, Cd_k> = 0$,on s'arrête : Le problème n'admet pas de solutions.

ii) $\lambda^k = - \dfrac{<\nabla f(x^k), d_k>}{<d_k, Cd_k>} = \dfrac{\|\nabla f(x^k)\|}{<d_k, Cd_k>}$

$x^{k+1} = x^k + \lambda^k d_k$.

iii) Si $g_{k+1} = \nabla f(x^{k+1}) = 0$, on s'arrête : x^{k+1} est une solution optimale du problème.

iv) $d_{k+1} = - \nabla f(x^{k+1}) + \beta_k d_k$

Avec $\beta_k = - \dfrac{< \nabla f(x^{k+1}, Cd_k>}{<d_k, Cd_k>} = \dfrac{\|\nabla f(x^{k+1})\|^2}{\|\nabla f(x^k)\|^2}$.

$k = k + 1$, et on retourne à (1).

Remarques :

Pour calculer λ^k, il faut la choisir de telle façon qu'il minimise la fonction suivant la direction d_k i.e., $\lambda^k = \min \{ f(x^k + \lambda d_k) ; \lambda \geq 0\}$.

Soit $g(\lambda) = f(x^k + \lambda d_k) = 1/2 < x^k + \lambda d_k, C.(x^k + \lambda d_k)> + <d, x^k + \lambda d_k>$

$= 1/2 <d_k, Cd_k> \lambda^2 - \lambda \|\nabla f(x^k)\|^2$ + terme constant par rapport à λ.

alors, $g'(\lambda) = <d_k, Cd_k> \lambda - \|\nabla f(x^k)\|^2$

i) Si $<d_k, Cd_k> = 0$, alors $g'(\lambda) = - \|\nabla f(x^k)\|^2$ et par suite, la fonction g est strictement décroissante d'où $\min\{f(x): x \in \mathbb{R}^n\} = - \infty$. Le problème n'admet pas de solutions.

ii) Sinon, alors λ^k doit annuler la dérivée $g'(\lambda)$ d'où : $\lambda^k = \|\nabla f(x^k)\|^2 / <d_k, Cd_k>$.

Théorème 2 ([2, 6, 20]) :

La méthode du gradient conjugué décrite ci-dessus converge en n itérations au plus .

Le principe est formé sur la remarque suivante : les g_i successifs vont en s'orthogonalisant, c'est-à-dire qu'on aura toujours un g_k égal à zéro. ◊

2.4. Opérateur de projection :

Soit $I=\{1,.....,m\}$ et J un sous-ensemble de I. Formons la matrice A_J dont les lignes sont les vecteurs a_i ($i \in J$), de manière à obtenir une matrice $p \times n$, p étant le cardinal de J .

Lemme 1:

Les vecteurs a_i ($i \in J$) sont linéairement indépendants si et seulement si $(A_J A_J{}^t)$ est régulière.

Remarque :
On suppose que les vecteurs a_i ($i \in J$) sont linéairement indépendants. Si on pose : $P = A_J^t (A_J A_J^t)^{-1} A_J$, on voit clairement que :
1) $P^t = P$
2) $P.P = P$
3) $P(I - P) = (I - P)P = 0$.
L'opérateur P est un opérateur de projection orthogonale dans le sous espace engendré par les vecteurs a_i ($i \in J$) .

2.5. Minimisation d'une forme quadratique sur une variété affine :

Soit le problème :

$$(P_2) : \begin{cases} \min\ f(x) = 1/2\langle x, Cx\rangle + \langle d, x\rangle \\ \text{sous les contraintes :} \\ \langle a_i, x\rangle = b_i \ ; \ i \in J \end{cases}$$

où J est un sous ensemble de $I=\{1,.....,m\}$.
Les vecteurs a_i ($i \in J$) sont supposés linéairement indépendants.
De ce fait, la matrice $(A_J A_J^t)$ serait régulière. On définit alors la projection orthogonale sur le sous-espace engendré par les vecteurs a_i ($i \in J$) par :

$$P = A_J^t (A_J A_J^t)^{-1} A_J .$$

Soit $x^0 \in S = \{ x \in \mathbb{R}^n ; \langle a_i, x\rangle = b_i ; i \in J\}$, on introduit une nouvelle variable y par la formule:

$$x = x^0 + (I - P)y \qquad (*)$$

Soit la fonction quadratique

$$g(y) = f(x) = f(x^0 + (I - P)y) \qquad (**)$$

alors, $\nabla g(y) = (I - P)\, \nabla f(x)$.

Il est facile de voir que le problème (P_2) se ramène à la minimisation de la fonction quadratique g(y) sans contrainte. La recherche du minimum de g(y) se fera par la méthode du gradient conjugué décrite ci-dessus.

Lemme 2 [2] :
On considère le problème :

$$(Q) : \begin{cases} \min\ g(y) \\ y \in \mathbb{R}^n \end{cases}$$

Soit y^* le minimum libre de la fonction g(y) alors le point correspondant $x^* = x^0 + (I - P)\, y^*$ est la solution de (P_2). ◊

La méthode du gradient conjugué pour minimiser g(y) est donnée par le processus itératif suivant (voir **2.3**):

On initialise $y^0 = 0$, $d_0 = -\nabla g(y^0) = -(I-P)\nabla f(x^0)$, $k = 0$.

$y^{k+1} = y^k + \lambda^k d_k$ avec :

$$\lambda^k = -\frac{\langle \nabla g(y^k), d_k \rangle}{\langle d_k, (I-P).C.(I-P).d_k \rangle}$$

$$d_{k+1} = -\nabla g(y^{k+1}) + \beta_k d_k \quad \text{avec :}$$

$$\beta_k = \frac{\|\nabla g(y^{k+1})\|^2}{\|\nabla g(y^k)\|^2}$$

Car la matrice qui définit le terme quadratique de g(y) est (I-P)C(I-P). Les formules ci-dessus définissent le processus par rapport aux variables auxiliaires. Il est cependant nécessaire d'utiliser les variables originelles. La formule (*) entraine

$$x^{k+1} = x^0 + (I-P)\, y^{k+1} \quad \forall k \geq 0$$

or

$$y^{k+1} = y^k + \lambda^k d_k \quad \text{, d'où}$$

$$x^{k+1} = x^0 + (I-P)(y^k + \lambda^k d_k) = x^0 + (I-P)y^k + \lambda^k (I-P)d_k = x^k + \lambda^k (I-P)d_k$$

d'où

$$x^{k+1} = x^k + \lambda^k (I-P)d_k \qquad (***)$$

Montrons alors par récurrence que

$$(I-P)\, d_k = d_k \qquad \forall k \geq 0. \qquad (****)$$

En effet, pour k=0 on a $(I-P)\, d_0 = -(I-P)\nabla g(0) = -(I-P)(I-P)\nabla f(x^0)$
or $(I-P)(I-P) = I - P - P(I-P) = I - P - P + P^2 = I - P$
d'où $(I-P)\, d_0 = d_0$.

Supposons que la relation (****) est vraie pour k et démontrons-le pour (k+1) :

$$(I-P)\, d_{k+1} = -(I-P)\nabla g(y^{k+1}) + \beta_k (I-P)\, d_k = -(I-P)(I-P)\nabla f(x^{k+1}) + \beta_k d_k$$

$$= -(I-P)\nabla f(x^{k+1}) + \beta_k d_k = -\nabla g(y^{k+1}) + \beta_k d_k = d_{k+1}$$

alors la formule (***) devient : $x^{k+1} = x^k + \lambda^k d_k$

on peut alors citer le théorème suivant :

Théorème 3 :

Connaissant un point x^0 vérifiant les contraintes $< a_i,x> = b_i$; $i \in J$, alors le problème (P_2) est résolu en un nombre fini de pas par l'algorithme suivant :

0) On prend $x^0 \in S = \{ x \in \mathbb{R}^n : < a_i,x> = b_i , i \in J\}$ comme point de départ, $d_0 = - (I -P)\nabla f(x^0)$, $k = 0$. Si $(I -P)\nabla f(x^0) = 0$, on s'arrête : x^0 est une solution optimale du problème (P_2).

1) i) Si $<d_k,Cd_k>=0$, on s'arrête: le problème (P_2) n'admet pas de solutions.

ii) $$\lambda^k = - \frac{<(I-P)\nabla f(x^k),d_k>}{<d_k,Cd_k>} = \frac{\|(I-P)\nabla f(x^k)\|^2}{<d_k,Cd_k>}.$$

$$x^{k+1} = x^k + \lambda^k d_k .$$

iii) Si $(I -P)\nabla f(x^{k+1}) = 0$, on s'arrête : x^{k+1} est une solution optimale du problème (P_2).

iv) $$d_{k+1} = - (I -P) \nabla f(x^{k+1}) + \beta_k d_k$$

Avec $$\beta_k = \frac{\|(I-P).\nabla f(x^{k+1})\|^2}{\|(I-P).\nabla f(x^k)\|^2}.$$

$k = k + 1$, et on retourne à (1). ◊

2.6. Description de l'algorithme de résolution du problème (1.1)-(1.2) :

Soit le problème (1.1)-(1.2) qui consiste à minimiser

$$f(x) = 1/2<x,Cx> + <d,x> \qquad (7.1)$$

sous les contraintes

$$Ax \leq b \qquad (7.2)$$

Posons $\mathcal{J}(x)= \{i : 1 \leq i \leq m$ tel que $<a_i,x> - b_i = 0\}$, et supposons que, pour tout x, les vecteurs , $i \in \mathcal{J}(x)$, sont linéairement indépendants.

Soit x^0 un point quelconque vérifiant (1.2) et constituant l'approximation initiale. Soit $\mathcal{J}_0 = \mathcal{J}(x^0)$ et $P_{\mathcal{J}0} = A_{\mathcal{J}0}{}^t(A_{\mathcal{J}0} A_{\mathcal{J}0}{}^t)^{-1} A_{\mathcal{J}0}$

Calculons les quantités :

$v_0 = - (A_{\mathcal{J}0} A_{\mathcal{J}0}{}^t)^{-1} A_{\mathcal{J}0} \nabla f(x^0)$ et $(I- P_{\mathcal{J}0})\nabla f(x^0) = \nabla f(x^0) + A_{\mathcal{J}0}{}^t v_0$.

Deux cas peuvent se présenter :

I) $(I- P_{\mathcal{J}0})\nabla f(x^0) = 0$ alors $\nabla f(x^0) + A_{\mathcal{J}0}{}^t v_0 =0.$ (7.3)

et x^0 réalise le minimum de f(x) sur une face définie par le système d'équations : $<a_i,x> - b_i = 0$ $i \in \mathcal{J}_0$.

Si le vecteur v_0 n'a pas des composantes négatives $v_0^i, i \in \mathcal{J}_0$, le point x^0 est solution du problème original, car les relations (7.3) sont alors les conditions nécessaires et suffisantes d'existence d'un minimum de f(x) avec les contraintes $Ax \leq b$.

Admettons qu'il existe un indice $j \in \mathcal{J}_0$ tel que $v_0^j < 0$.

Soit $\mathcal{J}_0' = \mathcal{J}_0 \setminus \{j\}$. On résoud alors le problème

$$\min f(x)$$

avec

$$\langle a_i, x\rangle - b_i = 0 \quad i \in \mathcal{J}_0' \qquad (7.4)$$

par la méthode du gradient conjugué. Le point trouvé ne doit pas quitter le domaine admissible défini par $Ax \leq b$. Pour éviter ce phénomène, on effectue donc, à chaque pas une vérification en calculant :

$$\lambda_k^* = \min\{ \frac{b_i - \langle a_i, x^k\rangle}{\langle a_i, d_k\rangle} : i \in J\} \quad \text{où } J=\{i : \langle a_i, d_k\rangle > 0\}.$$

Ici x^k est un point de la suite générée par l'algorithme et d_k la direction conjuguée en ce point. Soit λ^k la grandeur correspondante au pas de la méthode du gradient conjugué.

Si $\lambda^k < \lambda_k^*$ alors $x^{k+1} = x^k + \lambda^k d_k$, et le processus continue.

Si $\lambda^k \geq \lambda_k^*$ alors $x^{k+1} = x^k + \lambda_k^* d_k$, et l'algorithme s'arrête.

Conclusion :
ou bien on réalise le minimum de f(x) sous les conditions (7.4), ou bien le processus s'arrête quand $\lambda^k \geq \lambda_k^*$.
Dans les deux cas, on initialise avec le point obtenu comme on l'a fait avec le point initial.

II) $(I - P_{\mathcal{J}_0})\nabla f(x^0) \neq 0$.

Dans ce cas, on résoud par la technique du gradient conjugué, le problème de minimisation de f(x) sous les contraintes:

$$\langle a_i, x\rangle - b_i = 0 \quad i \in \mathcal{J}_0 \qquad (7.5)$$

avec x^0 comme point de départ. Comme dans le premier cas, on teste à chaque pas l'admissibilité des points obtenus, i.e. on calcule

$$\lambda_k^* = \min\{ \frac{b_i - \langle a_i, x^k\rangle}{\langle a_i, d_k\rangle} : i \in J\} \quad \text{où } J=\{i : \langle a_i, d_k\rangle > 0\}.$$

Comme dans le cas I) on aura :

Si $\lambda^k < \lambda_k^*$ alors $x^{k+1} = x^k + \lambda^k d_k$, et le processus continue.

Si $\lambda^k \geq \lambda_k^*$ alors $x^{k+1} = x^k + \lambda_k^* d_k$, et l'algorithme s'arrête.

Conclusion [2] :
ou bien on réalise le minimum de f(x) sous les conditions (7.5), ou bien le processus s'arrête quand $\lambda^k \geq \lambda_k^*$.
L'algorithme ainsi décrit converge en un nombre fini d'itérations. ◊

2.7. Schéma de l'itération :

Soient x^0 un point quelconque tel que $Ax^0 \leq b$, $\mathcal{J}_0 = \{1 \leq i \leq m : \langle a_i, x\rangle - b_i = 0\}$ et $A_{\mathcal{J}_0}$ la matrice dont les colonnes sont les vecteurs a_i, $i \in \mathcal{J}_0$.

Posons : $P_{\mathcal{J}_0} = A_{\mathcal{J}_0}{}^t(A_{\mathcal{J}_0} A_{\mathcal{J}_0}{}^t)^{-1} A_{\mathcal{J}_0}$ et $v_0 = -(A_{\mathcal{J}_0} A_{\mathcal{J}_0}{}^t)^{-1} A_{\mathcal{J}_0} \nabla f(x^0)$.

1) Si $(I - P_{\mathcal{J}_0})\nabla f(x^0)=0$ alors on passe à 2), sinon (i.e., $(I - P_{\mathcal{J}_0})\nabla f(x^0)\neq 0$) on passe à 3).

2) Si $v_0^j \geq 0$, $i \in \mathcal{J}_0$ alors x^0 est une solution du problème (1.1)-(1.2).

Sinon, i.e. il existe $j \in \mathcal{J}_0$ tel que $v_0^j < 0$, on pose $\mathcal{J}_0 = \mathcal{J}_0 \setminus \{j\}$ et on passe à 3).

3) Procédure du gradient conjugué pour la minimisation de

$f(x) = 1/2\langle x, Cx\rangle + \langle d, x\rangle$

sous les contraintes ($)

$\langle a_i, x\rangle - b_i = 0 \quad i \in \mathcal{J}_0$.

a) On prend x^0 comme point de départ, et on pose $d_0 = -(I - P_{\mathcal{J}_0})\nabla f(x^0)$

b) A l'itération k, on obtient x^k ; pour déterminer x^{k+1}, on procède de la façon suivante :

on définit $d_k = -(I - P_{\mathcal{J}_0})\nabla f(x^k) + \beta_k d_{k-1}$

avec

$$\beta_k = \frac{\|(I-P)\nabla f(x^{k+1})\|^2}{\|(I-P)\nabla f(x^k)\|^2} \quad , \quad \lambda^k = \frac{\|\nabla f(x^k)\|^2}{\langle d_k, Cd_k\rangle} \qquad k=0,1,2,.....$$

et

$$\lambda_k^* = \min\left\{ \frac{b_i - \langle a_i, x^k\rangle}{\langle a_i, d_k\rangle} \ : \ i \in J \right\} \quad \text{où } J=\{i : \langle a_i, d_k\rangle > 0\}.$$

-Si $\lambda^k < \lambda_k^*$ alors $x^{k+1} = x^k + \lambda^k d_k$.

Faire $k \leftarrow k+1$ et on retourne à b) jusqu'à l'obtention d'un point qui réalise le minimum du problème ($) qu'on notera x^* et on passe à 4).

Si $\lambda^k \geq \lambda_k^*$, alors $x^{k+1} = x^k + \lambda_k^* d_k$, on pose $x^* = x^{k+1}$ et on passe à 4).

4) On pose $x^0 = x^*$ et $\mathcal{J}_0 = \{1 \leq i \leq m : \langle a_i, x^*\rangle - b_i \leq 0\}$ et on recommence le processus (passage à 1)).

3. DUALITE ET FONCTION DE LAGRANGE

Soit le problème

$$\min f(x) \qquad (8.1)$$

sous les contraintes

$$g_i(x) \leq 0 \ \text{ pour } 1 \leq i \leq m \qquad (8.2)$$

Le Lagrangien associé au problème (8.1)-(8.2) est défini par :

$$L(x,u) = \begin{cases} f(x) + \langle u, g(x)\rangle & \text{si } u \geq 0 \\ -\infty & \text{sinon} \end{cases}$$

où $u=(u_1,u_2,.....,u_m)^t$ et $g(x)=(g_1(x),g_2(x),....., g_m(x))^t$.
Le vecteur $u=(u_1,u_2,...,u_m)^t$ est le vecteur de la variable duale.
Cette fonction joue un rôle fondamental dans la théorie de dualité.

La fonction primale est définie par :

$$L_p(x) = \left\{ \begin{array}{ll} f(x) & \text{si } g(x) \leq 0 \\ +\infty & \text{sinon} \end{array} \right\}$$

et la fonction duale par

$$L_d(x) = \inf\{L(x,u) : x \in \mathbb{R}^n\}$$

Comme la fonction primale (resp. duale) est une fonction en x (resp. en u), les deux problèmes d'optimisation suivants peuvent être formulés:

Problème primal :

$$\min\{L_p(x) ; g_i(x) \leq 0 ; 1 \leq i \leq m\} = \min\{f(x) ; g_i(x) \leq 0 ; 1 \leq i \leq m\}. \quad (8.3)$$

Problème dual :

$$\max\{L_d(x) ; u \geq 0\} \quad (8.4)$$

Définition :
On dit que (x^*,u^*), $u^* \geq 0$ est un point-selle du Lagrangien si :

$$L(x^*,u) \leq L(x^*,u^*) \leq L(x,u^*) \quad \forall u \geq 0 \text{ et } \forall x \in \mathbb{R}^n . \quad (8.5)$$

On notera D l'ensemble des solutions réalisables de (8.1)-(8.2), c'est-à-dire:

$$D=\{x \in \mathbb{R}^n : g_i(x) \leq 0 ; 1 \leq i \leq m\} \quad (8.6)$$

On dit que les contraintes (8.2) vérifient l'hypothèse de qualification (condition de Slater) si il existe x^0 tel que $g_i(x^0)<0$ pour i=1,...,m.

Théorème 1 [10, 11] :
Supposons que les fonctions f et g_i $(1 \leq i \leq m)$ sont convexes et différentiables, et que la condition de Slater est vérifiée, alors le point x^* est solution du problème primal si et seulement si il existe $u^* \geq 0$ tel que (x^*,u^*) soit un point-selle de la fonction du Lagrangien. ◊

Dans ce cas u^* est solution du problème dual. Autrement dit, (x^*,u^*) est un point-selle de L défini par (8.5), si et seulement si :

$$\nabla_x L(x^*,u^*) = 0 \qquad (8.7)$$

$$\nabla_u L(x^*,u^*) \leq 0 \qquad (8.8)$$

$$\langle u^*, \nabla_u L(x^*,u^*)\rangle = 0 \qquad (8.9)$$

$$u^* \geq 0 \qquad (8.10)$$

où ∇_x, ∇_u représentent respectivement les gradients par rapport à x et u.

Ajoutons que si les contraintes g_i (i=1,...,m), sont linéaires affines alors la condition de Slater n'est pas nécessaire dans le Théorème 1 (Rockafellar [11]). D'autre part la simplicité des contraintes (de non-négativité) du problème dual permettent d'obtenir explicitement les projections intervenant dans la méthode du gradient conjugué conditionnel (cf. § 2.6). L'inconvénient du problème dual (sauf si C^{-1} se calcule facilement, par exemple si C est diagonale ou tridiagonale, cf. §.5).

La théorie de la dualité nous permet de formuler et de résoudre le problème dual au lieu du problème primal (i.e. résoudre le primal à partir du dual), ce qui est parfois plus avantageux et plus commode (cf. Chapitre VI).

3.1. Le dual du programme quadratique convexe (1.1)-(1.2) :

Pour simplifier la présentation, nous supposons que C est définie positive. Selon l'étude faite précédemment, le dual du problème quadratique (1.1)-(1.2) est :

$$\max\{\min\{1/2\langle x,Cx\rangle+\langle d,x\rangle+\langle u,Ax-b\rangle \ ; x\in \mathbb{R}^n\} \ ; u\geq 0\} \qquad (8.11)$$

Le problème sans contraintes

$$\min\{1/2\langle x,Cx\rangle+\langle d,x\rangle+\langle u,Ax-b\rangle \ ; x\in \mathbb{R}^n\}$$

a pour solution

$$x^* = -C^{-1}(d+A^t u). \qquad (8.12)$$

Substituons (8.12) dans (8.11), on aura :

$$\min\{1/2\langle u, AC^{-1}A^t u\rangle+\langle u,AC^{-1}d+b\rangle \ ; u\geq 0\}. \qquad (8.13)$$

Ce problème dual est lui aussi un programme quadratique, mais le nombre de variables est m au lieu de n ce qui est avantageux dans le cas où m<n.

3.2. Méthode du gradient conjugué conditionnel appliqué au problème dual (8.13) :

Le problème dual est de minimiser

$$F(u) = 1/2\langle u, AC^{-1}A^t u\rangle+\langle u,AC^{-1}d+b\rangle \qquad (9.1)$$

sous les contraintes

$$u_i \geq 0 \text{ pour } i=1,.....,m \qquad (9.2)$$

avec $u=(u_1,u_2,.....,u_m)^t \in \mathbb{R}^n$.

Pour résoudre (9.1)-(9.2), nous décrivons ci-dessous la forme simplifié de la méthode du gradient conjugué conditionnel.

Soit $u^0 \geq 0$ un point arbitraire vérifiant $u_i^0 \geq 0$ pour i=1,.....,m, posons:

$\mathfrak{J}(u) = \{1 \leq i \leq m \ ; \ u_i = 0\}$.

Décrivons les opérations à partir de u^0, et calculons l'ensemble $\mathfrak{J}(u^0)$. Deux cas peuvent se présenter :

I)$[\nabla F(u^0)]^i = 0, i \notin \mathfrak{J}(u^0)$, où $[\nabla F(u^0)]^i$ est la $i^{ème}$composante du vecteur $\nabla F(u^0)$. Dans ce cas, u^0 réalise le minimum de F(u) sous les contraintes $u_i=0$, $i \in \mathfrak{J}(u^0)$. Si en plus $[\nabla F(u^0)]^i > 0$, $\forall i \in \mathfrak{J}(u^0)$, alors u^0 est solution du problème, car, en ce point, les conditions nécessaires et suffisantes du minimum sont vérifiées.

Soit $[\nabla F(u^0)]^i < 0$ pour certains $i \in \mathfrak{J}(u^0)$.

Posons $\mathfrak{J}' = \{ i \in \mathfrak{J}(u^0) : [\nabla F(u^0)]^i \geq 0\}$, pour résoudre le problème

$$\min F(u)$$
$$u_i = 0 \quad i \in \mathfrak{J}'.$$

Par la méthode du gradient conjugué, il faut toujours calculer λ_k^* :

$$\lambda_k^* = \min\{ -\frac{u_i^k}{d_k^i} : i \in J\} \quad \text{avec} \quad J=\{i \notin \mathfrak{J}' : d_k^i < 0\},$$

puis comparer λ^k et λ_k^*;

Si $\lambda^k < \lambda_k^*$ alors $u_i^{k+1} = u_i^k + \lambda^k d_k^i$, $i \notin \mathfrak{J}'$ et $u_i^{k+1} = u_i^k = 0$, $i \in \mathfrak{J}'$

Si $\lambda^k \geq \lambda_k^*$ alors $u_i^{k+1} = u_i^k + \lambda_k^* d_k^i$, $i \notin \mathfrak{J}'$ et $u_i^{k+1} = u_i^k = 0$, $i \in \mathfrak{J}'$.

Le processus s'arrête après un nombre fini de pas. On trouve u^{k+1} tel que, soit $\lambda^k \geq \lambda_k^*$, soit F(u) atteint en ce point son minimum sous les conditions $u_i=0$; $i \in \mathfrak{J}'$.

Dans chacun des cas, le processus recommence avec u^{k+1} au départ.

II) Il existe des indices i tels que $[\nabla F(u^0)]^i \neq 0, i \notin \mathfrak{J}(u^0) = \mathfrak{J}_0$. On minimise alors

$$F(u)$$
$$u_i = 0 \quad i \in \mathfrak{J}_0$$

Par la méthode du gradient conjugué. De plus, on calcule à chaque pas (comme pour I)) la quantité:

$$\lambda_k^* = \min\{ -\frac{u_i^k}{d_k^i} : i \in J\} \quad \text{avec} \quad J=\{i \notin \mathfrak{J}_0 \ : \ d_k^i < 0\}$$

pour ensuite comparer λ^k et λ_k^*;

Si $\lambda^k < \lambda_k^*$ alors $u_i^{k+1} = u_i^k + \lambda^k d_k^i$, $i \notin \mathfrak{J}_0$ et $u_i^{k+1} = u_i^k = 0$, $i \in \mathfrak{J}_0$

Si $\lambda^k \geq \lambda_k^*$ alors $u_i^{k+1} = u_i^k + \lambda_k^* d_k^i$, $i \notin \mathfrak{J}_0$ et $u_i^{k+1} = u_i^k = 0$, $i \in \mathfrak{J}_0$.

Le processus se termine comme dans la cas I).

3.3. Schéma de l'itération

Soit u^0 un point tel que $u_i^0 \geq 0$ pour $1 \leq i \leq m$ et $\mathcal{J}_0 = \{1 \leq i \leq m : u_i^0 = 0\}$.

1) Si $[\nabla F(u^0)]^i = 0, \forall i \notin \mathcal{J}_0$ on passe à 2)

Sinon, c'est-à-dire il existe $i \notin \mathcal{J}_0$ tel que $[\nabla F(u^0)]^i \neq 0$: on passe à 3).

2) Si $[\nabla F(u^0)]^i \geq 0, \forall i \in \mathcal{J}_0$,alors u^0 est une solution du problème (9.1)-(9.2).

Sinon, c'est-à-dire il existe $i \in \mathcal{J}_0$ tel que $[\nabla F(u^0)]^i < 0$, on pose :

$\mathcal{J}_0 = \mathcal{J}_0 - \{i \in \mathcal{J}_0 : [\nabla F(u^0)]^i < 0\}$ et on passe à 3).

3) Procédure du gradient conjugué pour la résolution de la minimisation de:

$$F(u) = 1/2\langle u, AC^{-1}A^t u\rangle + \langle u, AC^{-1}d + b\rangle$$

sous les contraintes ($$)

$$u_i = 0 \quad i \in \mathcal{J}_0$$

a) On prend u^0 comme point de départ et on pose $d_0 = -(I - P_{\mathcal{J}_0})\nabla F(u^0)$.

b) A l'itération k, on a u^k. Pour déterminer u^{k+1} on procède de la manière suivante : on calcule $d_k = -\nabla F(u^k) + \beta_{k-1} d_{k-1}$

$$\text{avec} \qquad \beta_{k-1} = \frac{\|\nabla F(u^k)\|^2}{\|\nabla F(u^{k-1})\|^2}$$

Le pas de déplacement est défini par:

$$\lambda^k = -\frac{\langle \nabla F(u^k), d_k\rangle}{\langle d_k, Cd_k\rangle}$$

et

$$\lambda_k^* = \min\{-\frac{u_i^k}{d_k^i} : i \in J\} \quad \text{avec} \quad J = \{i \notin \mathcal{J}_0 \ : \ d_k^i < 0\}$$

Si $\lambda^k < \lambda_k^*$ alors $u_i^{k+1} = u_i^k + \lambda^k d_k^i$, $i \notin \mathcal{J}_0$ et $u_i^{k+1} = u_i^k = 0$, $i \in \mathcal{J}_0$

faire $k \leftarrow k+1$ et retourner à b), jusqu'à obtention d'un point, noté u^*, qui réalise le minimum du problème ($$) et on passe à 4).

Si $\lambda^k \geq \lambda_k^*$ alors $u_i^{k+1} = u_i^k + \lambda_k^* d_k^i$, $i \notin \mathcal{J}_0$ et $u_i^{k+1} = u_i^k = 0$, $i \in \mathcal{J}_0$.

On pose $u^* = u^{k+1}$ et on passe à 4).

4) On pose $u^0 = u^*$ et $\mathcal{J}_0 = \{1 \leq i \leq m : u_i^* = 0\}$,puis on recommence le processus (passage à 1)).

3.4. Aspect numérique

L'algorithme du gradient conjugué conditionnel, décrit dans le paragraphe §.2.6, comprend en fait une seule itération compliquée, à savoir la projection du gradient sur un sous espace c'est-à-dire le calcul de

$$(I - P)\nabla f(x) = \nabla f(x^k) - P.\nabla f(x^k)$$

La première façon d'agir consiste à calculer directement la matrice P_J, c'est-à-dire $P_J = A_J^t (A_J A_J^t)^{-1} A_J$. Dans ce cas $(A_J A_J^t)^{-1}$ doit être calculer sans cesse.

Cette façon de faire produit l'accumulation des erreurs de calcul. Pour remédier à ce phénomène, on procède comme suit :

en posant $v^k = - (A_J A_J^t)^{-1} A_J \nabla f(x)$, le calcul à chaque itération de la quantité $(I-P_J)\nabla f(x^k)$ se ramène à celui de l'expression :

$$\nabla f(x^k) + A_J^t v^k \qquad (10.1)$$

Dans (10.1), la seule opération compliquée à effectuer est le calcul du vecteur v^k. Or, calculer v^k revient à résoudre le système suivant

$$(A_J A_J^t) v^k = -A_J \nabla f(x) \qquad (10.2)$$

Le vecteur $(I-P_J)\nabla f(x^k)$ est obtenu facilement par la formule

$$(I-P_J)\nabla f(x^k) = \nabla f(x^k) - P_J \nabla f(x^k) = \nabla f(x^k) - A_J^t (A_J A_J^t)^{-1} A_J \nabla f(x^k) = \nabla f(x^k) + A_J^t v^k.$$

Ainsi, en procédant de cette façon, le problème se ramène à une répétition d'applications d'une procédure standard pour la résolution du système (10.2).

L'utilisation du problème dual permet d'éviter la résolution d'un système d'équations à chaque itération et d'obtenir une bonne convergence de l'algorithme du gradient conjugué.

4. PROBLEME DE PROGRAMMATION QUADRATIQUE CONVEXE ET SA TRANSFORMATION EN UN PROBLEME LINEAIRE COMPLEMENTAIRE :

Dans ce paragraphe nous présentons tout d'abord la transformation d'un programme quadratique convexe (basée sur les conditions de Kuhn-Tucker) en un problème linéaire complémentaire.

Nous distinguons deux cas suivant qu'il y a ou non des contraintes de non-négativité pour la variable x.

Nous donnerons ensuite les conditions assurant la convergence de l'algorithme de pivotage de Lemke pour la résolution de ces problèmes linéaires complémentaire, faisant ainsi le lien avec le chapitre II qui traite des programmes linéaires et non linéaires complémentaires.

4.1. Transformation d'un programme quadratique convexe avec contraintes x≥0 en un programme linéaire complémentaire.

Soit

$$(P_3) : \begin{cases} \min\ 1/2 \langle x, Cx \rangle + \langle d, x \rangle \\ \text{sous les contraintes :} \\ Ax \leq b,\ x \geq 0 \end{cases}$$

où d est un n-vecteur, b est un m-vecteur.

A est une matrice de type m×n.

C est une matrice carrée d'ordre n, symétrique, semi-définie positive.

Les conditions de Kuhn-Tucker relatives au problème (P_3) permettent de dire que x est une solution de (P_3) si et seulement si il existe $u \in \mathbb{R}^m_+$ et $v \in \mathbb{R}^n_+$ tels que :

$$(*) \begin{cases} Cx + A^t u + d - v = 0 \\ u^t(b - Ax) = 0,\ v^t x = 0 \\ Ax \leq b,\ x \geq 0 \end{cases}$$

Soit $q = \begin{bmatrix} b \\ d \end{bmatrix}$, $z = \begin{bmatrix} u \\ x \end{bmatrix}$, et M la matrice de type (m+n)x(m+n) définie par :

$$M = \begin{bmatrix} 0 & -A \\ A^t & C \end{bmatrix}$$

on considère le problème linéaire complémentaire suivant (P_2) :

Chercher $z \geq 0$, $Mz + q \geq 0$ tel que $z^t(Mz+q) = 0$

Puisque la relation entre les solutions de (P_2) et (P_3) est immédiate, la résolution (P_3) se ramène à celle de (P_2) dont la résolution peut se faire à l'aide de l'algorithme de pivotage de Lemke.

4.2. Conditions assurant la convergence de l'algorithme de pivotage de Lemke pour la résolution de (P_2).

L'étude du programme linéaire complémentaire et de l'algorithme de pivotage de Lemke (pour sa résolution) présentée dans le chapitre II montre que si la matrice M est copositive plus (cf. Chapitre II) alors l'algorithme de Lemke conduit à une solution de (P_2) (et donc de (P_3)) ou à une conclusion de la vacuité de l'ensemble de solutions de (P_2) (et donc de (P_3)).

Proposition

M est copositive plus si C est symétrique semi-définie positive.

Preuve

Soit $z = \begin{bmatrix} x \\ y \end{bmatrix} \geq 0$ où $x \in \mathbb{R}_+^m$ et $y \in \mathbb{R}_+^n$, alors $z^t = (x^t \; y^t) \geq 0$

$$z^t M z = (x^t \; y^t) \cdot \begin{bmatrix} 0 & -A \\ A^t & C \end{bmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} = y^t C y$$

Puisque C est semi-définie positive alors $y^t.C.y \geq 0$ et par suite $z^t.M.z \geq 0$ pour tout $z \geq 0$, ce qui entraine que M est copositive.

$$\text{or} \quad M + M^t = \begin{bmatrix} 0 & 0 \\ 0 & 2C \end{bmatrix} \Rightarrow (M + M^t).z = \begin{bmatrix} 0 & 0 \\ 0 & 2C \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = 2Cy$$

donc : $z^t.M.z = 0 \Rightarrow y^t.C.y = 0 \Rightarrow C.y = 0 \Rightarrow (M+M^t).z = 0 \Rightarrow$ M est copositive plus. ◊

4.3. Transformation d'un proramme quadratique convexe général (sans contrainte $x \geq 0$) en un programme linéaire complémentaire.

Soit

$$(P_4) : \begin{cases} \min \; 1/2 \langle x, Cx \rangle + \langle d, x \rangle \\ \text{sous les contraintes linéaires :} \\ Ax \leq b \end{cases}$$

où d est un n-vecteur, b est un m-vecteur.

A est une matrice de type $m \times n$.

C est une matrice carrée d'ordre n, symétrique, semi définie positive.

D'après les conditions de Kuhn-Tucker relatives au problème (P_4) : x est une solution de (P_4) si et seulement si il existe $u \in \mathbb{R}^m_+$ tel que :

$$(**)\begin{cases} Cx + A^t u + d = 0 \\ \langle u, b - Ax \rangle = 0 \\ Ax \leq b \end{cases}$$

Supposons que C est définie positive. considérons alors le problème linéaire complémentaire suivant (P_5) :

Trouver $u \in \mathbb{R}^m_+$ tel que :

$$Mu + q \geq 0, \quad u^t(Mu+q) = 0$$

où $\quad M = AC^{-1}A^t, \quad q = AC^{-1}d + b$.

Puisque la relation entre les solutions de (P_4) et (P_5) est immédiate, la résolution (P_4) se ramène à celle de (P_5) dont la résolution peut se faire à l'aide de l'algorithme de pivotage de Lemke.

Il est facile de montrer les résultats suivants :

(i) u^* est solution du problème PLC (P_5) si et seulement si $x^* = (-C^{-1}A^t u^* - C^{-1}d)$ est solution du problème (P_4).
(ii) Puisque $M=AC^{-1}A^t$ est semi-définie positive, l'algorithme de pivotage de Lemke conduit à une solution de (P_5) (et donc de (P_4)) ou à une conclusion de la vacuité de l'ensemble de solutions de (P_5) (et donc de (P_4)).

Remarque :

Dans le cas 4.3. la transformation suppose que C soit définie positive. De plus l'application de l'algorithme de Lemke nécessite le calcul de C^{-1}.

Ce sont les inconvénients de la résolution de (P_4) (plus exactement la résolution du problème dual de (P_4)) par l'algorithme de pivotage de Lemke.

Dans le cas 4.1. on n'a pas ces inconvénients, par contre on est obligé de travailler dans $\mathbb{R}^{n+m}$ (au lieu de $\mathbb{R}^m$ comme dans 4.3.).
L'algorithme de Lemke serait plus coûteux lorsque n est assez grand.

5. APPLICATIONS [19] :

5.1. Régression Isotone :

Le problème de la régression isotone consiste à minimiser

$$\sum_{i=1}^{n} w_i (g_i - x_i)^2 \qquad (5.1)$$

sous les contraintes

$$x_i - x_{i-1} \leq 0 \quad \text{pour } i=2,\ldots,n \qquad (5.2)$$

où $w_i > 0$, $1 \leq i \leq n$, $g = (g_1, g_2, \ldots, g_n)^t \in \mathbb{R}^n$, $x = (x_1, x_2, \ldots, x_n)^t \in \mathbb{R}^n$.

Posons

$$W = \begin{bmatrix} w_1 & 0 & \ldots & 0 \\ 0 & w_2 & 0 & \ldots & 0 \\ 0 & 0 & w_3 & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & \ldots & \ldots & w_n \end{bmatrix} \quad \text{et} \quad A = \begin{bmatrix} 1 & -1 & 0 & \ldots & \ldots & 0 \\ 0 & 1 & -1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & -1 & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & \ldots & 1 & -1 \end{bmatrix}$$

La matrice W est une matrice diagonale de type nxn, symétrique, définie positive, tandis que la matrice A est de type (n-1)xn et de rang (n-1). Ainsi le problème (5.1)-(5.2) est équivalent au programme quadratique

$$\min 1/2\langle x, Wx\rangle + \langle x, -Wg\rangle \qquad (5.3)$$

sous les contraintes

$$Ax \leq 0 \qquad (5.4)$$

Selon l'étude faite au paragraphe **3.**, le problème dual du problème de régression isotone s'écrit ainsi :

$$\min 1/2\langle u, AW^{-1}A^t u\rangle + \langle u, -Ag\rangle \qquad (5.5)$$

sous les contraintes

$$u_i \geq 0, \quad 1 \leq i \leq n-1 \qquad (5.6)$$

La matrice qui définit le terme quadratique est donnée par

$$AW^{-1}A^t = \begin{bmatrix} \frac{1}{w_1}+\frac{1}{w_2} & -\frac{1}{w_2} & 0 & \cdots & 0 \\ -\frac{1}{w_2} & \frac{1}{w_2}+\frac{1}{w_3} & -\frac{1}{w_3} & 0 \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & -\frac{1}{w_{n-1}} \\ 0 & 0 & 0 & -\frac{1}{w_{n-1}} & \frac{1}{w_{n-1}}+\frac{1}{w_n} \end{bmatrix}$$

Une fois la solution duale u* est déterminée, on déduit immédiatement la solution du problème de la régression isotone (cf. § 3.1) :

$$x^* = -W^{-1}A^t u^* + W^{-1}g.$$

Tout calcul fait, la valeur explicite de x* est égale à :

$$x^* = \left(\frac{g_1-\dot{u_1}}{w_1}, \frac{\dot{u_1}}{w_2}+\frac{g_2-\dot{u_2}}{w_2}, \ldots\ldots\ldots, \frac{\dot{u}_{n-2}}{w_{n-1}}+\frac{g_{n-1}-\dot{u}_{n-1}}{w_{n-1}}, \frac{g_n+\dot{u}_n}{w_n}\right)^t$$

Pour la résolution du problème de la régression isotone, il est préférable de passer au problème dual, qui présente des avantages sur le problème primal.

5.2. Régression concave :

Le problème de la régression concave s'écrit comme suit :

$$\min \sum_{i=1}^{n} w_i (g_i-x_i)^2 \qquad (5.7)$$

sous les contraintes

$$\frac{x_i-x_{i-1}}{y_i-y_{i-1}} \leq \frac{x_{i+1}-x_i}{y_{i+1}-y_i} \quad \text{pour } i = 2,\ldots\ldots\ldots,n-1 \qquad (5.8)$$

avec $w_i > 0$, $1 \leq i \leq n$, $g = (g_1, g_2, \ldots, g_n)^t \in \mathbb{R}^n$, $x = (x_1, x_2, \ldots, x_n)^t \in \mathbb{R}^n$.
Les variables $y_i > 0$, $1 \leq i \leq n$, vérifient la relation $0 < y_1 < y_2 < \ldots\ldots\ldots < y_n$.

Posons

$$A = \begin{bmatrix} y_3-y_2 & y_3-y_1 & y_2-y_1 & 0 & 0 & 0.....0 \\ 0 & y_4-y_3 & y_4-y_2 & y_3-y_2 & 0 & 0.....0 \\ \multicolumn{6}{c}{.....} \\ \multicolumn{6}{c}{.....} \\ \multicolumn{6}{c}{.....} \\ 0 & 0 & 0..... & y_n-y_{n-1} & y_n-y_{n-2} & y_{n-1}-y_{n-2} \end{bmatrix}$$

La matrice A est de type (n-2)xn et de rang n-2.

Comme dans le cas (5.1), le problème de la régression concave est équivalent au problème de programmation quadratique suivant :

$$\min 1/2\langle x,Wx\rangle + \langle x,-Wg\rangle \qquad (5.9)$$

sous les contraintes

$$Ax \leq 0 \qquad (5.10)$$

Comme c'est indiqué dans §3.1, les expérimentations numériques (cf. Chapitre VI) sur une série de problèmes de régression isotone et de régression concave prouvent qu'il est beaucoup plus économique de résoudre ces problèmes par l'application du gradient conjugué conditionnel aux problèmes duaux.

Références

[1] L.COLLATZ and W.WETTERLING (1975)
Optimisation problems.
Springer-verlag New York Heidelberg Berlin.
[2] B.PCHENITCHNY et Y.DANILINE (1977)
Méthodes numériques dans les problèmes d'extrémum.
Edition Mir.
[3] P.JOLY (1986)
Présentation de synthèse des méthodes de gradient conjugué.
Mathématical Modelling and Numerical Analysis, Vol 20, 4, pp. 639-665.
[4] G.ZOUTENDIJK (1966)
Nonlinear Programming : a numerical survey.
SIAM Journal on Control 4, 1 , pp 194-210.
[5] L.S.JACOBY , J.S.KOWALIK and J.T.PIZZO (1972)
Iterative Methods for Nonlinear Optimisation Problems.
Prentice-Hall,Inc, New-Jersey.
[6] N.GASTINEL (1966)
Analyse Numérique Linéaire.
Hermann, Paris.
[7] AUSLENDER (1976)
Optimisation : Méthodes numériques.
Masson, Paris.
[8] J.CEA (1971)
Optimisation : Théorie et Algorithmes.
Dunod, Paris.
[9] M.MINOUX (1983)
Programmation mathématique : théorie et algorithmes.
Dunod, Paris.
[10] P. J. LAURENT
Approximation et Optimisation.
Herman, Paris.
[11] T.ROCKAFELLAR (1972)
Convex Analysis.
Princeton University Press, Princeton.
[12] P.E.GILL, W.MURRAY and M.H.WRIGHT (1981)
Pratical Optimisation.
Academic Press Londin New York Toronto .
[13] I. EKELAND ET R. TEMAM (1974)
Analyse convexe et problème variationnels.
Dunod, Paris.
[14] P. LANCASTER [1969]
Theory of Matrices.
Academic Press.

[15] A. LENT and Y. CENSOR (1980)
Extentions of Hildreth's Row-Action Method foe Quadratic Programming. SIAM J.Control and Optimization, vol 18, 4, pp. 444-454.
[16] M. A. BOUGHAZI (1987)
Contribution à l'etude des algorithmes d'optimisation en Analyse des Données.
Thèse de Doctorat à l'Université Scientifique, Technologique, et Médicale de Grenoble (USTMG).
[17] G. DROUET D'AUBIGNY, T. PHAM DINH and A. YASSINE (1988)
A conditional conjugate gradient method for solving the isotonic and concave regression problems.
to appear in Computational Statistics & Data Analysis.
[18] A. YASSINE, T. PHAM DINH and G. DROUET D'AUBIGNY (1988)
Numerical Experimentation of conditional conjugate gradient algorithm for solving isotonic and concave regression problems.
Technical Report, TIM3, Univ. of Grenoble.
[19] G. DROUET D'AUBIGNY (1989)
L'analyse Multidimensionnelle des données de dissimilarité. Thèse d'Etat, Université Joseph Fourier - Grenoble I. Grenoble.
[20] D. G. LUENBERGER (1972)
Introduction to linear and nonlinear programming. Addison-Welsey.
[21] M.S.BAZARAA and C.M.SHETTY (1972)
Non Linear Programming.

CHAPITRE IV

ETUDE ADAPTATIVE ET IMPLEMENTATION EFFECTIVE DE CERTAINS ALGORITHMES POUR LA MINIMISATION D'UNE FONCTION CONCAVE SUR UN ENSEMBLE CONVEXE. EXPERIMENTATIONS NUMERIQUES COMPARATIVES.

1. Introduction.

Ce chapitre est consacré à l'étude de certaines méthodes qui permettent d'obtenir les solutions globales du problème d'optimisation suivant :

$$(P) : \begin{cases} \min \ f(x) \\ x \in S = \{x / g_i(x) \leq 0 \ ; i=1, \ldots, m\} \end{cases}$$

où f est une fonction concave définie sur $\mathbb{R}^n$ et g_i (i = 1, ..., m) sont des fonctions convexes sur $\mathbb{R}^n$. On suppose que S est un ensemble compact et que son intérieur (absolu) est non vide (i.e. int(S) $\neq \emptyset$).

Il s'agit des problèmes d'optimisation non convexe dont la difficulté majeure réside dans le fait qu'il n'existe pas de caractérisation complète de solution (i.e. sous forme de condition nécessaire et suffisante) comme en optimisation convexe.

Nous présentons dans les deux paragraphes de ce chapitre les deux algorithmes dus à H. Tuy et K. Hofman respectivement pour résoudre le problème (P).

(i) La première méthode (méthode de partitionnement conique) basée sur l'introduction du procédé de subdivision de S et sur le calcul d'un certain minorant de f(x) dans chaque sous-domaine. Ceci, jusqu'à ce que l'on trouve une solution globale.

(ii) La deuxième utilise le principe de l'approximation extérieure de l'ensemble admissible S et la technique de coupes planes. L'idée générale de cette méthode est de remplacer (P) par un sous problème (P_k) de la forme $\min\{f(x) : x \in S_k\}$ (où $S \subset S_k$, qui est en général un polyèdre convexe) plus simples à mettre en oeuvre.

On fera alors une comparaison entre la méthode de résolution par partition de l'ensemble admissible et la méthode d'approximation extérieure.

Dans la classification de problème d'optimisation non convexe faite par Pham Dinh Tao & El Bernoussi Souad [20,23], le problème (P) est un cas particulier du problème suivant :

(P*) $\quad \min\{f(x) : x \in C, g(x) \geq 0\}$ (f, g, C convexes).

L'étude des algorithmes pour la résolution du problème (P*), en dehors de l'aspect unificateur, permet d'obtenir un algorithme original (avec ses différentes versions). La description de cet algorithme est présentée dans §4.

Nous avons implémenté ces algorithmes sur PC (Olivetti_380) en langage PASCAL et les expérimentations numériques comparatives sont données à la fin de ce chapitre.

2. Algorithme de résolution par partition de l'ensemble admissible.

Soit le problème (P) : min { f(x) ; x ∈ D } où f : $\mathbb{R}^n \mapsto \mathbb{R}$ est une fonction concave et D est un ensemble fermé, convexe, non nécessairement borné sur $\mathbb{R}^n$.

Nous ferons les hypothèses suivantes

H1- L'ensemble D est défini par :
D = {x∈ $\mathbb{R}^n$ / $g_i(x) \leq 0$, i= 1,...,m}
où g_i(i= 1,...,m) : $\mathbb{R}^n \mapsto \mathbb{R}$ sont des fonctions convexes dans $\mathbb{R}^n$.

H2- L'origine O de $\mathbb{R}^n$ appartient à l'intérieur de l'ensemble D i.e. O est un point admissible (condition de Slater) .

2.1 Description de l'algorithme général

On initialise avec :

- Un n-simplexe $T[s^1,s^2,...,s^{n+1}]$ (de dimension n et de sommets s^i(i=1,...,n+1) contenant l'origine O (i.e. O ∈ int(T)).

- $\mathcal{M}_0 = \{M_{0,1}, M_{0,2}, ..., M_{0,n+1}\}$ l'ensemble fini des cônes polyédriques de sommet l'origine est tel que chaque cône $M_{0,i}(s^1, s^2,..., s^{i-1}, s^{i+1},...,s^{n+1})$ engendré par les sommets $(s^1, s^2,..., s^{i-1}, s^{i+1},...,s^{n+1})$, soit défini par :
$M_{0,i} = \{x\in \mathbb{R}^n / x = \lambda_1 s^1 + ... + \lambda_{i-1}s^{i-1} + \lambda_{i+1}s^{i+1} + ... + \lambda_{n+1}s^{n+1} ; \lambda_i \geq 0\}$

- Pour tout cône M∈ $\mathcal{M}_0$, on définit une minorante d'estimation μ(M) de f(x) dans M∩D, i.e. $\mu(M) \leq \inf \{ f(x) / x\in M\cap D \}$.
- On remarque que D ⊂ $\mathcal{M}_0$ et pour tout cône M∈ $\mathcal{M}_0$ si l'arête de M n'est pas contenue dans D alors elle coupe D en un segment de droite [O,z*] d'origine O ; z* ≠ O .

Si pour un certain j∈ J={1,...,n+1}, l'arête passant par s^j est inclue dans D et si la fonction f n'est pas bornée sur cette arête, alors inf {f(x) / x∈ D} = -∞ et le problème (P) n'admet pas une solution finie. Sinon on prend: k = 0;
x^0 = argmin { f(x) / x=θs^j ; θ ≥0 ; θs^j∈ D ; j∈J } et on pose $\gamma_0 = f(x^0) > -\infty$.

Itération k = 0,1,2,.....

On a x^k et $\gamma_k = f(x^k)$, $\mathcal{M}_k$ l'ensemble des cônes restants et, pour chaque cône $M \in \mathcal{M}_k$, sa minorante d'estimation $\mu(M)$.

étape 1: On pose $\mathcal{R}_k = \{ M \in \mathcal{M}_k / \mu(M) < \gamma_k \}$.
Si $\mathcal{R}_k = \varnothing$ on s'arrête : x^k est un minimum global de f sur D .
Sinon on choisit le cône $M_k(v^1,...,v^n) \in \mathcal{R}_k$ associé à la plus petite minorante d'estimation i.e. $M_k = \text{argmin} \{ \mu(M) ; M \in \mathcal{R}_k \}$.

étape 2: On subdivise le cône $M_k(v^1,...,v^n)$ en deux sous cônes $M_{k,1}$ et $M_{k,2}$ de la manière suivante :
On calcule $w=(v^r+v^s)/2$ où $||v^r - v^s|| = \max(||v^i - v^j|| ; i \neq j)$ et on pose:
$M_k(v^1,..,v^n) = M_{k,1}(v^1,..,v^{r-1},w,v^{r+1},..,v^n) \cup M_{k,2}(v^1,..,v^{s-1},w,v^{s+1},v^n)$.

étape 3: Pour i=1,2 on calcule $\mu(M_{k,i})$ et on détermine :
$x^* = \rho^*.w$ où $\rho^* = \max\{ \rho \geq 0 ; \rho.w \in \partial D \}$ (∂D est la frontière de D).

étape 4: Si pour certaines arêtes x* n'existe pas (i.e. elle est contenue dans D) et f(x)<f(0) pour certains x appartenant a cette arête alors on s'arrête : f n'est pas bornée sur cette arête et le problème (P) n'admet pas une solution finie. Sinon on calcule :

x^{k+1} = argmin $\{f(x^k),f(x^*)\}$, $\gamma_{k+1} = f(x^{k+1})$

$\mathcal{M}_{k+1} = (\mathcal{R}_k - M_k) \cup \{ M_{k,1} , M_{k,2} \}$, k = k+1 et on retourne à l' itération k.

2.2. Conditions de convergence:

Définition 1:
Etant donné un cône polyédrique M de sommet l'origine,on dit que le processus de partition du cône M est **exhaustif** si toute sous-suite infinie, décroissante (M_i^k) de la suite de cônes $M_i \subset M$, tend vers une demi-droite d'origine O, i.e. $\cap M_i^k = \{x = \rho u / u \in M, \rho \geq 0\}$.

Définition 2:
Etant donné un cône polyédrique M de sommet l'origine,on dit que le procédé de minorante d'estimation de f(x) dans $D \cap M$ est **compatible** si, pour toute sous-suite infinie décroissante (M_i^k) de la suite (M_i) qui tend vers une demi-droite d'origine O on a : $\mu(M_i^k) - \gamma_i^k \mapsto 0$ quand $k \mapsto \infty$.

Définition 3:

On dit que le procédé de la subdivision sur l'ensemble admissible est **complet**, si le processus de partition par les cônes est **exhaustif** et le calcul de minorante d'estimation est **compatible.**

Théorème 1

Si le procédé de la subdivision de l'ensemble admissible D, décrit dans l'algorithme ci-dessus, est complet, alors on a les propriétés suivantes :

1- Si l'algorithme s'arrête à l'itération k, alors x^k est un minimum global du problème (P).

2- Tout point d'adhérence de la suite infinie $\{x^k\}$ est un minimum global.

3- Si x^* est un minimum global du problème (P) alors :

$f(x^k) - f(x^*) \leq \gamma_k - \mu(M_k)$.

Preuve :

Soient $\mu^* = \min \{ f(x) ; x \in D \}$ et $\mu(M_k) = \min \{ \mu(M) ; M \in \mathfrak{R}_k \}$.

Par définition de $\mu(M_k)$ on a : $\mu(M_k) \leq \mu^*$.

1) On s'arrête à la $k+1^{ème}$ itération si $\mathfrak{M}_{k+1}=\emptyset$ et dans ce cas on aura : $\mu(M_k) \geq \mu^* \geq \gamma_k = f(x^k)$. D'où par définition de μ^* : $\mu^* = \gamma_k = f(x^k)$.

2) Puisque le processus de subdivision est complet, alors il existe une suite infinie (M_k^q) de cônes dont l'intersection tend vers une demi-droite d'origine O et $\mu(M_k^q) - \gamma_k^q \mapsto 0$ quand $q \mapsto \infty$.

Par construction, on a : $f(x) \geq \mu(M_k^q)$, $\forall\, x \in D$.

Puisque f est continue et D compact,alors on déduit que :

la limite x^* de la sous-suite $\{x_k^i\}$ de la suite $\{x^k\}$ appartient à D ;

$f(x_k^i) = \gamma_k^i \mapsto f(x^*)=\gamma^*$ et $f(x) \geq f(x^*)=\gamma^*=\mu^*$.

3) Par construction. ◊

Théorème 2: [6]

Cet algorithme se termine après un nombre fini d'itérations par l'une des deux possibilités:

1- Avec une solution optimale du problème (P)

2- Avec une arête dans D sur laquelle la fonction f n'est pas bornée.

ou bien il est infini, dans ce cas si le procédé de la subdivision est complet alors : $f(x^k) \mapsto \inf\{ f(x) ; x \in D \}$. ◊

2.3.Mise en oeuvre de l'algorithme.

A) Règle de choix.

On choisit le cône $M_k = \text{argmin} \{ \mu(M) ; M \in \mathfrak{R}_k \}$.

B) Règle de subdivision.

Soit $M_k(v^1,...,v^n)$ le cône que l'on va subdiviser .
On calcule $w=(v^r+v^s)/2$ où $||v^r - v^s|| = \max(||v^i - v^j|| ; i \neq j)$ et on pose:
$M_{k,1}(v^1,..,v^{r-1},w,v^{r+1},..,v^n)$ et $M_{k,2}(v^1,..,v^{s-1},w,v^{s+1},v^n)$ on aura :

Lemme 1. [6,16]

$M_k(v^1,...,v^n) = M_{k,1}(v^1,..,v^{r-1},w,v^{r+1},..,v^n) \cup M_{k,2}(v^1,..,v^{s-1},w,v^{s+1},v^n)$.
Le processus de subdivision est exhaustif.

C) Règle de limitation.

Il est très important de savoir comment on calcule pour chaque cône M, la valeur : $\mu(M) \leq \inf \{ f(x) / x \in M \cap D \}$. L'idée est très simple, on coupe le cône M par l'hyperplan tangent sur D en n'importe quel point $z \in M \cap \partial D$, ce qui revient alors à définir un simplexe contenant $M \cap D$ tel que $\mu(M)$ est le minimum de f sur ce simplexe .

Soient $v^1,...,v^n$ les points d'intersection des n arêtes du cône M avec ∂T (∂T est la frontière du simplexe T) .
Pour j=1,..n, on calcule $\theta_j = \sup \{ \theta; \theta \geq 0 \text{ et } g_i(\theta v^j) \leq 0 \ (i=1,..,m)\}$
Soit $z \in M \cap \partial D$ tel que $\theta.z \notin D$ pour $\theta > 1$ alors il existe un indice i tel que :

$g_i(0) < 0$ et $g_i(z) = 0$ (*).

On distingue trois cas :
1- Il existe un j tel que $\theta_j = +\infty$ et $f(\theta v^j) < f(0)$ pour un certain $\theta > 0$, alors la $j^{ème}$ arête de M est inclue dans D et f n'est pas bornée sur cette arête. Par suite, $\mu^* = -\infty$ et l'algorithme s'arrête .
2- Si $f(\theta v^j) \geq f(0)$ pour tout $\theta > 0$ et pour chaque indice j, alors $\mu(M) = f(0)$.
3- Si aucun des deux cas précédents n'a lieu, alors il existe un j tel que $\theta_j < +\infty$. On prend $z=z(M) \in M \cap \partial D$ tel que $\theta.z \notin D$ pour $\theta > 1$.
Soit g_i une contrainte vérifiant (*) on choisit un $t \in \partial g_i(z)$ (un sous-gradient de g_i en z) . Ainsi $g_i(z) = 0 \Rightarrow g_i(x) \geq \langle t, x-z \rangle \quad \forall x \in \mathbb{R}^n$ (**).
D'aprés (**) on a :
$D \subset H = \{ x \in \mathbb{R}^n / \langle t, x-z \rangle \leq 0 \}$, ce qui justifie l'estimation suivante :

cas 3a). $<t,v^j> >0 \ \forall\ j=1,...,n$. Alors l'hyperplan $<t,x-z> =0$ coupe chaque arête de M, on prend :

$z^j = \tau_j.v^j$ où $\tau_j = <t,z>/<t,v^j> \ \forall\ j=1,...,n$, alors :

$\alpha(M) = \min\{f(x) / x \in M \cap H\} = \min\{f(0),f(z^1),f(z^2),..,f(z^n)\}$.

$$\text{De plus : } \begin{cases} \alpha(M) & \text{Si } M \in \mathcal{M}_0 \\ \max\{\alpha(M),\mu(M_{anc})\} & \text{Sinon} \end{cases}$$

où M_{anc} désigne l'ancien cône M .

cas 3b). Il existe un j tel que $<t,v^j> \leq 0$ alors la j^{ème} arête de M est inclue dans H avec $M \cap H$ non borné .Dans ce cas :

$$\mu(M) = \begin{cases} -\infty & \text{Si } M \in \mathcal{M}_0 \\ \mu(M_{anc}) & \text{Sinon} \end{cases}$$

Lemme 2. Le procédé de minorante estimation est compatible .

Preuve: **[6**, lemme 5]. ◊

2.4. Remarques

Si on suppose que l'origine O n'est pas un point admissible,i.e. n'est pas à l'intérieur de D. Deux cas se représentent :

1- Si D est un polytope et admet un point extrême non dégénéré (sans perdre de généralité, on suppose que ce point est l'origine O),alors on prend $\mathcal{M}_0 = \{M_0\}$ où M_0 est le plus petit cône de sommet O contenant D et on applique l'algorithme précédent.

2- Si D est un ensemble compact, convexe, défini par (H1) tel que $O \notin int(D)$, alors on considère le problème auxilliaire :

$$(P') : \min\{ y / g_j(x) - y \leq 0 \ \ (i = 1,...,m)\} .$$

Soit $x^0 \in \mathbb{R}^n$ et un nombre z^0 très grand tel que :
$g_j(x^0) - z^0 \leq 0$ $(i = 1,...,m)$, alors $(x^0,z^0) \in \mathbb{R}^{n+1}$ est un point intérieur à l'ensemble admissible de (P') .

A partir de ce point, on résoud (P') par n'importe quelle méthode de programmation convexe (car (P') est équivalent à trouver le minimum de la fonction convexe : $g(x) = \max [g_j(x) ; i = 1,...,m]$).

Si d'après un nombre fini d'étapes, on obtient un point (x^+,z^+) tel que $g_i(x^+) - z^+ < 0$ $(i = 1,...,m)$, alors x^+ est un point intérieur à D (l'ensemble admissible du problème original (P)).

On part de x^+ en appliquant l'algorithme précédent pour résoudre (P). Si la valeur optimale z^* de (P') est positive alors (P) n'admet pas des solutions .

Si $z^*=0$, alors $int(D)=\emptyset$; dans ce cas on peut trouver une solution approchée pour (P) si z^* est trop petit ($z^*< \varepsilon$).

2.5. Application à un problème complémentaire convexe.

Un problème complémentaire convexe consiste à trouver un vecteur $x\in \mathbb{R}^n$ tel que :

$$(C) \quad : \quad x\geq 0 \ ; \ w(x)\geq 0 \ ; \ \langle x,w(x)\rangle = 0 .$$

où $w : \mathbb{R}^n \mapsto \mathbb{R}^n$ est une fonction concave i.e.
$w_i : \mathbb{R}^n \mapsto \mathbb{R}$ est une fonction concave dans $\mathbb{R}^n$ $\forall i=1,...,n.$ (si $w(x) = M.x+q$ où $M\in \mathbb{R}^{n\times n}$ et $q \in \mathbb{R}^n$ on obtient le problème linéaire complémentaire).

Si avec le problème (C) on associe la fonction : $f(x) = \sum_{i=1}^{n} \min (x_i ,w_i(x))$ et on prend:
$D = \{x \in \mathbb{R}^n / x\geq 0$ et $w(x)\geq 0\}$, alors le problème (C) se représente par :

$$(Q) : \quad \text{Trouver} \quad x \in D \ ; f(x) =0 .$$

Donc le problème (C) se ramène à résoudre le problème :
$0 = \min \{f(x) ; x \in D\}$ où $f : \mathbb{R}^n \mapsto \mathbb{R}$ est une fonction concave à valeurs réelles et D est un ensemble convexe sur $\mathbb{R}^n$.

Remarques.

1- Pour tout $x \in \mathbb{R}^n$, on a $f(x) \geq 0$; et si, à l'étape k on a $f(x^k) = 0$, alors on s'arrête : x^k est une solution globale optimale de (C) .

2- On utilise une règle spéciale de choix :
Soit $\varrho_j = \sup\{ \theta \geq 0 ; f(A + \theta.(v^j - A)) >0\}$; $j=1,...,n.$
$\varrho(M) = \inf\{ \varrho_1,\varrho_2,...,\varrho_n\}$; on prend $M_k \in \operatorname{argmin} \{ \varrho(M) ; M\in \mathfrak{R}_k \}$ (***).

3- Pour calculer $\mu(M)$, on distingue deux cas :
1er cas) $\rho_j = +\infty \ \forall\ j=1,...,n$ alors $\mu(M) > 0$.
2èmcas) S'il existe un indice $j \in \{1,..,n\}$ alors la jème arête de M coupe ∂D en un point z (on prend z quelconque dans $M \cap \partial D$ et soit H est le plan tangent en z ($H=\{ x \in \mathbb{R}^n / \langle t, x-z \rangle = 0 \}$) alors :

$$\mu(M) \begin{cases} >0 & \text{Si } \inf \{ f(x) / x \in D \cap H \} > 0 \\ =0 & \text{Sinon} \end{cases}$$

4- $\mu(M) \geq 0 \ \forall M$. Dans chaque étape, on élimine les cônes M tel que: $\mu(M) > 0$.

5- Si dans l'étape k, on a $\mu(M) > 0 \ \forall M \in \mathfrak{R}_k$ alors le problème (Q) n'admet pas de solution .

Algorithme

A) Initialisation:
On prend un point A appartenant à l'intérieur de D .
On définit un n-simplexe $T[s^1, s^2, ..., s^{n+1}]$ qui contient A (i.e. $A \in int(T)$).
$\mathcal{M}_0 = \{M_{0,1}, M_{0,2}, ..., M_{0,n+1}\}$ l'ensemble fini des cônes polyédriques de sommet A et engendré par les sommets ($s^1, .., s^{i-1}, s^{i+1}, .., s^{n+1}$) .
Pour tout cône $M \in \mathcal{M}_0$, on calcule $\mu(M) \leq \inf \{ f(x) / x \in M \cap D \}$. Soient:
$x^0 \in \text{argmin} \{ f(x) / x = \theta s^j ; \theta \geq 0 ; \theta s^j \in D ; j \in J \}$, $\gamma_0 = f(x^0)$ et $k=0$

Itération k = 0,1,2,.....

étape 1: Si $\gamma_k = f(x^k)$, on s'arrête : x^k est une solution globale du problème (Q) et $(x^k, w(x^k))$ est une solution du problème (C).
Sinon, on supprime chaque cône $M \in \mathcal{M}_k$ tel que $\mu(M) > 0$.

étape 2 : Soit $\mathfrak{R}_k = \{ M \in \mathcal{M}_k / \mu(M) = 0 \}$.
Si $\mathfrak{R}_k = \varnothing$ on s'arrête : (Q) n'admet pas de solution et par suite le problème original (C) n'en admet pas .
Sinon, on choisit $M_k \in \mathfrak{R}_k$ (vérifiant la règle spéciale (***)).
On divise le cône $M_k(v^1, ..., v^n)$ en deux sous-cônes $M_{k,1}$ et $M_{k,2}$ par la règle de subdivision et on pose:

$$M_k(v^1,..,v^n) = M_{k,1}(v^1,..,v^{r-1},w,v^{r+1},..,v^n) \cup M_{k,2}(v^1,..,v^{s-1},w,v^{s+1},v^n)$$

étape 3 : Pour chaque i=1,2 on calcule $\mu(M_{k,i})$ et on détermine :
$x^* = A+\varrho^*.(w-A)$ où $\varrho^* = \max\{ \varrho \geq 0 ; A+\varrho^*.(w-A) \in \partial D \}$ (∂D est la frontière de D)

étape 4 : $x^{k+1} = \text{argmin } \{f(x^k), f(x^*)\}$, $\gamma_{k+1} = f(x^{k+1})$
$\mathcal{M}_{k+1} = (\mathcal{R}_k - M_k) \cup \{ M_{k,1} , M_{k,2} \}$, k =k+1 et on retourne à l' itération k.

Proposition 1.

Si le problème (Q) admet une solution, alors l'algorithme termine après un nombre fini d'itérations avec une solution de (C) ou il génère une suite $\{x^k\} \subset D$ tel que $f(x^k) \to 0$. ◊
La démonstration de cette proposition résulte de deux lemmes suivants:

Lemme 3. [6]

Si l'algorithme génère une suite infinie décroissante des cônes $\{M_{kq}\}$ dont l'intersection est une arête $\Gamma \subset D$, alors le problème (Q) n'admet pas de solution. ◊

Lemme 4. [6]

Si l'algorithme génère une suite infinie décroissante des cônes $\{M_{kq}\}$ dont l'intersection est une arête Γ telle que $\Gamma \cap D$ est un segment de droite [A,z*] avec $z^* \neq A$, alors z^* est la solution de (Q). ◊

Proposition2.[6]

Pour chacun des deux nombres positifs ε et N, l'algorithme donne après un nombre fini d'itérations une ε -approximation solution du (C) i.e. un point $x^k \in D$ tel que : $f(x^k) < \varepsilon$ ou il établit que le problème n'admet pas de solutions dans la boule $B=\{ x \in \mathbb{R}^n / ||x|| < N \}$. ◊

3. Méthodes de résolution par approximation extérieure de l'ensemble admissible.

On considère le problème :

$$(P) : \begin{cases} \min \ f(x) \\ g_i(x) \leq 0 \ ; \ i=1, \ldots, m \end{cases}$$

où f est une fonction concave définie et continue sur $\mathbb{R}^n$ et g_i (i=1,..., m) sont des fonctions convexes sur $\mathbb{R}^n$ dont les gradients sont continus. On suppose que $S = \{x / g_i(x) \leq 0 ; i = 1, \ldots, m\}$ est compact non vide.

On notera par E(D) l'ensemble des points extrémaux de tout polyèdre convexe non vide $D \subset \mathbb{R}^n$.

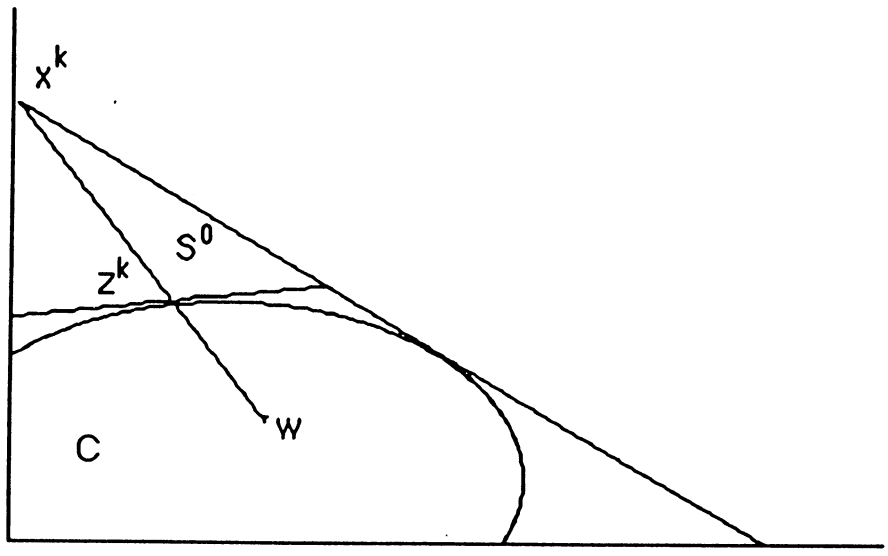
3.1. Description de l'algorithme

On initialise avec un polyèdre borné S^0 contenant S et un point w de l'intérieur de S.

Itération k: (k=0,1,.....)

étape 1: On calcule $f(x^k) = \max\{f(x) / x \in E(S^k)\}$. Si $x^k \in S$, alors on s'arrête et x^k est une solution globale du problème (P).
Sinon, on détermine le point $z^k = \theta w + (1- \theta) x^k$ tels que $0< \theta <1$ et $h(z^k) = \max(g_i(z^k) : i= 1,...,m)=0$.

étape 2 : On prend le sous différentiel ρ^k de h en z^k et on calcule l'ensemble $E(S^{k+1})$ du polyèdre $S^{k+1} = S^k \cap \{x \in \mathbb{R}^n / < \rho^k, x- z^k> \leq 0\}$ (cf. Chap. I, algorithme S.G.G.P). On pose k=k+1 et on retourne à l'étape 1.



3.2. Mise en oeuvre de l'algorithme

Le sous-problème principal dans cet algorithme consiste à déterminer, à chaque itération, l'ensemble des sommets du polyèdre S^{k+1} (cf. Chap. I, algorithme S.G.G.P). Par construction de cette suite de polyèdres, la résolution de chaque sous-problème est basée sur le lemme suivant.

Lemme 2.1. [7]

Soient $D \subset \mathbb{R}^n$ un polyèdre borné non vide et $H^- \subset \mathbb{R}^n$ un demi espace fermé tel que $D \cap H^+ \neq \emptyset$, où $H^+ = \mathbb{R}^n - H^-$. Le point $v \in \mathbb{R}^n$ est un sommet du polyèdre $D \cap H^-$ et non du polyèdre D si et seulement si il est de la forme $v =]v^1, v^2[\cap \partial H^-$, où v^1 et v^2 sont deux sommets adjacents du polyèdre D tels que $v^1 \in H^+$, $v^2 \in (H^- - \partial H^-)$ et $]v^1, v^2[$ est l'arête joignant ces deux sommets. ◊

En pratique, on détermine les éléments de l'ensemble $E(S^{k+1})$ par l'algorithme S.G.G.P (cité dans le Chap. I) ou par une succession d'opérations de pivotage [8] de la manière suivante:

On pose : $D = S^k$, $H^- = \{x \in \mathbb{R}^n / < \rho^k, x - z^k > \leq 0\}$,

$H^+ = \{x \in \mathbb{R}^n / < \rho^k, x - z^k > > 0\}$ et $\partial H^- = \{x \in \mathbb{R}^n / < \rho^k, x - z^k > = 0\}$.

* On détermine l'ensemble $V = \{v \in E(S^k) / v \notin S^{k+1}\}$.
* Pour chaque $v \in V$, on calcule les éléments de l'ensemble A(v) suivant : $A(v) = \{u / u =]v, y[\cap \partial H^- ; y \in E(v)\}$, où E(v) est l'ensemble des sommets adjacents au sommet v appartenant à $(H^- - \partial H^-)$.
* On prend $E(S^{k+1}) = E(S^k) \cup_{v \in V} A(v) - V$.

3.3. Convergence de l'algorithme

Théorème [6, 16] :

Pour toute suite (x^k) générée par l'algorithme ci-dessus, on a les propriétés suivantes:

1-Si l'algorithme s'arrête à l'itération k, alors x^k est un minimum global du problème (P).

2- Si les contraintes $g_i(x) \leq 0$ $(i = 1,...,m)$ sont linéaires, alors l'algorithme converge en un nombre fini d'itérations.

3- Tout point d'adhérence de la suite (pour tout $k>0$) est solution du problème (P). ◊

Preuve

1- Par construction, on a : $S \subset ... \subset S^{k+1} \subset S^k \subset ... \subset S^0$ pour tout $k>0$
Comme f est concave alors pour tout $k\geq 0$, x^k réalise le minimum global de f(x) dans S^k. Donc x^k est une solution globale si on s'arrête à l'itération k.
2- Si les fonctions $g_i (i= 1,...,m)$ sont linéaires alors chaque $k^{ème}$ itération consiste à ajouter une contrainte violée g_j en x^k.Comme le nombre des contraintes est fini alors le nombre des itérations est inférieur ou égal à m.
3- D'après (1),il suffit de montrer que tout point d'adhérence de la suite (x^k) appartient à S.
Comme les suites $(\rho^k),(z^k),(x^k)$ sont bornées alors elles possèdent toujours des points d'adhérances .Soit $a^k(x) = < \rho^k, x- z^k >$.

Par construction, on a les inégalités suivantes :

$a^k(x^k) > 0$ pour tout $k\geq 0$ et $a^k(x^j) > 0$ pour tout $j>k$ (1)

Supposons qu'il existe un point limite x^* de la suite (x^k) tel que $x^* \in S$.
Dans le cas contraire,il existe une sous-suite (k_i) d'indices telle que :
$x^{k_i} \mapsto x^*$, $a^{k_i}(x^{k_i}) \mapsto a(x^*) > 0$ et $a^{k_i}(x^{k_j}) \mapsto a(x^*)$ quand $i,j \mapsto \infty$
or par passage à la limite dans (1) on a: $a(x^*) < 0$ ce qui contredit $a(x^*) > 0$.

Comme application de ce genre d'algorithmes nous avons adapté l'algorithme de Falk-Hofman décrit ci-dessous :

3.4. Algorithme de Falk-Hofman.

Soit p un point intérieur du domaine admissible, i.e. $p \in \{x \in \mathbb{R}^n : g_i(x) < 0 ; i = 1, ... , m\}$.

étape 0 : Choisir $\varepsilon > 0$ un réel suffisamment petit et un point p intérieur à S, soit $u^0 = f(p)$. Construire un polyèdre borné $S^0 = \{x / Ax \leq b\}$ contenant S.
Soit V^0 l'ensemble des sommets de S^0 . Initialiser k =1et passer à l'étape 1.

étape k (k = 1,2,...) :

Résoudre le problème

$$(P_k) : \begin{cases} \min f(x) \\ x \in S^{k-1} \end{cases} \quad \text{et ceci en détérminant } \min \{ f(v) / v \in V^{k-1} \}$$

Soit x^k la solution du problème (P_k) et λ_k la solution du problème (Q_k) suivant:

$$(Q_k): \begin{cases} \min \quad \lambda \\ 0 \le \lambda \le 1 \ ; \ x^k + \lambda(p - x^k) \in S \end{cases}$$

Si $\lambda_k = 0$, alors on s'arrête : $x^k \in S$ est la solution de (P).
Si $\lambda_k > 0$, on calcule $z^k = x^k + \lambda_k(p - x^k)$. Soit :

$$u^k = \begin{cases} f(z^k) & \text{Si } f(z^k) < u^{k-1} \\ u^{k-1} & \text{Sinon} \end{cases}$$

Si $z^k - x^k < \varepsilon$ (ou simplement $u^k - f(x^k) < \varepsilon$) on s'arrête : x^k est une solution de (P).Sinon, on choisit une contrainte de S saturée par z^k ,on considère :
$J = \{ i / g_i(z^k) = 0 ; i = 1, \ldots , m \}$, et un indice $j \in J$.
Soit $G_j^k(x) = g_j(z^k) + [\nabla g_j(z^k)]^T.(x - z^k)$
$S^k = S^{k-1} \cap \{ x / G_j^k(x) \le 0 \}$; $V^k = \{$ des sommets de $S^k \}$.
$k = k+1$, retourner à l'étape k .
Les Lemmes suivants montrent la convergence de cet algorithme :

Lemme 1. La suite $\{ f(x^k)\}$ est décroissante.
preuve:
La fonction objective f est minimisée successivement sur une suite décroissante des ensembles $S^1, S^2, \ldots, S^k, S^{k+1}$. On a
$\min \{ f(x) / x \in S^{k+1}\} \ge \min \{ f(x) / x \in S^k\}$, d'où $f(x^{k+1}) \ge f(x^k)$. ◊

Lemme 2 [1] . Soit x^* la limite de la suite bornée $\{x^k\}$, λ^* la solution du problème :

$$(Q^*): \begin{cases} \min \quad \lambda \\ x^* + \lambda(p - x^*) \in S \ ; \ 0 \le \lambda \le 1. \end{cases}$$

Alors λ^* est la limite de la suite $\{\lambda_k\}$. ◊

Lemme 3 [1] . Soit $z^k = x^k + \lambda_k(p - x^k)$. Avec x^*, λ^* données dans le Lemme 2, la suite $\{z^k\}$ admet $z^* = x^* + \lambda^*(p - x^*)$ comme limite. ◊

Théorème 2 [1]. Chaque limite de la suite bornée x^k résoud le problème (P).

3.5. Essais numériques

Ces méthodes ont été mises au point sur micro PC (Olivetti M380) en langage Pascal pour la résolution du problème (P), dans le cas où C est un polyèdre convexe borné. A partir d'une série d'exemples testés, nous présentons quelques résultats numériques qui prennent en compte les principales remarques sur la mise en oeuvre de ces algorithmes.

Nous avons pris le problème de maximisation d'une forme quadratique symétrique définie positive sur un polytope.

$$\text{Max}\ \{\ f(x) = \langle x,Cx\rangle + \langle d,x\rangle + r\ /\ Ax \le b\ ,\ x \ge 0\ \} \qquad (P)$$

Exemple 1

$$C = \begin{bmatrix} 0.75 & -2.25 \\ -2.25 & 8.75 \end{bmatrix}\ ;\ A = \begin{bmatrix} -1 & 3 \\ 1 & -1 \\ -1 & 7 \\ 1 & -5 \end{bmatrix}\ ;\ d = [\,9, -27\,]\ ;\ b = [\,6\ ,\ 14\ ,\ 22\ ,2\,]\ ;\ r = 27.$$

Le maximum global est $x^* = (17\ ,\ 3)$ et $f(x^*) = 165$.

Exemple 2. [9]

Nous prenons le contre-exemple de Zwart [9] pour la première méthode de H.Tuy [10] .

$$C = I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\ ;\ A = \begin{bmatrix} 1 & 1 & -1 \\ -1 & 1 & -1 \\ 12 & 5 & 12 \\ 12 & 12 & 7 \\ -6 & 1 & 1 \end{bmatrix}\ ;\ d = [-2,0,-2]\ ;\ b = [1,-1,34.8,29.1,-4.1]\ ;\ r = 2.$$

Le maximum global est $x^* = [\,1\ ,\ 0\ ,\ 0\,]$ et $f(x^*) = 1.0$

Exemple 3. [11,12]

Calcul du maximum d'une forme quadratique définie positive sur la boule unité de la norme φ_∞ : $\text{Max}\ \{\ \langle y,Cy\rangle\ /\ -1 \le y \le +1\ \}$.

On fait le changement de variable $x = y+e$ où $e = (1,1,...,1)$, on obtient

$$\text{Max}\ \{\ \langle x,Cx\rangle - 2\langle Ce,x\rangle + \langle e,Ce\rangle\ /\ 0 \le x \le 2\ \}\ .$$

$$C = \begin{bmatrix} 22 & 1 & -7 & 6 \\ 1 & 33 & 4 & -7 \\ -7 & 4 & 27 & -1 \\ 6 & -7 & -1 & 38 \end{bmatrix} ; A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} ; d = [-44,-62,-46,-72] ; b = [2,2,2,2] ; r = 112.$$

Les solutions globales sont :
$x_1^* = [2,0,0,2]$, $x_2^* = [0,2,2,0]$ et $f(x_i^*) = 168.0$ (i=1,2).

3.6. Comparaison

Ces algorithmes ont été testés sur plusieurs exemples . La comparaison entre eux est basée sur deux critères principaux :

* L'occupation de la place mémoire nécessaire.
* Le temps de calcul.

On constate que :

- L'inconvénient principal des algorithmes de résolution par les subdivisions est la taille de la place mémoire. Signalons que l'occupation de mémoire se fait à l'exécution, lors des subdivisions .Ne pouvant pas définir la taille nécessaire au départ, nous avons utilisé les pointeurs qui permettent de créer, lors de la subdivision, la mémoire nécessaire de chaque cône (par l'instruction "new") et de détruire les mémoires des cônes $M \in \mathcal{M}_k$ où $\mu(M) \geq \gamma_k$ $(k \geq 0)$ 'par l'instruction "dispose"). Le temps de calcul de l'algorithme général avec les deux procédures, est très grand; ceci peut s'expliquer par le choix de la subdivision et le calcul de la minorante estimation utilisée. Ainsi cette méthode n'est pas appliquable pour les problèmes de grande dimension.

- L'algorithme d'approximation extérieure est plus simple à mettre en oeuvre, nous utilisons seulement des opérations de pivotages dans la méthode du simplexe ou on applique l'algorithme SGGP (cf. Chapitre I). Le temps de calcul et la taille de mémoire augmentent très rapidement avec la taille des problèmes, le choix du simplexe initial contenant le domaine.

4. Minimisation d'une fonction concave sur un polyèdre borné.

4.1. Préliminaires.

Soit D un polyèdre convexe borné de $\mathbb{R}^n$ et f une fonction concave définie sur D.

Définition 1 : une fonction f est dite concave sur D si :

$$f(\lambda x + (1-\lambda)y) \geq \lambda f(x) + (1-\lambda)f(y) \qquad (1)$$

pour tout $x,y \in D$ et pour tout $\lambda \in [0,1]$.
Autrement dit : f est concave si et seulement si la fonction (-f) est convexe.

La difficulté dans ce genre de problèmes réside essentiellement dans le fait qu'un minimum local n'est pas nécessairement un minimum global. Pour surmonter cette difficulté, on propose une méthode basée sur l'utilisation des deux propriétés suivantes:

Propriété 1. Le minimum d'une fonction concave sur un polyèdre convexe, s'il existe, est atteint au moins en un sommet. ◊

Définition 2. Une extension (prolongement) concave de la fonction f(x) est une fonction concave sur tout l'espace $\mathbb{R}^n$ qui coïncide avec f(x) sur D.

Propriété 2. (extention maximale) Si f est concave sur D, alors il existe une extention concave F(x) telle que $F(x) \geq g(x)$ pour tout $x \in \mathbb{R}^n$ et pour toute extention concave g(x) de f. ◊

En effet; la propriété 1 montre que pour résoudre le problème, il suffit de considérer seulement les sommets du polyèdre D, et la propriété 2 nous permet d'utiliser les valeurs de F(x) à l'extérieur de D.

4.2. Algorithmes.

On propose des algorithmes finis (de type d'approximation extérieure) qui nous permettent de résoudre le problème original :

$$\min\{ f(x) ; x \in D\} \qquad (P)$$

D est un polyèdre convexe borné et f est une fonction concave définie sur D. On peut supposer (sans perte de généralité) que le polyèdre D est "non dégénéré" (dans le cas contraire on fait une perturbation).

4.2.1. Algorithme 1 (sans changement du sommet de départ) [20, 22, 23].

I. Initialisation:
(i) On détermine par n'importe quelle méthode de programmation linéaire (S.G.G.P, méthode du simplexe) un sommet de départ x^0 du polyèdre D et on initialise k=0.
(ii) On construit un n-simplexe S^k du sommet x^k contenant le polyèdre D. Soit $V(S^k)$ l'ensemble des sommets de S^k.

II. Etape principale:
(i) Si pour tout $x \in V(S^k)$, $f(x^k) \leq f(x)$, alors x^k est une solution optimale de (P).
(ii) S'il existe $x^* \in \text{argmin}\{ f(x) ; x \in V(S^i) \}$ tel que $x^* \in D$ alors x^* est une solution optimale de (P).
(iii) Choisir un $x^* \in \text{argmin}\{ f(x) ; x \in V(S^i) \}$ et une contrainte violée en x^* (Soit $dx \leq b'$ cette contrainte violée en x^* c'est-à-dire $dx^* > b'$); on construit $S^{k+1} = S^k \cap \{ x \in \mathbb{R}^n : dx \leq b'\}$ et $k = k+1$, on calcule $V(S^k)$ (cf. Chap.I, algorithme S.G.G.P) et on retourne de nouveau à l'étape principale.

Description de l'algorithme1:

Phase 0 Déterminer un sommet quelconque x^0 de D. i:=0.

Phase 1 Construire un n-simplexe S^i de sommet x^i contenant D.

Phase 2

Pas 2.1 Si pour tout $x \in V(S^i)$, $f(x^i) \leq f(x)$, alors x^i est une solution optimale de (P). Sinon, aller à 2.2.

Pas 2.2 S'il existe $x^* \in \text{argmin}\{ f(x); x \in V(S^i) \}$ tel que $x^* \notin D$, alors faire $S^{i+1} = S^i \cap \{ x \in \mathbb{R}^n : dx \leq b'\}$; ($dx \leq b'$ étant une contrainte violée en x^*), $i = i+1$, calculer $V(S^i)$ et aller à 2.1.
Sinon, alors $\text{argmin}\{f(x) : x \in V(S^i) \}$ est contenu dans l'ensemble des solutions de (P).

4.2.2. Algorithme2 (avec changement du sommet de départ) [20, 22, 23].

I. Initialisation:

(i) On détermine par n'importe quelle méthode de programmation linéaire un sommet de départ x^0 du polyèdre D et on initialise k=0 et i=0.

(ii) On construit le n-simplexe S_k^i du sommet x^k contenant le polyèdre D. Soit $V(S_k^i)$ l'ensemble des sommets de S_k^i.

II. Etape principale:

(i) Si pour tout $x \in V(S_k^i)$, $f(x^k) \leq f(x)$, alors x^k est une solution optimale de (P).

(ii) S'il existe $x^* \in V(S^i)$ tel que $x^* \in D$ et $f(x^*) < f(x^k)$ alors $x^k = x^*$, $k = k+1$, et passer à la phase (ii) de l'étape I (l'initialisation).

(iii) Choisir $x^* \in$ argmin $\{f(x): x \in V(S_k^i)\}$ et une contrainte violée en x^*, (soit $dx \leq b'$ cette contrainte i.e. $dx^* > b'$). Prendre $S_k^{i+1} = S_k^i \cap \{x \in \mathbb{R}^n : dx \leq b'\}$, $i = i+1$. Calculer $V(S_k^i)$ (cf. Chap.I, algorithme S.G.G.P) et retourner à l'étape principale.

Déscription de l'algorithme 2 :

Phase 0 Soit $k = 1$; $i = 0$, déterminer un sommet quelconque x^k de D.

Phase 1 Déterminer un n-simplexe S_k^i de dimension n et de sommet x^k contenant D.

Phase 2 La phase 2 se décompose en deux pas :

Pas 2.1 Si pour tout $x \in V(S_k^i)$, $f(x^k) \leq f(x)$, alors x^k est une solution optimale de (P).

Pas 2.2 (i) S'il existe $v^* \in V(S_k^i)$ tel que $x^* \in D$ et $f(v^*) < f(x^k)$ alors $x^k = v^*$, $k = k+1$, aller à la phase 1.

(ii) Choisir $x^* \in$ argmin$\{f(x): x \in V(S_k^i)\}$ et une contrainte violée en x^*) ; (soit $dx \leq b'$ cette contrainte violée c'est-à-dire $dx^* \leq b'$).
Prendre $S_k^{i+1} = S_k^i \cap \{ x \in \mathbb{R}^n ; dx \leq b'\}$; $i = i+1$.
calculer $V(S_k^i)$ et aller à la phase 2.

4.2.3. Description de l'algorithme 3 [20, 22, 23]:

Phase 0 Soit $k = 1$, $i = 0$; déterminer un sommet quelconque x^k de D.

Phase 1 Soient $x^{k,j}$ $j=1,..n$ les sommets adjacents de x^k dans le polyèdre D. Prendre $x^* \in$ argmin$\{ f(x^k), f(x^{k,j}); 1 \leq j \leq n\}$ et calculer pour $1 \leq j \leq n$

$u^{k,j} = x^k + \lambda_{k,j}.(x^{k,j} - x^k)$ tel que $\lambda_{k,j} = \text{Max}\{1 \leq \lambda \leq M : f(u^{k,j}) \geq f(x^*)\}$
où M étant un réel assez grand .
Soit $\Sigma_k^i = \text{conv}\{ x^k, u^{k,j} ; j=1,..n\}$ le n-simplexe engendré par
$\{ x^k, u^{k,j} ; j=1,..n\}$. Calculer le vecteur direction γ_k défini par :
$\gamma_k(u^{k,j} - u^{k,1}) = 0$ pour $j=1,..n$ et $\gamma_k x^k < \gamma_k u^{k,1} = \mu_k$.
Calculer $m_k = \max\{ < \gamma_k, x> ; x \in D\}$
Si $\mu_k \geq m_k$ on s'arrête : x^* est une solution optimale de (P).
Sinon, calculer $z^{k,j} = x^k + \mu_{k,j}.(x^{k,j} - x^k)$, $(j=1,..n)$ où $\mu_{k,j}$ sont données par $\gamma_k z^{k,j} = m_k$.
Soit $S_k^i = \text{conv}\{ x^k, z^{k,j} ; j=1,..n\}$ le simplexe engendré par $\{x^k, z^{k,j}; 1 \leq j \leq n\}$, (c'est un n-simplexe du sommet x^k contenant D).

Phase 2 La phase 2 se décompose en deux pas :

Pas 2.1 Si pour tout $x \in V(S_k^i)$, $f(x^*) \leq f(x)$, alors x^* est une solution optimale de (P).

Pas 2.2 (i) S'il existe $w \in V(S_k^i)$ tel que $x \in D$ et $f(w) < f(x^*)$ alors, $x^{k+1} = w$, $k = k+1$, aller à la phase 1.
(ii) Choisir $v \in \text{argmin}\{f(x) : x \in V(S_k^i)\}$ et une contrainte violée en v) ; (soit $dx \leq b'$ cette contrainte violée c'est-à-dire $dv > b'$).
Prendre $S_k^{i+1} = S_k^i \cap \{ x \in \mathbb{R}^n ; dx \leq b'\}$; $i = i+1$.
Calculer $V(S_k^i)$ et aller à la phase2.

4.2.4. Description de l'algorithme 4 (l'algorithme général) [20, 22, 23] :

Phase 0 Initialiser $k=1$, $i=0$, déterminer un sommet quelconque x^k de D.

Phase 1 Soient $x^{k,j}$ $j=1,..n$ les sommets adjacents de x^k dans le polyèdre D.
Prendre $x^* \in \text{argmin}\{f(x^k), f(x^{k,j}); 1 \leq j \leq n\}$ et on calcule pour $1 \leq j \leq n$
$u^{k,j} = x^k + \lambda_{k,j}.(x^{k,j} - x^k)$ tel que $\lambda_{k,j} = \text{Max}\{ 1 \leq \lambda \leq M ; f(u^{k,j}) \geq f(x^*) \}$
où M étant un réel assez grand .
Soit $\Sigma_k^i = \text{conv}\{ x^k, u^{k,j} ; j=1,..n\}$ le n-simplexe engendré par
$\{ x^k, u^{k,j} ; j=1,..n\}$. Calculer le vecteur direction γ_k défini par :
$\gamma_k(u^{k,j} - u^{k,1}) = 0$ pour $j=1,..n$ et $\gamma_k x^k < \gamma_k u^{k,1} = \mu_k$.

Calculer $m_k = \max\{ < \gamma_k, x> ; x \in D\}$

Si $\mu_k \geq m_k$ on s'arrête : x^* est une solution optimale de (P).

Sinon on calcule $z^{k,j} = x^k + \mu_{k,j}.(x^{k,j} - x^k), (j=1,..n)$ où $\mu_{k,j}$ sont données par $\gamma_k z^{k,j} = m_k$. Soit $S_k^i = \text{conv}\{x^k, z^{k,j} ; j=1,..n\}$ le simplexe engendré par $\{x^k, z^{k,j}; 1 \leq j \leq n\}$ (c'est un n-simplexe du sommet x^k contenant D).

Phase 2 Calculer $V(S_k^i)$ (cf. Chap.I, algorithme S.G.G.P).
Si pour tout $x \in V(S_k^i)$, $f(x^*) \leq f(x)$, alors x^* est une solution optimale de (P).

Phase 3 S'il existe $v^* \in \text{argmin}\{f(x); x \in V(S_k^i)\}$ tel que $v^* \in D$
(i.e., $\text{argmin}\{f(x); x \in V(S_k^i)\} \cap D \neq \emptyset$) alors, on s'arrête : chaque élément de cette intersection est une solution optimale de (P).

Phase 4 Si $\text{argmin}\{f(x); x \in V(S_k^i) \cap D\} \neq \emptyset$.
Choisir $y \in \text{argmin}\{f(x); x \in V(S_k^i) \cap D\}$. Si $f(y) < f(x^*)$ alors:
$x^{k+1} = y$, $k = k+1$, aller à la phase 1.

Phase 5 Choisir $v \in \text{argmin}\{ f(x) ; x \in V(S_k^i) \}$ et une contrainte violée en v; (soit $dx \leq b'$ cette contrainte violée en v i.e. $dv > b'$).
Prendre $S_k^{i+1} = S_k^i \cap \{ x \in \mathbb{R}^n : dx \leq b'\}$, $i = i+1$, aller à la phase2.

4.3. Implémentation

Deux problèmes fondamentaux se posent dans ces algorithmes:

(i) détermination de l'ensemble des sommets $V(S_k^i)$
(ii) construction du polyèdre S_k^i

La résolution de deuxième problème (construction du polyèdre S_k^i) est écrite en détail dans les algorithmes 3 et 4 (en utilisant la technique de l'algorithme SGGP pour la détermination d'un sommet réalisable et pour calculer les voisins de ce sommet).

Détermination de l'ensemble des sommets $V(S_k)$

On connait $S^k = \{ x \in \mathbb{R}^n / g_i(x) = \langle A_i, x \rangle - b_i \leq 0 \}$ et $V(S^k)$, on veut couper S^k par l'hyperplan

$$H = \{ x \in \mathbb{R}^n / h(x) = \langle \alpha, x \rangle + \beta = 0 \} \text{ avec } \alpha \in \mathbb{R}^n, \beta \in \mathbb{R},$$

pour obtenir $S^{k+1} = S^k \cap \{ x \in \mathbb{R}^n / h(x) \leq 0 \}$.

On démontre que [22,23], si on pose

$S = S^k \cap H$

$V^+ = \{ x \in V(S^k) / h(x) > 0 \}$

$V^- = \{ x \in V(S^k) / h(x) < 0 \}$

$V^\circ = \{ x \in V(S^k) / h(x) = 0 \}$

alors on a $V(S^{k+1}) = V^- \cup V(S)$.
On peut obtenir facilement V^- , il suffit donc de calculer V(S):

on a $V(S) = V^\circ \cup W$

et $W = \{ w \in S^k / w \in [u,v]$ où $u \in V^-$ et $v \in V^+$, u et v voisins et $h(w)=0 \}$

Il y a alors deux façons de calculer W :

1ère méthode pour calculer W :

Pour chaque sommet u de V^-, on teste chaque sommet v de V^+. Si u et v sont 2 sommets voisins, alors ils sont de part et d'autre de l'hyperplan $h(x) = 0$ et on peut calculer alors directement $w \in [u,v]$ tel que $h(w) = 0$.
Complexité de cet algorithme: $|V^+| \times |V^-|$
Cette complexité ne cesse d'augmenter, car le nombre de sommets de S^k ne cesse pas d'augmenter au cours des itérations.

2ème méthode pour calculer W:

On va travailler avec l'ensemble V^- ou V^+ qui comporte le moins de sommets pour réduire la complexité.

Pour chaque sommet u de cet ensemble, on peut utiliser soit l'algorithme de Horst-Vries-Thoai [15], soit l'algorithme SGGP dû à Pham Dinh Tao [cf. Chapitre 1] pour déterminer tous les sommets voisins de u. Dans ce logiciel c'est le deuxième algorithme qui est implémenté à cause de sa simplicité. Parmi ces voisins, pour tous ceux qui sont de l'autre côté de l'hyperplan par rapport à u, on peut calculer directement

$w \in$ [u,voisin_opposé de u] tel que $h(w) = 0$.

Complexité de cet algorithme: $\min (|V^+| , |V^-|) \times O$(algorithme simplexe)
La complexité de l'algorithme SGGP est en $O(n^4)$ donc la complexité de cet algorithme pour calculer W est de $\min (|V^+| , |V^-|) \times n^4$.

Ceci est intéressant car même si n^4 est assez grand, il est constant alors que V^+ augmente toujours en général.

Remarque:

L'algorithme basé sur l'algorithme SGGP ne permet d'obtenir tous les sommets de u que si u est non dégénéré (ce qui est souvent le cas). Dans le cas contraire, on devrait utiliser la première méthode . Les 2 méthodes sont donc nécessaires, elles cohabitent et on utilise celle qui est le plus rapide en fonction du nombre de sommets courants.

La coupe par l'hyperplan H est ainsi effectuée et l'on obtient tous les sommets de S^{k+1}. Cependant, à cause de cette nouvelle coupe, il est possible que certaines des contraintes antérieures soient devenues superflues. Il faut donc supprimer les contraintes redondantes.
Pour cela, on utilise le résultat de [16] qui démontre que :

si $V^- = \{ x \in V(S^k) \ / \ h(x) < 0 \} \neq \emptyset$
alors la contrainte $g_k(x) \leq 0$ est redondante pour S^k relativement à h
si et seulement si $g_k(u) < 0 \ \forall u \in V^-$.

Il faut alors supprimer cette contrainte de la matrice représentant le polyèdre S^{k+1} et de la liste des contraintes qui saturent les sommets.

5. Expérimentations numériques.

5.1. Problème non linéaire complémentaire (Algorithme de §2.).

Soit : $f : \mathbb{R}^n \mapsto \mathbb{R}^n$
Trouver $x \in \mathbb{R}^n$: $x \geq 0$, $f(x) \geq 0$, $x^t f(x) = 0$.

Exemple 1.1

$$f : \mathbb{R}^2 \mapsto \mathbb{R}^2$$

$$f_1(x) = -x_1^2 - x_2^2 + 2x_1 + 2x_2 + 2$$

$$f_2(x) = -x_1^2 - x_2^2 + 4x_1 + 4x_2 - 4$$

la solution est : $x^* = (0, 2)$, $f(x^*) = (2, 0)$.

Le temps de calcul en secondes : 4.78

Exemple 1.2

$$f : \mathbb{R}^3 \mapsto \mathbb{R}^3$$

$$f_1(x) = -x_1^2 - x_2^2 - x_3^2 + 3x_1 + x_2 + 4x_3 - 1$$

$$f_2(x) = -\frac{3}{2}x_1^2 - \frac{3}{2}x_2^2 - \frac{3}{2}x_3^2 + 4x_1 + 3x_2 + 5x_3 - 2$$

$$f_3(x) = -\frac{x_1^2}{2} - \frac{x_2^2}{2} - \frac{x_3^2}{2} + x_1 - 2x_3 + 6$$

la solution est : $x^* = (0, 0, 2)$, $f(x^*) = (3, 2, 0)$.
Le temps de calcul en secondes : 7.69

Exemple 1.3

$$f : \mathbb{R}^4 \mapsto \mathbb{R}^4$$

$$f_1(x) = -x_1^2 - x_2^2 - x_3^2 - x_4^2 + 2x_1 + 4x_4$$

$$f_2(x) = -2x_1^2 - 2x_2^2 - 2x_3^2 - 2x_4^2 + 6x_2 + 2x_3 + 4x_4 + 35$$

$$f_3(x) = -3x_1^2 - 3x_2^2 - 3x_3^2 - 3x_4^2 + 3x_1 + 3x_3 + 24x_4 - 4.5$$

$$f_4(x) = -\frac{x_1^2}{2} - \frac{x_2^2}{2} - \frac{x_3^2}{2} - \frac{x_4^2}{2} + x_1 + 2x_2 + 3x_3 + x_4 + 4$$

la solution est : $x^* = (0, 0, 0, 4)$, $f(x^*) = (0, 19, 43.5, 0)$.
Le temps de calcul en secondes : 14.19

5.2. Problème linéaire complémentaire (Algorithme 4, §4.2.4.).

Etant donné une matrice carrée M d'ordre n et un vecteur $q \in \mathbb{R}^n$.
On considère le problème linéaire complémentaire :

Trouver $x \in \mathbb{R}^n$: $x \geq 0$, $Mx+q \geq 0$, $x^t(Mx+q) = 0$.

qui se ramène à :

$$(P)\begin{cases} 0 = \min f(x) \\ x \in D \end{cases}$$

où $f(x) = \sum_{i=1}^{n} (x_i , (Mx+q)_i)$ et $D = \{x \in \mathbb{R}^n : x \geq 0 , Mx+q \geq 0\}$.

Exemple 2.1

```
        2   2   3  -3  -2   0  -1   1  -3
       -1   5  -4   2  -4  -2   2  -2   2
        2  -3   1   1  -3  -3   4   1  -1
       -1   1  -4   3  -2   2  -3   2  -3
M =     1   2   3  -2   1  -1   4  -4  -2
        2  -3  -1   3  -2   4   3  -2   1
        3  -1   3  -2   4  -3   1  -3   2
       -2   3   1  -3   1  -2  -4   2  -1
       -3  -1   3  -2  -2   3  -2   3   2

q = (-4, -8, 2, 2, 6, 4, -2, -4, -2)

vecteur solution globale          : (2, 2, 0, 0, 0, 0,  0, 2, 2)
valeur de la fonction objective   :  0.00000000
Le temps de calcul en secondes    :  7.31
```

Exemple 2.2

```
       -2  -2   1  -1   1  -3   2  -1
        1  -4   2  -2   2  -2   3  -4
        2  -2   4  -1  -1   3   1  -2
M =     1   2  -3   4  -2  -1  -1   2
       -1  -1   1  -1   4  -2   2  -3
       -1   2  -2   2  -2   4  -2   3
       -2   1  -3   1   1  -2   4  -2
        2  -1   1  -1   1   2  -2   3

q = (3, 0, -2, -4, 2, 1, -3, -2)

vecteur solution globale          :   (2, 1, 0, 0, 0, 0, 2, 1)
valeur de la fonction objective   :    5.52891066263328E-015
Le temps de calcul en secondes    :    5.739
```

Exemple 2.3

```
        2   2   1  -1   1  -3
       -1  -4  -2   2   2   2
M =     1   1   4  -1  -1   3
        1   2  -2   4  -2  -1
        1   1   2  -3   2  -3
        1   2  -1   0  -4  -2

q = (3, 3, -12, -2, 6, 1)

vecteur solution globale          :    (1, 2, 0, 0, 0, 3)
valeur de la fonction objective   :    5.28191066263328E-016
Le temps de calcul en secondes    :    2.8
```

5.3. Programmation quadratique (Algorithme 4, §4.2.4.).

$$\min_{x \in D} f(x) = \langle x, Mx \rangle + \langle q, x \rangle + r$$

où $D = \{x \in \mathbb{R}^n : 0 \leq x_i \leq 1,\ 1 \leq i \leq n\}$, M est une matrice carrée d'ordre n, symétrique, semi-définie négative, q est un n-vecteur et r est un réel.

Exemple 3.1

```
     -4   -2   -1    0    0    0
     -2   -4   -2   -1    0    0
M =  -1   -2   -4   -2   -1    0
      0   -1   -2   -4   -2   -1
      0    0   -1   -2   -4   -2
      0    0    0   -1   -2   -4

q = (1, 1, 1, 1, 1, 1)  et  r = -16

vecteur solution globale:  (1, 1, 1, 1, 1, 1)


valeur de la fonction objective :   -62.000000000
Le temps de calcul en secondes  :    12.78
```

Exemple 3.2

```
     -3  -1   1   0   0   0   0   0
     -1  -3  -1   1   0   0   0   0
      1  -1  -3  -1   1   0   0   0
M =   0   1  -1  -3  -1   1   0   0
      0   0   1  -1  -3  -1   1   0
      0   0   0   1  -1  -3  -1   1
      0   0   0   0   1  -1  -3  -1
      0   0   0   0   0   1  -1  -3

q = (1, 2, 3, 4, 4, 3, 2, 1)  et   r = -10

vecteur solution globale        :  (1, 1, 1, 0, 0, 1, 1, 1)

valeur de la fonction objective :  -20.000000000

Le temps de calcul en secondes  :   15.68
```

Exemple 3.3

On considère le problème suivant :

(P) Min $\{f(x) = -\langle x, Mx\rangle + \langle q, x\rangle : Ax \leq b, x \geq 0\}$

où: A est une matrice de type mxn et $b \in \mathbb{R}^m$
m est le nombre de contraintes
n est le nombre de variables

$$M = \begin{pmatrix} 7.6 & 6.2 & 4.4 & 5.2 & 6.4 & 5.6 & 6.6 & 4.4 & 5.6 & 7 \\ 6.2 & 13.4 & 4.6 & 8.8 & 11.8 & 6.4 & 7.8 & 5.6 & 8 & 10 \\ 4.4 & 4.6 & 3.4 & 4.8 & 4.8 & 4.6 & 5.6 & 3.2 & 4.6 & 5.2 \\ 5.2 & 8.8 & 4.8 & 7.8 & 8 & 7 & 7 & 4.8 & 6.6 & 7.4 \\ 6.4 & 11.8 & 4.8 & 8 & 12.2 & 5.4 & 8.6 & 5.6 & 9 & 9.2 \\ 5.6 & 6.4 & 4.6 & 7 & 5.4 & 7.6 & 6.2 & 4.4 & 5.2 & 6.4 \\ 6.6 & 7.8 & 5.6 & 7 & 8.6 & 6.2 & 13.4 & 4.6 & 8.8 & 6.4 \\ 4.4 & 5.6 & 3.2 & 4.8 & 5.6 & 4.4 & 4.6 & 3.4 & 4.6 & 4.8 \\ 5.6 & 8 & 4.6 & 6.6 & 9 & 5.2 & 8.8 & 4.6 & 7.8 & 8 \\ 7 & 10 & 5.2 & 7.4 & 9.2 & 6.4 & 11.8 & 4.8 & 8 & 12.2 \end{pmatrix} \quad q = \begin{pmatrix} 10 \\ 10 \\ 10 \\ 10 \\ 10 \\ 10 \\ 10 \\ 10 \\ 10 \\ 10 \end{pmatrix}$$

et

$$A = \begin{pmatrix} 20 & 20 & 60 & 60 & 60 & 60 & 5 & 45 & 55 & 65 \\ 5 & 7 & 3 & 8 & 13 & 13 & 2 & 14 & 14 & 14 \\ 100 & 130 & 50 & 70 & 70 & 70 & 20 & 80 & 80 & 80 \\ 200 & 280 & 100 & 200 & 250 & 280 & 100 & 180 & 200 & 220 \\ 2 & 2 & 4 & 4 & 4 & 4 & 2 & 6 & 6 & 6 \\ 4 & 8 & 2 & 6 & 10 & 10 & 5 & 10 & 10 & 10 \\ 60 & 110 & 20 & 40 & 60 & 70 & 10 & 4 & 5 & 50 \\ 150 & 210 & 40 & 70 & 90 & 105 & 60 & 100 & 140 & 180 \\ 80 & 100 & 6 & 16 & 20 & 22 & 60 & 20 & 30 & 30 \\ 40 & 40 & 12 & 20 & 24 & 28 & 60 & 20 & 40 & 50 \end{pmatrix} \quad b = \begin{pmatrix} 600.1 \\ 310.5 \\ 1800 \\ 3850 \\ 18.6 \\ 198.7 \\ 882 \\ 4200 \\ 40.25 \\ 327 \end{pmatrix}$$

La solution globale est : $x^* = (0, 0, 3.415, 1.235, 0, 0, 0, 0, 0, 0)$
Le temps d'execution est : 33.29
La valeur de la fonction objective : $f^* = -45.53656$

Nous avons fait des tests comparatifs entre l'algorithme 4 (cf. §4.2.4.) et l'algorithme de Falk-Hoffman (cf. §3.4.) pour résoudre le problème (P), les résultats numériques sont présentés dans le tableau suivant :

Tableau comparatif :

Pb	m	n	Mét : Falk-Hoffman				Algorithme 4			
			NI	NSG	NSS	T	NI	NSG	NSS	T
1	6	6	4	33	19	6.32	5	24	11	6.08
2	10	6	6	98	57	8.85	5	78	29	9.72
3	12	6	7	102	45	9.45	7	105	32	9.42
4	6	10	6	173	84	39.04	6	228	71	38.2
5	10	10	8	312	117	46.3	7	307	103	44.33
6	12	12	10	304	98	57.2	11	321	87	56.1

où :

NI est le nombre d'itérations
NSG est le nombre de sommets générés par l'algorithme
NSS est le nombre de sommets stockés
T est le temps c.p.u (en secondes)

Les tests numériques sont faits sur PC (Olivetti_380).

Références

[1]. Tuy.H. (1964).
Concave Programming under Linear constraints. Dokl. Akad. Nauk ***159*** *32-35.*
[2]. R.Horst (1976).
An Algorithm for Nonconvex Programming Problems.
Math.Programming. Vol ***10*** *, P 312-321.*
[3]. Mangasarian.O.L. (1978).
Characterization of Linear Complementarity Problems as Linear Programs.
Math.Programming Study 7 74-88.
[4]. Tuy.H and Ng.V.Thoai. (1980).
Convergent Algorithms for Minimizing a Concave Function.
Math.Oper.Res. Vol 7, P 556-566 .
[5]. Ban.VuThien (1982).
A Finite Algorithm for Minimizing a Concave Function under Linear Constraints and Its Applications. Preprint.Institute of Mathematics, Hanoi.
[6]. Tuy.H, T.V.Thieu and Ng.Q.Thai. (1985).
A Conical Algorithm for Globally Minimisation a Concave Function over a Closed Convex Set. Math .Oper .Res . P 498-514.
[7] E.Balas and C.E.Burdet (1975).
Nonconvex quadratic programming via generalized polars.
SIAM J.Appl.Math. Vol 28 , p 325-349.
[8] P.B.Zwart (1974)
Global maximization of a convex function with linear inequality constraints.
Oper .Res ; Vol 22 , p 602-608.
[9] P.B.Zwart (1973]
Nonlinear programming : counterexamples to two global optimization algorithms.
Oper .Res., N 21, Vol 6 ,p 1960-1966.
[10] J.F.Toland (1979]
A duality principle for non-convex optimization and calculus of variations.
Arch.Rational.Mech.Analy , Vol 71,p 41-61.
[11] Pham Dinh Tao (1981)
Contribution à la théorie de normes et ses applications à l'analyse numérique.
Thèse d'Etat. USMG (Grenoble).
[12] Pham Dinh Tao (1984)
Algorithmes de calcul du maximum d'une forme quadratique sur la boule unité de la norme du maximum.Num.Math. Vol 45, pp 163-183.
[13] J.E.Falk and K.Hofman (1973)
A successive under-estimation method for concave minimization.
Math. Oper.Res, vol 21, p 123-134
[14] K.Hofman (1981)
A methode for globally minimizing concave functions over convex sets.
Math.Prog , vol 20, p 22-32 .
[15] R.Horst., N.V.Thoai and J.Vries
On finding new vertices and redundant constraints in cutting plane algorithms for global optimization. To appear in O.R.Letters.

[16] R.Benacer (1986)
*Contribution à l'étude des algorithmes de l'optimisation non convexe et non differentiable.Thèse de l'**USTMG** (Grenoble).*
[17] M.Sakarovitch (1984)
Programmation Linéaire. Hermann .
[18] T.V.Thieu, B.T.Tam and V.T.Ban (1983)
An outer approximation metod for globallyminimizing a concave function over a compact convex set. Acta.Math. Vietnamica , 8, 1, 21-40.
[19] H. Tuy (1983)
On outer approximation methods for solving concave minimization problems".
Acta Mathematica Vietnamica, 8 , 2 , 3-34 .
[20] Pham Dinh Tao & El Bernoussi Souad (1989)
Numerical algorithms for solving a class of global nonconvex optimization problems. International Series of Numerical Mathematics: "New Methods in Optimization and Their Industrial Use", Birkhauser Verlag.
[21] Pham Dinh Tao (1986)
Global maximization of a nondefinite quadratic function over a convex polyhedron. Math. for optimization. North Holland. J.B.Hiriart Urruty, (editor) (Elsevier Science Publishers B.V.)
[22] Pham Dinh Tao, Jamal Chaarani and Souad El Bernoussi
Global numerical algorithms for solving a class of nonconvex optimization problems. To appear in Zeitschrift für Operations Research (ZOR).
[23] Pham Dinh Tao & El Bernoussi Souad (1988)
Duality in D.C. (difference of convex functions) Optimization. Subgradient-Methods. International Series of Numerical Mathematics: "New Methods in Optimization and Their Industrial Use", Birkhauser Verlag.

CHAPITRE V

METHODES DE REGION DE CONFIANCE. THEORIE, ALGORITHMES ET APPLICATIONS

1. Introduction.

Soit le problème de minimisation sans contrainte :

$$(P) : \begin{cases} \min f(x) \\ x \in \mathbb{R}^n \end{cases}$$

où f est une fonction de $\mathbb{R}^n$ dans $\mathbb{R}$, bornée inférieurement et deux fois continûment différentiable.

Nous désignons par g_k le gradient de f en x_k et par H_k le Hessien de f en x_k ou une matrice symétrique qui approxime le Hessien de f en x_k (quasi-Hessien).

En général, le problème (P) est traité par des variantes de la méthode de Newton qui consiste à trouver un zéro du gradient de la fonction f.
Ces variantes recouvrent aussi bien les méthodes utilisant l'approximation d'ordre 1 de f que celles qui utilisent les approximations de la matrice Hessienne de f.

Le but principal de ce chapitre est de décrire et d'analyser une technique pour la résolution de ce problème. L'approche que nous allons présenter est bien connue (Moré 1983, Sorensen 1981). Elle est communément appelée "méthode de région de confiance" dans laquelle le pas déterminant un nouvel itéré x_{k+1} est obtenu par la minimisation d'une approximation quadratique locale de la fonction objective sur une région restreinte (qui est la boule euclidienne centrée en x_k).

Le diamètre de cette région est agrandi ou réduit selon la qualité d'approximation du modèle quadratique local. Il est possible de contrôler l'itération de manière à forcer la convergence de la méthode sous certaines conditions raisonnables portant sur la fonction objective. La méthode de région de confiance génère une suite d'itérés en résolvant à chaque pas un problème quadratique de la forme:

$$(P_k) \quad : \quad \min \{q_k(d) : \|d\| \leq \delta_k\}.$$

où q_k désigne l'approximation quadratique de la variation de f en x_k définie par: $q_k(d) = \langle g_k, d \rangle + 1/2 \langle H_k d, d \rangle$

Le nombre strictement positif δ_k est appelé rayon de confiance. Dans les algorithmes de région de confiance, le calcul du gradient de la fonction objective est requis. L'utilisation des informations du premier ordre nous amène en général à des points stationnaires du premier ordre.
En y incorporant des informations du second ordre ces algorithmes pourraient satisfaire les conditions nécessaires du second ordre pour le problème (P) : dans ce cas la méthode de région de confiance peut être regardée comme une méthode de Newton modifiée appliquée à la recherche d'un zéro du gradient de la fonction objective.

La méthode "générale" de région de confiance peut être décrite comme suit:

En calculant la direction d_k, solution de (P_k), nous pouvons facilement contrôler la qualité de l'approximation locale $q_k(d)$ et par suite prendre la décision appropriée:

* Si l'approximation est satisfaisante, alors la solution de (P_k) établit un nouveau itéré et le rayon de confiance est augmenté.

* Dans le cas contraire, l'itéré est inchangé, en plus le rayon de confiance est diminué jusqu'à ce que q_k établisse une approximation satisfaisante à l'intérieur de la région de confiance (qui a nécessairement lieu pour de petits rayons car le gradient est supposé connu exactement et le terme linéaire en g_k devient dominant si $||g_k|| \neq 0$).

Naturellement le schéma général peut être implémenté de différentes manières. La qualité de l'approximation est en général examinée à travers la qualité suivante appelée "coefficient de qualité":

$$r_k = \frac{f(x_k) - f(x_k+d_k)}{q_k(0) - q_k(d_k)}$$

Le numérateur représente l'actuelle réduction de la fonction f quand on se déplace de x_k à x_{k+1}, le dénominateur représente la réduction relative à l'approximation quadratique.

Ainsi dans un algorithme de région de confiance, les principaux efforts de calcul, à part les évaluations de la fonction requises, sont centrés dans la résolution de (P_k) pour déterminer le pas de déplacement.

Les algorithmes de région de confiance diffèrent dans leurs stratégies de résoudre approximativement (P_k).

Pour les besoins d'ordre pratique, nous avons implémenté diverses versions de la méthode de region de confiance sur différents matériels (PC, VAX, Station APOLLO DN 4000) en langage Pascal et Fortran.

Les expérimentations numériques comparatives sont très satisfaisantes et seront présentées dans le chapitre VI à travers les applications concrètes :

(i) Optimisation (non convexe) dans les réseaux neuronaux
(ii) Optimisation (non convexe) dans les problèmes d'écoulement bi et tridimensionnels à géométrie simplement connexe.
(iii) Optimisation (non convexe) dans les problèmes d'analyse multidimensionnelle des données de dissimilarité (MDS).

2. Algorithme général de la région de confiance.

0. Soient $0<\mu<\eta<1$; $0<\gamma_1<\gamma_2<1<\gamma_3$; $x_0 \in \mathbb{R}^n$ arbitrairement choisi et $\delta_0>0$.
1. On calcule $f_k = f(x_k)$, $g_k = \nabla f(x_k)$ et la matrice symétrique H_k .
2. Si $g_k = 0$ on s'arrête : x_k est une solution optimale.
3. On calcule la direction d_k solution du problème :

$$(P_k) : \begin{cases} \min\ q_k(d) \\ \|d\| \le \delta_k \end{cases}$$

Soit $r_k = \dfrac{f(x_k) - f(x_k+d_k)}{q_k(0) - q_k(d_k)}$

4. Si $r_k \le \mu$ alors : $\delta_k = \Delta \in [\gamma_1 \delta_k, \gamma_2 \delta_k]$ et on retourne à l'étape 3.
   $x_{k+1} = x_k + d_k$
   Si $r_k \le \eta$ alors : $\delta_{k+1} \in [\gamma_2 \delta_k, \delta_k]$,sinon $\delta_{k+1} \in [\delta_k, \gamma_3 \delta_k]$
   $k \leftarrow k+1$ et on retourne à l'étape 1.

3. Calcul de H_{k+1} [20] [21]

Soient $\gamma_k = \nabla f(x_{k+1}) - \nabla f(x_k)$ et $s_k = x_{k+1} - x_k$.
Prenons H_0 une matrice symétrique, définie positive.

i) Formule de rang 1 (DFP) (Davidon,Fletcher,Powell)

$$H_{k+1} = H_k + \alpha_k . u_k . u_k^t$$

Avec $\alpha_k = \dfrac{-1}{s_k^t . (H_k s_k - \gamma_k)}$ et $u_k = H_k s_k - \gamma_k$.

ii) Formule de rang 2 (BFGS) (Broyden,Fletcher,Goldfarb,shanno)

$$H_{k+1} = H_k + \alpha_k . u_k . u_k^t + \beta_k . v_k . v_k^t$$

Avec : $\alpha_k = \dfrac{1}{s_k^t . \gamma_k}$, $u_k = \gamma_k$ et $\beta_k = \dfrac{-1}{s_k^t . H_k . s_k}$, $v_k = H_k . s_k$.

Il est prouvé (Minoux 1983, Gill & Murray 1972) que, si la fonction f est quadratique et $A=\nabla^2 f(x)$ est définie positive, la suite $\{H_k\}$, calculée par les formules précédentes, converge vers $\nabla^2 f(x^*)$.

4. Calcul des valeurs et vecteurs propres de H_{k+1}

4.1.Théorème 1. [22, 27]

On suppose que $H_{k+1} = H_k + c.w.w^t$, c est un réel quelconque non nul et w est un n-vecteur.

Soient $\lambda_1 < \lambda_2 < \lambda_3 < ..... < \lambda_n$ les valeurs propres de H_k et $v_1, v_2, v_3, ..., v_n$ les vecteurs propres de H_k associés respectivement à $\lambda_1, \lambda_2, \lambda_3, ..., \lambda_n$.

1. Soit $F(\beta) = \sum_{i=1}^{n} \frac{\langle v_i, w \rangle}{\lambda_i - \beta} + \frac{1}{c}$

Les valeurs propres β_i $(1 \le i \le n)$ de la matrice H_{k+1} sont les racines de l'équation $F(\beta) = 0$ avec $\lambda_i < \beta_i < \lambda_{i+1}$ pour $1 \le i \le n-1$.

2. Le vecteur propre z_i de H_{k+1} associé à β_i est donné par: $(H_k - \beta_i I).z_i = w$. ◊

5. Mise à jour du rayon de confiance δ_k

Le rayon de confiance est calculé suivant la règle suivante :
Soient $0 < \mu < \eta < 1$ et $0 < \gamma_1 < \gamma_2 < 1 < \gamma_3$ des constantes spécifiées.

1. Si $r_k \le \mu$ alors : $\delta_{k+1} = \Delta \in [\gamma_1 \delta_k, \gamma_2 \delta_k]$
2. Si $r_k \le \eta$ alors : $\delta_{k+1} \in [\gamma_2 \delta_k, \delta_k]$,sinon $\delta_{k+1} \in [\delta_k, \gamma_3 \delta_k]$

Pratiquement, on applique cette règle comme suit :

Si $r_k < 0.25$ alors $\delta_{k+1} = \delta_k / 2$
Si $r_k > 0.75$ alors $\delta_{k+1} = 2.\delta_k$
sinon $(0.25 \le r_k \le 0.75)$ alors $\delta_{k+1} = \delta_k$.

6. Calcul de l'itéré x_k

Dans la méthode de région de confiance, l'itéré est calculé à partir d'un réel donné s tel que $0 \le s < 0.25$, de la façon suivante :
On prend $x_0 \in \mathbb{R}^n$ arbitrairement choisi :

1. Si $r_k < s$ alors $x_{k+1} = x_k$
2. Si $r_k \ge s$ alors $x_{k+1} = x_k + d_k$

7. Résolution du problème local (Q) (Calcul de la direction d_k).

Le problème local se réduit à la minimisation d'une fonction quadratique sur une sphère : $\min \{q(d) = \langle g,d \rangle + 1/2 \langle Hd,d \rangle : \|d\| \le \delta\}$ (Q)
avec g est un n-vecteur, H est une matrice symétrique, δ est un réel positif.
Puisque le domaine admissible $\{d \in \mathbb{R}^n : \|d\| \le \delta\}$ est compact, le problème (Q) admet une solution. Si, de plus, H est définie positive, alors la solution est unique.

Ici notre but est de donner une étude complète des aspects théoriques du problème (Q) et d'exposer la nature des difficultés rencontrées dans sa résolution. Certaines propriétés sont connues ([15],[16],[18]), cependant nous voudrions les formuler et les démontrer d'une manière plus complète et mieux adaptée .

Les algorithmes pour la résolution de (Q) sont basés sur les résultats théoriques suivants :

Lemme 1. [15,18]

d* est une solution de (Q) si et seulement si il existe $\mu \geq 0$ tel que :

(i) $(H + \mu I)$ est semi-définie positive

(ii) $(H + \mu I)\, d^* = -g$ (*)

(iii) $\|d^*\| \leq \delta$ et $\mu(\delta - \|d^*\|) = 0$.

Un tel μ est unique. ◊

Preuve

Il est clair que (ii) et (iii) sont les conditions d'optimalité de Kuhn-Tucker du problème (Q) **[5]**.

Si la matrice H est semi-définie positive alors (Q) est un problème d'optimisation convexe. Dans ce cas, (i) est toujours vérifié et **Lemme 1** est connu.

Soit d* vérifiant (ii) et (iii), par un simple calcul de (ii) on obtient:

$q(d) = \langle g,d\rangle + 1/2 \langle Hd,d\rangle = q(d^*) - \mu/2[d^t d - d^{*t} d^*] + 1/2(d-d^*)^t(H+\mu I)(d-d^*)$ (**)

Condition nécéssaire:

Soit d* une solution de (Q), alors d'après (**):

$(d-d^*)^t(H+\mu I)(d-d^*) \geq \mu[d^t d - d^{*t} d^*]$ pour chaque d tel que $\|d\| \leq \delta$.

Si $d^* = 0$ alors $g = 0$ et on a $\min\{ d^t H d : \|d\| \leq \delta\} = 0$. Autrement dit H est semi-définie positive, ce qui montre que $(H+\mu I)$ est semi-définie positive pour chaque $\mu \geq 0$.

Si $d^* \neq 0$, alors on a $(d-d^*)^t(H+\mu I)(d-d^*) \geq 0 \ \forall d : d^t d = d^{*t} d^*$, ce qui est équivalent à $p^t(H+\mu I)p \geq 0 \ \forall p, p^t d^* \neq 0$. Finalement la continuité de la forme quadratique q permet d'obtenir $p^t(H+\mu I)p \geq 0 \ \forall p$.

Condition suffisante:

Soit d* qui vérifie les trois conditions du **Lemme 1**.

Si $\|d^*\| < \delta$ alors (iii) implique que $\mu = 0$ et par suite d* est une solution de (Q) en vertu de (**).

Si $\|d^*\| = \delta$ alors (**) implique que d* est une solution de (Q) car $d^t d \leq d^{*t} d^*$ pour chaque d tel que $\|d\| \leq \delta$ et $(H+\mu I)$ est semi-définie positive.

L'unicité de μ est démontrée dans le **Lemme 4**.

En fait (Q) admet une solution d avec $\|d\| = \delta$ si et seulement si H est définie positive et $\|H^{-1}g\| < \delta$. μ étant le multiplicateur de Lagrange associé à la contrainte $\|d\|^2 \leq \delta^2$.

On définit la fonction ϕ sur $\{\mu \in \mathbb{R} : (H+\mu I)$ est non singulière$\}$ par :

$$\phi(\mu) = \|d(\mu)\|$$

où $d(\mu)$ est la solution de $(H + \mu I)d = -g$.

Soient $\lambda_1 \leq \lambda_2 \leq \lambda_3 \leq \ldots\ldots \leq \lambda_n$ les valeurs propres de la matrice H et $v_1, v_2, v_3, \ldots\ldots, v_n$ les vecteurs propres de H associés respectivement à λ_1, λ_2, $\lambda_3, \ldots\ldots, \lambda_n$, on désigne par H^+ le pseudo-inverse de la matrice H et par $\mathcal{N}(H-\lambda_1 I)$ le noyau de $(H-\lambda_1 I)$ (i.e. $\mathcal{N}(H-\lambda_1 I) = \{x \in \mathbb{R}^n : (H-\lambda_1 I)\, x=0\}$), et $J_1 = \{i : \lambda_i=\lambda_1\}$. Il est clair que la solution de (Q) est intimement liée à l'équation non linéaire $\phi(\mu) = \delta$ pour $\mu \in]-\lambda_1, +\infty[$. Plus précisément c'est le cas où (Q) admet une solution sur la frontière du domaine admissible et $\mu > \max(0, -\lambda_1)$ dans le Lemme 1, dans ce cas l'algorithme de Hebden (présenté ci-dessous) est connu comme étant un algorithme efficace pour résoudre $\phi(\mu) - \delta = 0$.
Le "cas critique" correspond à $\mu = -\lambda_1$.
Considèrons maintenant la résolution de l'équation:

$$(H + \mu I)d = -g \quad \text{pour } \mu \geq -\lambda_1 \qquad (***)$$

à l'aide des éléments propres de H, si $\mu \geq -\lambda_1$ alors la solution de (Q) est définie par

$$v_i^t d = \frac{-v_i^t g}{(\lambda_i+\mu)} \qquad \text{pour } i=1,\ldots,n.$$

$$\text{Donc } \phi(\mu) = \|d(\mu)\| = \left[\sum_{i=1}^{n} \frac{(v_i^t g)^2}{(\lambda_i+\mu)^2}\right]^{1/2}$$

La fonction $\phi(\mu)$ est positive et strictement décroissante sur $]-\lambda_1, +\infty[$ avec $\lim_{\mu \to +\infty} \phi(\mu) = 0$, alors l'équation $\phi(\mu) = \delta$ admet au moins une solution dans cet intervallé.

Lemme 2. [31,32]
1°. Si g est perpendiculaire à $\mathcal{N}(H-\lambda_1 I)$ alors le problème $(***)$ admet toujours une solution.
2°. Le problème $(***)$ avec $\mu = -\lambda_1$ admet une solution si et seulement si g est perpendiculaire à $\mathcal{N}(H-\lambda_1 I)$.
De plus, l'ensemble de ses solutions est $d^* + \mathcal{N}(H-\lambda_1 I)$ avec d^* est définie par:

$$v_i^t d^* = 0 \qquad \text{si } i \in J_1$$

$$v_i^t d^* = \frac{-v_i^t g}{(\lambda_i+\mu)} \qquad \text{sinon } \Diamond$$

Lemme 3. [31,32]
On suppose que la matrice H est non nulle.
1) Si le vecteur g est non nul alors la fonction $\phi(\mu)$ est positive et strictement décroissante sur $]-\lambda_1, +\infty[$ avec $\lim_{\mu \to +\infty} \phi(\mu) = 0$.

2) Si g n'est pas perpendiculaire à $\mathcal{N}(H-\lambda_1 I)$ alors:

$$\lim_{\substack{\mu \to -\lambda_1 \\ \mu > -\lambda_1}} \phi(\mu) = +\infty$$

3) Si g est perpendiculaire à $\mathcal{N}(H-\lambda_1 I)$ alors:

$$\phi(\mu) = \left[\sum_{i \notin J_1} \frac{(v_i^t g)^2}{(\lambda_i+\mu)^2} \right]^{1/2}$$

$$\lim_{\substack{\mu \to \lambda_1 \\ \mu > -\lambda_1}} \phi(\mu) = \left[\sum_{i \notin J_1} \frac{(v_i^t g)^2}{(\lambda_i-\lambda_1)^2} \right]^{1/2} = \|(H-\lambda_1 I)^{+} g\|$$

Dans ce cas $\phi(\mu)$ peut être prolongée par continuité en $-\lambda_1$ en posant : $\phi(-\lambda_1) = \|(H-\lambda_1 I)^{+} g\|$. ◊

Remarque:

Dans la construction de l'algorithme pour résoudre (Q), Le **Lemme 3.** est utilisé (voir méthode de courbure négative) de la manière suivante:

1) Pour $\delta>0$ fixé, si l'on ne peut pas trouver $\varepsilon>0$ suffisamment petit (par la technique de dichotomie par exemple) tel que $\phi(-\lambda_1+\varepsilon) \geq \delta$ alors g est perpendiculaire à $\mathcal{N}(H-\lambda_1 I)$.

2) Si g est perpendiculaire à $\mathcal{N}(H-\lambda_1 I)$ alors pour $\varepsilon'>0$ suffisamment petit, $d(-\lambda_1+\varepsilon')$ et $\phi(-\lambda_1+\varepsilon')$ représentent une acceptable approximation de $-(H-\lambda_1 I)^{+} g$ et de $\|(H-\lambda_1 I)^{+} g\|$ respectivement. Ainsi on évite le calcul coûteux de $(H-\lambda_1 I)^{+} g$ dans le cas critique.

Lemme 4.

On suppose $g \neq 0$ dans (Q), (voir **Lemme 5.** pour le cas g=0)

1°. Si $\lambda_1 > 0$ (i.e. H est définie positive) alors :

(i) Si $\|H^{-1}g\| \leq \delta$ alors $d_k = - H^{-1}g$ est une solution unique de (Q) ($\mu=0$ dans le **Lemme 1** et $\phi(\mu) - \delta = \|d(\mu)\| - \delta \leq 0$).

(ii) Sinon, $\|H^{-1}g\| > \delta$ et $\phi(0)-\delta = \|d(0)\| - \delta > 0$, alors il existe une solution unique $\mu>0$ telle que : $\phi(\mu) = \delta$.

2°. Si $\lambda_1 = 0$ (i.e. H est seulement semi-définie positive) alors :

(i) Si $\|(H-\lambda_1 I)^{+} g\| \leq \delta$ et $(H-\lambda_1 I)(H-\lambda_1 I)^{+} g = g$ (cette égalité est vérifiée si et seulement si g n'est pas perpendiculaire à $\mathcal{N}(H-\lambda_1 I)$), alors:

$\{d \in \mathbb{R}^n : d = - (H-\lambda_1 I)^{+} g + u$, avec $u \in \mathcal{N}(H-\lambda_1 I)$ et $\|d\|^2 \leq \delta^2\}$ est l'ensemble des solutions du problème (Q) et $\mu^* = 0$ est l'unique réel positif qui vérifie le **Lemme 1**)(voir **Lemme 3.**).

(ii) Sinon $\|(H-\lambda_1 I)+g\| > \delta$ ou $(H-\lambda_1 I)(H-\lambda_1 I)+g \neq g$, alors $\phi(\mu)-\delta = 0$ admet une solution unique $\mu^* > -\lambda_1$ telle que μ^* est l'unique réel positif qui vérifie le **Lemme 1**)(voir **Lemme 3.**).

3°. Si $\lambda_1 < 0$ alors :

(i) Si $\|(H-\lambda_1 I)+g\| \leq \delta$ et $(H-\lambda_1 I)(H-\lambda_1 I)+g = g$ (cette égalité est vérifiée si et seulement si g n'est pas perpendiculaire à $\mathcal{N}(H-\lambda_1 I)$) alors:

$\{d \in \mathbb{R}^n : d = -(H-\lambda_1 I)+g+u$, avec $u \in \mathcal{N}(H-\lambda_1 I)$ et $\|d\|^2 = \delta^2\}$ est l'ensemble des solutions du problème (Q) et $\mu^* = -\lambda_1$ est l'unique réel positif qui vérifie le **Lemme 1**) (voir **Lemme 3.**).

(ii) Sinon $\|(H-\lambda_1 I)+g\| > \delta$ ou $(H-\lambda_1 I)(H-\lambda_1 I)+g \neq g$, alors $\phi(\mu)-\delta = 0$ admet une solution unique $\mu^* > -\lambda_1$ telle que μ^* est l'unique réel positif qui vérifie le **Lemme 1**)(voir **Lemme 3.**).

4°. Si $\lambda_n \leq 0$ (la fonction quadratique q(d) est concave) alors (Q) admet uniquement des solutions sur la frontière de $\{d : \|d\| \leq \delta\}$ à moins que g et H ne soient nulles (Rockafellar 1970).

(i) Si $\lambda_1 = 0$ alors H = 0, dans ce cas la solution de (Q) est:

$\{-(g/\|g\|)\,\delta\}$ si $g \neq 0$

$\{d : \|d\| \leq \delta\}$ si $g=0$ (cf. **Lemme 5.**)

(ii) Si $\lambda_1 < 0$ alors on applique les résultats de **3°.** ◊

La situation g = 0 est intimement liée à la théorie variationnelle spectrale **[1]**.

Lemme 5.

Si g=0 dans le problème (Q) alors l'ensemble des solutions de (Q) est :

* $\{0\}$ si $\lambda_1 > 0$
* $\{d \in \mathcal{N}(H-\lambda_1 I) : \|d\| \leq \delta\}$ si $\lambda_1 = 0$
* $\{d \in \mathcal{N}(H-\lambda_1 I) : \|d\| = \delta\}$ si $\lambda_1 < 0$. ◊

8. Algorithmes pour la résolution du problème local (Q) :

Nous allons présenter deux algorithmes pour la résolution du problème local (Q). Le premier, l'algorithme de Hebden est généralement appliqué quand la matrice H est semi-définie positive et le deuxième, méthode de courbure négative est utilisée dans le "cas critique" qui correspond à 2° (i) et 3° (i) dans le Lemme 4.

Ce dernier algorithme nécessite le calcul de la plus petite valeur propre λ_1 de H et un vecteur propre de H associé à λ_1.

Dans le cas critique, sauf si un système d'éléments propres de H est facilement obtenu (cf. Lemme 2), reconnaître chacune des situations 2° (i) et 3° (i) (dans le Lemme 4) peut être très coûteux.

Heureusement, il y a une acceptable alternative due à (Shultz, Schnabel et Byrd 1985). En pratique, leur technique d'approximation de la solution paraît être aussi performante que la méthode exacte la plus coûteuse.

a- La méthode de Hebden

Nous allons décrire maintenant l'algorithme pour la résolution de (Q) dans le cas où l'équation non linéaire

$$\phi(\mu) - \delta = 0,$$

avec $\phi(\mu) = \left[\sum_{i=1}^{n} \left(\frac{v_i^t g}{\lambda_i + \mu} \right)^2 \right]^{1/2}$

admet un unique zéro dans $]-\lambda_1, +\infty[$. Reinsch[36] et Hebden [10] ont observé indépendamment que la fonction $\phi^2(\mu)$ est une fonction rationnelle en μ avec un pôle d'ordre 2 sur l'ensemble $\{-\lambda_i : 1 \leq i \leq n\}$.
La méthode de Newton qui est basée sur une approximation linéaire locale de $\phi(\mu)$ paraît alors ne pas être la meilleure méthode pour résoudre (***), car la structure rationnelle de $\phi^2(\mu)$ n'est pas prise en compte.
A sa place, pour résoudre (***), une itération peut être dérivée d'une approximation rationnelle locale ϕ^* de ϕ.
L'itération est obtenue en imposant $\phi^*(\mu) = \gamma/(\alpha+\mu)$ à satisfaire les conditions suivantes :

$$\phi(\mu) = \phi^*(\mu) \quad \text{et} \quad \phi'(\mu) = \phi^{*\prime}(\mu)$$

où nous prenons μ comme l'approximation courante de μ^*.

Cet approximation est alors améliorée par μ°, solution de $\phi^*(\mu^\circ) = \delta$. L'itération résultante est donc :

$$\mu_{k+1} = \mu_k + \left(\frac{\phi(\mu_k)}{\phi'(\mu_k)} \right) \left(\frac{\delta - \phi(\mu_k)}{\delta} \right) \qquad (\$)$$

En fait, l'algorithme de Hebden peut être considéré comme l'algorithme de Newton appliqué à l'équation :

$$\psi(\mu) = \frac{1}{\delta} - \frac{1}{\phi(\mu)} = 0 \qquad \text{pour } \mu \in]-\lambda_1, +\infty[$$

Cette méthode admet une convergence quadratique, mais la plus importante caractéristique de ($) est qu'en général le nombre d'itérations démandé, pour obtenir une approximation acceptable de μ^*, est très petit car l'itération est dérivée d'une approximation rationnelle locale de ϕ.

L'itération ($) peut être implémentée sans la connaissance explicite du système d'éléments propres de H. Cette importante observation due à Hebden permet d'implémenter ($) tout simplement en résolvant des systèmes linéaires définis par $(H+\mu I)$.

En effet, il est facile de voir que :

$$\phi(\mu) = \|d(\mu)\| \text{ et } \phi'(\mu) = -(1/\phi(\mu))d(\mu)^t(H+\mu I)^{-1}d(\mu)$$

où $(H+\mu I)d(\mu) = -g$.

Par suite l'algorithme de Hebden peut s'écrire comme suit :
Soit $\mu_0 \geq 0$ avec $(H+\mu_0 I)$ définie positive et $\phi(\mu_0) > \delta$.
0. i = 0, 1, 2, ... jusqu'à convergence
1. On factorise $(H_k+\mu I) = R^tR$.
2. On résout $R^tR\, d = -g$.
3. On résout $R^tq = d$.
4. $\mu_{i+1} = \mu_i + \left[\frac{\|d\|}{\|q\|}\right]^2\left[\frac{\|d\| - \delta_k}{\delta_k}\right]$.

i = i+1 et on retourne à 0).

Dans cet algorithme R^tR est la factorisation de Cholesky avec R une matrice triangulaire supérieure.

b- Méthode de courbure négative [17]

Si H est indéfinie, on cherche par la méthode de courbure négative la direction qui donne une bonne décroissance de la forme quadratique q_k. On développe dans ce cas l'algorithme suivant :

1) Choisissant un $\alpha \in]-\lambda_1, c.\max(|\lambda_1|,\lambda_n)]$, où c>1 est une constante qui dépend de chaque problème.
2) Calculer $p_k = -(H + \alpha I)^{-1} g$.
3) si $\|p_k\| > \delta$, on applique la méthode de Hebden citée ci-dessus sur la matrice $H^* = H + \alpha I$ pour trouver la direction d_k.
4) Si $\|p_k\| = \delta$, alors $d_k = p_k$.
5) Si $\|p_k\| < \delta$, alors $d_k = p_k + \zeta v_1$, où v_1 est le vecteur propre associé à λ_1 et ζ est un réel choisit tel que $\|d_k\| = \delta$ et $\text{signe}(\zeta) = \text{signe}(v_1^t p_k)$.

Ainsi on peut appliquer les méthodes de projection (citées ci-dessous) pour calculer la direction d_k dans le cas où la matrice H est semi définie positive (la fonction f est convexe) :

c - Méthodes de projection :

l) La méthode du gradient projeté :

Soit H une matrice carrée d'ordre n symétrique semi-définie positive et $g \in \mathbb{R}^n$. Soit $f(x) = 1/2\langle x,Hx\rangle + \langle g,x\rangle$ la fonction quadratique convexe correspondante. On considère le problème d'optimisation convexe suivant :

(P) $\alpha = \inf\{f(x) : x \in C\}$ où C est un ensemble convexe fermé dans $\mathbb{R}^n$.

Soit S l'ensemble des solutions de **(P)**. On cite la caractérisation suivante concernant la solution de **(P)** (Ekelande & Temam 1974 **[6]**):
x^* est solution de **(P)** si et seulement si $x^* = \mathrm{Proj}_C [x^* - \rho \nabla f(x^*)]$, ρ étant un scalaire strictement positif. Proj_C désigne la projection orthogonale sur C :
$x = \mathrm{Proj}_C (y) \Leftrightarrow \|y-x\| = \min\{\|y-u\| : u \in C\}$.

La méthode du gradient projeté pour résoudre **(P)** est ainsi définie par : Partant d'un point initial $x_0 \in C$ arbitrairement choisi, on construit la suite $\{x_k\}$ tel que : $x_{k+1} = \mathrm{Proj}_C [x_k - \rho_k \nabla f(x_k)]$ (1)
où la suite des scalaires positifs $\{\rho_k\}$ est choisie de manière à assurer la convergence de $\{x_k\}$ vers une solution de **(P)**.
Nous allons décrire un choix particulièrement intéressant de (ρ_k) qui est basé sur les propriétés de contraction. Pour les autres choix, voir **[3,8,33]**.

Convergence de la méthode du gradient projeté :

Il est clair que si $\rho_k = \rho$ pour chaque k, alors (1) est l'itération du point fixe relative à $F_\rho(x) = \mathrm{Proj}_C [x - \rho \nabla f(x)]$. Dans ce cas, les résultats de convergence de $\{x_k\}$ sont fournis par l'étude de la contraction de F_ρ.

Théorème 2. [29]
On suppose H définie positive. Soit λ_1 (resp. λ_n) la plus petite (resp. la plus grande) valeur propre de H, alors **(P)** admet une solution unique et on a:

(i) $\|F_\rho(x) - F_\rho(y)\| \leq q(\rho)\|x-y\|$ où $q(\rho) = \mathrm{Max}\{|1-\lambda_1|, |1-\lambda_n|\}$

(ii) $q(\rho) < 1$ si et seulement si $0 < \rho < 2/\lambda_n$, de plus la valeur optimale ρ^* de ρ (i.e. ρ^* minimise $q(\rho)$sur $]0,2/\lambda_n[$) est :

$\rho^* = 2/(\lambda_1 + \lambda_n)$ et $q^* = q(\rho^*) = (\lambda_n - \lambda_1)/(\lambda_n + \lambda_1)$.

(iii) finalement, la suite $\{x_k\}$ définie par (1) avec $\rho_k = \rho$ pour chaque k, converge vers une solution de **(P)**. ◊

Remarque :

Pour calculer le pas d_k de notre problème, soit $C = \{x \in \mathbb{R}^n : \|x\| \leq \delta_k\}$.

On rappelle que si $x \in \mathbb{R}^n$, alors :

$$Proj_C(x) = \begin{cases} x & \text{Si } x \in C \\ \dfrac{\delta_k . x}{\|x\|} & \text{Sinon} \end{cases}$$

On prend $x_0 \in C$ arbitrairement choisi; $r = 0$;

$y_r = Proj_C [x_r - \rho \nabla q_k(x_r)] = Proj_C [x_r - \rho.(H_k x_r + g_k)] = Proj_C [(I - \rho H_k) x_r + g_k)]$

avec ρ est un scalaire strictement positif.

Par un simple calcul, on remarque que pour assurer la convergence il faut que : $0 < \rho < 2/ \lambda_n$. (λ_n est la plus grande valeur propre de H_k).

D'où : $x_{r+1} = x_r + \lambda(y_r - x_r)$ avec $0 \leq \lambda \leq 1$ choisi de telle façon que :

$q_k(x_{r+1}) = \inf\{ q_k(x) ; x \in [x_r, y_r] \}$. La suite $\{x_r\}$ converge vers d_k.

II) La méthode Primal Dual (Méthode d' UZAWA) : [5],[6]

Après l'introduction de la dualité (relative au problème primal **(P)**) et le Lagrangien $L(x,\mu)$, on définit le problème dual **(D)** et les relations entre les solutions duales et primales. L'algorithme d'Uzawa est basé sur cette dualité.

On suppose que C est un ensemble convexe fermé dans $\mathbb{R}^n$ défini analytiquement par : $C=\{x \in \mathbb{R}^n : f_i(x) \leq 0, i=1,...,m\}$ où f_i, $i=1,...,m$, sont des fonctions convexes continues sur $\mathbb{R}^n$.

Soit $\phi(x)=(f_1(x),..., f_m(x))$ pour chaque $x \in \mathbb{R}^n$. On a : $C=\{x \in \mathbb{R}^n : \phi(x) \leq 0\}$.

On définit le Lagrangien $L(x,\mu)$ sur $\mathbb{R}^n \times \mathbb{R}^m_+$ par :

$L(x,\mu) = f(x) + \langle \mu, \phi(x) \rangle$ où $\mu=(\mu_i)_{1 \leq i \leq m}$ et $\mathbb{R}^m_+ = \{\mu \in \mathbb{R}^m : \mu \geq 0\}$.

Il est clair que $L(x,\mu)$ est convexe en x (resp. concave en μ) pour μ fixé (resp. pour x fixé). La fonction $g(\mu) = Inf\{L(x,\mu) : x \in \mathbb{R}^n\}$ est concave.

Le problème dual **(D)** s'écrit alors :

(D) : $\beta = \sup\{g(\mu) : \mu \in \mathbb{R}^m_+\}$

Un élément $(x^*,\mu^*) \in \mathbb{R}^n \times \mathbb{R}^m_+$ est un point selle du Lagrangien $L(x,\mu)$ si:

$L(x^*,\mu) \leq L(x^*,\mu^*) \leq L(x,\mu^*)$ pour chaque $(x,\mu) \in \mathbb{R}^n \times \mathbb{R}^m_+$.

On note $S(\mu)$ ($\mu \in \mathbb{R}^m_+$) l'ensemble des solutions du problème suivant :

(P_μ) : $g(\mu) = Inf\{L(x,\mu) : x \in \mathbb{R}^n\}$

L'algorithme d'Uzawa est défini par :

Partant d'un point initial arbitraire $\mu_0 \in \mathbb{R}^m_+$, on calcule un élément x_0 de $S(\mu_0)$.

Si x_k et μ_k sont connus alors μ_{k+1} est défini par :

$$\mu_{k+1} = \text{Proj}_{\mathbb{R}^m_+}(\mu_k + \rho_k \phi(x_k))$$

où ρ_k est choisi sur l'intervalle $]0,\rho^*[$. (voir méthode du gradient projeté citée ci-dessus).

Remarques:

(i) $x = \text{Proj}_{\mathbb{R}^m_+}(y) \Leftrightarrow x_i = \begin{cases} y_i & \text{si } y_i \geq 0 \\ 0 & \text{sinon} \end{cases}$ $\quad i=1,...,m$

(ii) Dans la méthode de région de confiance, l'ensemble convexe fermé C est de la forme $C = \{x \in \mathbb{R}^n : 1/2\|x\|^2 \leq 1/2.\delta^2\}$ et m = 1 avec $S = \{x \in \mathbb{R}^n : (H+\mu I)x = -g\}$. Il est clair que $S(\mu)$ est un singleton si H est définie positive ou $\mu > 0$.

Convergence de la méthode d'Uzawa: [3],[6].

Théorème 3.

(i) $\alpha \geq \beta$

(ii) Si (x^*,μ^*) est un point selle de $L(x,\mu)$ alors x^* est solution de **(P)**, μ^* solution de **(D)** et on a : $\alpha = \beta$

(iii) Si il existe x_0 tel que $\phi(x_0)<0$ alors $\alpha = \beta$ et x^* est une solution de **(P)** si et seulement si il existe $\mu^* \in \mathbb{R}^m_+$ tel que (x^*,μ^*) est un point selle de $L(x,\mu)$.

(iv) Si H est définie positive ou C est borné alors S est non vide. ◊

Corollaire 1:

Si il existe x_0 tel que $\phi(x_0)<0$ alors x^* est une solution de **(P)** si et seulement si il existe $\mu^* \in \mathbb{R}^m_+$ tel que :

(i) $f(x^*) + \langle \mu^*, \phi(x^*) \rangle = \text{Inf}\{f(x) + \langle \mu^*, \phi(x) \rangle : x \in \mathbb{R}^n\}$

(ii) $\phi(x^*) \leq 0$ et $\langle \mu^*, \phi(x^*) \rangle = 0$

alors μ^* est nécessairement solution de **(D)**. ◊

Corollaire 2:

(i) Si μ^* est une solution de **(D)** alors $S(\mu^*)$ contient l'ensemble S des solutions de **(P)**. Précisement $S = \{x^* \in S(\mu^*) : \phi(x^*) \leq 0 \text{ et } \langle \mu^*, \phi(x^*) \rangle = 0\}$.

(ii) Si H est définie positive alors $S(\mu)$ est non vide et il se réduit à un seul élément: $S(\mu) = \{s(\mu)\}$. ◊

Théorème 4.

Si H est définie positive et il existe x_0 tel que $\phi(x_0)<0$ alors :

(i) $x_k = s(\mu_k)$

(ii) La fonction g est différentiable et $\nabla g(x) = \phi(s(\mu))$

(iii) La suite $\{\mu_k\}$ converge vers une solution μ^* de **(D)** et la suite $\{x_k\}$ converge vers une solution x^* de **(P)**. ◊

Ainsi, l'algorithme d'Uzawa résout les deux problèmes **(P)** et **(D)** en même temps. Notons que l'algorithme d'Uzawa est la méthode du gradient projeté appliquée au problème dual **(D)**.

9. Algorithme pratique dans le cas général.

0. Initialisation :

Soient x_0 arbitrairement choisi , $\delta_0 > 0$, $s = 0.2$, H_0 une matrice symétrique, dont on calcule la plus petite valeur propre λ_1 et un vecteur propre v_1 associé à λ_1 ; $\varepsilon > 0$ un réel suffisamment petit , $k = 0$.

1. On calcule $f_k = f(x_k)$, $g_k = \nabla f(x_k)$.

2. Si $g_k = 0$ on s'arrête : x_k est une solution optimale.

3. On calcule la direction d_k solution du problème :

$$(P_k) : \begin{cases} \min\ q_k(d) \\ \|d\| \le \delta_k \end{cases}$$

(i) Si $\lambda_1 > 0$, on résout $H_k d = -g_k$; si ($\|d\| \le \delta_k$) alors $d_k = d$; sinon, on utilise la méthode de Hebden (ou une autre méthode citée ci-dessus si la fonction est convexe) pour trouver d_k vérifiant :

$(H_k + \mu I)d = -g_k$, $\mu \ge 0$, $\mu(\delta_k - \|d\|) = 0$ avec $(H_k + \mu I)$ est semi-définie positive.

(ii) Sinon (si $\lambda_1 \le 0$) alors on prend $\alpha_k = -\lambda_1 + 0.0001$.

On calcule $p_k = -(H_k + \alpha_k I)^{-1} g_k$.

si $\|p_k\| > \delta_k$, on applique la méthode de Hebden sur la matrice $H = H_k + \alpha_k I$ pour trouver $\mu > 0$ tel que $\|(H_k + (\mu+\alpha_k)I)^{-1} g_k\| = \delta_k$. Dans ce cas, la direction d_k est donnée par : $d_k = -(H_k + (\mu+\alpha_k)I)^{-1} g_k$; sinon (i.e., $\|p_k\| \le \delta_k$) si $\|p_k\| = \delta_k$, alors $d_k = p_k$. Sinon (i.e., $\|p_k\| < \delta_k$) alors $d_k = p_k + \zeta v_1$, où v_1 est le vecteur propre associé à λ_1 et ζ est un réel choisi tel que : $\|d_k\| = \delta_k$ et $\text{sign}(\zeta) = \text{sign}(v_1^t p_k)$.

4. Soit $r_k = \dfrac{f(x_k) - f(x_k + d_k)}{q_k(0) - q_k(d_k)}$

   Si $r_k \le 0.25$ alors : $\delta_{k+1} = \delta_k / 2$

   Si $r_k \ge 0.75$ alors : $\delta_{k+1} = 2\delta_k$

   Sinon ($0.25 \le r_k \le 0.75$ alors : $\delta_{k+1} = \delta_k$

5. Si $r_k \le s$ alors $x_{k+1} = x_k$, $k = k+1$ et on passe à l'étape 3.
   Sinon, alors $x_{k+1} = x_k + d_k$

6. On calcule H_{k+1} (par l'un de deux formules **1** ou **2**) avec sa plus petite valeur propre λ_1 et le vecteur propre v_1 associé, $k \leftarrow k+1$ et on retourne à l'étape 1.

10. Résultats de convergence [16, 17, 18].

Avant de citer les conditions de convergence on définit une notation additionnelle : Soit $d = p(g,H,\delta) = \text{argmin} \{ \langle g,w\rangle + \langle w,Hw\rangle ; \|w\| \leq \delta\}$ et $pred(g,H,\delta) = -\langle g,p(g,H,\delta)\rangle - 1/2 \langle p(g,H,\delta),H.p(g,H,\delta)\rangle$.

Condition 1 : Il existe $c_1,s_1 > 0$ tel que $\forall g \in \mathbb{R}^n$, $\forall H \in \mathbb{R}^{n\times n}$ symétrique et $\forall \delta > 0$ on a : $pred(g,H,\delta) \geq c_1.\|g\|.\min(\delta,s_1.\|g\|/\|H\|)$.

Condition 2 : Il existe $c_2 > 0$ tel que $\forall g \in \mathbb{R}^n$, $\forall H \in \mathbb{R}^{n\times n}$ symétrique et $\forall \delta > 0$ alors on a : $pred(g,H,\delta) \geq c_2.(-\lambda_1(H)).\delta^2$.
(λ_1 étant la plus petite valeur propre de H).

Condition 3 : Si la matrice H est définie positive et $\| H^{-1}.g \| \leq \delta$, alors : $p(g,H,\delta) = - H^{-1}g$.

Théorème 5 [16, 18].

Si la fonction f est de classe C^1, bornée inférieurement dans $\mathbb{R}^n$, alors :

$$\lim_{k\to+\infty} \|g_k\| = 0$$

Théorème 6 [16].

Si la fonction f est de classe C^2, bornée dans $\mathbb{R}^n$ et si $\nabla^2 f$ est bornée dans l'ensemble $S= \{x\in \mathbb{R}^n : f(x)\leq f(x_0)\}$ alors l'algorithme de région de confiance avec $H_k = \nabla^2 f(x_k)$ possède les propriétés suivantes :

1. $\lim_{k\to+\infty} \|g_k\| = 0$
2. Si x^* est une valeur d'adhérence de la suite $\{x_k\}$ alors $\nabla^2 f(x^*)$ est semi-définie positive.
3. Si x^* est une valeur d'adhérence de la suite $\{x_k\}$ et si $\nabla^2 f(x^*)$ est non singulière, alors :
   - a - $\nabla^2 f(x^*)$ est définie positive
   - b - $\lim x_k = x^*$, et il existe $\varepsilon>0$ telle que : $\delta_k > \varepsilon$ pour k assez grand.
   - c - La convergence est superlinéaire.
4. Si la suite $\{x_k\}$ est bornée, il existe une valeur d'adhérence x^* de la suite $\{x_k\}$ telle que $\nabla^2 f(x^*)$ soit définie positive (dans ce cas les propriétés a,b et c citées en 3. sont vérifiées). ◊

Référence

[1] P. Lancaster [1969], Theory of Matrix, Academic Press, New York and London.
[2] R. T. Rockafellar [1970],
Convex Analysis, Princeton University Press, Princeton, New Jersey.
[3] J. Cea [1971], Optimisation : Théories et algorithmes. Masson, Paris.
[4] P. E. Gill and W. Murray [1972],
Quasi-Newton methods for unconstrained optimization.
The Journal of the Institute of Mathematics and its Applications, vol 9, pp. 91-108.
[5] D. G. Luenberger [1974], A combined penalty function and gradient projection method for nonlinear programming, J. Opt. Th. Applics, pp. 251-263.
[6] I. Ekeland & R. Temam [1974],
Analyse Convexe et problèmes variationnels. Dunod, Gauthier-Villars.
[7] M. J. D. Powell [1975], Convergence proprerties of a class of minimization algorithms. O. L. Mangazarian, R. R. Meyer, S. M. Robinson Editors, Non linear programming 2, pp 1-27, Academic press, New York.
[8] A. Auslender [1976], Optimisation, méthodes numériques. Masson, Paris.
[9] H. Mukal and E. Polak [1978], A second order method for uncontrained optimization. J.O.T.A. vol 26, pp. 501-513.
[10] M. D. Hebden [1978],
An algorithm for minimization using exact second derivatives.
Atomic Energy Research Establishment report T.P. 515, Harwell, England.
[11] J. J. Moré [1978], The Levenberg-Marquart algorithm : implementation and theory, Lecture Notes in Mathematics 630, G. A. Waston, ed., Springer-Verlag, Berlin-Heidelberg-New York,pp. 105-116.
[12] S. Kaniel and A. Dax [1979],
A modified Newton's method for unconstrained minimization.
SIAM J. Num. Anal., pp. 324-331.
[13] J. J. Moré and D. C. Sorensen [1979], On the use of directions of negative curvature in a modified Newton method. Math.Prog. 16, pp. 1-20.
[14] R. Fletcher [1980],
Pratical Methods of Optimization, vol 1, Jhon Wiley, New York.
[15] D. M. Gay [1981], Computing optimal constrained steps. SIAM J. Sci. Stat. Comput. 2, pp 186-197.
[16] J. J. Moré and D. C. Sorensen [1981], Computing a trust region step.
Argonne National Laboratory report, Argonne, Illinois.
[17] G. A. Schultz, R. B. Schnabel and R. H. Byrd [1982],
A family of trust region based algorithms for unconstrained minimization with strong global convergence properties. Departement of Computer Science , University of Colorado at Boulder, Boulder, Colorado 80309.
[18] D. C. Sorensen [1982],
Newton's method with a model trust region modification, SIAM J.
Numer. Anal. vol 19. N° 2, pp 409-426.
[19] J. J. Moré [1983],
Recent developments in algorithm and software for Trust Region Methods.
Mathematical Programming, The State of the Art, Springer, Berlin, pp 258-287.
[20] J. E. Dennis, R. B. Schnabel [1983],Numerical methods for unconstrained optimization and nonlinear equations. Printice-Hall.
[21] M. Minoux [1983], Programmation Mathematique. Tome1, Dunod.

[22] J. P. Penot, A. Roger, Updating the spectrum of a real matrix. Mathematics of Computation.

[23] Y. Yuan [1984],
An example of only linear convergence of trust region algorithms for nonsmooth optimization, IMA Journal of Numerical Analysis 4, pp. 327-335.

[24] Y. Yuan [1985],
On the superlinear convergence of a trust region algorithm for nonsmooth optimization, Mathematical Programming, vol 3, pp. 269-285.North-Holland.

[25] A. R. Conn, N. Gould, Ph. Toint [1986], Testing a class of methods for solving minimization problems with simple bounds on the variables. Report n° 86-3, University of Waterloo.

[26] I. S. Duff, J. Nocedal and J. K. Reid [1987],
The use linear programming for solution of sparse sets of nonlinear equations. SIAM J. SCI. STAT.COMPUT. vol 8, N°2, pp. 99-108.

[27] A. Roger [1987], Mise à jour du spectre d'une matrice symétrique. Rapport de recherche SNEA(P), n° AR/87-970.

[28] G. A. Schultz, R. B. Schnabel and R. H. Byrd [1988],
Approximate solution of the trust region problem by minimization over two-dimensional subspaces. Mathematical Programming, vol 40, pp.247-263, North-Holland.

[29] Pham Dinh Tao, S. Wang, A.Yassine [1989],
Training multi-layered neural network with a Trust-Region based algorithm. To appear in Mathematical Modelling & Numerical Analysis.

[30] J.R. Clermont, M. E. de la Lande, Pham Dinh Tao, A.Yassine [1989a],
Numerical simulation of axisymmetric converging flows using the stream tube method and a Trust Region optimization algorithm. Submitted to Structural Optimization.

[31] J.R. Clermont, M. E. de la Lande, Pham Dinh Tao, A.Yassine [1989b],
Analysis of plane and axisymmetric flows of incompressible fluids with the stream tube method. Numerical simulation by Trust Region optimization algorithm. Submitted in Numer. Math.

[32] Pham Dinh Tao [1989],
Numerical method for minimizing the Hermitian quadratic from on the unit Euclidean sphere of C^n. Submitted to SIAM Journal on Matrix Analysis and Applications.

[33] B. Pchenitchny & Y. Daniline [1977],
Méthodes numériques dans les problèmes d'extrémum. Edition Mir.

[34] J.R. Clermont, M. E. de la Lande, Pham Dinh Tao, A.Yassine [1989c],
A numerical method for solving converging flows using a stream tube analysis and Trust Region technique. To appear in Proceedings of 6th International Conference on Numerical Methods in Laminar and Turbulent Flow. Swansea, U.K.

[35] H. C. Reinsch [1967], Smoothing by spline functions, Numer. Math. 10, pp. 177-183.

[36] H. C. Reinsch [1971], Smoothing by spline functions II, Numer. Math. 16, pp. 451-454.

CHAPITRE VI

EXPERIMENTATIONS NUMERIQUES

VI.1. Applications de la méthode de région de confiance.

1. Réseaux Neuronaux.

Dans ce paragraphe, nous allons présenter tout d'abord la modélisation du problème d'apprentissage par un réseau de neurones en un problème d'optimisation et l'algorithme d'apprentissage le plus utilisé (GBP).

Thème de recherche actuel important mené dans l'Equipe de Calcul Parallèle du Labo TIM 3 (F. Robert, S. Wang), dont il s'agit d'une collaboration intéressante et fructueuse sur la modélisation et la résolution des problèmes d'optimisation convexe et non convexe.

Ensuite nous donnerons les résultats des expérimentations numériques obtenues par GBP et par le nouvel algorithme basé sur le principe de région de confiance (notée TR "Trust Region" ici) que nous avons adapté à ce problème (cf. Chapitre V).

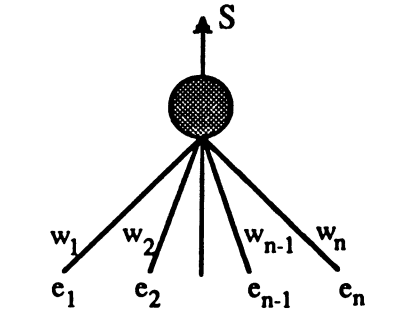
Ces résultats montrent que ce dernier est beaucoup plus rapide et robuste par rapport à GBP. Et en plus, il rend la conception des réseaux beaucoup plus indépendante du problème particulier traité. (pour plus de détail, voir [9]).

1.1. RESEAUX MULTI-COUCHES :

Un réseau multi-couches est un réseau de cellules réparties en couches interconnectées. Le fonctionnement détaillé d'une cellule est illustré par la **figure 1**. La sortie d'une cellule est calculée comme la fonction sigmoïdale de son entrée totale, qui elle même est une somme pondérée de toutes les entrées à cette cellule. Le réseau est construit par interconnection des cellules en les structurant en couche (**figure 2**).

On suppose que les cellules des couches du haut ne peuvent recevoir leurs entrées qu'à partir des cellules des couches du bas. Le poids de la connexion de la cellule j à la cellule i est représenté par w_{ij}.

En général, $w_{ij}=0$ signifie que la cellule i ne reçoit pas d'information de la cellule j, mais le fait que la cellule i ne reçoive pas d'information de la cellule j peut signifier aussi qu'il n'y pas de lien entre ces deux cellules. Donc ici on distingue le cas où $w_{ij}=0$ et le cas où les deux cellules sont disconnectées, puisque dans le premier cas w_{ij} est susceptible d'être modifié alors que dans ce dernier il ne peut pas l'être.



$S_i = f(A_i)$ où
$A_i = \sum_i w_i * e_i$: Entrée totale
$f(x) = (1-\exp(-\alpha x))/(1+\exp(-\alpha x))$

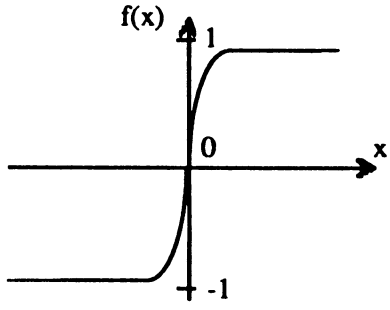


figure 1

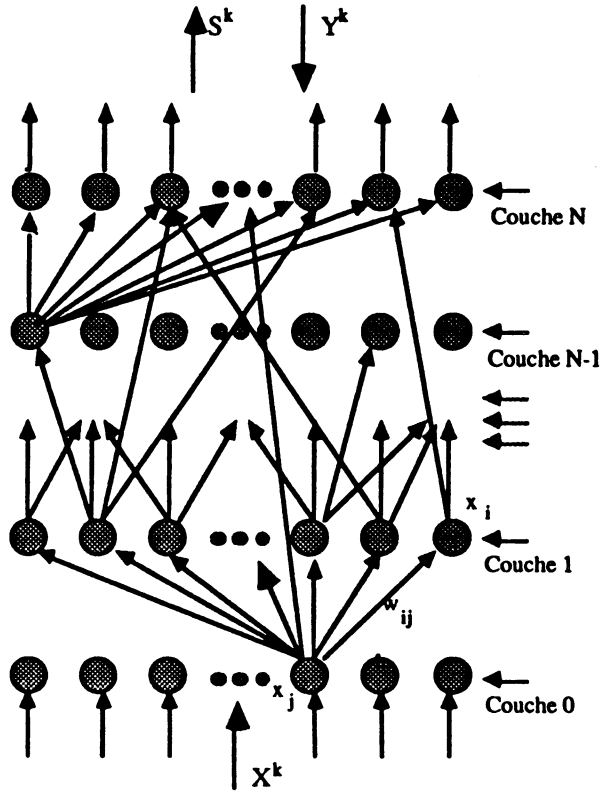


figure 2

Le principe de fonctionnement du réseau est le suivant : Etant donné un pattern d'entrée sous forme d'un vecteur (qui pourrait représenter une unité d'information réelle), les cellules dans la couche 0 prennent les composantes du vecteur d'entrée comme leurs états; puis chaque cellule dans les couches suivantes calcule son état dès que toutes les cellules d'entrée ont calculé leurs propres états. On pourrait imaginer une progression successive des états : d'abord ce sont les cellules de la couche 1 qui calculent leurs états, ensuite les cellules de la couche 2, ainsi de suite... A la fin de cette progression, le vecteur récupéré, constitué des états des cellules dans la dernière couche, est considéré comme la sortie du réseau. Il est évident que l'objectif de la construction du réseau est de rendre la sortie (qui dépend de l'entrée) significative.

Supposons que l'on ait K associations entrée-sortie à réaliser $(X^{(k)}, Y^{(k)})$, pour k=1,...,K. Ici $X^{(k)}$ appartenant à $[-1,1]^n$, $Y^{(k)}$ à $[-1,1]^m$ représentent respectivement les informations à classifier et les "étiquettes" représentant ces informations. $X^{(k)}$ est souvent choisi comme un des exemples le plus "représentatif" d'un ensemble d'informations. Bien sûr ils peuvent être plusieurs, dans ce cas les $Y^{(k)}$ correspondants doivent être les mêmes. L'état d'une cellule est déterminée par

$$S_i = f(A_i) \quad \text{avec} \quad A_i = \sum_j w_{ij} * x_j \qquad (1)$$

ici f_i est une fonction sigmoïdale définie par

$$f_i[x] = \frac{1-\exp(-\alpha_i x)}{1+\exp(-\alpha_i x)} \qquad \text{Avec } \alpha_i>0 \qquad (1.2)$$

La fonction de coût dépendant des poids de toutes les connexions est définie par

$$C(W)=\sum_{k=1}^{K} \| S^{(k)}-Y^{(k)} \|_2^2 \qquad (1.3)$$

avec $S^{(k)}$ définie comme la sortie effective du réseau pour l'entrée $X^{(k)}$.

Etablir un réseau qui réalise cet ensemble d'associations signifie :

1). Choisir une structure de réseau, ce qui comprend le nombre de couches, le nombre de cellules dans chaque couche, le mode de lien entre les cellules (existe-t-il une connexion d'une cellule à une autre ?).

2). Trouver les poids des connexions.

Pour le premier problème, il n'y a pas de théorie qui nous permette de faire un choix optimal. La solution est souvent très empirique.

On choisit généralement beaucoup trop de cellules pour augmenter la dimension de l'espace de décision afin de réaliser facilement les associations désirées.

Pour le deuxième problème, il existe certaines méthodes dérivées des celles conçues pour les problèmes d'optimisation. La plus connue est fondée sur la méthode du gradient. Elle s'appelle GBP (Gradient Back-Propagation).

Appliquer la méthode du gradient consiste à calculer les dérivées partielles de C par rapport à tous les poids w_{ij}, et à modifier w_{ij} selon la formule :

$$w_{ij} = w_{ij} - \beta * \frac{\partial C}{\partial w_{ij}} \qquad (1.4)$$

β doit être calculé afin de minimiser la fonction C dans la direction du gradient. Et le procédé doit être répété plusieurs fois pour que C(W) atteigne un de ses minimums (local ou global).

Ce procédé est en effet un apprentissage global puisque l'amélioration des poids des connexions s'effectue suivant les "erreurs" que le réseau commet pour l'ensemble des patterns d'entrée. On voudrait que le réseau puisse adapter ses poids petit à petit pour chaque association. Si on reprend C(W) sous la forme :

$$C(W)=\sum_{k=1}^{K} C_k(W) \qquad \text{Avec } C_k(W) = \| S^{(k)}-Y^{(k)} \|_2^2 \qquad (1.5)$$

ceci revient à appliquer la méthode d'optimisation sur C_k une fois seulement à chaque étape, ce qui consiste à calculer

$$w_{ij} = w_{ij} - \beta * \frac{\partial C_k}{\partial w_{ij}} \qquad (1.6)$$

Pour la simplicité, β ici est choisit comme un petit réel positif, ou une suite décroissante de réels positifs.

Le problème crucial de ce procédé est la convergence!. Il n'y pas d'analyse théorique qui l'affirme, mais l'experience montre que l'on pourrait effectivement conduire C(W) vers un minimum (souvent zéro) par le choix approprié de la structure du réseau, de l'ordre de présentation des patterns et du taux d'apprentissage α. Ces choix sont très empiriques.

L'algorithme "GBP" est un algorithme d'adaptation progressive du réseau à l'ensemble des patterns d'association entrée-sortie. Donc l'ordre de présentation des patterns, souvent appelé ordre d'apprentissage, doit être précisé. Deux stratégies sont fréquemment utilisées : la stratégie séquentielle et la stratégie aléatoire. Dans le mode séquentiel, on présente les patterns au réseau selon un ordre fixé a priori. Dans le mode aléatoire, on tire au hasard un entier entre 1 et K (nombre de patterns), on fait alors l'apprentissage sur le pattern qui correspond à l'entier tiré. Le terme "une itération d'apprentissage" correspond dans le cas séquentiel à un parcours de tous les patterns, et dans le cas aléatoire à K tirages. Il faut ajouter aussi que dans certaines applications (notamment la nôtre), on désire que certains patterns soient appris plus souvent que d'autres; pour cela il faut dupliquer ces patterns et les placer dans des endroits différents de la liste d'apprentissage. Enfin, pour chaque pattern pris dans l'ordre ou au hasard, on n'effectue qu'une seule fois l'apprentisage pour équilibrer l'influence de chaque pattern. Il est déconseillé de répéter l'apprentissage sur le même pattern.

1.2. Applications.

Trois expérimentations numériques sont présentées ci-dessous.

La première représente une comparaison de l'efficacité entre l'algorithme de gradient (GBP) et l'algorithme de région de confiance (TR) appliqués sur deux réseaux différents.

La deuxième montre l'avantage de l'algorithme TR, en remarquant les cas "critiques" où l'algorithme GBP ne converge pas.

La troisième vérifie l'indépendance de la convergence de l'algorithme TR de l'architecture du réseau et sa robustess par rapport à l'algorithme GBP.

1) Comparaison de l'efficacité :

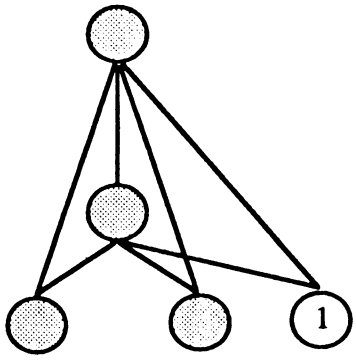


Figure 3 : xor_r_211, réseau de xor avec deux cellules d'entrée, une cellule intermédiaire et une cellule de sortie.

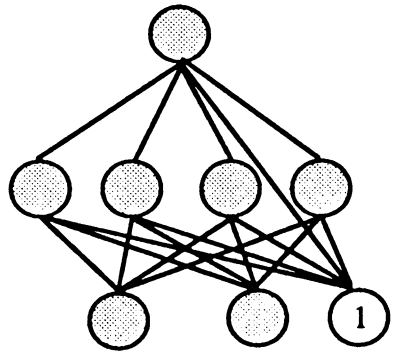


Figure 4 : xor_r_241, réseau de xor avec deux cellules d'entrée, 4 cellules intermédiaires et une cellule de sortie.

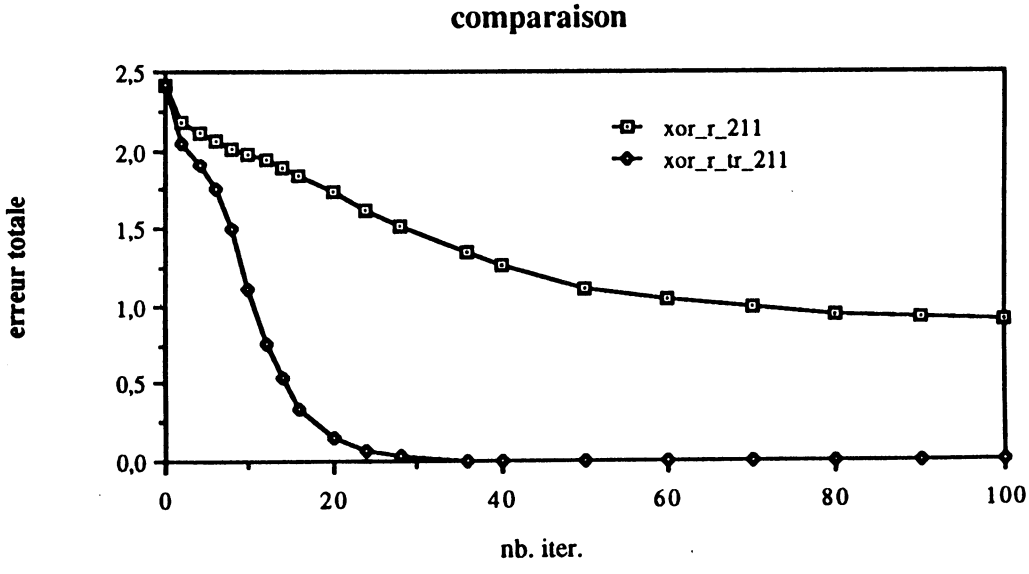


Figure 5 : efficacité de deux algorithmes appliqués sur le réseau illustré dans (Figure 3). xor_r_211 correspond à l'algorithme GBP

xor_r_tr_211 correspond à l'algorithme TR

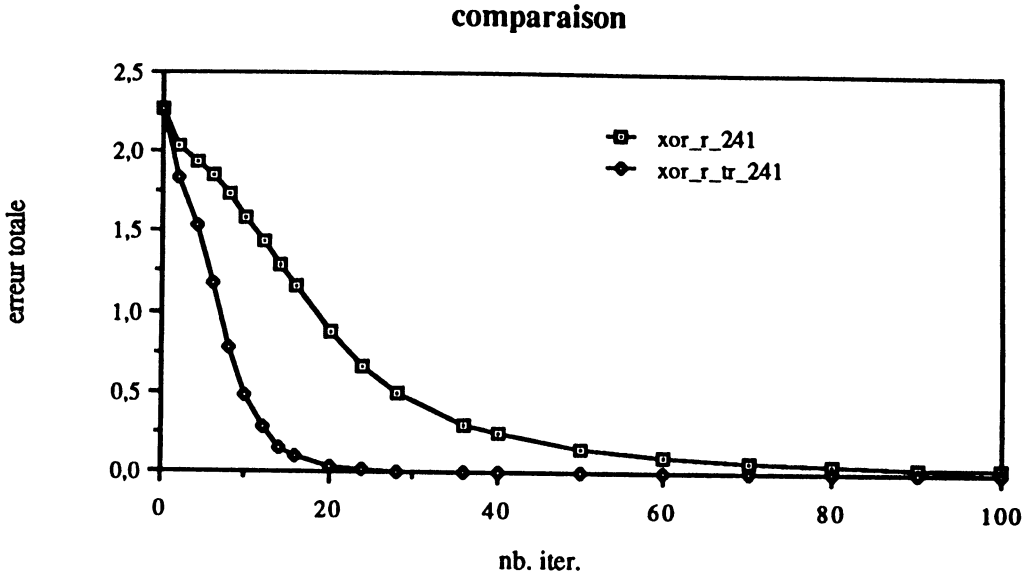


Figure 6 : efficacité de deux algorithmes appliqués sur le réseau illustré dans (Figure 4).
xor_r_241 correspond à l'algorithme GBP
xor_r_tr_241 correspond à l'algorithme TR.

2) Avantage de l'algorithme TR.

En appliquant les deux algorithmes sur deux réseaux xor_S_1 et xor_S_2 présentés ci-dessous (Figure 7 et Figure 8), les cas "critiques" de l'algorithmes GBP (Figure 9 et Figure 10) ne sont pas rares, ils se présentent dans le cas où le nombre de connexions est petit par rapport au nombre de patterns et si la majorité de poids de connexion ont le même signe. Dans ce cas on remarque aussi la difficulté de convergence de l'algorithme TR (Figure 10).

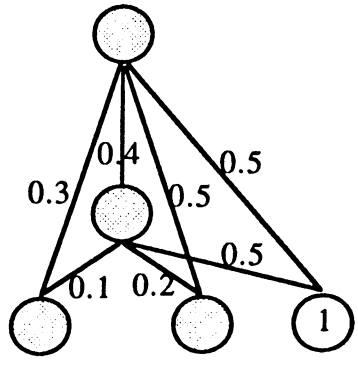


Figure 7 : réseau xor_S_1, réseau de xor avec deux cellules d'entrée,
deux cellules intermédiaires et une cellule de sortie avec des poids de connexion initiaux.

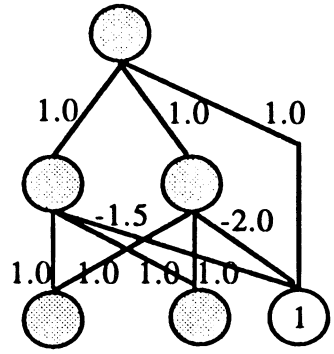


Figure 8 : réseau xor_S_2, réseau de xor avec deux cellules d'entrée,
deux cellules intermédiaires et une cellule de sortie avec des poids de connexion initiaux.

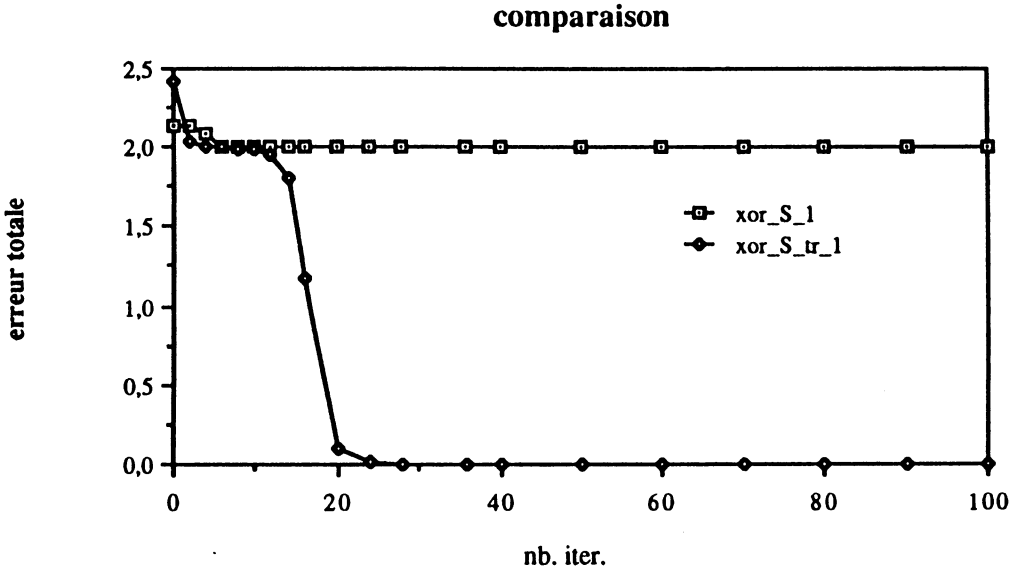


Figure 9 : efficacité de deux algorithmes appliqués sur le réseau illustré dans (Figure 7).
xor_S_1 correspond à l'algorithme GBP
xor_S_tr_1 correspond à l'algorithme TR.

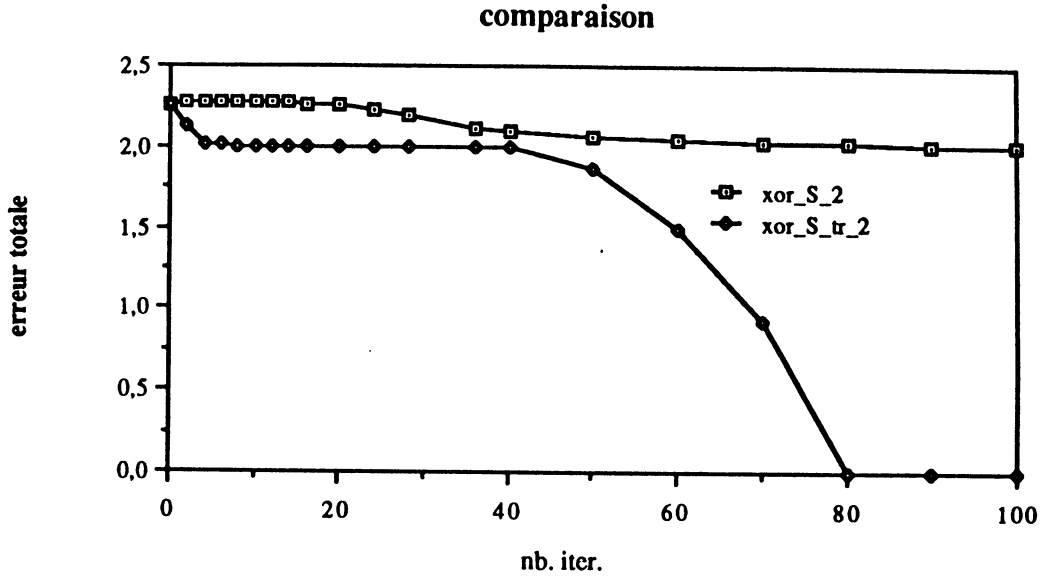


Figure 10 : efficacité de deux algorithmes appliqués sur le réseau illustré dans (Figure 8).
xor_S_2 correspond à l'algorithme GBP
xor_S_tr_2 correspond à l'algorithme TR.

3) Indépendence de la structure du réseau.

L'exemple suivant montre la robustesse de notre algorithme en respectant toujours la structure du réseau. Il s'agit de réaliser une fonction vectorielle de type boléenne par un réseau de 6-3-3. La fonction F est définie comme suit :

$$F : \{-1, 1\}^6 \rightarrow \{-1, 1\}^3$$
$$F(x_1, x_2, x_3, x_4, x_5, x_6)^t = (\bar{x}_1 x_2 + \bar{x}_2 x_1, \bar{x}_3 x_4 + \bar{x}_4 x_3, \bar{x}_5 x_6 + \bar{x}_6 x_5)^t.$$

Elle est composée en fait à partir de trois fonctions de XOR. Il y a, en total, 64 patterns d'association à apprendre.

Si on utilise l'algorithme de GBP, un des meilleurs choix pour l'architecture du réseau est celui illustré par la Figure 11. C'est un choix logique et naturel parce que chaque cellule de sortie ne dépend que de deux 2 cellules d'entrée et n'a besoin d'aucune information prevenant d'autres cellules d'entrée. Les trois cellules intermédiaires sont équalement "attribuées à" chacune des trois partie afin de rendre à chaque partie la même capacité de traiter le problème de séparabilité de XOR. Cette architecture est aussi la plus simple : toutes les connexions supplémentaires ne peuvent créer que des circulations des informations inutiles. On peut dire que dans ce réseau, il n'y a pas de connexions

abérrantes. Pourtant, dans une application réelle nous ne pouvons pas obtenir suffisament des informations pour déterminer un tel réseau "intelligent". Par contre, si on choisit une architecture du réseau avec beaucoup de connexions inutiles, le problème d'apprentissage devient très difficile. La Figure 12 illustre un cas où chaque cellule intermédiaire est connectée à toutes les cellules d'entrée, et chaque cellule de sortie est connectée à toutes les cellules d'entrée et à toutes les cellules intermédiaires. Nous avons remarqué que l'algorithme de RC fonctionne bien pour toutes les deux architectures, mais l'algorithme de GBP fonctionne déjà très mal avec la première (Figure 11) et pas du tout avec la seconde (Figures 12 et 13).

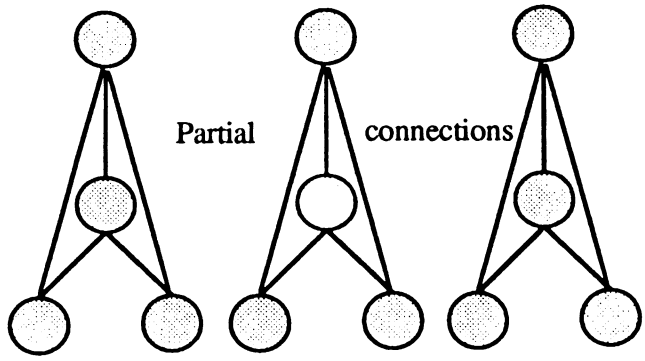


Figure 11 : un réseau 6-3-3 qui contient six cellules d'entrée, trois cellules intermédiaires et trois cellules de sortie. Il est composé de trois réseaux 2_1_1 .

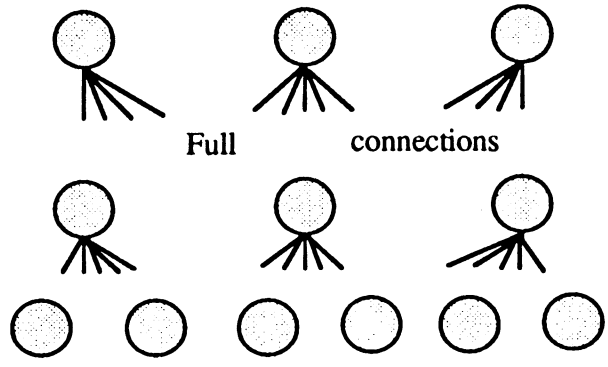


Figure 12 : un réseau 6_3_3 qui contient six cellules d'entrée, trois cellules intermédiaires et trois cellules de sortie. Chaque cellule reçoit des informations de toutes les cellules en dessous.

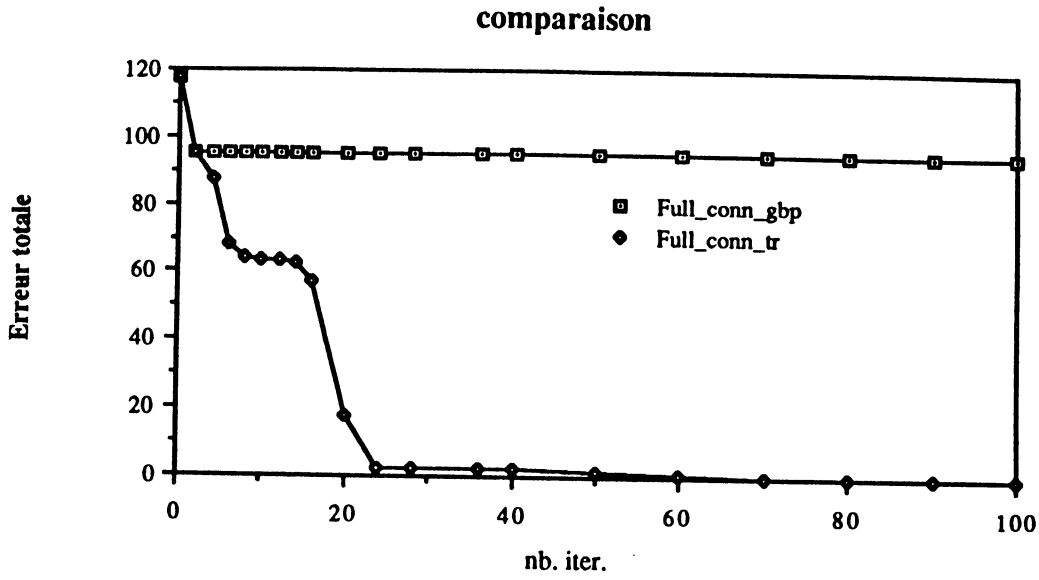


Figure 13 : le réseau xor de poids complets correspond à (Figure 8).
Full_conn_gbp correspond à l'algorithme GBP
Full_conn_tr correspond à l'algorithme TR.

2. Calcul d'écoulements de fluides incompressibles dans les conduites planes ou axisymétriques.

La simulation des écoulements en rapport avec des procédés industriels est d'un intérêt croissant depuis quelques années. Afin de résoudre ces écoulements complexes, les méthodes classiques appliquent en général des schémas de différences finies ou d'éléments finis.

Dans l'approche présentée ici, une méthode basée sur le concept de tube de courant est utilisée, pour traiter des écoulements de fluides incompressibles dans des géométries planes ou axisymétriques [23, 24].

Sous l'hypothèse d'absence de recirculations (écoulements secondaires, vortex), le domaine physique D est transformé en un domaine D_1 dans lequel les transformées des lignes de courant sont des droites parallèles (Figure I). Cette méthode permet de calculer l'écoulement sur des tubes de courant successifs, dans le domaine transformé, de la paroi vers la région centrale de l'écoulement.

Dans le cas isotherme, les inconnues sont la transformation f (de D_1 dans D) et la pression p. Avec cette transformation, la conservation de la masse est automatiquement vérifiée et seules les équations de la dynamique sont à prendre en compte, sous la forme :

$$\text{div } \sigma[f,p] = \mathbf{0} \qquad (1),$$

avec les conditions aux limites associées :

$$\mathfrak{L}[f,p] = \mathbf{0} \qquad (2).$$

σ désigne le tenseur des contraintes totales, tel que :

$$\sigma = -p\mathbf{I} + \mathbf{T} \qquad (3),$$

où $-p\mathbf{I}$ est la contribution sphérique de σ et $\mathbf{T}$ le tenseur des contraintes donné par la loi de comportement du matériau.

La discrétisation de (1) et (2) sur un tube de courant conduit à un système non-linéaire de n équations à n inconnues, de la forme :

$$\phi(X) = \mathbf{0} \qquad (4),$$

avec $X = (x_1, \ldots, x_n)$ et $\phi(X) = (\phi_1(X), \ldots, \phi_n(X))$ deux fois continûment différentiable de $\mathbb{R}^n$ dans $\mathbb{R}^n$.

Il est clair que le système non-linéaire (4) est équivalent au problème de moindres carrés non-linéaire suivant :

$$0 = \min\{f(X) = \frac{1}{2}\sum_{i=1}^{n}\phi_i(X)^2 \ : x\in \mathbb{R}^n\} \qquad (5)$$

Par simple calcul on obtient :

$$\nabla f(X) = \sum_{i=1}^{n}\phi_i(X)\nabla\phi_i(X) = J(X)^t\phi(X) \qquad (6)$$

$$\nabla^2 f(X) = \sum_{i=1}^{n}\nabla\phi_i(X)\nabla\phi_i(X)^t + \sum_{i=1}^{n}\phi_i(X)\nabla^2\phi_i(X) = J(X)^tJ(X) + \sum_{i=1}^{n}\phi_i(X)\nabla^2\phi_i(X) \qquad (7)$$

On constate qu'au voisinage d'une solution le dernier terme du second membre est négligeable, ce qui conduit à poser $H(X) = J(X)^tJ(X)$ comme une approximation de $\nabla^2 f(X)$ dans la méthode de région de confiance appliquée à ce problème.

Cette approximation connue sous le nom de Gauss-Newton est d'autant plus intéressante que le Jacobien $J(X)$ de notre problème est non singulière : $H(X)$ est donc définie positive.

Dans la suite, nous considérons l'application à un fluide newtonien dans une région axisymétrique, plus exactement un convergent d'angle α (Figure I).

Dans un premier temps, on a utilisé la méthode de Newton-Raphson pour résoudre le système (4), et cet algorithme converge jusqu'à un angle limite α^* de 29.3°; au-delà on observe une divergence.

L'algorithme de région de confiance a alors été appliqué et s'est avéré beaucoup plus stable et efficace que la méthode de Newton-Raphson.
Ainsi cet algorithme converge pour les angles $\alpha \leq \alpha^*$, vers une solution numérique identique à celle obtenue par la méthode de Newton. Au-delà de α^*, l'algorithme converge également pour $\alpha \leq 55°$; à partir de $\alpha = 60°$ on observe la divergence (qui est compatible avec la théorie).

La figure II indique, pour divers angles de convergents, la première ligne de courant calculée, ainsi que l'évolution de la pression le long de cette ligne de courant. (tests effectués sur une station APOLLO DN 4000, en double précision).

Ces résultats numériques sont tout à fait cohérents du point de vue physique. Par ailleurs, le seuil de convergence obtenu pour $\alpha_{lim} = 60°$ correspond physiquement à l'apparition de recirculations dans l'écoulement, ce qui est contraire à l'hypothèse de la méthode des tubes de courant (il est dans ce cas impossible de définir une transformation bi-univoque f entre D_1 et D).

Ainsi dans ce contexte, l'application de la méthode de région de confiance permet de calculer l'écoulement dans des convergents et également de prédire le seuil d'apparition de recirculations, celles-ci étant en général à éviter dans les procédés industriels (car elles engendrent des impuretés).

On présente ci-dessous les résultats graphiques :

Figure I Méthode de tube de courant : Transformation du domaine physique D en D_1.

Figure II Première ligne de courant calculée, et évolution de la pression adimentionnelle p/p_0 sur cette ligne de courant, pour $\alpha = 40°$, 50°, et 55°.

Figure I

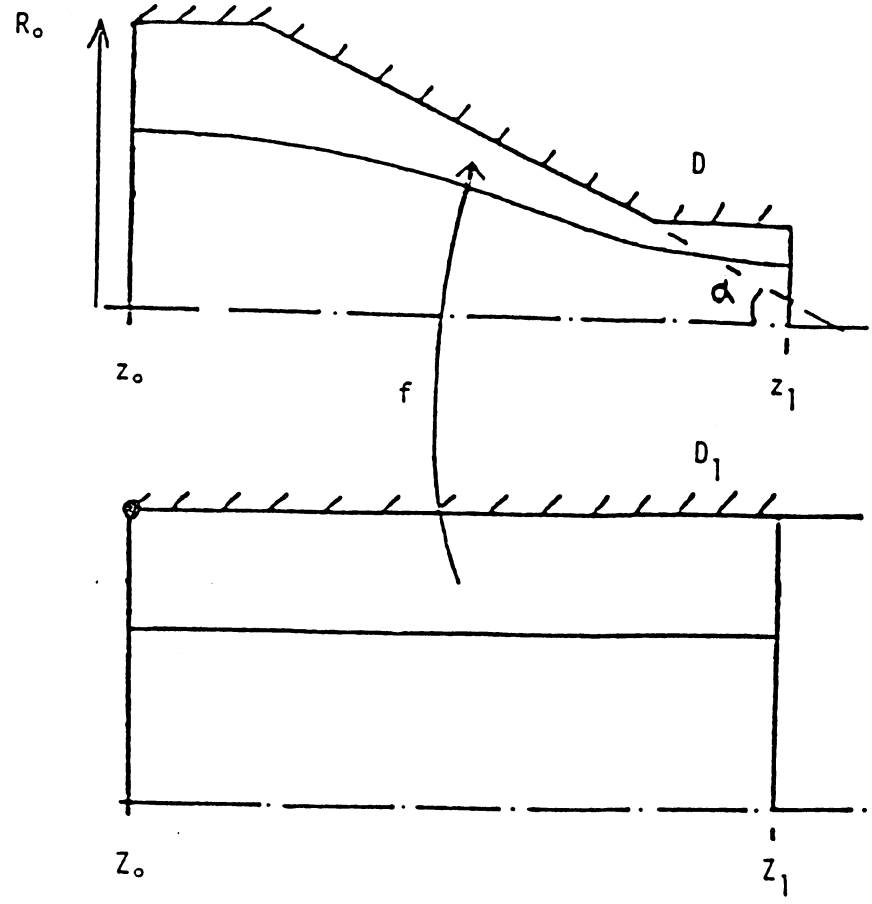
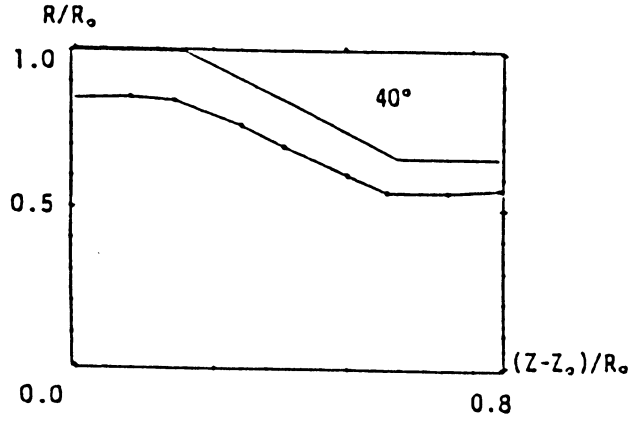
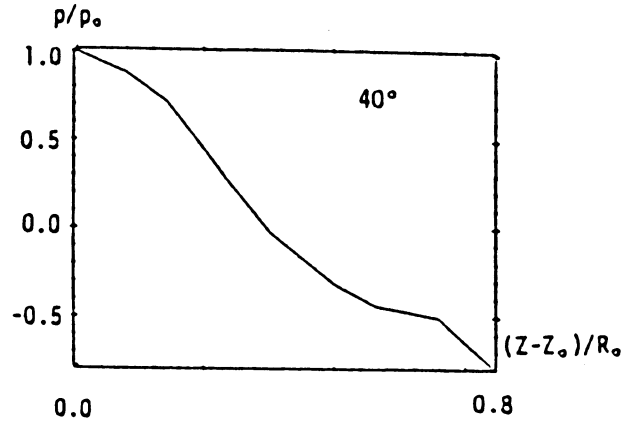
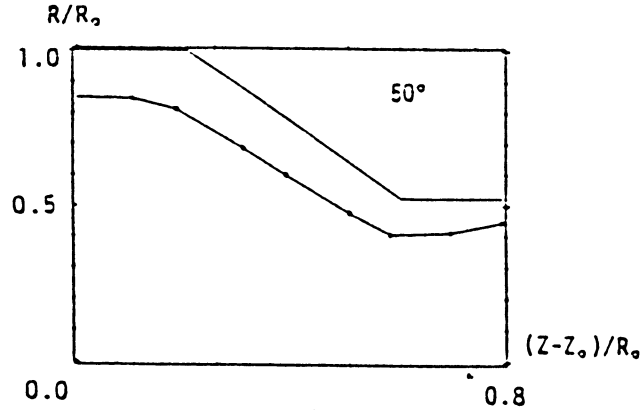
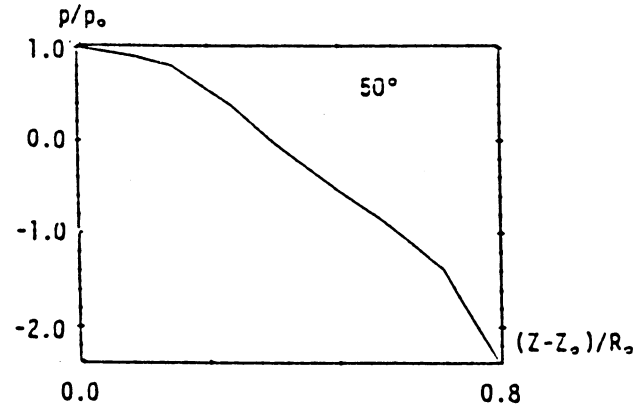


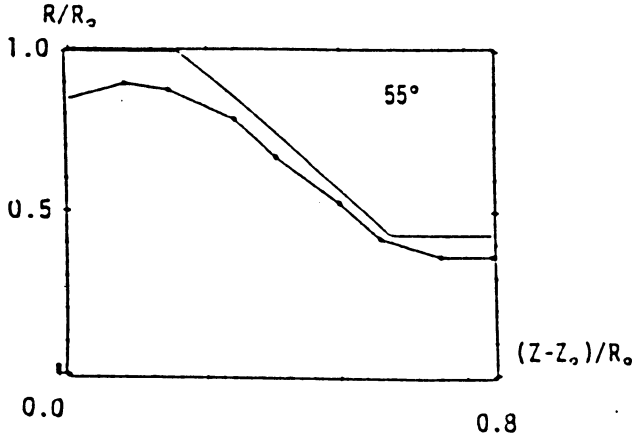
Figure II

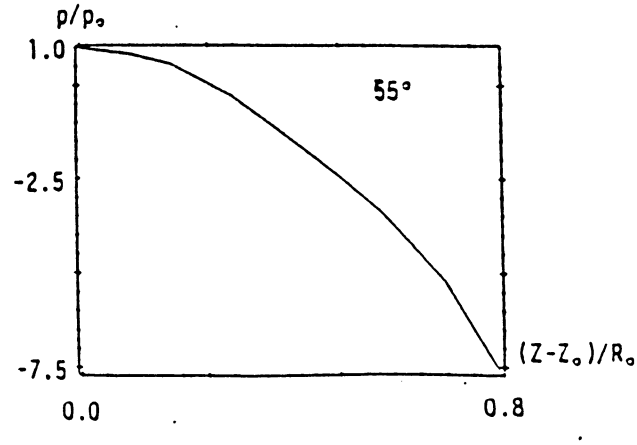

R/R₀
1.0
0.5
0.0
40°
(Z-Z₀)/R₀
0.8


p/p₀
1.0
0.5
0.0
-0.5
40°
(Z-Z₀)/R₀
0.0
0.8


R/R₀
1.0
0.5
0.0
50°
(Z-Z₀)/R₀
0.8


p/p₀
1.0
0.0
-1.0
-2.0
50°
(Z-Z₀)/R₀
0.0
0.8


R/R₀
1.0
0.5
0.0
55°
(Z-Z₀)/R₀
0.8


p/p₀
1.0
-2.5
-7.5
55°
(Z-Z₀)/R₀
0.0
0.8

3. Le problème de type MDS (Multidimensional Scaling Problem)

En statistique, cet important problème peut être formulé en termes matriciels :
$\Delta = (\delta_{ij})$ et $W = (w_{ij})$ étant deux matrices symétriques d'ordre n telles que : $\delta_{ij} > 0$, $w_{ij} > 0$, $\forall\ i \neq j$ et $\delta_{ii} = w_{ii} = 0$.

On désigne par $\mathcal{M}_{n,p}(\mathbb{R})$ l'ensemble des matrices de type nxp à valeurs réelles. Soit ϕ une norme de $\mathbb{R}^p$, pour tout $X \in \mathcal{M}_{n,p}(\mathbb{R})$ on considère les semi-normes $d_{ij}(X)$ définies par : $d_{ij}(X) = \phi[(X_{i.})^T - (X_{j.})^T]$, ($X_{i.}$ étant la $i^{ème}$ ligne de X). On pose alors :

$$\nu(X) = \sum_{i<j} w_{ij} d_{ij}^2(X) \ , \ \rho(X) = \sum_{i<j} w_{ij}\ \delta_{ij}\ d_{ij}(X)$$

Il est clair que ρ et ν sont deux semi-normes sur $\mathcal{M}_{n,p}(\mathbb{R})$ dont leur noyau commun est : $\{ X \in \mathcal{M}_{n,p}(\mathbb{R}) : X_{i.} = x \ , \ \forall\ i = 1,2,...,n\}$.
Le problème MDS est alors : Max $\{\rho(x)/\nu(x) : \nu(x) \neq 0\}$

On démontre que ce problème est équivalent au suivant ([1,5]) :

$$(Q) : \begin{cases} \min \sigma(X) = \sum_{i<j} w_{ij}\ (\delta_{ij} - d_{ij}(X))^2 \\ X \in \mathcal{M}_{n,p}(\mathbb{R}) \end{cases}$$

C'est sur le dernier problème que nombreux auteurs se sont penchés (J. B. Kruskal [2], L. Guttman [3], J.C. Linguoes et E. E. Roskan[4]). Leurs algorithmes sont de type "méthode du gradient [1,5]) qui sont certes simples mais ne peuvent fournir que des minimums locaux.

L'intérêt du problème suivant, intimement lié au précédent, réside dans le fait que sa fonction objective est deux fois continûment différentiable et que les expressions analytiques du gradient $\nabla f(X)$ et du Hessien $\nabla^2 f(X)$ sont très simple :

$$(P) : \begin{cases} \min f(X) = \sum_{i<j} w_{ij}\ (\delta_{ij}^2 - d_{ij}^2(X))^2 \\ X \in \mathcal{M}_{n,p}(\mathbb{R}) \end{cases}$$

où ϕ est la norme euclidienne de $\mathbb{R}^p$.

Il s'agit bien évidement des problèmes d'optimisation non convexe dont une étude complète de leur compléxité et des méthodes de résolution est en cours.

Ici nous nous contenterons de présenter l'application de la méthode de région de confiance à la résolution de (P) dans deux cas. Les résultats obtenus sont très satisfaisants (Pour plus de détails concernant la signification, l'importance et la compléxité du problème MDS, cf. [8]).

On signale que dans l'application de cette méthode à ce problème, on prend $H(X) = \nabla^2 f(X)$ qui est toujours définie positive.

On cite ci-dessous deux exemples numériques (PC, Olivetti_380) :

Exemple 3.1.

Soit $W = (w_{ij})$ une matrice symétrique d'ordre n telle que :
$w_{ij} = 1$, $\forall$ $i \neq j$ et $w_{ii} = 0$.

La matrice $\Delta = (\delta_{ij})$ est donnée par :

$$\Delta = \begin{bmatrix} 0.00 & 0.01 & 0.02 & 0.06 & 0.07 & 0.03 & 0.04 & 0.08 & 0.09 & 0.05 \\ 0.01 & 0.00 & 0.05 & 0.09 & 0.08 & 0.04 & 0.03 & 0.07 & 0.06 & 0.02 \\ 0.02 & 0.05 & 0.00 & 0.06 & 0.07 & 0.03 & 0.04 & 0.08 & 0.09 & 0.05 \\ 0.06 & 0.09 & 0.06 & 0.00 & 0.01 & 0.01 & 0.01 & 0.02 & 0.02 & 0.02 \\ 0.07 & 0.08 & 0.07 & 0.01 & 0.00 & 0.03 & 0.03 & 0.03 & 0.04 & 0.04 \\ 0.03 & 0.04 & 0.03 & 0.01 & 0.03 & 0.00 & 0.04 & 0.04 & 0.04 & 0.05 \\ 0.04 & 0.03 & 0.04 & 0.01 & 0.03 & 0.04 & 0.00 & 0.01 & 0.02 & 0.03 \\ 0.08 & 0.07 & 0.08 & 0.02 & 0.03 & 0.04 & 0.01 & 0.00 & 0.01 & 0.02 \\ 0.09 & 0.06 & 0.09 & 0.02 & 0.04 & 0.04 & 0.02 & 0.01 & 0.00 & 0.01 \\ 0.05 & 0.02 & 0.05 & 0.02 & 0.04 & 0.05 & 0.03 & 0.02 & 0.01 & 0.00 \end{bmatrix}$$

n = 10 , p = 3 .

La solution optimale est donnée par ses composantes :
****

0.44360880	0.46874703	0.49082587
0.47393030	0.44908519	0.51775471
0.47838074	0.49868070	0.45388530
0.48962149	0.52794728	0.52780433
0.51592905	0.52994419	0.53549578
0.47147904	0.49731562	0.51564157
0.47549518	0.49849576	0.52477291
0.48803727	0.51693397	0.54695895
0.48945207	0.51447462	0.55327744
0.47406605	0.49837564	0.53358314

Le nombre des iterations est : 5

Le temps de calcul en secondes : 168.13

La valeur de la fonction en x* est : 0.00010149

La norme du gradient en x* est : 0.00623386

Exemple 3.2.

Soit $W = (w_{ij})$ une matrice symétrique d'ordre n telle que : $w_{ij} = 1$, $\forall$ $i \neq j$ et $w_{ii} = 0$.

La matrice $\Delta = (\delta_{ij})$ est donnée par :

$$\Delta = \begin{bmatrix} 0.0 & 0.2 & 0.4 & 0.6 & 0.8 & 1.0 \\ 0.2 & 0.0 & 0.1 & 0.3 & 0.5 & 0.7 \\ 0.4 & 0.1 & 0.0 & 0.4 & 0.7 & 1.0 \\ 0.6 & 0.3 & 0.4 & 0.0 & 0.1 & 0.2 \\ 0.8 & 0.5 & 0.7 & 0.1 & 0.0 & 0.1 \\ 1.0 & 0.7 & 1.0 & 0.2 & 0.1 & 0.0 \end{bmatrix}$$

n = 6 , p = 3.

La solution optimale est donnée par ses composantes :
****

0.12900118	0.19561469	0.31453840
0.37662253	0.37918455	0.38246386
0.40309311	0.33580131	0.11371065
0.54758120	0.56087224	0.56429421
0.63771058	0.66436419	0.70213965
0.68376918	0.75305191	0.92285324

Le nombre des iterations est : 24

Le temps de calcul en secondes : 102.23

La valeur de la fonction en x* est : 0.02801032

La norme du gradient en x* est : 0.00041860

4. Minimisation d'une forme quadratique hermitienne sur la sphère euclidienne unité de C^n.

Le problème (P_0) : Min $\{\phi(z) = 1/2(z-c)^H C(z-c) : z \in C^n, z^H z = 1\}$ (C étant une matrice Hermitienne) a été étudié par Forsythe & Golub sur une idée de Rao. Ces auteurs ont établi un certain nombre de résultats (sur la condition nécessaire du 1er ordre de ses solutions) qui sont basés essentiellement sur des propriétés matricielles adéquates. Ces résultats ne sont pas simples et surtout peu convenables à l'élaboration des algorithmes de résolution de (P_0). Forsythe & Golub ont souligné l'importance de ce problème dans les applications [6], sa non convexité et par conséquent la complexité de sa résolution.

Récemment Pham Dinh Tao est parvenu à formuler une caractérisation complète (i.e. sous forme de condition nécessaire et suffisante) très simple qui permet de construire un algorithme très performant pour la résolution de (P_0) [7].

En utilisant la décomposition canonique $C = A + iB$, où A et B sont deux matrices réelles de type nxn, on aura :

$z = x + iy$, $c = a + ib$

$$\phi(z)=1/2\{(x-a)^T[A(x-a) - B(y-b)] + (y-b)^T[A(y-b)+B(x-a)]\} +$$
$$i/2\{(x-a)^T[A(y-b) - B(x-a)] - (y-b)^T [A(x-a) + B(y-b)]\}$$

Il est facile de voir que C est hermitienne si et seulement si les deux conditions suivantes sont vérifiées :

1) La partie complexe de $\phi(z)$ est nulle.
2) $A = A^T$ et $B = -B^T$.

Soit donc la matrice $\mathfrak{C}$ de type 2nx2n définie par :

$$\mathfrak{C} = \begin{array}{|c|c|} \hline A & -B \\ \hline B & A \\ \hline \end{array}$$

Théorème 1 [7].

1) C est hermitienne si et seulement si $\mathfrak{C}$ est symétrique.
2) Les valeurs propres de $\mathfrak{C}$ sont les mêmes que C mais avec les multiplicités doublées.
3) C est semi-définie positive (resp. définie positive) si et seulement si $\mathfrak{C}$ est semi-définie positive (resp. définie positive).
4) C est semi-définie négative (resp. définie négative) si et seulement si $\mathfrak{C}$ est semi-définie négative (resp. définie négative). ◊

Le problème (P_0) est équivalent au problème $(\mathcal{P}_0)$ suivant :

$$(\mathcal{P}_0) : \text{Min } \psi(x,y) = 1/2[(x,y) - (a,b)]^T C[(x,y) - (a,b)]$$
$$(x,y)^T(x,y) = 1.$$

Pour simplifier la formule on considère le problème (P') de même type que $(\mathcal{P}_0)$:

$$(P') : \text{Min}\{f(x) = 1/2x^T \mathcal{C} x + b^T x : x \in \mathbb{R}^{2n},\ x^T x = 1\}$$

où $\mathcal{C}$ est une matrice symétrique de type 2nx2n de la forme :

$$\mathcal{C} = \begin{array}{|c|c|}\hline A & -B \\ \hline B & A \\ \hline \end{array}$$

Théorème 2 [7].

Soit ρ un nombre réel tel que $\mathcal{C}+\rho I$ soit définie négative, alors le problème (P') est équivalent au problème (P) :

$$(P) : \text{Min } \{g(x) = 1/2x^T(\mathcal{C}+\rho I)x + b^T x : x \in \mathbb{R}^{2n},\ x^T x \leq 1\}. \quad \Diamond$$

Preuve:

Puisque g(x) est strictement concave, la solution de (P) appartient à $\{x \in \mathbb{R}^n,\ x^T x = 1\}$. Cette équivalence de (P) et (P') est déduite du fait que $\rho x^T x$ est constamment égal à ρ dans $\{ x \in \mathbb{R}^n : x^T x = 1\}$.

Remarques importantes concernant la résolution de (P') :

Considérons le problème

$$(Q') : \beta = \text{Min } \{g(x) = 1/2x^T \mathcal{C} x + b^T x : x^T x \leq 1\}$$

que l'on sait résoudre (cf. Chapitre V).

Soit α la valeur optimale de (P'). On a les propriétés suivantes :

(i) $\alpha \leq \beta$.

(ii) Si x est une solution de (Q') tel que $x^T x = 1$ alors x est une solution de (P') et $\alpha = \beta$.

(iii) (P') admet une solution x telle que $||x|| < 1$ si et seulement si $\mathcal{C}$ est semi-définie positive et $||\mathcal{C}^+ b|| < 1$ ($\mathcal{C}^+$ désigne le pseudo-inverse de $\mathcal{C}$) (cf. Chapitre V). D'où le

Corollaire

Soit ρ un nombre réel tel que la matrice $\mathfrak{C}+\rho I$ ne soit semi-définie positive. Alors le problème (P') est équivalent au problème (P)

(P) : Min $\{g(x) = 1/2x^T(\mathfrak{C}+\rho I)x + b^Tx : x^Tx \leq 1\}$. $\Diamond$

La résolution de (P') se ramène donc à celle du programme quadratique local dans la méthode de région de confiance (cf. Chapitre V).

Remarques

1) Pour satisfaire le Théorème 2, on peut prendre $\rho > -||\mathfrak{C}||_1 \geq -\rho(\mathfrak{C})$ où $\rho(\mathfrak{C})$ est le rayon spectral de $\mathfrak{C}$ et $||\mathfrak{C}||_1$ désigne la norme d'opérateur de matrice relative à la norme de vecteurs : $||x||_1 = \Sigma |x_i|$:

$$||\mathfrak{C}||_1 = \text{Max}\{\sum_{i=1}^{n} |\mathfrak{C}_{ij}| : j = 1, ..., n\}$$

Il est intéressant de souligner qu'une valeur trop grande de ρ peut avoir des influences néfastes à la qualité de l'algorithme utilisé pour résoudre (P).

2) Pour satisfaire le Corollaire il suffit de prendre ρ tel que $(\rho+\lambda_1) < 0$ où λ_1 est la plus petite valeur propre de $\mathfrak{C}$.

Nous donnons ci-dessous deux exemples numériques de la résolution de (P) (Matériel utilisé : PC (Olivetti_380)) :

Exemple 4.1 :

$$A = \begin{pmatrix}
1 & 1 & 2 & 4 & 7 & -1 & -6 & 0.1 & -0.8 & 0 \\
1 & 1 & 3 & 5 & 8 & -2 & -7 & 0.2 & -0.7 & 0 \\
2 & 3 & 1 & 6 & 9 & -3 & -8 & 0.3 & -0.1 & 0 \\
4 & 5 & 6 & 1 & 10 & -4 & -9 & 0.4 & -0.5 & 0 \\
7 & 8 & 9 & 10 & 1 & -5 & -10 & 0.5 & -0.6 & 2 \\
-1 & -2 & -3 & -4 & -5 & 1 & -1 & 0.6 & -0.8 & 1 \\
-6 & -7 & -8 & -9 & -10 & -1 & 1 & 0.7 & -1 & 3 \\
0.1 & 0.2 & 0.3 & 0.4 & 0.5 & 0.6 & 0.7 & 1 & -2 & 0 \\
-0.8 & -0.7 & -0.1 & -0.5 & -0.6 & -0.8 & -1 & -2 & 1 & 4 \\
0.0 & 0.0 & 0.0 & 0.0 & 2 & 1 & 3 & 0.0 & 4 & 1
\end{pmatrix}$$

$$B = \begin{pmatrix}
0 & 1 & 4 & 2 & 3 & 2 & 0 & -1 & 0 & 1 \\
-1 & 0 & 7 & 5 & 6 & 6 & 0 & -2 & 0 & 2 \\
-4 & -7 & 0 & 8 & 9 & 8 & 0 & -3 & 0 & 3 \\
-2 & -5 & -8 & 0 & 10 & 4 & 0 & -4 & 0 & 4 \\
-3 & -6 & -9 & -10 & 0 & 5 & -2 & 0 & -4 & 5 \\
-2 & -6 & -8 & -4 & -5 & 0 & -6 & 0 & -5 & 6 \\
0 & 0 & 0 & 0 & 2 & 6 & 0 & 0 & -6 & 0 \\
1 & 2 & 3 & 4 & 0 & 0 & 0 & 0 & -7 & 0 \\
0 & 0 & 0 & 0 & 4 & 5 & 6 & 7 & 0 & 0 \\
-1 & -2 & -3 & -4 & -5 & -6 & 0 & 0 & 0 & 0
\end{pmatrix}$$

b = (1,-2,3,-4,5,-6,7,-8,9,10,-1,2,-3,4,-5,6,-7,8 ,-9,10).

Les termes diagonaux de A ne sont pas tous non négatifs, par suite la matrice $\mathfrak{C}$ n'est pas semi-définie positive.
D'après le Corollaire précédent on peut prendre $\rho = 0$.

La dimension n = 10.(2n = 20)

Le temps de calcul en secondes : 14.28

le vecteur solution z* est donné par ses composantes :

```
**********************************************************
  -0.0852930          0.0541222
  -0.0201917          0.0149128
  -0.1084653          0.2079834
   0.1331107         -0.0689694
   0.0630400          0.2718379
   0.4028995         -0.0498939
  -0.1065321          0.4276607
   0.2378432         -0.1494351
  -0.2502773          0.3602305
  -0.1956148         -0.4064094
```

norme(z*) = 1.0000000

La valeur de la fonction objective est : f(z*) = -31.1935497

Exemple 4.2 :

```
      1
      1  1
      2  3  1
      4  5  6  1
      7  8  9  10 1
     -1  -2  -3  -4  -5    1
     -6  -7  -8  -9  -10  -1    1
      0.1   0.2   0.3    0.4   0.5   0.6  0.7   1
     -0.8  -0.7  -0.1   -0.5  -0.6  -0.8  -1  -2   1
      0  0  0  0  2  1  3  0  4  1
A =  -1  2  3  4 -5  0  0  0  0  3  1
     -1  4  7 -2 -5  8  3 -6  9  0  0  1
     -4  0  0  0  0  1  2  0  0  2  3  1  1
     -2  2  2  1  1  0  0  2  0  0  0  0  0  1
     -3 -6 -9 -10 0  0  1  0 -2 -1  0  0  0  0   1
      8 -5 0  0  0  0  0  0  0  0  1  2  -1  -5  -3  1
      0  0  0  0  2  6 1  2  3  4  0  0  0  -2 -1 -3   1
      0  0  0  0  4  5  6  7  0  0  0  0  0  0  1 -2 -3  1
     -1 -2 -3 -4 -5 -6  0  0  0 -5 -1  0  0  0  0  0  0  0  1
      1  2  3  0  0  0  0  2  6  1  2  3  4  0  0  0 -2 -1 -3  1
```

```
        0
        1    0
        2    3    0
        4    5    6    0
        7    8    9   10   0
       -1   -2   -3   -4   -5    0
       -6   -7   -8   -9  -10   -1    0
        0.1   0.2   0.3   0.4   0.5   0.6   0.7   0
       -0.8  -0.7  -0.1  -0.5  -0.6  -0.8  -1    -2    0
        0    0    0    0    2    1    3    0    4    0
       -1    2    3    4   -5    0    0    0    0    3    0
B =    -1 -4  7 -2 -5  8 -3 -6  9  0  1   0
       -4  0  0  0  0  1  2  0  0  2 -3  1   0
       -2  2 -2  1  1  0  0  2  0  0  0  0  0   0
       -3 -6  9 -10 0  0  1  0 -2  1  0  0  0  0   0
        8 -5 0  0  0  0  0  0  0  0  1  2  -1   5  -3  0
        0  0  0  0  2 -6 1  2  3  4  0  0  0  -2  1 -3   0
        0  0  0  0  4  5 -6  7  0  0  0  0  0  0  1 -2 -3   0
       -1 -2 -3 -4 -5 -6  0  0  0 -5 -1  0  0  0  0  0  0  0   0
        1  2  3  0  0  0  0  2  6  1  2  3  4  0  0  0 -2 -1 -3  0
```

b = (1,-2,3,-4,5,-6,7,-8,9,-10,-11,12,-13,14,-15,16,-17,18,-19,20,
-1,2,-3,4,-5,6,-7,8,-9,10,11,-12,13,-14,15,-16,17,-18,19,-20).

Même remarque que pour l'exemple 4.1.

La dimension n = 20 (2n = 40)

Le temps de calcul en secondes : 41.09

le vecteur solution z* est donné par ses composantes :
****

0.0563988	0.0054869
0.0516088	-0.0634192
0.0069875	-0.0395105
0.1273584	-0.0609293
-0.0295843	0.1057468
0.1696762	-0.0631316
-0.1042216	0.0152741
0.1311432	-0.0978796
-0.0570686	0.1038999
0.1251101	-0.1484675
0.1678032	-0.0905313
-0.1288354	0.1531629
0.1772363	-0.1647695
-0.1717891	0.2089759
0.1989760	-0.2300123
-0.2327380	0.1759817
0.1658202	-0.1871674
-0.2377129	0.2618464
0.2336697	-0.2822978
-0.2590334	0.2435710

norme(z*) = 1.0000001

La valeur de la fonction objective est : f(z*) = -76.2414505

VI.2. Régression Isotone et concave [8,26].

Dans le chapitre III, nous avons décrit deux problèmes importants en Analyse des données :
problème de régression isotone et problème de régression concave.

De plus nous avons montré qu'il s'agit des programmes quadratiques. Actuellement les analystes des données disposent :

(i) La méthode de Kruskal (PAV) pour la résolution du problème de régression isotone. Cette méthode est reconnue comme étant la meilleure pour le traitement de ce problème.
Nos nombreuses expériences numériques l'ont confirmé.

(ii) La méthode de Dykstra pour la résolution du problème de régression concave.
Malheureusement nous avons constaté que la méthode de Dykstra n'est pas efficace : elle est très lente (par rapport aux autres méthodes que nous avons étudiées et adaptées à ce problème) et diverge pour les problèmes de taille ≥ 30.

Nous exposons donc dans ce paragraphe les expérimentations numériques comparatives concernant l'application de cinq méthodes (algorithme de Kruskal, algorithme de Dykstra et les algorithmes de pivotage de Lemke, du gradient conjugué conditionnel (cf. Chapitres II et III) et de l'inverse partiel).

Les cinq algorithmes (écrits en langage PASCAL) ont été implémentés sur compatible PC (Olivetti_M24) et BULL DPS 8/70 M. sous MULTICS.

Pour la régression isotone on présentera les solutions de tailles 100 calculées par la méthode de Kruskal, la méthode de pivotage de Lemke et la méthode du gradient conjugué en appliquant ces méthodes sur la fonction $f(x)=x^3$ (avec des bruits 10, 1, 0.5, 0.1 qui représentent le pourcentage d'erreurs).

Pour la régression concave on présentera les solutions de tailles 50 pour la fonction $f(x)=x^3$, calculées par la méthode de pivotage de Lemke et la méthode du gradient conjugué (pour différentes bruits).
La méthode de Dykstra et la méthode de l'inverse partiel divergent à partir de dimension 30.

On remarque que dans le cas de la régression isotone, les trois méthodes (Kruskal, Lemke et gradient conjugué) convergent toujours vers la même solution, et que la méthode de Kruskal dans ce cas est la plus rapide suivie par la méthode de pivotage de Lemke et la méthode du gradient conjugué conditionnel.

Dans le cas de la régression concave, on constate que les deux méthodes (Lemke et gradient conjugué) sont les meilleures méthodes appliquées à ce type de problèmes, elles convergent pour n'importe quelle dimension tandis que les autres méthodes (Dykstra et inverse partiel) donnent une solution approchée pour les petites tailles ($n \leq 30$) et divergent si la taille du problème est plus grande que 30.

A la fin on présente les résultats graphiques correspondants aux solutions de taille 100 et 50 pour les différentes bruits (on applique les méthodes sur la fonction $f(x)=x^3$ pour le cas isotone et sur les deux fonctions $f(x)=x^3$ et $g(x)=x^2$ pour le cas concave).

Pour terminer, il est intéressant de noter le rôle important de la dualité dans ces applications :
Pour résoudre le problème primal, les méthodes pivotage de Lemke et de l'inverse partiel travaillent directement dans le problème dual.
Quant à l'algorithme du gradient conjugué conditionnel, les experimentations numériques montrent que pour résoudre le problème primal, il est beaucoup plus rapide de l'appliquer au problème dual.

Régression Isotone

Méthode de Kruskal, Dimension 100.

Bruit 10

Poids W	Abscisses x	Val.Exactes y	Val.Brutées g	Sol.Optimale p
1	-2.478600	-15.227190	3.405070	-12.248973
1	-2.401220	-13.845090	-18.190830	-12.248973
1	-2.327790	-12.613390	-6.371890	-12.248973
1	-2.303150	-12.217000	-4.650910	-12.248973
1	-2.254060	-11.452360	-22.197000	-12.248973
1	-2.229650	-11.084330	-22.295680	-12.248973
1	-2.204990	-10.720590	-15.441570	-12.248973
1	-2.104590	-9.321840	-8.672230	-8.759250
1	-2.089260	-9.119660	-3.232550	-8.759250
1	-2.059630	-8.737070	-13.402560	-8.759250
1	-2.053590	-8.660490	-9.729660	-8.759250
1	-2.013150	-8.158860	-2.970530	-3.030533
1	-1.957600	-7.501870	0.200140	-3.030533
1	-1.952970	-7.448760	-3.845460	-3.030533
1	-1.819710	-6.025660	4.684260	-3.030533
1	-1.766430	-5.511740	5.657880	-3.030533
1	-1.764390	-5.492670	0.238440	-3.030533
1	-1.637410	-4.390090	-0.677060	-3.030533
1	-1.541860	-3.665510	4.718560	-3.030533
1	-1.530520	-3.585230	-23.006500	-3.030533
1	-1.457690	-3.097360	-4.112220	-3.030533
1	-1.433630	-2.946530	-2.546510	-3.030533
1	-1.429030	-2.918270	-14.707390	-3.030533
1	-1.410080	-2.803680	1.649270	-1.613812
1	-1.369210	-2.566880	9.847130	-1.613812
1	-1.306210	-2.228660	0.558510	-1.613812
1	-1.261370	-2.006910	-11.140820	-1.613812
1	-1.090230	-1.295840	12.817770	-1.613812
1	-1.086920	-1.284100	-13.729240	-1.613812
1	-1.033950	-1.105340	-1.205320	-1.613812
1	-0.979870	-0.940820	7.722790	-1.613812
1	-0.859820	-0.635670	8.103040	-1.613812
1	-0.853760	-0.622310	-3.992910	-1.613812
1	-0.739180	-0.403870	-12.209670	-1.613812
1	-0.725600	-0.382020	-9.989350	-1.613812
1	-0.725240	-0.381460	-9.410760	-1.613812
1	-0.715150	-0.365760	6.717560	-1.489722
1	-0.708280	-0.355310	3.882900	-1.489722
1	-0.641100	-0.263500	-8.355180	-1.489722
1	-0.619460	-0.237710	-0.078590	-1.489722
1	-0.524390	-0.144200	-0.278260	-1.489722
1	-0.333810	-0.037200	-9.558620	-1.489722
1	-0.283140	-0.022700	1.795610	-1.489722
1	-0.263340	-0.018260	3.222210	-1.489722
1	-0.249240	-0.015480	-5.514660	-1.489722
1	-0.100350	-0.001010	1.948060	-1.489722
1	0.057250	0.000190	-0.074010	-1.489722
1	0.060960	0.000230	1.749980	-1.489722

1	0.093150	0.000810	-14.823390	-1.489722
1	0.106210	0.001200	10.745960	-0.864470
1	0.339460	0.039120	3.181750	-0.864470
1	0.478150	0.109310	-1.711590	-0.864470
1	0.508150	0.131210	7.297420	-0.864470
1	0.549970	0.166350	-2.380920	-0.864470
1	0.591800	0.207270	-20.168070	-0.864470
1	0.657400	0.284110	-3.015840	-0.864470
1	0.659930	0.287400	10.500740	2.150043
1	0.733390	0.394460	-0.087600	2.150043
1	0.735650	0.398120	11.276670	2.150043
1	0.776220	0.467690	1.703890	2.150043
1	0.857520	0.630560	8.191340	2.150043
1	0.858410	0.632540	11.976190	2.150043
1	0.897390	0.722670	-1.104820	2.150043
1	0.918390	0.774610	8.189390	2.150043
1	0.926670	0.795750	-8.668170	2.150043
1	0.947190	0.849790	-5.318610	2.150043
1	0.963020	0.893120	-8.532850	2.150043
1	0.966390	0.902520	3.725990	2.150043
1	1.035790	1.111250	1.957320	2.150043
1	1.039700	1.123900	-3.708880	2.150043
1	1.041140	1.128570	6.093000	5.866725
1	1.144360	1.498600	5.981310	5.866725
1	1.233140	1.875170	6.209040	5.866725
1	1.291250	2.152940	6.677280	5.866725
1	1.359740	2.514040	11.133780	5.866725
1	1.381060	2.634150	3.278280	5.866725
1	1.391610	2.694940	14.958160	5.866725
1	1.410800	2.808000	15.380050	5.866725
1	1.497000	3.354810	12.841070	5.866725
1	1.535360	3.619370	7.247190	5.866725
1	1.572640	3.889420	-8.324030	5.866725
1	1.597610	4.077650	0.971130	5.866725
1	1.615800	4.218560	6.799780	5.866725
1	1.622760	4.273300	21.275420	5.866725
1	1.697870	4.894570	-0.228460	5.866725
1	1.728530	5.164530	-3.956070	5.866725
1	1.807970	5.909860	29.112930	5.866725
1	1.834050	6.169250	8.959810	5.866725
1	1.882900	6.675450	-11.336470	5.866725
1	1.917130	7.046210	2.838590	5.866725
1	1.992590	7.911360	-3.425220	5.866725
1	2.036970	8.451930	-1.793470	5.866725
1	2.134360	9.723090	21.843100	5.866725
1	2.193480	10.553590	2.516210	5.866725
1	2.236020	11.179580	3.076410	5.866725
1	2.259530	11.536040	1.704420	5.866725
1	2.267970	11.665680	12.737810	5.866725
1	2.443020	14.580730	-3.146080	5.866725
1	2.459680	14.881110	3.671110	5.866725
1	2.466600	15.007050	2.905670	5.866725

Dimension : 100
Valeur de la fonction f(x) : 3215.760
Nombre des iterations : 7
Temps du calcul en secondes : 3.3

Bruit 1

Poids W	Abscisses x	Val.Exactes y	Val.Brutées g	Sol.Optimale p
1	-2.478600	-15.227190	-13.363960	-13.821810
1	-2.401220	-13.845090	-14.279660	-13.821810
1	-2.327790	-12.613390	-11.989240	-12.045483
1	-2.303150	-12.217000	-11.460390	-12.045483
1	-2.254060	-11.452360	-12.526830	-12.045483
1	-2.229650	-11.084330	-12.205470	-12.045483
1	-2.204990	-10.720590	-11.192690	-11.192690
1	-2.104590	-9.321840	-9.256880	-9.256880
1	-2.089260	-9.119660	-8.530950	-8.867285
1	-2.059630	-8.737070	-9.203620	-8.867285
1	-2.053590	-8.660490	-8.767400	-8.767400
1	-2.013150	-8.158860	-7.640020	-7.640020
1	-1.957600	-7.501870	-6.731670	-6.910050
1	-1.952970	-7.448760	-7.088430	-6.910050
1	-1.819710	-6.025660	-4.954670	-4.954670
1	-1.766430	-5.511740	-4.394780	-4.657165
1	-1.764390	-5.492670	-4.919550	-4.657165
1	-1.637410	-4.390090	-4.018790	-4.124417
1	-1.541860	-3.665510	-2.827100	-4.124417
1	-1.530520	-3.585230	-5.527360	-4.124417
1	-1.457690	-3.097360	-3.198850	-3.400850
1	-1.433630	-2.946530	-2.906520	-3.400850
1	-1.429030	-2.918270	-4.097180	-3.400850
1	-1.410080	-2.803680	-2.358380	-2.358380
1	-1.369210	-2.566880	-1.325480	-2.065240
1	-1.306210	-2.228660	-1.949940	-2.065240
1	-1.261370	-2.006910	-2.920300	-2.065240
1	-1.090230	-1.295840	0.115520	-1.206545
1	-1.086920	-1.284100	-2.528610	-1.206545
1	-1.033950	-1.105340	-1.115340	-1.115340
1	-0.979870	-0.940820	-0.074460	-0.834535
1	-0.859820	-0.635670	0.238210	-0.834535
1	-0.853760	-0.622310	-0.959370	-0.834535
1	-0.739180	-0.403870	-1.584450	-0.834535
1	-0.725600	-0.382020	-1.342750	-0.834535
1	-0.725240	-0.381460	-1.284390	-0.834535
1	-0.715150	-0.365760	0.342580	-0.338387
1	-0.708280	-0.355310	0.068510	-0.338387
1	-0.641100	-0.263500	-1.072670	-0.338387
1	-0.619460	-0.237710	-0.221800	-0.338387
1	-0.524390	-0.144200	-0.157600	-0.338387
1	-0.333810	-0.037200	-0.989340	-0.338387
1	-0.283140	-0.022700	0.159130	-0.174317
1	-0.263340	-0.018260	0.305790	-0.174317
1	-0.249240	-0.015480	-0.565400	-0.174317
1	-0.100350	-0.001010	0.193900	-0.174317
1	0.057250	0.000190	-0.007230	-0.174317
1	0.060960	0.000230	0.175200	-0.174317
1	0.093150	0.000810	-1.481610	-0.174317
1	0.106210	0.001200	1.075670	0.034224
1	0.339460	0.039120	0.353380	0.034224
1	0.478150	0.109310	-0.072780	0.034224
1	0.508150	0.131210	0.847840	0.034224
1	0.549970	0.166350	-0.088380	0.034224

1	0.591800	0.207270	-1.830270	0.034224
1	0.657400	0.284110	-0.045890	0.034224
1	0.659930	0.287400	1.308730	0.815878
1	0.733390	0.394460	0.346250	0.815878
1	0.735650	0.398120	1.485980	0.815878
1	0.776220	0.467690	0.591310	0.815878
1	0.857520	0.630560	1.386640	0.815878
1	0.858410	0.632540	1.766900	0.815878
1	0.897390	0.722670	0.539920	0.815878
1	0.918390	0.774610	1.516090	0.815878
1	0.926670	0.795750	-0.150640	0.815878
1	0.947190	0.849790	0.232950	0.815878
1	0.963020	0.893120	-0.049470	0.815878
1	0.966390	0.902520	1.184870	1.007117
1	1.035790	1.111250	1.195860	1.007117
1	1.039700	1.123900	0.640620	1.007117
1	1.041140	1.128570	1.625010	1.625010
1	1.144360	1.498600	1.946870	1.946870
1	1.233140	1.875170	2.308560	2.308560
1	1.291250	2.152940	2.605380	2.605380
1	1.359740	2.514040	3.376020	3.037290
1	1.381060	2.634150	2.698560	3.037290
1	1.391610	2.694940	3.921270	3.784527
1	1.410800	2.808000	4.065210	3.784527
1	1.497000	3.354810	4.303440	3.784527
1	1.535360	3.619370	3.982160	3.784527
1	1.572640	3.889420	2.668080	3.784527
1	1.597610	4.077650	3.767000	3.784527
1	1.615800	4.218560	4.476680	4.476680
1	1.622760	4.273300	5.973510	4.869417
1	1.697870	4.894570	4.382270	4.869417
1	1.728530	5.164530	4.252470	4.869417
1	1.807970	5.909860	8.230160	6.517577
1	1.834050	6.169250	6.448310	6.517577
1	1.882900	6.675450	4.874260	6.517577
1	1.917130	7.046210	6.625450	6.625450
1	1.992590	7.911360	6.777700	6.777700
1	2.036970	8.451930	7.427390	7.427390
1	2.134360	9.723090	10.935090	10.342470
1	2.193480	10.553590	9.749850	10.342470
1	2.236020	11.179580	10.369260	10.369260
1	2.259530	11.536040	10.552880	10.552880
1	2.267970	11.665680	11.772890	11.772890
1	2.443020	14.580730	12.808050	12.808050
1	2.459680	14.881110	13.760110	13.760110
1	2.466600	15.007050	13.796910	13.796910

Dimension : 100
Valeur de la fonction f(x) : 18.983
Nombre des iterations : 5
Temps du calcul en secondes : 3.35

Bruit 05

Poids W	Abscisses x	Val.Exactes y	Val.Brutées g	Sol.Optimale p
1	-2.478600	-15.227190	-14.295580	-14.295580
1	-2.401220	-13.845090	-14.062380	-14.062380
1	-2.327790	-12.613390	-12.301310	-12.301310
1	-2.303150	-12.217000	-11.838690	-11.914145
1	-2.254060	-11.452360	-11.989600	-11.914145
1	-2.229650	-11.084330	-11.644900	-11.644900
1	-2.204990	-10.720590	-10.956640	-10.956640
1	-2.104590	-9.321840	-9.289360	-9.289360
1	-2.089260	-9.119660	-8.825300	-8.897825
1	-2.059630	-8.737070	-8.970350	-8.897825
1	-2.053590	-8.660490	-8.713950	-8.713950
1	-2.013150	-8.158860	-7.899440	-7.899440
1	-1.957600	-7.501870	-7.116770	-7.192685
1	-1.952970	-7.448760	-7.268600	-7.192685
1	-1.819710	-6.025660	-5.490170	-5.490170
1	-1.766430	-5.511740	-4.953260	-5.079685
1	-1.764390	-5.492670	-5.206110	-5.079685
1	-1.637410	-4.390090	-4.204440	-4.204440
1	-1.541860	-3.665510	-3.246310	-3.901300
1	-1.530520	-3.585230	-4.556290	-3.901300
1	-1.457690	-3.097360	-3.148100	-3.194113
1	-1.433630	-2.946530	-2.926520	-3.194113
1	-1.429030	-2.918270	-3.507720	-3.194113
1	-1.410080	-2.803680	-2.581030	-2.581030
1	-1.369210	-2.566880	-1.946180	-2.166360
1	-1.306210	-2.228660	-2.089300	-2.166360
1	-1.261370	-2.006910	-2.463600	-2.166360
1	-1.090230	-1.295840	-0.590160	-1.248260
1	-1.086920	-1.284100	-1.906360	-1.248260
1	-1.033950	-1.105340	-1.110340	-1.110340
1	-0.979870	-0.940820	-0.507640	-0.697782
1	-0.859820	-0.635670	-0.198730	-0.697782
1	-0.853760	-0.622310	-0.790840	-0.697782
1	-0.739180	-0.403870	-0.994160	-0.697782
1	-0.725600	-0.382020	-0.862390	-0.697782
1	-0.725240	-0.381460	-0.832930	-0.697782
1	-0.715150	-0.365760	-0.011590	-0.286165
1	-0.708280	-0.355310	-0.143400	-0.286165
1	-0.641100	-0.263500	-0.668080	-0.286165
1	-0.619460	-0.237710	-0.229750	-0.286165
1	-0.524390	-0.144200	-0.150900	-0.286165
1	-0.333810	-0.037200	-0.513270	-0.286165
1	-0.283140	-0.022700	0.068220	-0.091176
1	-0.263340	-0.018260	0.143760	-0.091176
1	-0.249240	-0.015480	-0.290440	-0.091176
1	-0.100350	-0.001010	0.096440	-0.091176
1	0.057250	0.000190	-0.003520	-0.091176
1	0.060960	0.000230	0.087710	-0.091176
1	0.093150	0.000810	-0.740400	-0.091176
1	0.106210	0.001200	0.538440	0.078328
1	0.339460	0.039120	0.196250	0.078328
1	0.478150	0.109310	0.018270	0.078328
1	0.508150	0.131210	0.489530	0.078328

1	0.549970	0.166350	0.038980	0.078328
1	0.591800	0.207270	-0.811500	0.078328
1	0.657400	0.284110	0.119110	0.119110
1	0.659930	0.287400	0.798070	0.584215
1	0.733390	0.394460	0.370360	0.584215
1	0.735650	0.398120	0.942050	0.735775
1	0.776220	0.467690	0.529500	0.735775
1	0.857520	0.630560	1.008600	0.752960
1	0.858410	0.632540	1.199720	0.752960
1	0.897390	0.722670	0.631300	0.752960
1	0.918390	0.774610	1.145350	0.752960
1	0.926670	0.795750	0.322560	0.752960
1	0.947190	0.849790	0.541370	0.752960
1	0.963020	0.893120	0.421820	0.752960
1	0.966390	0.902520	1.043700	1.026503
1	1.035790	1.111250	1.153550	1.026503
1	1.039700	1.123900	0.882260	1.026503
1	1.041140	1.128570	1.376790	1.376790
1	1.144360	1.498600	1.722740	1.722740
1	1.233140	1.875170	2.091870	2.091870
1	1.291250	2.152940	2.379160	2.379160
1	1.359740	2.514040	2.945030	2.805695
1	1.381060	2.634150	2.666360	2.805695
1	1.391610	2.694940	3.308100	3.308100
1	1.410800	2.808000	3.436610	3.436610
1	1.497000	3.354810	3.829130	3.636213
1	1.535360	3.619370	3.800760	3.636213
1	1.572640	3.889420	3.278750	3.636213
1	1.597610	4.077650	3.922330	3.922330
1	1.615800	4.218560	4.347620	4.347620
1	1.622760	4.273300	5.123400	4.823440
1	1.697870	4.894570	4.638420	4.823440
1	1.728530	5.164530	4.708500	4.823440
1	1.807970	5.909860	7.070010	6.384547
1	1.834050	6.169250	6.308780	6.384547
1	1.882900	6.675450	5.774850	6.384547
1	1.917130	7.046210	6.835830	6.835830
1	1.992590	7.911360	7.344530	7.344530
1	2.036970	8.451930	7.939660	7.939660
1	2.134360	9.723090	10.329090	10.240405
1	2.193480	10.553590	10.151720	10.240405
1	2.236020	11.179580	10.774420	10.774420
1	2.259530	11.536040	11.044460	11.044460
1	2.267970	11.665680	11.719290	11.719290
1	2.443020	14.580730	13.694390	13.694390
1	2.459680	14.881110	14.320610	14.320610
1	2.466600	15.007050	14.401980	14.401980

Dimension : 100
Valeur de la fonction f(x) : 3.437
Nombre des iterations : 4
Temps du calcul en secondes : 3.85

Bruit 01

Poids W	Abscisses x	Val.Exactes y	Val.Brutées g	Sol.Optimale p
1	-2.478600	-15.227190	-15.040870	-15.040870
1	-2.401220	-13.845090	-13.888550	-13.888550
1	-2.327790	-12.613390	-12.550970	-12.550970
1	-2.303150	-12.217000	-12.141340	-12.141340
1	-2.254060	-11.452360	-11.559810	-11.559810
1	-2.229650	-11.084330	-11.196440	-11.196440
1	-2.204990	-10.720590	-10.767800	-10.767800
1	-2.104590	-9.321840	-9.315340	-9.315340
1	-2.089260	-9.119660	-9.060790	-9.060790
1	-2.059630	-8.737070	-8.783730	-8.783730
1	-2.053590	-8.660490	-8.671180	-8.671180
1	-2.013150	-8.158860	-8.106970	-8.106970
1	-1.957600	-7.501870	-7.424850	-7.424850
1	-1.952970	-7.448760	-7.412730	-7.412730
1	-1.819710	-6.025660	-5.918560	-5.918560
1	-1.766430	-5.511740	-5.400040	-5.417695
1	-1.764390	-5.492670	-5.435350	-5.417695
1	-1.637410	-4.390090	-4.352960	-4.352960
1	-1.541860	-3.665510	-3.581670	-3.680555
1	-1.530520	-3.585230	-3.779440	-3.680555
1	-1.457690	-3.097360	-3.107510	-3.107510
1	-1.433630	-2.946530	-2.942520	-2.989340
1	-1.429030	-2.918270	-3.036160	-2.989340
1	-1.410080	-2.803680	-2.759150	-2.759150
1	-1.369210	-2.566880	-2.442740	-2.442740
1	-1.306210	-2.228660	-2.200790	-2.200790
1	-1.261370	-2.006910	-2.098250	-2.098250
1	-1.090230	-1.295840	-1.154710	-1.281630
1	-1.086920	-1.284100	-1.408550	-1.281630
1	-1.033950	-1.105340	-1.106340	-1.106340
1	-0.979870	-0.940820	-0.854180	-0.854180
1	-0.859820	-0.635670	-0.548280	-0.602145
1	-0.853760	-0.622310	-0.656010	-0.602145
1	-0.739180	-0.403870	-0.521930	-0.521930
1	-0.725600	-0.382020	-0.478090	-0.478090
1	-0.725240	-0.381460	-0.471760	-0.471760
1	-0.715150	-0.365760	-0.294920	-0.317423
1	-0.708280	-0.355310	-0.312930	-0.317423
1	-0.641100	-0.263500	-0.344420	-0.317423
1	-0.619460	-0.237710	-0.236120	-0.236120
1	-0.524390	-0.144200	-0.145540	-0.145540
1	-0.333810	-0.037200	-0.132410	-0.132410
1	-0.283140	-0.022700	-0.004520	-0.024663
1	-0.263340	-0.018260	0.014140	-0.024663
1	-0.249240	-0.015480	-0.070480	-0.024663
1	-0.100350	-0.001010	0.018480	-0.024663
1	0.057250	0.000190	-0.000550	-0.024663
1	0.060960	0.000230	0.017720	-0.024663
1	0.093150	0.000810	-0.147430	-0.024663
1	0.106210	0.001200	0.108650	0.089595
1	0.339460	0.039120	0.070540	0.089595
1	0.478150	0.109310	0.091110	0.091110
1	0.508150	0.131210	0.202880	0.115753

1	0.549970	0.166350	0.140870	0.115753
1	0.591800	0.207270	0.003510	0.115753
1	0.657400	0.284110	0.251110	0.251110
1	0.659930	0.287400	0.389530	0.389530
1	0.733390	0.394460	0.389640	0.389640
1	0.735650	0.398120	0.506910	0.493485
1	0.776220	0.467690	0.480060	0.493485
1	0.857520	0.630560	0.706170	0.706170
1	0.858410	0.632540	0.745970	0.725185
1	0.897390	0.722670	0.704400	0.725185
1	0.918390	0.774610	0.848760	0.774935
1	0.926670	0.795750	0.701110	0.774935
1	0.947190	0.849790	0.788100	0.788100
1	0.963020	0.893120	0.798860	0.798860
1	0.966390	0.902520	0.930760	0.930760
1	1.035790	1.111250	1.119710	1.097640
1	1.039700	1.123900	1.075570	1.097640
1	1.041140	1.128570	1.178210	1.178210
1	1.144360	1.498600	1.543430	1.543430
1	1.233140	1.875170	1.918510	1.918510
1	1.291250	2.152940	2.198190	2.198190
1	1.359740	2.514040	2.600240	2.600240
1	1.381060	2.634150	2.640590	2.640590
1	1.391610	2.694940	2.817580	2.817580
1	1.410800	2.808000	2.933720	2.933720
1	1.497000	3.354810	3.449680	3.449680
1	1.535360	3.619370	3.655650	3.655650
1	1.572640	3.889420	3.767290	3.767290
1	1.597610	4.077650	4.046590	4.046590
1	1.615800	4.218560	4.244370	4.244370
1	1.622760	4.273300	4.443320	4.443320
1	1.697870	4.894570	4.843340	4.843340
1	1.728530	5.164530	5.073330	5.073330
1	1.807970	5.909860	6.141890	6.141890
1	1.834050	6.169250	6.197160	6.197160
1	1.882900	6.675450	6.495330	6.495330
1	1.917130	7.046210	7.004140	7.004140
1	1.992590	7.911360	7.798000	7.798000
1	2.036970	8.451930	8.349480	8.349480
1	2.134360	9.723090	9.844290	9.844290
1	2.193480	10.553590	10.473220	10.473220
1	2.236020	11.179580	11.098550	11.098550
1	2.259530	11.536040	11.437730	11.437730
1	2.267970	11.665680	11.676400	11.676400
1	2.443020	14.580730	14.403460	14.403460
1	2.459680	14.881110	14.769010	14.769010
1	2.466600	15.007050	14.886030	14.886030

Dimension : 100
Valeur de la fonction f(x) : 0.061
Nombre des iterations : 4
Temps du calcul en secondes : 3.95

Dimension 100, méthode de Lemke

Bruit 10

Poids W	Abscisses x	Val.Exactes y	Val.Brutées g	Sol.Optimale p	Sol.Duale d
1	-2.478600	-15.227190	3.405070	-12.247296	15.652366
1	-2.401220	-13.845090	-18.190830	-12.249367	9.707403
1	-2.327790	-12.613390	-6.371890	-12.249386	15.582949
1	-2.303150	-12.217000	-4.650910	-12.249391	23.181140
1	-2.254060	-11.452360	-22.197000	-12.249397	13.233538
1	-2.229650	-11.084330	-22.295680	-12.249412	3.188919
1	-2.204990	-10.720590	-15.441570	-12.249651	0.000000
1	-2.104590	-9.321840	-8.672230	-8.769975	0.087745
1	-2.089260	-9.119660	-3.232550	-8.768737	5.613932
1	-2.059630	-8.737070	-13.402560	-8.767063	0.968435
1	-2.053590	-8.660490	-9.729660	-8.761225	0.000000
1	-2.013150	-8.158860	-2.970530	-3.050789	0.060259
1	-1.957600	-7.501870	0.200140	-3.049386	3.299785
1	-1.952970	-7.448760	-3.845460	-3.049386	2.493712
1	-1.819710	-6.025660	4.684260	-3.047983	10.225955
1	-1.766430	-5.511740	5.657880	-3.047337	18.931172
1	-1.764390	-5.492670	0.238440	-3.040062	22.209674
1	-1.637410	-4.390090	-0.677060	-3.032718	24.565331
1	-1.541860	-3.665510	4.718560	-3.032718	32.316609
1	-1.530520	-3.585230	-23.006500	-3.025373	12.335482
1	-1.457690	-3.097360	-4.112220	-3.017958	11.241220
1	-1.433630	-2.946530	-2.546510	-3.010330	11.705040
1	-1.429030	-2.918270	-14.707390	-3.002350	0.000000
1	-1.410080	-2.803680	1.649270	-1.657254	3.306524
1	-1.369210	-2.566880	9.847130	-1.649375	14.803028
1	-1.306210	-2.228660	0.558510	-1.641539	17.003077
1	-1.261370	-2.006910	-11.140820	-1.633653	7.495910
1	-1.090230	-1.295840	12.817770	-1.625693	21.939373
1	-1.086920	-1.284100	-13.729240	-1.617674	9.827807
1	-1.033950	-1.105340	-1.205320	-1.617674	10.240160
1	-0.979870	-0.940820	7.722790	-1.609655	19.572605
1	-0.859820	-0.635670	8.103040	-1.601577	29.277223
1	-0.853760	-0.622310	-3.992910	-1.593458	26.877770
1	-0.739180	-0.403870	-12.209670	-1.585351	16.253451
1	-0.725600	-0.382020	-9.989350	-1.577322	7.841423
1	-0.725240	-0.381460	-9.410760	-1.509337	0.000000
1	-0.715150	-0.365760	6.717560	-1.508468	8.221028
1	-0.708280	-0.355310	3.882900	-1.506713	13.600641
1	-0.641100	-0.263500	-8.355180	-1.503977	6.747438
1	-0.619460	-0.237710	-0.078590	-1.503645	8.170493
1	-0.524390	-0.144200	-0.278260	-1.502322	9.394555
1	-0.333810	-0.037200	-9.558620	-1.494729	1.330664
1	-0.283140	-0.022700	1.795610	-1.488706	4.614980
1	-0.263340	-0.018260	3.222210	-1.487743	9.324934
1	-0.249240	-0.015480	-5.514660	-1.481824	5.292097
1	-0.100350	-0.001010	1.948060	-1.479557	8.716714
1	0.057250	0.000190	-0.074010	-1.479520	10.117224
1	0.060960	0.000230	1.749980	-1.478330	13.345234
1	0.093150	0.000810	-14.823390	-1.478156	0.000000
1	0.106210	0.001200	10.745960	-0.869558	11.609518

1	0.339460	0.039120	3.181750	-0.868956	15.660224
1	0.478150	0.109310	-1.711590	-0.865612	14.811246
1	0.508150	0.131210	7.297420	-0.865092	22.971759
1	0.549970	0.166350	-2.380920	-0.864701	21.455140
1	0.591800	0.207270	-20.168070	-0.864481	2.151550
1	0.657400	0.284110	-3.015840	-0.864290	0.000000
1	0.659930	0.287400	10.500740	2.110918	8.339822
1	0.733390	0.394460	-0.087600	2.119738	6.092484
1	0.735650	0.398120	11.276670	2.119920	15.207235
1	0.776220	0.467690	1.703890	2.120729	14.754395
1	0.857520	0.630560	8.191340	2.121532	20.784203
1	0.858410	0.632540	11.976190	2.122446	30.601947
1	0.897390	0.722670	-1.104820	2.124130	27.342997
1	0.918390	0.774610	8.189390	2.126487	33.385899
1	0.926670	0.795750	-8.668170	2.125799	22.571930
1	0.947190	0.849790	-5.318610	2.132161	15.115159
1	0.963020	0.893120	-8.532850	2.135548	4.446761
1	0.966390	0.902520	3.725990	2.136230	6.036520
1	1.035790	1.111250	1.957320	2.142145	5.851695
1	1.039700	1.123900	-3.708880	2.142815	0.000000
1	1.041140	1.128570	6.093000	5.871428	0.101572
1	1.144360	1.498600	5.981310	5.873630	0.099253
1	1.233140	1.875170	6.209040	5.875513	0.332780
1	1.291250	2.152940	6.677280	5.876933	1.043127
1	1.359740	2.514040	11.133780	5.877938	6.218969
1	1.381060	2.634150	3.278280	5.878648	3.548601
1	1.391610	2.694940	14.958160	5.889186	12.567575
1	1.410800	2.808000	15.380050	5.890648	22.017978
1	1.497000	3.354810	12.841070	5.890648	28.929400
1	1.535360	3.619370	7.247190	5.890109	30.256481
1	1.572640	3.889420	-8.324030	5.890495	16.021955
1	1.597610	4.077650	0.971130	5.890900	11.092185
1	1.615800	4.218560	6.799780	5.891392	12.000573
1	1.622760	4.273300	21.275420	5.891979	27.394014
1	1.697870	4.894570	-0.228460	5.912625	21.292930
1	1.728530	5.164530	-3.956070	5.913301	11.473559
1	1.807970	5.909860	29.112930	5.914003	34.732486
1	1.834050	6.169250	8.959810	5.924740	37.847556
1	1.882900	6.675450	-11.336470	5.955527	20.675559
1	1.917130	7.046210	2.838590	5.956368	17.687781
1	1.992590	7.911360	-3.425220	5.957266	8.445294
1	2.036970	8.451930	-1.793470	5.958223	0.843601
1	2.134360	9.723090	21.843100	5.979246	16.887455
1	2.193480	10.553590	2.516210	5.990352	13.613313
1	2.236020	11.179580	3.076410	5.991561	10.908162
1	2.259530	11.536040	1.704420	5.992899	6.839683
1	2.267970	11.665680	12.737810	5.994389	13.813103
1	2.443020	14.580730	-3.146080	5.996047	4.910977
1	2.459680	14.881110	3.671110	5.997878	2.834209
1	2.466600	15.007050	2.905670	5.999879	

Dimension : 100
Valeur de la fonction f(x) : 3222.227
Nombre des iterations : 72
Temps du calcul en secondes : 211.46

Bruit 1

Poids W	Abscisses x	Val.Exactes y	Val.Brutées g	Sol.Optimale p	Sol.Duale d
1	-2.478600	-15.227190	-13.363960	-13.821810	0.457850
1	-2.401220	-13.845090	-14.279660	-13.821810	0.000000
1	-2.327790	-12.613390	-11.989240	-12.046637	0.052397
1	-2.303150	-12.217000	-11.460390	-12.045019	0.637026
1	-2.254060	-11.452360	-12.526830	-12.041960	0.162156
1	-2.229650	-11.084330	-12.205470	-12.041314	0.000000
1	-2.204990	-10.720590	-11.192690	-11.192690	0.000000
1	-2.104590	-9.321840	-9.256880	-9.256880	0.000000
1	-2.089260	-9.119660	-8.530950	-8.867285	0.336335
1	-2.059630	-8.737070	-9.203620	-8.867285	0.000000
1	-2.053590	-8.660490	-8.767400	-8.767400	0.000000
1	-2.013150	-8.158860	-7.640020	-7.640020	0.000000
1	-1.957600	-7.501870	-6.731670	-6.910050	0.178380
1	-1.952970	-7.448760	-7.088430	-6.910050	0.000000
1	-1.819710	-6.025660	-4.954670	-4.954670	0.000000
1	-1.766430	-5.511740	-4.394780	-4.657165	0.262385
1	-1.764390	-5.492670	-4.919550	-4.657165	0.000000
1	-1.637410	-4.390090	-4.018790	-4.128559	0.109769
1	-1.541860	-3.665510	-2.827100	-4.126185	1.408855
1	-1.530520	-3.585230	-5.527360	-4.118505	0.000000
1	-1.457690	-3.097360	-3.198850	-3.405164	0.206314
1	-1.433630	-2.946530	-2.906520	-3.399645	0.699440
1	-1.429030	-2.918270	-4.097180	-3.397740	0.000000
1	-1.410080	-2.803680	-2.358380	-2.358380	0.000000
1	-1.369210	-2.566880	-1.325480	-2.065240	0.739760
1	-1.306210	-2.228660	-1.949940	-2.065240	0.855060
1	-1.261370	-2.006910	-2.920300	-2.065240	0.000000
1	-1.090230	-1.295840	0.115520	-1.206545	1.322065
1	-1.086920	-1.284100	-2.528610	-1.206545	0.000000
1	-1.033950	-1.105340	-1.115340	-1.115340	0.000000
1	-0.979870	-0.940820	-0.074460	-0.843800	0.765340
1	-0.859820	-0.635670	0.238210	-0.841529	1.845080
1	-0.853760	-0.622310	-0.959370	-0.841529	1.727239
1	-0.739180	-0.403870	-1.584450	-0.832712	0.975502
1	-0.725600	-0.382020	-1.342750	-0.829281	0.456032
1	-0.725240	-0.381460	-1.284390	-0.828358	0.000000
1	-0.715150	-0.365760	0.342580	-0.350224	0.692804
1	-0.708280	-0.355310	0.068510	-0.343437	1.104751
1	-0.641100	-0.263500	-1.072670	-0.340932	0.371013
1	-0.619460	-0.237710	-0.221800	-0.340292	0.489505
1	-0.524390	-0.144200	-0.157600	-0.333418	0.665324
1	-0.333810	-0.037200	-0.989340	-0.324016	0.000000
1	-0.283140	-0.022700	0.159130	-0.185192	0.339322
1	-0.263340	-0.018260	0.305790	-0.184810	0.829922
1	-0.249240	-0.015480	-0.565400	-0.175175	0.439697
1	-0.100350	-0.001010	0.193900	-0.174125	0.805722
1	0.057250	0.000190	-0.007230	-0.173788	0.972281
1	0.060960	0.000230	0.175200	-0.169342	1.316823
1	0.093150	0.000810	-1.481610	-0.164787	0.000000
1	0.106210	0.001200	1.075670	0.031012	1.044658
1	0.339460	0.039120	0.353380	0.031730	1.370309
1	0.478150	0.109310	-0.072780	0.031781	1.261248
1	0.508150	0.131210	0.847840	0.031976	2.074212

1	0.549970	0.166350	-0.088380	0.043945	1.944087
1	0.591800	0.207270	-1.830270	0.034223	0.081894
1	0.657400	0.284110	-0.045890	0.036004	0.000000
1	0.659930	0.287400	1.308730	0.817490	0.481240
1	0.733390	0.394460	0.346250	0.817490	0.000000
1	0.735650	0.398120	1.485980	0.821002	0.654178
1	0.776220	0.467690	0.591310	0.821046	0.422842
1	0.857520	0.630560	1.386640	0.821216	0.986835
1	0.858410	0.632540	1.766900	0.821218	1.938518
1	0.897390	0.722670	0.539920	0.822100	1.671337
1	0.918390	0.774610	1.516090	0.822283	2.386145
1	0.926670	0.795750	-0.150640	0.822484	1.433020
1	0.947190	0.849790	0.232950	0.826948	0.859022
1	0.963020	0.893120	-0.049470	0.829552	0.000000
1	0.966390	0.902520	1.184870	1.005748	0.179122
1	1.035790	1.111250	1.195860	1.005748	0.369233
1	1.039700	1.123900	0.640620	1.009853	0.000000
1	1.041140	1.128570	1.625010	1.625010	0.000000
1	1.144360	1.498600	1.946870	1.946870	0.000000
1	1.233140	1.875170	2.308560	2.308560	0.000000
1	1.291250	2.152940	2.605380	2.605380	0.000000
1	1.359740	2.514040	3.376020	3.037290	0.338730
1	1.381060	2.634150	2.698560	3.037290	0.000000
1	1.391610	2.694940	3.921270	3.763503	0.127767
1	1.410800	2.808000	4.065210	3.770152	0.398826
1	1.497000	3.354810	4.303440	3.770586	0.914680
1	1.535360	3.619370	3.982160	3.770691	1.116149
1	1.572640	3.889420	2.668080	3.775615	0.008615
1	1.597610	4.077650	3.767000	3.775615	0.000000
1	1.615800	4.218560	4.476680	4.476680	0.000000
1	1.622760	4.273300	5.973510	4.869417	1.104093
1	1.697870	4.894570	4.382270	4.869417	0.616947
1	1.728530	5.164530	4.252470	4.869417	0.000000
1	1.807970	5.909860	8.230160	6.517577	1.712583
1	1.834050	6.169250	6.448310	6.517577	1.643317
1	1.882900	6.675450	4.874260	6.517577	0.000000
1	1.917130	7.046210	6.625450	6.625450	0.000000
1	1.992590	7.911360	6.777700	6.777700	0.000000
1	2.036970	8.451930	7.427390	7.427390	0.000000
1	2.134360	9.723090	10.935090	10.342470	0.592620
1	2.193480	10.553590	9.749850	10.342470	0.000000
1	2.236020	11.179580	10.369260	10.369260	0.000000
1	2.259530	11.536040	10.552880	10.552880	0.000000
1	2.267970	11.665680	11.772890	11.772890	0.000000
1	2.443020	14.580730	12.808050	12.808050	0.000000
1	2.459680	14.881110	13.760110	13.760110	0.000000
1	2.466600	15.007050	13.796910	13.796910	

Dimension : 100
Valeur de la fonction f(x) : 19.082
Nombre des iterations : 64
Temps du calcul en secondes : 43.12

Bruit 05

Poids W	Abscisses x	Val.Exactes y	Val.Brutées g	Sol.Optimale p	Sol.Duale d
1	-2.478600	-15.227190	-14.295580	-14.295580	0.000000
1	-2.401220	-13.845090	-14.062380	-14.062380	0.000000
1	-2.327790	-12.613390	-12.301310	-12.301310	0.000000
1	-2.303150	-12.217000	-11.838690	-11.914145	0.075422
1	-2.254060	-11.452360	-11.989600	-11.914145	0.000000
1	-2.229650	-11.084330	-11.644900	-11.644900	0.000000
1	-2.204990	-10.720590	-10.956640	-10.956640	0.000000
1	-2.104590	-9.321840	-9.289360	-9.289360	0.000000
1	-2.089260	-9.119660	-8.825300	-8.897825	0.076820
1	-2.059630	-8.737070	-8.970350	-8.897825	0.000000
1	-2.053590	-8.660490	-8.713950	-8.713950	0.000000
1	-2.013150	-8.158860	-7.899440	-7.899440	0.000000
1	-1.957600	-7.501870	-7.116770	-7.192685	0.075882
1	-1.952970	-7.448760	-7.268600	-7.192685	0.000000
1	-1.819710	-6.025660	-5.490170	-5.490170	0.000000
1	-1.766430	-5.511740	-4.953260	-5.079685	0.126370
1	-1.764390	-5.492670	-5.206110	-5.079685	0.000000
1	-1.637410	-4.390090	-4.204440	-4.204440	0.000000
1	-1.541860	-3.665510	-3.246310	-3.901300	0.654704
1	-1.530520	-3.585230	-4.556290	-3.901300	0.000000
1	-1.457690	-3.097360	-3.148100	-3.194113	0.046778
1	-1.433630	-2.946530	-2.926520	-3.194113	0.311939
1	-1.429030	-2.918270	-3.507720	-3.194113	0.000000
1	-1.410080	-2.803680	-2.581030	-2.581030	0.000000
1	-1.369210	-2.566880	-1.946180	-2.166360	0.220343
1	-1.306210	-2.228660	-2.089300	-2.166360	0.297353
1	-1.261370	-2.006910	-2.463600	-2.166360	0.000000
1	-1.090230	-1.295840	-0.590160	-1.248260	0.657813
1	-1.086920	-1.284100	-1.906360	-1.248260	0.000000
1	-1.033950	-1.105340	-1.110340	-1.110340	0.000000
1	-0.979870	-0.940820	-0.507640	-0.697782	0.196812
1	-0.859820	-0.635670	-0.198730	-0.697782	0.697235
1	-0.853760	-0.622310	-0.790840	-0.697782	0.607013
1	-0.739180	-0.403870	-0.994160	-0.697782	0.308335
1	-0.725600	-0.382020	-0.862390	-0.697782	0.140580
1	-0.725240	-0.381460	-0.832930	-0.697782	0.000000
1	-0.715150	-0.365760	-0.011590	-0.286165	0.272519
1	-0.708280	-0.355310	-0.143400	-0.286165	0.415510
1	-0.641100	-0.263500	-0.668080	-0.286165	0.036564
1	-0.619460	-0.237710	-0.229750	-0.286165	0.090529
1	-0.524390	-0.144200	-0.150900	-0.286165	0.224715
1	-0.333810	-0.037200	-0.513270	-0.286165	0.000000
1	-0.283140	-0.022700	0.068220	-0.091176	0.154195
1	-0.263340	-0.018260	0.143760	-0.091176	0.389442
1	-0.249240	-0.015480	-0.290440	-0.091176	0.186116
1	-0.100350	-0.001010	0.096440	-0.091176	0.372772
1	0.057250	0.000190	-0.003520	-0.091176	0.463228
1	0.060960	0.000230	0.087710	-0.091176	0.645592
1	0.093150	0.000810	-0.740400	-0.091176	0.000000
1	0.106210	0.001200	0.538440	0.078328	0.471519
1	0.339460	0.039120	0.196250	0.078328	0.594880
1	0.478150	0.109310	0.018270	0.078328	0.535366
1	0.508150	0.131210	0.489530	0.078328	0.946033
1	0.549970	0.166350	0.038980	0.078328	0.901285

1	0.591800	0.207270	-0.811500	0.078328	0.000000
1	0.657400	0.284110	0.119110	0.119110	0.000000
1	0.659930	0.287400	0.798070	0.584215	0.213762
1	0.733390	0.394460	0.370360	0.584215	0.000000
1	0.735650	0.398120	0.942050	0.735775	0.206185
1	0.776220	0.467690	0.529500	0.735775	0.000000
1	0.857520	0.630560	1.008600	0.752960	0.248770
1	0.858410	0.632540	1.199720	0.752960	0.687932
1	0.897390	0.722670	0.631300	0.752960	0.568509
1	0.918390	0.774610	1.145350	0.752960	0.956897
1	0.926670	0.795750	0.322560	0.752960	0.528762
1	0.947190	0.849790	0.541370	0.752960	0.324001
1	0.963020	0.893120	0.421820	0.752960	0.000000
1	0.966390	0.902520	1.043700	1.026503	0.016488
1	1.035790	1.111250	1.153550	1.026503	0.145288
1	1.039700	1.123900	0.882260	1.026503	0.000000
1	1.041140	1.128570	1.376790	1.376790	0.000000
1	1.144360	1.498600	1.722740	1.722740	0.000000
1	1.233140	1.875170	2.091870	2.091870	0.000000
1	1.291250	2.152940	2.379160	2.379160	0.000000
1	1.359740	2.514040	2.945030	2.805695	0.139274
1	1.381060	2.634150	2.666360	2.805695	0.000000
1	1.391610	2.694940	3.308100	3.308100	0.000000
1	1.410800	2.808000	3.436610	3.436610	0.000000
1	1.497000	3.354810	3.829130	3.636213	0.193116
1	1.535360	3.619370	3.800760	3.636213	0.357558
1	1.572640	3.889420	3.278750	3.636213	0.000000
1	1.597610	4.077650	3.922330	3.922330	0.000000
1	1.615800	4.218560	4.347620	4.347620	0.000000
1	1.622760	4.273300	5.123400	4.823440	0.305383
1	1.697870	4.894570	4.638420	4.823440	0.116934
1	1.728530	5.164530	4.708500	4.823440	0.000000
1	1.807970	5.909860	7.070010	6.384547	0.685784
1	1.834050	6.169250	6.308780	6.384547	0.612081
1	1.882900	6.675450	5.774850	6.386547	0.000000
1	1.917130	7.046210	6.835830	6.835830	0.000000
1	1.992590	7.911360	7.344530	7.344530	0.000000
1	2.036970	8.451930	7.939660	7.939660	0.000000
1	2.134360	9.723090	10.329090	10.240405	0.088646
1	2.193480	10.553590	10.151720	10.240405	0.000000
1	2.236020	11.179580	10.774420	10.774420	0.000000
1	2.259530	11.536040	11.044460	11.044460	0.000000
1	2.267970	11.665680	11.719290	11.719290	0.000000
1	2.443020	14.580730	13.694390	13.694390	0.000000
1	2.459680	14.881110	14.320610	14.320610	0.000000
1	2.466600	15.007050	14.401980	14.401980	

Dimension : 100
Valeur de la fonction f(x) : 3.437
Nombre des iterations : 56
Temps du calcul en secondes : 31.58

Bruit 01

Poids W	Abscisses x	Val.Exactes y	Val.Brutées g	Sol.Optimale p	Sol.Duale d
1	-2.478600	-15.227190	-15.040870	-15.040870	0.000000
1	-2.401220	-13.845090	-13.888550	-13.888550	0.000000
1	-2.327790	-12.613390	-12.550970	-12.550970	0.000000
1	-2.303150	-12.217000	-12.141340	-12.141340	0.000000
1	-2.254060	-11.452360	-11.559810	-11.559810	0.000000
1	-2.229650	-11.084330	-11.196440	-11.196440	0.000000
1	-2.204990	-10.720590	-10.767800	-10.767800	0.000000
1	-2.104590	-9.321840	-9.315340	-9.315340	0.000000
1	-2.089260	-9.119660	-9.060790	-9.060790	0.000000
1	-2.059630	-8.737070	-8.783730	-8.783730	0.000000
1	-2.053590	-8.660490	-8.671180	-8.671180	0.000000
1	-2.013150	-8.158860	-8.106970	-8.106970	0.000000
1	-1.957600	-7.501870	-7.424850	-7.424850	0.000000
1	-1.952970	-7.448760	-7.412730	-7.412730	0.000000
1	-1.819710	-6.025660	-5.918560	-5.918560	0.000000
1	-1.766430	-5.511740	-5.400040	-5.417695	0.019335
1	-1.764390	-5.492670	-5.435350	-5.417695	0.000000
1	-1.637410	-4.390090	-4.352960	-4.352960	0.000000
1	-1.541860	-3.665510	-3.581670	-3.680555	0.099929
1	-1.530520	-3.585230	-3.779440	-3.680555	0.000000
1	-1.457690	-3.097360	-3.107510	-3.107510	0.000000
1	-1.433630	-2.946530	-2.942520	-2.989340	0.051275
1	-1.429030	-2.918270	-3.036160	-2.989340	0.000000
1	-1.410080	-2.803680	-2.759150	-2.759150	0.000000
1	-1.369210	-2.566880	-2.442740	-2.442740	0.000000
1	-1.306210	-2.228660	-2.200790	-2.200790	0.000000
1	-1.261370	-2.006910	-2.098250	-2.098250	0.000000
1	-1.090230	-1.295840	-1.154710	-1.281630	0.128585
1	-1.086920	-1.284100	-1.408550	-1.281630	0.000000
1	-1.033950	-1.105340	-1.106340	-1.106340	0.000000
1	-0.979870	-0.940820	-0.854180	-0.854180	0.000000
1	-0.859820	-0.635670	-0.548280	-0.602145	0.054104
1	-0.853760	-0.622310	-0.656010	-0.602145	0.000000
1	-0.739180	-0.403870	-0.521930	-0.521930	0.000000
1	-0.725600	-0.382020	-0.478090	-0.478090	0.000000
1	-0.725240	-0.381460	-0.471760	-0.471760	0.000000
1	-0.715150	-0.365760	-0.294920	-0.317423	0.024168
1	-0.708280	-0.355310	-0.312930	-0.317423	0.026444
1	-0.641100	-0.263500	-0.344420	-0.317423	0.000000
1	-0.619460	-0.237710	-0.236120	-0.236120	0.000000
1	-0.524390	-0.144200	-0.145540	-0.145540	0.000000
1	-0.333810	-0.037200	-0.132410	-0.132410	0.000000
1	-0.283140	-0.022700	-0.004520	-0.024663	0.017098
1	-0.263340	-0.018260	0.014140	-0.024663	0.052596
1	-0.249240	-0.015480	-0.070480	-0.024663	0.005149
1	-0.100350	-0.001010	0.018480	-0.024663	0.046661
1	0.057250	0.000190	-0.000550	-0.024663	0.072381
1	0.060960	0.000230	0.017720	-0.024633	0.119755
1	0.093150	0.000810	-0.147430	-0.024633	0.000000
1	0.106210	0.001200	0.108650	0.089595	0.020868
1	0.339460	0.039120	0.070540	0.089595	0.000000
1	0.478150	0.109310	0.091110	0.091110	0.000000
1	0.508150	0.131210	0.202880	0.115753	0.091362

1	0.549970	0.166350	0.140870	0.115753	0.116182
1	0.591800	0.207270	0.003510	0.115753	0.000000
1	0.657400	0.284110	0.251110	0.251110	0.000000
1	0.659930	0.287400	0.389530	0.389530	0.000000
1	0.733390	0.394460	0.389640	0.389640	0.000000
1	0.735650	0.398120	0.506910	0.493485	0.014702
1	0.776220	0.467690	0.480060	0.493485	0.000000
1	0.857520	0.630560	0.706170	0.706170	0.000000
1	0.858410	0.632540	0.745970	0.725185	0.022763
1	0.897390	0.722670	0.704400	0.725185	0.000000
1	0.918390	0.774610	0.848760	0.774935	0.074605
1	0.926670	0.795750	0.701110	0.775935	0.000000
1	0.947190	0.849790	0.788100	0.788100	0.000000
1	0.963020	0.893120	0.798860	0.798860	0.000000
1	0.966390	0.902520	0.930760	0.930760	0.000000
1	1.035790	1.111250	1.119710	1.095540	0.024170
1	1.039700	1.123900	1.075570	1.099740	0.000000
1	1.041140	1.128570	1.178210	1.178210	0.000000
1	1.144360	1.498600	1.543430	1.543430	0.000000
1	1.233140	1.875170	1.918510	1.918510	0.000000
1	1.291250	2.152940	2.198190	2.198190	0.000000
1	1.359740	2.514040	2.600240	2.600240	0.000000
1	1.381060	2.634150	2.640590	2.640590	0.000000
1	1.391610	2.694940	2.817580	2.817580	0.000000
1	1.410800	2.808000	2.933720	2.933720	0.000000
1	1.497000	3.354810	3.449680	3.449680	0.000000
1	1.535360	3.619370	3.655650	3.655650	0.000000
1	1.572640	3.889420	3.767290	3.767290	0.000000
1	1.597610	4.077650	4.046590	4.046590	0.000000
1	1.615800	4.218560	4.244370	4.244370	0.000000
1	1.622760	4.273300	4.443320	4.443320	0.000000
1	1.697870	4.894570	4.843340	4.843340	0.000000
1	1.728530	5.164530	5.073330	5.073330	0.000000
1	1.807970	5.909860	6.141890	6.141890	0.000000
1	1.834050	6.169250	6.197160	6.197160	0.000000
1	1.882900	6.675450	6.495330	6.495330	0.000000
1	1.917130	7.046210	7.004140	7.004140	0.000000
1	1.992590	7.911360	7.798000	7.798000	0.000000
1	2.036970	8.451930	8.349480	8.349480	0.000000
1	2.134360	9.723090	9.844290	9.844290	0.000000
1	2.193480	10.553590	10.473220	10.473220	0.000000
1	2.236020	11.179580	11.098550	11.098550	0.000000
1	2.259530	11.536040	11.437730	11.437730	0.000000
1	2.267970	11.665680	11.676400	11.676400	0.000000
1	2.443020	14.580730	14.403460	14.403460	0.000000
1	2.459680	14.881110	14.769010	14.769010	0.000000
1	2.466600	15.007050	14.886030	14.886030	

Dimension : 100
Valeur de la fonction f(x) : 0.061
Nombre des iterations : 54
Temps du calcul en secondes : 21.28

Dimension 100, méthode du gradient conjugué.

Bruit 10

Poids W	Abscisses x	Val.Exactes y	Val.Brutées g	Sol.Optimale p	Sol.Duale d
1	-2.478600	-15.227190	3.405070	-12.247296	15.652366
1	-2.401220	-13.845090	-18.190830	-12.249367	9.707403
1	-2.327790	-12.613390	-6.371890	-12.249386	15.582949
1	-2.303150	-12.217000	-4.650910	-12.249391	23.181140
1	-2.254060	-11.452360	-22.197000	-12.249397	13.233538
1	-2.229650	-11.084330	-22.295680	-12.249412	3.188919
1	-2.204990	-10.720590	-15.441570	-12.249651	0.000000
1	-2.104590	-9.321840	-8.672230	-8.769975	0.087745
1	-2.089260	-9.119660	-3.232550	-8.768737	5.613932
1	-2.059630	-8.737070	-13.402560	-8.767063	0.968435
1	-2.053590	-8.660490	-9.729660	-8.761225	0.000000
1	-2.013150	-8.158860	-2.970530	-3.050789	0.060259
1	-1.957600	-7.501870	0.200140	-3.049386	3.299785
1	-1.952970	-7.448760	-3.845460	-3.049386	2.493712
1	-1.819710	-6.025660	4.684260	-3.047983	10.225955
1	-1.766430	-5.511740	5.657880	-3.047337	18.931172
1	-1.764390	-5.492670	0.238440	-3.040062	22.209674
1	-1.637410	-4.390090	-0.677060	-3.032718	24.565331
1	-1.541860	-3.665510	4.718560	-3.032718	32.316609
1	-1.530520	-3.585230	-23.006500	-3.025373	12.335482
1	-1.457690	-3.097360	-4.112220	-3.017958	11.241220
1	-1.433630	-2.946530	-2.546510	-3.010330	11.705040
1	-1.429030	-2.918270	-14.707390	-3.002350	0.000000
1	-1.410080	-2.803680	1.649270	-1.657254	3.306524
1	-1.369210	-2.566880	9.847130	-1.649375	14.803028
1	-1.306210	-2.228660	0.558510	-1.641539	17.003077
1	-1.261370	-2.006910	-11.140820	-1.633653	7.495910
1	-1.090230	-1.295840	12.817770	-1.625693	21.939373
1	-1.086920	-1.284100	-13.729240	-1.617674	9.827807
1	-1.033950	-1.105340	-1.205320	-1.617674	10.240160
1	-0.979870	-0.940820	7.722790	-1.609655	19.572605
1	-0.859820	-0.635670	8.103040	-1.601577	29.277223
1	-0.853760	-0.622310	-3.992910	-1.593458	26.877770
1	-0.739180	-0.403870	-12.209670	-1.585351	16.253451
1	-0.725600	-0.382020	-9.989350	-1.577322	7.841423
1	-0.725240	-0.381460	-9.410760	-1.509337	0.000000
1	-0.715150	-0.365760	6.717560	-1.508468	8.221028
1	-0.708280	-0.355310	3.882900	-1.506713	13.600641
1	-0.641100	-0.263500	-8.355180	-1.503977	6.747438
1	-0.619460	-0.237710	-0.078590	-1.503645	8.170493
1	-0.524390	-0.144200	-0.278260	-1.502322	9.394555
1	-0.333810	-0.037200	-9.558620	-1.494729	1.330664
1	-0.283140	-0.022700	1.795610	-1.488706	4.614980
1	-0.263340	-0.018260	3.222210	-1.487743	9.324934
1	-0.249240	-0.015480	-5.514660	-1.481824	5.292097
1	-0.100350	-0.001010	1.948060	-1.479557	8.716714
1	0.057250	0.000190	-0.074010	-1.479520	10.117224
1	0.060960	0.000230	1.749980	-1.478330	13.345234
1	0.093150	0.000810	-14.823390	-1.478156	0.000000
1	0.106210	0.001200	10.745960	-0.869558	11.609518

1	0.339460	0.039120	3.181750	-0.868956	15.660224
1	0.478150	0.109310	-1.711590	-0.865612	14.811246
1	0.508150	0.131210	7.297420	-0.865092	22.971759
1	0.549970	0.166350	-2.380920	-0.864701	21.455140
1	0.591800	0.207270	-20.168070	-0.864481	2.151550
1	0.657400	0.284110	-3.015840	-0.864290	0.000000
1	0.659930	0.287400	10.500740	2.110918	8.339822
1	0.733390	0.394460	-0.087600	2.119738	6.092484
1	0.735650	0.398120	11.276670	2.119920	15.207235
1	0.776220	0.467690	1.703890	2.120729	14.754395
1	0.857520	0.630560	8.191340	2.121532	20.784203
1	0.858410	0.632540	11.976190	2.122446	30.601947
1	0.897390	0.722670	-1.104820	2.124130	27.342997
1	0.918390	0.774610	8.189390	2.126487	33.385899
1	0.926670	0.795750	-8.668170	2.125799	22.571930
1	0.947190	0.849790	-5.318610	2.132161	15.115159
1	0.963020	0.893120	-8.532850	2.135548	4.446761
1	0.966390	0.902520	3.725990	2.136230	6.036520
1	1.035790	1.111250	1.957320	2.142145	5.851695
1	1.039700	1.123900	-3.708880	2.142815	0.000000
1	1.041140	1.128570	6.093000	5.871428	0.101572
1	1.144360	1.498600	5.981310	5.873630	0.099253
1	1.233140	1.875170	6.209040	5.875513	0.332780
1	1.291250	2.152940	6.677280	5.876933	1.043127
1	1.359740	2.514040	11.133780	5.877938	6.218969
1	1.381060	2.634150	3.278280	5.878648	3.548601
1	1.391610	2.694940	14.958160	5.889186	12.567575
1	1.410800	2.808000	15.380050	5.890648	22.017978
1	1.497000	3.354810	12.841070	5.890648	28.929400
1	1.535360	3.619370	7.247190	5.890109	30.256481
1	1.572640	3.889420	-8.324030	5.890495	16.021955
1	1.597610	4.077650	0.971130	5.890900	11.092185
1	1.615800	4.218560	6.799780	5.891392	12.000573
1	1.622760	4.273300	21.275420	5.891979	27.394014
1	1.697870	4.894570	-0.228460	5.912625	21.292930
1	1.728530	5.164530	-3.956070	5.913301	11.473559
1	1.807970	5.909860	29.112930	5.914003	34.732486
1	1.834050	6.169250	8.959810	5.924740	37.847556
1	1.882900	6.675450	-11.336470	5.955527	20.675559
1	1.917130	7.046210	2.838590	5.956368	17.687781
1	1.992590	7.911360	-3.425220	5.957266	8.445294
1	2.036970	8.451930	-1.793470	5.958223	0.843601
1	2.134360	9.723090	21.843100	5.979246	16.887455
1	2.193480	10.553590	2.516210	5.990352	13.613313
1	2.236020	11.179580	3.076410	5.991561	10.908162
1	2.259530	11.536040	1.704420	5.992899	6.839683
1	2.267970	11.665680	12.737810	5.994389	13.813103
1	2.443020	14.580730	-3.146080	5.996047	4.910977
1	2.459680	14.881110	3.671110	5.997878	2.834209
1	2.466600	15.007050	2.905670	5.999879	

Dimension : 100
Valeur de la fonction f(x) : 3222.227
Nombre des iterations : 122
Temps du calcul en secondes : 301.23

Bruit 1

Poids W	Abscisses x	Val.Exactes y	Val.Brutées g	Sol.Optimale p	Sol.Duale d
1	-2.478600	-15.227190	-13.363960	-13.821810	0.457850
1	-2.401220	-13.845090	-14.279660	-13.821810	0.000000
1	-2.327790	-12.613390	-11.989240	-12.046637	0.052397
1	-2.303150	-12.217000	-11.460390	-12.045019	0.637026
1	-2.254060	-11.452360	-12.526830	-12.041960	0.162156
1	-2.229650	-11.084330	-12.205470	-12.041314	0.000000
1	-2.204990	-10.720590	-11.192690	-11.192690	0.000000
1	-2.104590	-9.321840	-9.256880	-9.256880	0.000000
1	-2.089260	-9.119660	-8.530950	-8.867285	0.336335
1	-2.059630	-8.737070	-9.203620	-8.867285	0.000000
1	-2.053590	-8.660490	-8.767400	-8.767400	0.000000
1	-2.013150	-8.158860	-7.640020	-7.640020	0.000000
1	-1.957600	-7.501870	-6.731670	-6.910050	0.178380
1	-1.952970	-7.448760	-7.088430	-6.910050	0.000000
1	-1.819710	-6.025660	-4.954670	-4.954670	0.000000
1	-1.766430	-5.511740	-4.394780	-4.657165	0.262385
1	-1.764390	-5.492670	-4.919550	-4.657165	0.000000
1	-1.637410	-4.390090	-4.018790	-4.128559	0.109769
1	-1.541860	-3.665510	-2.827100	-4.126185	1.408855
1	-1.530520	-3.585230	-5.527360	-4.118505	0.000000
1	-1.457690	-3.097360	-3.198850	-3.405164	0.206314
1	-1.433630	-2.946530	-2.906520	-3.399645	0.699440
1	-1.429030	-2.918270	-4.097180	-3.397740	0.000000
1	-1.410080	-2.803680	-2.358380	-2.358380	0.000000
1	-1.369210	-2.566880	-1.325480	-2.065240	0.739760
1	-1.306210	-2.228660	-1.949940	-2.065240	0.855060
1	-1.261370	-2.006910	-2.920300	-2.065240	0.000000
1	-1.090230	-1.295840	0.115520	-1.206545	1.322065
1	-1.086920	-1.284100	-2.528610	-1.206545	0.000000
1	-1.033950	-1.105340	-1.115340	-1.115340	0.000000
1	-0.979870	-0.940820	-0.074460	-0.843800	0.765340
1	-0.859820	-0.635670	0.238210	-0.841529	1.845080
1	-0.853760	-0.622310	-0.959370	-0.841529	1.727239
1	-0.739180	-0.403870	-1.584450	-0.832712	0.975502
1	-0.725600	-0.382020	-1.342750	-0.829281	0.456032
1	-0.725240	-0.381460	-1.284390	-0.828358	0.000000
1	-0.715150	-0.365760	0.342580	-0.350224	0.692804
1	-0.708280	-0.355310	0.068510	-0.343437	1.104751
1	-0.641100	-0.263500	-1.072670	-0.340932	0.371013
1	-0.619460	-0.237710	-0.221800	-0.340292	0.489505
1	-0.524390	-0.144200	-0.157600	-0.333418	0.665324
1	-0.333810	-0.037200	-0.989340	-0.324016	0.000000
1	-0.283140	-0.022700	0.159130	-0.185192	0.339322
1	-0.263340	-0.018260	0.305790	-0.184810	0.829922
1	-0.249240	-0.015480	-0.565400	-0.175175	0.439697
1	-0.100350	-0.001010	0.193900	-0.174125	0.805722
1	0.057250	0.000190	-0.007230	-0.173788	0.972281
1	0.060960	0.000230	0.175200	-0.169342	1.316823
1	0.093150	0.000810	-1.481610	-0.164787	0.000000
1	0.106210	0.001200	1.075670	0.031012	1.044658
1	0.339460	0.039120	0.353380	0.031730	1.370309
1	0.478150	0.109310	-0.072780	0.031781	1.261248
1	0.508150	0.131210	0.847840	0.031976	2.074212

1	0.549970	0.166350	-0.088380	0.043945	1.944087
1	0.591800	0.207270	-1.830270	0.034223	0.081894
1	0.657400	0.284110	-0.045890	0.036004	0.000000
1	0.659930	0.287400	1.308730	0.817490	0.481240
1	0.733390	0.394460	0.346250	0.817490	0.000000
1	0.735650	0.398120	1.485980	0.821002	0.654178
1	0.776220	0.467690	0.591310	0.821046	0.422842
1	0.857520	0.630560	1.386640	0.821216	0.986835
1	0.858410	0.632540	1.766900	0.821218	1.938518
1	0.897390	0.722670	0.539920	0.822100	1.671337
1	0.918390	0.774610	1.516090	0.822283	2.386145
1	0.926670	0.795750	-0.150640	0.822484	1.433020
1	0.947190	0.849790	0.232950	0.826948	0.859022
1	0.963020	0.893120	-0.049470	0.829552	0.000000
1	0.966390	0.902520	1.184870	1.005748	0.179122
1	1.035790	1.111250	1.195860	1.005748	0.369233
1	1.039700	1.123900	0.640620	1.009853	0.000000
1	1.041140	1.128570	1.625010	1.625010	0.000000
1	1.144360	1.498600	1.946870	1.946870	0.000000
1	1.233140	1.875170	2.308560	2.308560	0.000000
1	1.291250	2.152940	2.605380	2.605380	0.000000
1	1.359740	2.514040	3.376020	3.037290	0.338730
1	1.381060	2.634150	2.698560	3.037290	0.000000
1	1.391610	2.694940	3.921270	3.763503	0.127767
1	1.410800	2.808000	4.065210	3.770152	0.398826
1	1.497000	3.354810	4.303440	3.770586	0.914680
1	1.535360	3.619370	3.982160	3.770691	1.116149
1	1.572640	3.889420	2.668080	3.775615	0.008615
1	1.597610	4.077650	3.767000	3.775615	0.000000
1	1.615800	4.218560	4.476680	4.476680	0.000000
1	1.622760	4.273300	5.973510	4.869417	1.104093
1	1.697870	4.894570	4.382270	4.869417	0.616947
1	1.728530	5.164530	4.252470	4.869417	0.000000
1	1.807970	5.909860	8.230160	6.517577	1.712583
1	1.834050	6.169250	6.448310	6.517577	1.643317
1	1.882900	6.675450	4.874260	6.517577	0.000000
1	1.917130	7.046210	6.625450	6.625450	0.000000
1	1.992590	7.911360	6.777700	6.777700	0.000000
1	2.036970	8.451930	7.427390	7.427390	0.000000
1	2.134360	9.723090	10.935090	10.342470	0.592620
1	2.193480	10.553590	9.749850	10.342470	0.000000
1	2.236020	11.179580	10.369260	10.369260	0.000000
1	2.259530	11.536040	10.552880	10.552880	0.000000
1	2.267970	11.665680	11.772890	11.772890	0.000000
1	2.443020	14.580730	12.808050	12.808050	0.000000
1	2.459680	14.881110	13.760110	13.760110	0.000000
1	2.466600	15.007050	13.796910	13.796910	

Dimension : 100
Valeur de la fonction f(x) : 19.082
Nombre des iterations : 9
Temps du calcul en secondes : 132.42

Bruit 05

Poids W	Abscisses x	Val.Exactes y	Val.Brutées g	Sol.Optimale p	Sol.Duale d
1	-2.478600	-15.227190	-14.295580	-14.295580	0.000000
1	-2.401220	-13.845090	-14.062380	-14.062380	0.000000
1	-2.327790	-12.613390	-12.301310	-12.301310	0.000000
1	-2.303150	-12.217000	-11.838690	-11.914112	0.075422
1	-2.254060	-11.452360	-11.989600	-11.914100	0.000000
1	-2.229650	-11.084330	-11.644900	-11.644900	0.000000
1	-2.204990	-10.720590	-10.956640	-10.956640	0.000000
1	-2.104590	-9.321840	-9.289360	-9.289360	0.000000
1	-2.089260	-9.119660	-8.825300	-8.902120	0.076820
1	-2.059630	-8.737070	-8.970350	-8.893530	0.000000
1	-2.053590	-8.660490	-8.713950	-8.713950	0.000000
1	-2.013150	-8.158860	-7.899440	-7.899440	0.000000
1	-1.957600	-7.501870	-7.116770	-7.192752	0.075882
1	-1.952970	-7.448760	-7.268600	-7.192718	0.000000
1	-1.819710	-6.025660	-5.490170	-5.490170	0.000000
1	-1.766430	-5.511740	-4.953260	-5.079830	0.126370
1	-1.764390	-5.492670	-5.206110	-5.079740	0.000000
1	-1.637410	-4.390090	-4.204440	-4.204440	0.000000
1	-1.541860	-3.665510	-3.246310	-3.901614	0.654704
1	-1.530520	-3.585230	-4.556290	-3.901586	0.000000
1	-1.457690	-3.097360	-3.148100	-3.196878	0.046778
1	-1.433630	-2.946530	-2.926520	-3.196681	0.311939
1	-1.429030	-2.918270	-3.507720	-3.195781	0.000000
1	-1.410080	-2.803680	-2.581030	-2.581030	0.000000
1	-1.369210	-2.566880	-1.946180	-2.166523	0.220343
1	-1.306210	-2.228660	-2.089300	-2.166311	0.297353
1	-1.261370	-2.006910	-2.463600	-2.166247	0.000000
1	-1.090230	-1.295840	-0.590160	-1.249973	0.657813
1	-1.086920	-1.284100	-1.906360	-1.248547	0.000000
1	-1.033950	-1.105340	-1.110340	-1.110340	0.000000
1	-0.979870	-0.940820	-0.507640	-0.704452	0.196812
1	-0.859820	-0.635670	-0.198730	-0.700753	0.697235
1	-0.853760	-0.622310	-0.790840	-0.700618	0.607013
1	-0.739180	-0.403870	-0.994160	-0.695482	0.308335
1	-0.725600	-0.382020	-0.862390	-0.694635	0.140580
1	-0.725240	-0.381460	-0.832930	-0.692350	0.000000
1	-0.715150	-0.365760	-0.011590	-0.294109	0.272519
1	-0.708280	-0.355310	-0.143400	-0.289391	0.415510
1	-0.641100	-0.263500	-0.668080	-0.289134	0.036564
1	-0.619460	-0.237710	-0.229750	-0.289115	0.090529
1	-0.524390	-0.144200	-0.150900	-0.289086	0.224715
1	-0.333810	-0.037200	-0.513270	-0.288555	0.000000
1	-0.283140	-0.022700	0.068220	-0.097975	0.154195
1	-0.263340	-0.018260	0.143760	-0.097187	0.389442
1	-0.249240	-0.015480	-0.290440	-0.097114	0.186116
1	-0.100350	-0.001010	0.096440	-0.094916	0.372772
1	0.057250	0.000190	-0.003520	-0.094875	0.463228
1	0.060960	0.000230	0.087710	-0.094854	0.645592
1	0.093150	0.000810	-0.740400	-0.094808	0.000000
1	0.106210	0.001200	0.538440	0.066921	0.471519
1	0.339460	0.039120	0.196250	0.072889	0.594880
1	0.478150	0.109310	0.018270	0.077784	0.535366
1	0.508150	0.131210	0.489530	0.078863	0.946033

1	0.549970	0.166350	0.038980	0.083729	0.901285
1	0.591800	0.207270	-0.811500	0.089785	0.000000
1	0.657400	0.284110	0.119110	0.119110	0.000000
1	0.659930	0.287400	0.798070	0.584308	0.213762
1	0.733390	0.394460	0.370360	0.584322	0.000000
1	0.735650	0.398120	0.942050	0.735865	0.206185
1	0.776220	0.467690	0.529500	0.735885	0.000000
1	0.857520	0.630560	1.008600	0.759830	0.248770
1	0.858410	0.632540	1.199720	0.760458	0.687932
1	0.897390	0.722670	0.631300	0.760623	0.568509
1	0.918390	0.774610	1.145350	0.766662	0.956897
1	0.926670	0.795750	0.322560	0.760695	0.528762
1	0.947190	0.849790	0.541370	0.760731	0.324001
1	0.963020	0.893120	0.421820	0.760721	0.000000
1	0.966390	0.902520	1.043700	1.027212	0.016488
1	1.035790	1.111250	1.153550	1.027750	0.145288
1	1.039700	1.123900	0.882260	1.027848	0.000000
1	1.041140	1.128570	1.376790	1.376790	0.000000
1	1.144360	1.498600	1.722740	1.722740	0.000000
1	1.233140	1.875170	2.091870	2.091870	0.000000
1	1.291250	2.152940	2.379160	2.379160	0.000000
1	1.359740	2.514040	2.945030	2.805756	0.139274
1	1.381060	2.634150	2.666360	2.805764	0.000000
1	1.391610	2.694940	3.308100	3.308100	0.000000
1	1.410800	2.808000	3.436610	3.436610	0.000000
1	1.497000	3.354810	3.829130	3.636014	0.193116
1	1.535360	3.619370	3.800760	3.636318	0.357558
1	1.572640	3.889420	3.278750	3.636319	0.000000
1	1.597610	4.077650	3.922330	3.922330	0.000000
1	1.615800	4.218560	4.347620	4.347620	0.000000
1	1.622760	4.273300	5.123400	4.818017	0.305383
1	1.697870	4.894570	4.638420	4.825469	0.116934
1	1.728530	5.164530	4.708500	4.826834	0.000000
1	1.807970	5.909860	7.070010	6.384226	0.685784
1	1.834050	6.169250	6.308780	6.384483	0.612081
1	1.882900	6.675450	5.774850	6.386931	0.000000
1	1.917130	7.046210	6.835830	6.835830	0.000000
1	1.992590	7.911360	7.344530	7.344530	0.000000
1	2.036970	8.451930	7.939660	7.939660	0.000000
1	2.134360	9.723090	10.329090	10.240444	0.088646
1	2.193480	10.553590	10.151720	10.240366	0.000000
1	2.236020	11.179580	10.774420	10.774420	0.000000
1	2.259530	11.536040	11.044460	11.044460	0.000000
1	2.267970	11.665680	11.719290	11.719290	0.000000
1	2.443020	14.580730	13.694390	13.694390	0.000000
1	2.459680	14.881110	14.320610	14.320610	0.000000
1	2.466600	15.007050	14.401980	14.401980	

Dimension : 100
Valeur de la fonction f(x) : 3.464
Nombre des iterations : 5
Temps du calcul en secondes : 67.58

Bruit 01

Poids W	Abscisses x	Val.Exactes y	Val.Brutées g	Sol.Optimale p	Sol.Duale d
1	-2.478600	-15.227190	-15.040870	-15.040870	0.000000
1	-2.401220	-13.845090	-13.888550	-13.888550	0.000000
1	-2.327790	-12.613390	-12.550970	-12.550970	0.000000
1	-2.303150	-12.217000	-12.141340	-12.141340	0.000000
1	-2.254060	-11.452360	-11.559810	-11.559810	0.000000
1	-2.229650	-11.084330	-11.196440	-11.196440	0.000000
1	-2.204990	-10.720590	-10.767800	-10.767800	0.000000
1	-2.104590	-9.321840	-9.315340	-9.315340	0.000000
1	-2.089260	-9.119660	-9.060790	-9.060790	0.000000
1	-2.059630	-8.737070	-8.783730	-8.783730	0.000000
1	-2.053590	-8.660490	-8.671180	-8.671180	0.000000
1	-2.013150	-8.158860	-8.106970	-8.106970	0.000000
1	-1.957600	-7.501870	-7.424850	-7.424850	0.000000
1	-1.952970	-7.448760	-7.412730	-7.412730	0.000000
1	-1.819710	-6.025660	-5.918560	-5.918560	0.000000
1	-1.766430	-5.511740	-5.400040	-5.419375	0.019335
1	-1.764390	-5.492670	-5.435350	-5.416015	0.000000
1	-1.637410	-4.390090	-4.352960	-4.352960	0.000000
1	-1.541860	-3.665510	-3.581670	-3.681599	0.099929
1	-1.530520	-3.585230	-3.779440	-3.679511	0.000000
1	-1.457690	-3.097360	-3.107510	-3.107510	0.000000
1	-1.433630	-2.946530	-2.942520	-2.993795	0.051275
1	-1.429030	-2.918270	-3.036160	-2.984885	0.000000
1	-1.410080	-2.803680	-2.759150	-2.759150	0.000000
1	-1.369210	-2.566880	-2.442740	-2.442740	0.000000
1	-1.306210	-2.228660	-2.200790	-2.200790	0.000000
1	-1.261370	-2.006910	-2.098250	-2.098250	0.000000
1	-1.090230	-1.295840	-1.154710	-1.283295	0.128585
1	-1.086920	-1.284100	-1.408550	-1.279965	0.000000
1	-1.033950	-1.105340	-1.106340	-1.106340	0.000000
1	-0.979870	-0.940820	-0.854180	-0.854180	0.000000
1	-0.859820	-0.635670	-0.548280	-0.602384	0.054104
1	-0.853760	-0.622310	-0.656010	-0.601906	0.000000
1	-0.739180	-0.403870	-0.521930	-0.521930	0.000000
1	-0.725600	-0.382020	-0.478090	-0.478090	0.000000
1	-0.725240	-0.381460	-0.471760	-0.471760	0.000000
1	-0.715150	-0.365760	-0.294920	-0.319088	0.024168
1	-0.708280	-0.355310	-0.312930	-0.317976	0.026444
1	-0.641100	-0.263500	-0.344420	-0.315206	0.000000
1	-0.619460	-0.237710	-0.236120	-0.236120	0.000000
1	-0.524390	-0.144200	-0.145540	-0.145540	0.000000
1	-0.333810	-0.037200	-0.132410	-0.132410	0.000000
1	-0.283140	-0.022700	-0.004520	-0.021618	0.017098
1	-0.263340	-0.018260	0.014140	-0.021358	0.052596
1	-0.249240	-0.015480	-0.070480	-0.023033	0.005149
1	-0.100350	-0.001010	0.018480	-0.023033	0.046661
1	0.057250	0.000190	-0.000550	-0.023033	0.072381
1	0.060960	0.000230	0.017720	-0.023033	0.119755
1	0.093150	0.000810	-0.147430	-0.023033	0.000000
1	0.106210	0.001200	0.108650	0.087782	0.020868
1	0.339460	0.039120	0.070540	0.091408	0.000000
1	0.478150	0.109310	0.091110	0.091110	0.000000
1	0.508150	0.131210	0.202880	0.111518	0.091362

1	0.549970	0.166350	0.140870	0.116050	0.116182
1	0.591800	0.207270	0.003510	0.119692	0.000000
1	0.657400	0.284110	0.251110	0.251110	0.000000
1	0.659930	0.287400	0.389530	0.389530	0.000000
1	0.733390	0.394460	0.389640	0.389640	0.000000
1	0.735650	0.398120	0.506910	0.492208	0.014702
1	0.776220	0.467690	0.480060	0.494762	0.000000
1	0.857520	0.630560	0.706170	0.706170	0.000000
1	0.858410	0.632540	0.745970	0.723207	0.022763
1	0.897390	0.722670	0.704400	0.727163	0.000000
1	0.918390	0.774610	0.848760	0.774155	0.074605
1	0.926670	0.795750	0.701110	0.775715	0.000000
1	0.947190	0.849790	0.788100	0.788100	0.000000
1	0.963020	0.893120	0.798860	0.798860	0.000000
1	0.966390	0.902520	0.930760	0.930760	0.000000
1	1.035790	1.111250	1.119710	1.095540	0.024170
1	1.039700	1.123900	1.075570	1.099740	0.000000
1	1.041140	1.128570	1.178210	1.178210	0.000000
1	1.144360	1.498600	1.543430	1.543430	0.000000
1	1.233140	1.875170	1.918510	1.918510	0.000000
1	1.291250	2.152940	2.198190	2.198190	0.000000
1	1.359740	2.514040	2.600240	2.600240	0.000000
1	1.381060	2.634150	2.640590	2.640590	0.000000
1	1.391610	2.694940	2.817580	2.817580	0.000000
1	1.410800	2.808000	2.933720	2.933720	0.000000
1	1.497000	3.354810	3.449680	3.449680	0.000000
1	1.535360	3.619370	3.655650	3.655650	0.000000
1	1.572640	3.889420	3.767290	3.767290	0.000000
1	1.597610	4.077650	4.046590	4.046590	0.000000
1	1.615800	4.218560	4.244370	4.244370	0.000000
1	1.622760	4.273300	4.443320	4.443320	0.000000
1	1.697870	4.894570	4.843340	4.843340	0.000000
1	1.728530	5.164530	5.073330	5.073330	0.000000
1	1.807970	5.909860	6.141890	6.141890	0.000000
1	1.834050	6.169250	6.197160	6.197160	0.000000
1	1.882900	6.675450	6.495330	6.495330	0.000000
1	1.917130	7.046210	7.004140	7.004140	0.000000
1	1.992590	7.911360	7.798000	7.798000	0.000000
1	2.036970	8.451930	8.349480	8.349480	0.000000
1	2.134360	9.723090	9.844290	9.844290	0.000000
1	2.193480	10.553590	10.473220	10.473220	0.000000
1	2.236020	11.179580	11.098550	11.098550	0.000000
1	2.259530	11.536040	11.437730	11.437730	0.000000
1	2.267970	11.665680	11.676400	11.676400	0.000000
1	2.443020	14.580730	14.403460	14.403460	0.000000
1	2.459680	14.881110	14.769010	14.769010	0.000000
1	2.466600	15.007050	14.886030	14.886030	

Dimension : 100
Valeur de la fonction f(x) : 0.063
Nombre des iterations : 4
Temps du calcul en secondes : 42.28

Regression concave

Méthode de Lemke, Dimension 50.

Bruit 10

Poids W	Abscisses x	Val.Exactes y	Val.Brutées g	Sol.Optimale p	Sol.Duale d
1	-2.478600	-15.227190	3.405070	-13.230897	8.317984
1	-2.401220	-13.845090	-18.190830	-12.509449	13.795277
1	-2.327790	-12.613390	-6.371890	-11.788001	21.980626
1	-2.303150	-12.217000	-4.650910	-11.066553	33.373796
1	-2.254060	-11.452360	-22.197000	-10.345105	38.841019
1	-2.229650	-11.084330	-22.295680	-9.623656	37.972230
1	-2.204990	-10.720590	-15.441570	-8.902208	33.833760
1	-2.104590	-9.321840	-8.672230	-8.180760	29.449555
1	-2.089260	-9.119660	-3.232550	-7.459312	27.178731
1	-2.059630	-8.737070	-13.402560	-6.737864	21.575559
1	-2.053590	-8.660490	-9.729660	-6.016415	14.115764
1	-2.013150	-8.158860	-2.970530	-5.294967	7.818188
1	-1.957600	-7.501870	0.200140	-4.573519	3.907441
1	-1.952970	-7.448760	-3.845460	-3.852071	0.000000
1	-1.819710	-6.025660	4.684260	-3.130622	0.000000
1	-1.766430	-5.511740	5.657880	-2.486819	4.072349
1	-1.764390	-5.492670	0.238440	-2.448157	9.487997
1	-1.637410	-4.390090	-0.677060	-2.409495	15.769862
1	-1.541860	-3.665510	4.718560	-2.370833	25.596423
1	-1.530520	-3.585230	-23.006500	-2.332171	25.085820
1	-1.457690	-3.097360	-4.112220	-2.293509	23.665861
1	-1.433630	-2.946530	-2.546510	-2.254847	22.100071
1	-1.429030	-2.918270	-14.707390	-2.216185	14.288678
1	-1.410080	-2.803680	1.649270	-2.177523	8.390682
1	-1.369210	-2.566880	9.847130	-2.138861	8.485681
1	-1.306210	-2.228660	0.558510	-2.100199	9.910035
1	-1.261370	-2.006910	-11.140820	-2.061537	6.794747
1	-1.090230	-1.295840	12.817770	-2.022875	11.099782
1	-1.086920	-1.284100	-13.729240	-1.984213	9.532303
1	-1.033950	-1.105340	-1.205320	-1.945551	8.334940
1	-0.979870	-0.940820	7.722790	-1.906889	11.952417
1	-0.859820	-0.635670	8.103040	-1.868227	20.555528
1	-0.853760	-0.622310	-3.992910	-1.829565	28.076966
1	-0.739180	-0.403870	-12.209670	-1.790903	30.389021
1	-0.725600	-0.382020	-9.989350	-1.752242	28.582522
1	-0.725240	-0.381460	-9.410760	-1.713580	22.927432
1	-0.715150	-0.365760	6.717560	-1.674918	21.468581
1	-0.708280	-0.355310	3.882900	-1.636256	22.769309
1	-0.641100	-0.263500	-8.355180	-1.597594	20.691243
1	-0.619460	-0.237710	-0.078590	-1.558932	19.353348
1	-0.524390	-0.144200	-0.278260	-1.520270	18.636457
1	-0.333810	-0.037200	-9.558620	-1.481608	13.881061
1	-0.283140	-0.022700	1.795610	-1.442946	10.744943
1	-0.263340	-0.018260	3.222210	-1.404284	9.922071
1	-0.249240	-0.015480	-5.514660	-1.365622	7.024681
1	-0.100350	-0.001010	1.948060	-1.326960	5.764801
1	0.057250	0.000190	-0.074010	-1.288298	5.112064
1	0.060960	0.000230	1.749980	-1.249636	5.959136
1	0.093150	0.000810	-14.823390	-1.210974	

1	0.106210	0.001200	10.745960	-1.172312

Dimension : 50
Valeur de la fonction f(x) : 1627.397
Nombre des iterations : 49
Temps du calcul en secondes : 58.88

Bruit 1

Poids W	Abscisses x	Val.Exactes y	Val.Brutées g	Sol.Optimale p	Sol.Duale d
1	-2.478600	-15.227190	-13.363960	-14.114515	0.375277
1	-2.401220	-13.845090	-14.279660	-13.513298	0.367374
1	-2.327790	-12.613390	-11.989240	-12.912082	0.820891
1	-2.303150	-12.217000	-11.460390	-12.310865	1.699645
1	-2.254060	-11.452360	-12.526830	-11.709649	2.169809
1	-2.229650	-11.084330	-12.205470	-11.108432	2.091454
1	-2.204990	-10.720590	-11.192690	-10.507216	1.670362
1	-2.104590	-9.321840	-9.256880	-9.905999	1.573830
1	-2.089260	-9.119660	-8.530950	-9.304783	1.864213
1	-2.059630	-8.737070	-9.203620	-8.703566	1.904570
1	-2.053590	-8.660490	-8.767400	-8.102350	1.612401
1	-2.013150	-8.158860	-7.640020	-7.501133	1.250789
1	-1.957600	-7.501870	-6.731670	-6.899917	0.973301
1	-1.952970	-7.448760	-7.088430	-6.298700	0.300947
1	-1.819710	-6.025660	-4.954670	-5.697484	0.000000
1	-1.766430	-5.511740	-4.394780	-5.096267	0.049797
1	-1.764390	-5.492670	-4.919550	-4.720363	0.000000
1	-1.637410	-4.390090	-4.018790	-4.344460	0.113038
1	-1.541860	-3.665510	-2.827100	-4.056083	0.840568
1	-1.530520	-3.585230	-5.527360	-3.767707	0.688271
1	-1.457690	-3.097360	-3.198850	-3.479330	0.676214
1	-1.433630	-2.946530	-2.906520	-3.190954	0.806374
1	-1.429030	-2.918270	-4.097180	-2.902577	0.339232
1	-1.410080	-2.803680	-2.358380	-2.614200	0.000000
1	-1.369210	-2.566880	-1.325480	-2.325824	0.160940
1	-1.306210	-2.228660	-1.949940	-2.045091	0.369456
1	-1.261370	-2.006910	-2.920300	-1.764357	0.000000
1	-1.090230	-1.295840	0.115520	-1.483624	0.430116
1	-1.086920	-1.284100	-2.528610	-1.268872	0.230363
1	-1.033950	-1.105340	-1.115340	-1.054119	0.000000
1	-0.979870	-0.940820	-0.074460	-0.839367	0.152090
1	-0.859820	-0.635670	0.238210	-0.790487	0.818528
1	-0.853760	-0.622310	-0.959370	-0.741607	1.376085
1	-0.739180	-0.403870	-1.584450	-0.692727	1.487780
1	-0.725600	-0.382020	-1.342750	-0.643848	1.250024
1	-0.725240	-0.381460	-1.284390	-0.594968	0.667557
1	-0.715150	-0.365760	0.342580	-0.546088	0.529425
1	-0.708280	-0.355310	0.068510	-0.497208	0.674151
1	-0.641100	-0.263500	-1.072670	-0.448329	0.506707
1	-0.619460	-0.237710	-0.221800	-0.399449	0.428087
1	-0.524390	-0.144200	-0.157600	-0.350569	0.445951
1	-0.333810	-0.037200	-0.989340	-0.301689	0.119991
1	-0.283140	-0.022700	0.159130	-0.252810	0.000000
1	-0.263340	-0.018260	0.305790	-0.203930	0.134869
1	-0.249240	-0.015480	-0.565400	-0.161842	0.067959
1	-0.100350	-0.001010	0.193900	-0.119754	0.157877
1	0.057250	0.000190	-0.007230	-0.077666	0.283012

1	0.060960	0.000230	0.175200	-0.035578	0.513536
1	0.093150	0.000810	-1.481610	0.006510	
1	0.106210	0.001200	1.075670	0.048598	

Dimension : 50
Valeur de la fonction f(x) : 15.446
Nombre des iterations : 45
Temps du calcul en secondes : 54.21

Bruit 05

Poids W	Abscisses x	Val.Exactes y	Val.Brutées g	Sol.Optimale p	Sol.Duale d
1	-2.478600	-15.227190	-14.295580	-14.503634	0.104027
1	-2.401220	-13.845090	-14.062380	-13.646271	0.000000
1	-2.327790	-12.613390	-12.301310	-12.788908	0.139772
1	-2.303150	-12.217000	-11.838690	-12.223796	0.472097
1	-2.254060	-11.452360	-11.989600	-11.658683	0.638964
1	-2.229650	-11.084330	-11.644900	-11.093570	0.530166
1	-2.204990	-10.720590	-10.956640	-10.528458	0.207276
1	-2.104590	-9.321840	-9.289360	-9.963345	0.221379
1	-2.089260	-9.119660	-8.825300	-9.398232	0.521948
1	-2.059630	-8.737070	-8.970350	-8.833120	0.753902
1	-2.053590	-8.660490	-8.713950	-8.268007	0.762884
1	-2.013150	-8.158860	-7.899440	-7.702894	0.673594
1	-1.957600	-7.501870	-7.116770	-7.137781	0.594809
1	-1.952970	-7.448760	-7.268600	-6.572669	0.168058
1	-1.819710	-6.025660	-5.490170	-6.007556	0.000000
1	-1.766430	-5.511740	-4.953260	-5.442443	0.076534
1	-1.764390	-5.492670	-5.206110	-4.899975	0.000000
1	-1.637410	-4.390090	-4.204440	-4.357507	0.000000
1	-1.541860	-3.665510	-3.246310	-3.944414	0.349052
1	-1.530520	-3.585230	-4.556290	-3.678645	0.259282
1	-1.457690	-3.097360	-3.148100	-3.412877	0.301900
1	-1.433630	-2.946530	-2.926520	-3.147108	0.454812
1	-1.429030	-2.918270	-3.507720	-2.881339	0.294534
1	-1.410080	-2.803680	-2.581030	-2.615571	0.151526
1	-1.369210	-2.566880	-1.946180	-2.349802	0.210329
1	-1.306210	-2.228660	-2.089300	-2.084033	0.266498
1	-1.261370	-2.006910	-2.463600	-1.818264	0.000000
1	-1.090230	-1.295840	-0.590160	-1.552496	0.214669
1	-1.086920	-1.284100	-1.906360	-1.296885	0.124601
1	-1.033950	-1.105340	-1.110340	-1.041274	0.000000
1	-0.979870	-0.940820	-0.507640	-0.785663	0.014410
1	-0.859820	-0.635670	-0.198730	-0.731834	0.295372
1	-0.853760	-0.622310	-0.790840	-0.678004	0.519916
1	-0.739180	-0.403870	-0.994160	-0.624175	0.559467
1	-0.725600	-0.382020	-0.862390	-0.570346	0.452996
1	-0.725240	-0.381460	-0.832930	-0.516516	0.188319
1	-0.715150	-0.365760	-0.011590	-0.462687	0.149189
1	-0.708280	-0.355310	-0.143400	-0.408858	0.242789
1	-0.641100	-0.263500	-0.668080	-0.355029	0.179863
1	-0.619460	-0.237710	-0.229750	-0.301199	0.152662
1	-0.524390	-0.144200	-0.150900	-0.247370	0.173696
1	-0.333810	-0.037200	-0.513270	-0.193541	0.034865
1	-0.283140	-0.022700	0.068220	-0.139711	0.000000
1	-0.263340	-0.018260	0.143760	-0.085882	0.079956

1	-0.249240	-0.015480	-0.290440	-0.068576	0.048980
1	-0.100350	-0.001010	0.096440	-0.051269	0.091859
1	0.057250	0.000190	-0.003520	-0.033963	0.149959
1	0.060960	0.000230	0.087710	-0.016656	0.260242
1	0.093150	0.000810	-0.740400	0.000650	
1	0.106210	0.001200	0.538440	0.017957	

Dimension : 50
Valeur de la fonction f(x) : 4.591
Nombre des iterations : 44
Temps du calcul en secondes : 53.17

Bruit 01

Poids W	Abscisses x	Val.Exactes y	Val.Brutées g	Sol.Primale p	Sol.Duale d
1	-2.478600	-15.227190	-15.040870	-15.053603	0.006366
1	-2.401220	-13.845090	-13.888550	-13.863084	0.000000
1	-2.327790	-12.613390	-12.550970	-12.672566	0.054432
1	-2.303150	-12.217000	-12.141340	-12.129341	0.102864
1	-2.254060	-11.452360	-11.559810	-11.586115	0.164449
1	-2.229650	-11.084330	-11.196440	-11.042890	0.149258
1	-2.204990	-10.720590	-10.767800	-10.499664	0.000000
1	-2.104590	-9.321840	-9.315340	-9.956438	0.171291
1	-2.089260	-9.119660	-9.060790	-9.416066	0.520219
1	-2.059630	-8.737070	-8.783730	-8.875693	0.915129
1	-2.053590	-8.660490	-8.671180	-8.335320	1.142109
1	-2.013150	-8.158860	-8.106970	-7.794947	1.213077
1	-1.957600	-7.501870	-7.424850	-7.254574	1.198907
1	-1.952970	-7.448760	-7.412730	-6.714201	0.835472
1	-1.819710	-6.025660	-5.918560	-6.173828	0.599672
1	-1.766430	-5.511740	-5.400040	-5.633455	0.480579
1	-1.764390	-5.492670	-5.435350	-5.093082	0.190352
1	-1.637410	-4.390090	-4.352960	-4.552709	0.000000
1	-1.541860	-3.665510	-3.581670	-4.012336	0.024981
1	-1.530520	-3.585230	-3.779440	-3.679516	0.000000
1	-1.457690	-3.097360	-3.107510	-3.346695	0.094612
1	-1.433630	-2.946530	-2.942520	-3.098399	0.267163
1	-1.429030	-2.918270	-3.036160	-2.850103	0.346685
1	-1.410080	-2.803680	-2.759150	-2.601806	0.347536
1	-1.369210	-2.566880	-2.442740	-2.353510	0.303771
1	-1.306210	-2.228660	-2.200790	-2.105214	0.212219
1	-1.261370	-2.006910	-2.098250	-1.856918	0.000000
1	-1.090230	-1.295840	-1.154710	-1.608621	0.014737
1	-1.086920	-1.284100	-1.408550	-1.360910	0.005654
1	-1.033950	-1.105340	-1.106340	-1.113199	0.000000
1	-0.979870	-0.940820	-0.854180	-0.865487	0.000000
1	-0.859820	-0.635670	-0.548280	-0.632161	0.041941
1	-0.853760	-0.622310	-0.656010	-0.580055	0.045904
1	-0.739180	-0.403870	-0.521930	-0.527949	0.052876
1	-0.725600	-0.382020	-0.478090	-0.475843	0.058725
1	-0.725240	-0.381460	-0.471760	-0.423737	0.040562
1	-0.715150	-0.365760	-0.294920	-0.371631	0.060755
1	-0.708280	-0.355310	-0.312930	-0.319524	0.084245
1	-0.641100	-0.263500	-0.344420	-0.267418	0.069234
1	-0.619460	-0.237710	-0.236120	-0.215312	0.043819

1	-0.524390	-0.144200	-0.145540	-0.163206	0.027237
1	-0.333810	-0.037200	-0.132410	-0.111100	0.000000
1	-0.283140	-0.022700	-0.004520	-0.058994	0.000000
1	-0.263340	-0.018260	0.014140	-0.022303	0.018222
1	-0.249240	-0.015480	-0.070480	-0.017701	0.010054
1	-0.100350	-0.001010	0.018480	-0.013098	0.017675
1	0.057250	0.000190	-0.000550	-0.008496	0.029269
1	0.060960	0.000230	0.017720	-0.003893	0.051669
1	0.093150	0.000810	-0.147430	0.000709	
1	0.106210	0.001200	0.108650	0.005312	

Dimension : 50
Valeur de la fonction f(x) : 1.175
Nombre des iterations : 42
Temps du calcul en secondes : 50.97

Méthode du gradient conjugué. Dimension 50.

Bruit 10

Poids W	Abscisses x	Val.Exactes y	Val.Brutées g	Sol.Optimale p	Sol.Duale d
1	-2.478600	-15.227190	3.405070	-13.240907	8.320084
1	-2.401220	-13.845090	-18.190830	-12.520049	13.800077
1	-2.327790	-12.613390	-6.371890	-11.769001	21.979926
1	-2.303150	-12.217000	-4.650910	-11.067053	33.373806
1	-2.254060	-11.452360	-22.197000	-10.344605	38.841019
1	-2.229650	-11.084330	-22.295680	-9.624056	38.002230
1	-2.204990	-10.720590	-15.441570	-8.905608	33.834160
1	-2.104590	-9.321840	-8.672230	-8.190060	29.450055
1	-2.089260	-9.119660	-3.232550	-7.459312	27.180731
1	-2.059630	-8.737070	-13.402560	-6.738064	21.575560
1	-2.053590	-8.660490	-9.729660	-6.016415	14.115764
1	-2.013150	-8.158860	-2.970530	-5.295067	7.818200
1	-1.957600	-7.501870	0.200140	-4.573519	3.907441
1	-1.952970	-7.448760	-3.845460	-3.852071	0.000000
1	-1.819710	-6.025660	4.684260	-3.130622	0.000000
1	-1.766430	-5.511740	5.657880	-2.486820	4.072350
1	-1.764390	-5.492670	0.238440	-2.450157	9.489000
1	-1.637410	-4.390090	-0.677060	-2.409495	15.769862
1	-1.541860	-3.665510	4.718560	-2.370833	25.596423
1	-1.530520	-3.585230	-23.006500	-2.332171	25.085820
1	-1.457690	-3.097360	-4.112220	-2.293509	23.665861
1	-1.433630	-2.946530	-2.546510	-2.254847	22.100071
1	-1.429030	-2.918270	-14.707390	-2.216185	14.288678
1	-1.410080	-2.803680	1.649270	-2.177523	8.390682
1	-1.369210	-2.566880	9.847130	-2.138861	8.485681
1	-1.306210	-2.228660	0.558510	-2.100199	9.910035
1	-1.261370	-2.006910	-11.140820	-2.061537	6.811747
1	-1.090230	-1.295840	12.817770	-2.022875	11.099782
1	-1.086920	-1.284100	-13.729240	-1.984213	9.532303
1	-1.033950	-1.105340	-1.205320	-1.945551	8.334940
1	-0.979870	-0.940820	7.722790	-1.906889	11.952417
1	-0.859820	-0.635670	8.103040	-1.868227	20.555528
1	-0.853760	-0.622310	-3.992910	-1.829565	28.076966
1	-0.739180	-0.403870	-12.209670	-1.790903	30.389021

1	-0.725600	-0.382020	-9.989350	-1.752242	28.582522
1	-0.725240	-0.381460	-9.410760	-1.713580	22.927432
1	-0.715150	-0.365760	6.717560	-1.674918	21.468581
1	-0.708280	-0.355310	3.882900	-1.636256	22.770309
1	-0.641100	-0.263500	-8.355180	-1.597594	20.691243
1	-0.619460	-0.237710	-0.078590	-1.558932	19.353348
1	-0.524390	-0.144200	-0.278260	-1.520270	18.636457
1	-0.333810	-0.037200	-9.558620	-1.481608	13.881061
1	-0.283140	-0.022700	1.795610	-1.443046	10.745043
1	-0.263340	-0.018260	3.222210	-1.404284	9.922071
1	-0.249240	-0.015480	-5.514660	-1.365622	7.024681
1	-0.100350	-0.001010	1.948060	-1.327060	5.765001
1	0.057250	0.000190	-0.074010	-1.288298	5.112064
1	0.060960	0.000230	1.749980	-1.250636	5.960136
1	0.093150	0.000810	-14.823390	-1.210974	
1	0.106210	0.001200	10.745960	-1.172312	

Dimension : 50
Valeur de la fonction f(x) : 1627.386
Nombre des iterations : 89
Temps du calcul en secondes : 492.11

Bruit 1

Poids W	Abscisses x	Val.Exactes y	Val.Brutées g	Sol.Optimale p	Sol.Duale d
1	-2.478600	-15.227190	-13.363960	-14.115015	0.376077
1	-2.401220	-13.845090	-14.279660	-13.513300	0.368574
1	-2.327790	-12.613390	-11.989240	-12.913082	0.820901
1	-2.303150	-12.217000	-11.460390	-12.311065	1.700045
1	-2.254060	-11.452360	-12.526830	-11.710649	2.170009
1	-2.229650	-11.084330	-12.205470	-11.108432	2.101454
1	-2.204990	-10.720590	-11.192690	-10.507216	1.670362
1	-2.104590	-9.321840	-9.256880	-9.960000	1.574030
1	-2.089260	-9.119660	-8.530950	-9.304783	1.864213
1	-2.059630	-8.737070	-9.203620	-8.703566	1.904570
1	-2.053590	-8.660490	-8.767400	-8.102350	1.612401
1	-2.013150	-8.158860	-7.640020	-7.501133	1.250789
1	-1.957600	-7.501870	-6.731670	-6.900017	0.973301
1	-1.952970	-7.448760	-7.088430	-6.308700	0.300947
1	-1.819710	-6.025660	-4.954670	-5.700084	0.000000
1	-1.766430	-5.511740	-4.394780	-5.100000	0.050797
1	-1.764390	-5.492670	-4.919550	-4.720363	0.000000
1	-1.637410	-4.390090	-4.018790	-4.344460	0.113038
1	-1.541860	-3.665510	-2.827100	-4.056083	0.840568
1	-1.530520	-3.585230	-5.527360	-3.767707	0.690271
1	-1.457690	-3.097360	-3.198850	-3.480000	0.676214
1	-1.433630	-2.946530	-2.906520	-3.190954	0.806374
1	-1.429030	-2.918270	-4.097180	-2.902577	0.340232
1	-1.410080	-2.803680	-2.358380	-2.614200	0.000000
1	-1.369210	-2.566880	-1.325480	-2.325824	0.159940
1	-1.306210	-2.228660	-1.949940	-2.045091	0.370456
1	-1.261370	-2.006910	-2.920300	-1.764357	0.000000
1	-1.090230	-1.295840	0.115520	-1.483624	0.430116
1	-1.086920	-1.284100	-2.528610	-1.268872	0.230363

1	-1.033950	-1.105340	-1.115340	-1.054119	0.000000
1	-0.979870	-0.940820	-0.074460	-0.840367	0.152090
1	-0.859820	-0.635670	0.238210	-0.800487	0.818528
1	-0.853760	-0.622310	-0.959370	-0.741607	1.376085
1	-0.739180	-0.403870	-1.584450	-0.692727	1.487780
1	-0.725600	-0.382020	-1.342750	-0.643848	1.250024
1	-0.725240	-0.381460	-1.284390	-0.594968	0.667557
1	-0.715150	-0.365760	0.342580	-0.546088	0.530425
1	-0.708280	-0.355310	0.068510	-0.497208	0.674151
1	-0.641100	-0.263500	-1.072670	-0.448329	0.506707
1	-0.619460	-0.237710	-0.221800	-0.400449	0.428087
1	-0.524390	-0.144200	-0.157600	-0.350569	0.446051
1	-0.333810	-0.037200	-0.989340	-0.301689	0.120001
1	-0.283140	-0.022700	0.159130	-0.252810	0.000000
1	-0.263340	-0.018260	0.305790	-0.203930	0.135069
1	-0.249240	-0.015480	-0.565400	-0.161842	0.068059
1	-0.100350	-0.001010	0.193900	-0.119754	0.157877
1	0.057250	0.000190	-0.007230	-0.077666	0.283012
1	0.060960	0.000230	0.175200	-0.035578	0.513536
1	0.093150	0.000810	-1.481610	0.006510	
1	0.106210	0.001200	1.075670	0.048598	

Dimension : 50
Valeur de la fonction f(x) : 15.492
Nombre des iterations : 83
Temps du calcul en secondes : 611.23

Bruit 05

Poids W	Abscisses x	Val.Exactes y	Val.Brutées g	Sol.Optimale p	Sol.Duale d
1	-2.478600	-15.227190	-14.295580	-14.508761	0.106590
1	-2.401220	-13.845090	-14.062380	-13.636019	0.000000
1	-2.327790	-12.613390	-12.301310	-12.761577	0.123543
1	-2.303150	-12.217000	-11.838690	-12.205729	0.430606
1	-2.254060	-11.452360	-11.989600	-11.654044	0.569890
1	-2.229650	-11.084330	-11.644900	-11.098329	0.435890
1	-2.204990	-10.720590	-10.956640	-10.542614	0.094876
1	-2.104590	-9.321840	-9.289360	-9.982493	0.100429
1	-2.089260	-9.119660	-8.825300	-9.420081	0.403373
1	-2.059630	-8.737070	-8.970350	-8.853558	0.647921
1	-2.053590	-8.660490	-8.713950	-8.284206	0.677597
1	-2.013150	-8.158860	-7.899440	-7.710886	0.612995
1	-1.957600	-7.501870	-7.116770	-7.138685	0.559351
1	-1.952970	-7.448760	-7.268600	-6.565498	0.154156
1	-1.819710	-6.025660	-5.490170	-5.992249	0.000000
1	-1.766430	-5.511740	-4.953260	-5.419783	0.079105
1	-1.764390	-5.492670	-5.206110	-4.889689	0.000000
1	-1.637410	-4.390090	-4.204440	-4.362651	0.000000
1	-1.541860	-3.665510	-3.246310	-3.954908	0.354299
1	-1.530520	-3.585230	-4.556290	-3.682305	0.271606
1	-1.457690	-3.097360	-3.148100	-3.411008	0.320366
1	-1.433630	-2.946530	-2.926520	-3.142911	0.477322
1	-1.429030	-2.918270	-3.507720	-2.874304	0.317570
1	-1.410080	-2.803680	-2.581030	-2.607399	0.171003
1	-1.369210	-2.566880	-1.946180	-2.344159	0.223424

1	-1.306210	-2.228660	-2.089300	-2.082266	0.272329
1	-1.261370	-2.006910	-2.463600	-1.824244	0.001555
1	-1.090230	-1.295840	-0.590160	-1.561930	0.216667
1	-1.086920	-1.284100	-1.906360	-1.297341	0.127270
1	-1.033950	-1.105340	-1.110340	-1.034596	0.000000
1	-0.979870	-0.940820	-0.507640	-0.771851	0.004836
1	-0.859820	-0.635670	-0.198730	-0.726578	0.273596
1	-0.853760	-0.622310	-0.790840	-0.677939	0.485905
1	-0.739180	-0.403870	-0.994160	-0.625625	0.513946
1	-0.725600	-0.382020	-0.862390	-0.573311	0.397448
1	-0.725240	-0.381460	-0.832930	-0.518189	0.123580
1	-0.715150	-0.365760	-0.011590	-0.467323	0.077578
1	-0.708280	-0.355310	-0.143400	-0.419905	0.169829
1	-0.641100	-0.263500	-0.668080	-0.367770	0.111924
1	-0.619460	-0.237710	-0.229750	-0.315319	0.096804
1	-0.524390	-0.144200	-0.150900	-0.258542	0.135505
1	-0.333810	-0.037200	-0.513270	-0.198908	0.017025
1	-0.283140	-0.022700	0.068220	-0.134691	0.000000
1	-0.263340	-0.018260	0.143760	-0.065578	0.087644
1	-0.249240	-0.015480	-0.290440	-0.056357	0.058247
1	-0.100350	-0.001010	0.096440	-0.044804	0.099471
1	0.057250	0.000190	-0.003520	-0.032293	0.155082
1	0.060960	0.000230	0.087710	-0.015786	0.262441
1	0.093150	0.000810	-0.740400	-0.000801	
1	0.106210	0.001200	0.538440	0.013559	

Dimension : 50
Valeur de la fonction f(x) : 4.576
Nombre des iterations : 82
Temps du calcul en secondes : 454.07

Bruit 01

Poids W	Abscisses x	Val.Exactes y	Val.Brutées g	Sol.Optimale p	Sol.Duale d
1	-2.478600	-15.227190	-15.040870	-15.053114	0.006122
1	-2.401220	-13.845090	-13.888550	-13.864062	0.000000
1	-2.327790	-12.613390	-12.550970	-12.674792	0.055789
1	-2.303150	-12.217000	-12.141340	-12.128735	0.105275
1	-2.254060	-11.452360	-11.559810	-11.583282	0.166497
1	-2.229650	-11.084330	-11.196440	-11.040919	0.149959
1	-2.204990	-10.720590	-10.767800	-10.500959	0.000000
1	-2.104590	-9.321840	-9.315340	-9.958966	0.171854
1	-2.089260	-9.119660	-9.060790	-9.416133	0.521379
1	-2.059630	-8.737070	-8.783730	-8.874335	0.916207
1	-2.053590	-8.660490	-8.671180	-8.336511	1.143701
1	-2.013150	-8.158860	-8.106970	-7.797408	1.216413
1	-1.957600	-7.501870	-7.424850	-7.254359	1.203879
1	-1.952970	-7.448760	-7.412730	-6.712725	0.841344
1	-1.819710	-6.025660	-5.918560	-6.172102	0.605579
1	-1.766430	-5.511740	-5.400040	-5.630848	0.485218
1	-1.764390	-5.492670	-5.435350	-5.090416	0.192391
1	-1.637410	-4.390090	-4.352960	-4.553834	0.000000
1	-1.541860	-3.665510	-3.581670	-4.018002	0.025775
1	-1.530520	-3.585230	-3.779440	-3.676339	0.000000

1	-1.457690	-3.097360	-3.107510	-3.337169	0.089054
1	-1.433630	-2.946530	-2.942520	-3.097832	0.255765
1	-1.429030	-2.918270	-3.036160	-2.855389	0.332090
1	-1.410080	-2.803680	-2.759150	-2.609739	0.333709
1	-1.369210	-2.566880	-2.442740	-2.360982	0.294450
1	-1.306210	-2.228660	-2.200790	-2.107238	0.208414
1	-1.261370	-2.006910	-2.098250	-1.853493	0.000000
1	-1.090230	-1.295840	-1.154710	-1.602572	0.015517
1	-1.086920	-1.284100	-1.408550	-1.358228	0.005873
1	-1.033950	-1.105340	-1.106340	-1.113883	0.000000
1	-0.979870	-0.940820	-0.854180	-0.865925	0.000000
1	-0.859820	-0.635670	-0.548280	-0.631018	0.041369
1	-0.853760	-0.622310	-0.656010	-0.581316	0.045391
1	-0.739180	-0.403870	-0.521930	-0.529002	0.052949
1	-0.725600	-0.382020	-0.478090	-0.475874	0.059400
1	-0.725240	-0.381460	-0.471760	-0.424020	0.041980
1	-0.715150	-0.365760	-0.294920	-0.370398	0.062299
1	-0.708280	-0.355310	-0.312930	-0.319413	0.085860
1	-0.641100	-0.263500	-0.344420	-0.265619	0.070020
1	-0.619460	-0.237710	-0.236120	-0.215469	0.043855
1	-0.524390	-0.144200	-0.145540	-0.163970	0.026905
1	-0.333810	-0.037200	-0.132410	-0.112501	0.000000
1	-0.283140	-0.022700	-0.004520	-0.058329	0.000000
1	-0.263340	-0.018260	0.014140	-0.025144	0.019642
1	-0.249240	-0.015480	-0.070480	-0.016728	0.012408
1	-0.100350	-0.001010	0.018480	-0.010343	0.019586
1	0.057250	0.000190	-0.000550	-0.007694	0.030335
1	0.060960	0.000230	0.017720	-0.003922	0.051906
1	0.093150	0.000810	-0.147430	-0.000477	
1	0.106210	0.001200	0.108650	0.004838	

Dimension : 50
Valeur de la fonction f(x) : 1.172
Nombre des iterations : 69
Temps du calcul en secondes : 455.28

REFERENCES

[1] De Leeuw, J. [1976]
Some convergence theorems for MDS. Psychonomic Society Meeting, Nunspeet.

[2] Kruskal, J. B. [1964]
Multidimensional Scaling by optimizing goodness-of-fit to a nonmetric hypothesis. Psychometrika, vol 19, p 1-28.

[3] Guttman, L. [1968]
A general nometric technique for finding the smallest coordinate space for a configuration of points. Psychometrika, vol 33, p 469-506.

[4] Lingoes, J.C. & Roskam, E.E. [1973]
A mathematical and empirical analysis of two multidimensional scaling algorithms. Psychometrika, vol 38, monograph supplement.

[5] De Leeuw, J. [1976]
Applications of convex analysis to multidimensional scaling. European meeting of statisticians, Grenoble.

[6] Forsythe, G., E. and Golub, G. H. (1965)
On the stationary values of a second-degree polynomial on the unit sphere. J. Soc. Indust. Appl. Math., Vol. 13, N°.4, Printed in U.S.A.

[7] Pham Dinh Tao (1989)
Numerical method for minimizing the Hermitian quadratic form on the unit Euclidean sphere of C^n. Submitted to SIAM Journal on Matrix Analysis and Applications.

[8] Drouet d'Aubigny. G. (1989)
L'analyse Multidimensionnelle des données de dissimilarité. Thèse d'Etat, Université Joseph Fourier - Grenoble I. Grenoble.

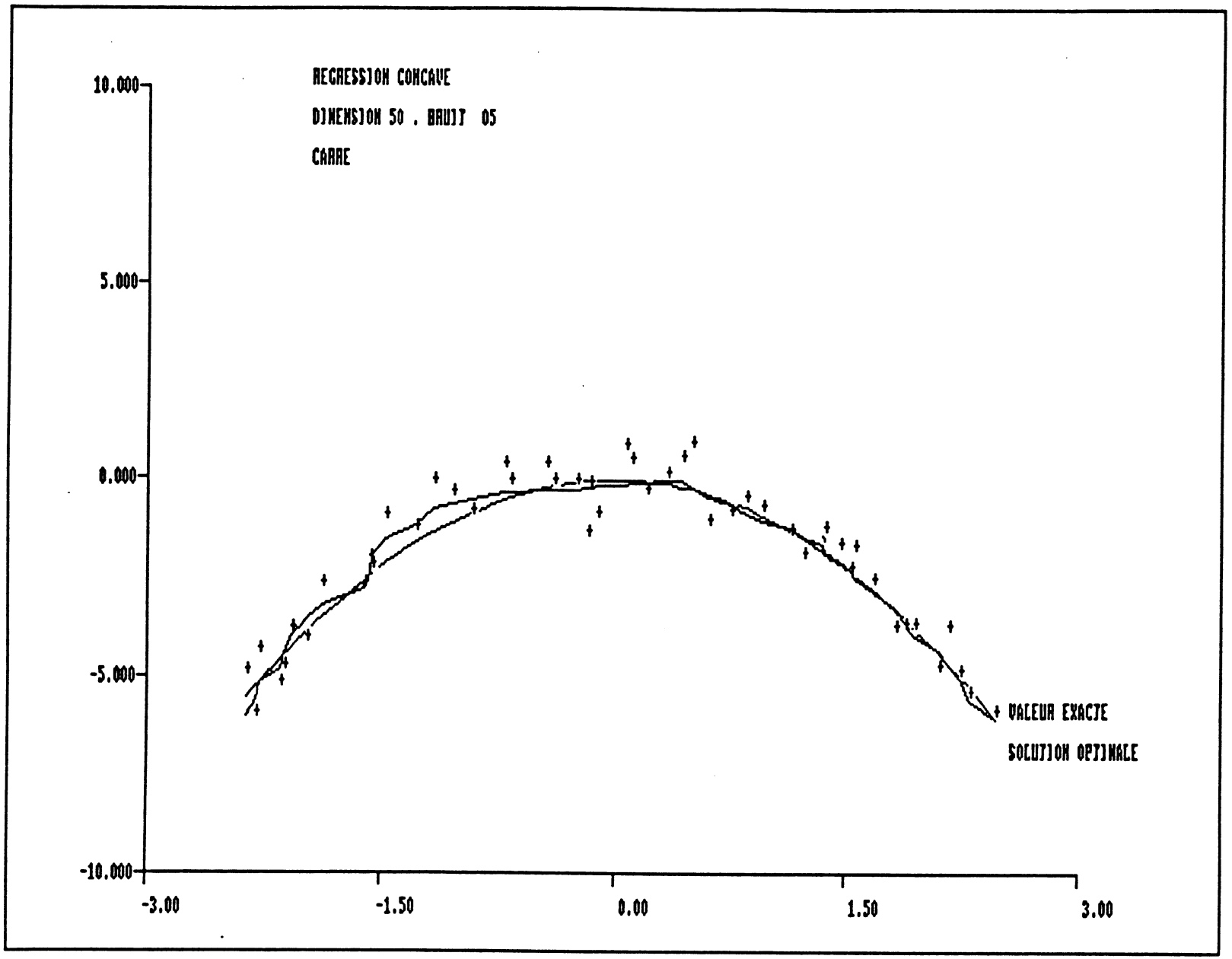
[9] Pham Dinh Tao, Wang. S, Yassine. A (1989)
Training multi-layered neural network with a Trust-Region based algorithm. To appear in Mathematical Modelling & Numerical Analysis.

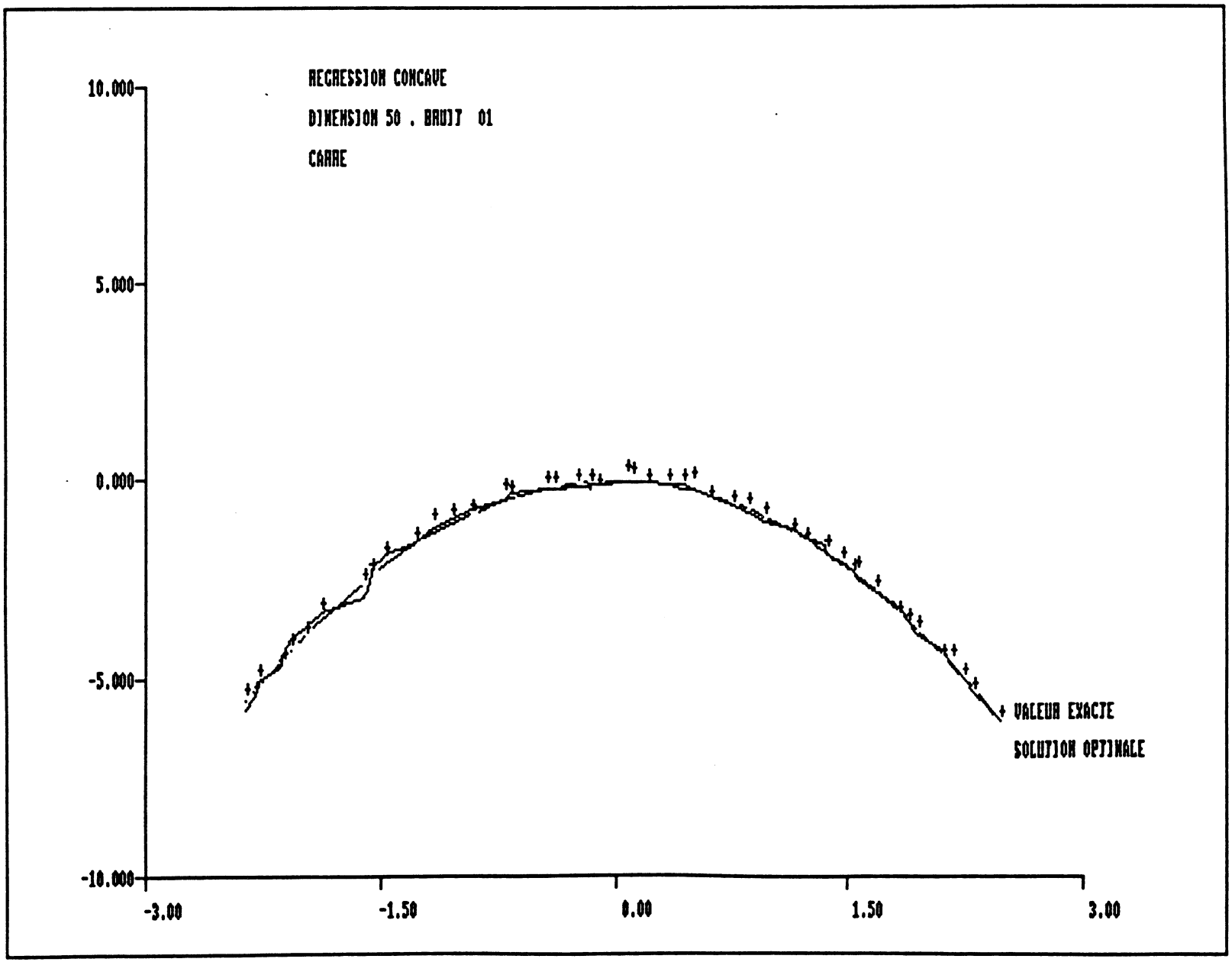
[10] Clermont. J. R., de la Lande. M. E., Pham Dinh Tao, Yassine. A. (1989)
Numerical simulation of axisymmetric converging flows using the stream tube method and a Trust Region optimization algorithm. Submitted to Structural Optimization.

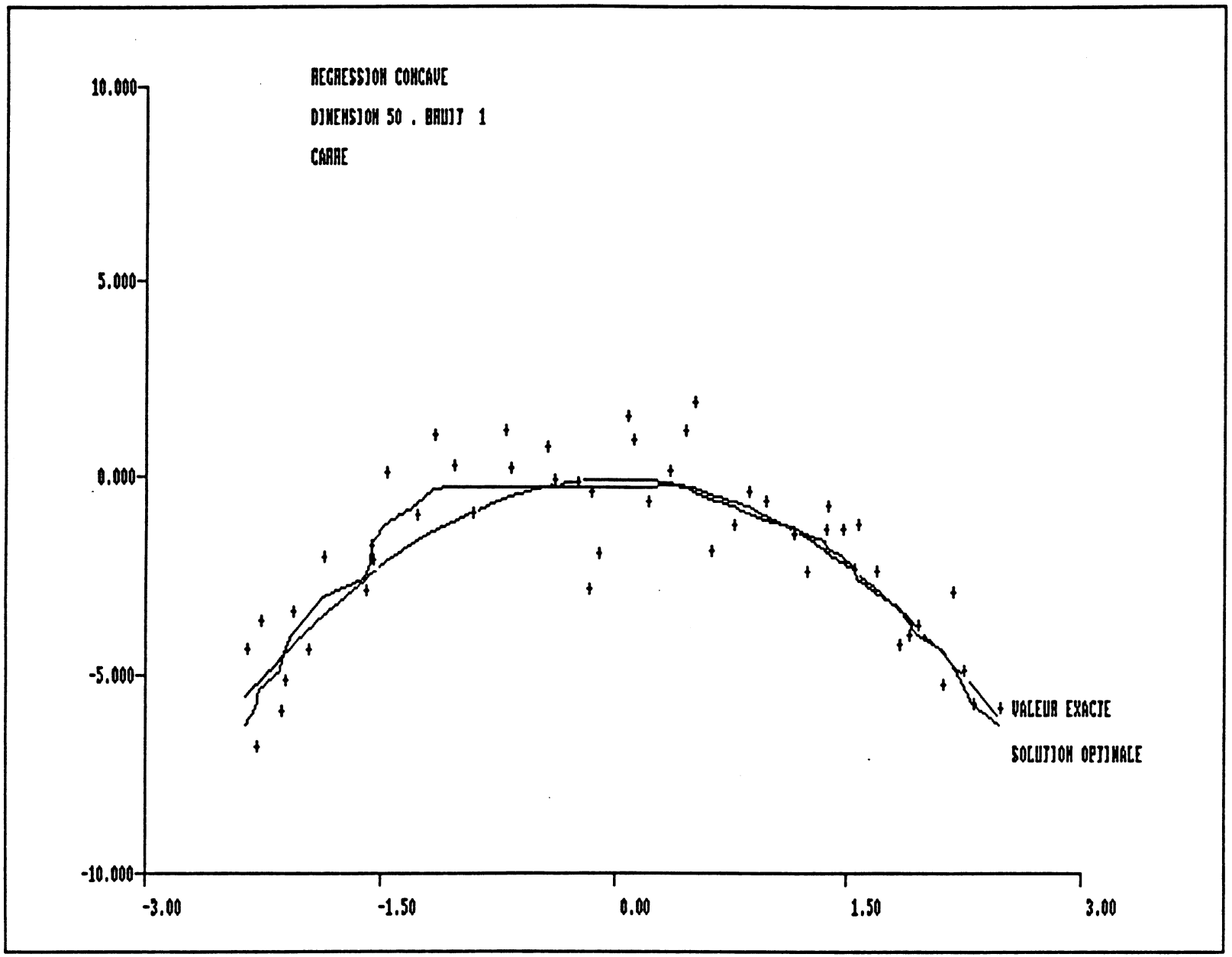
[11] Clermont. J. R., de la Lande. M. E., Pham Dinh Tao, Yassine. A. (1989)
Analysis of plane and axisymmetric flows of incompressible fluids with the stream tube method. Numerical simulation by Trust Region optimization algorithm. Submitted in Numer. Math.

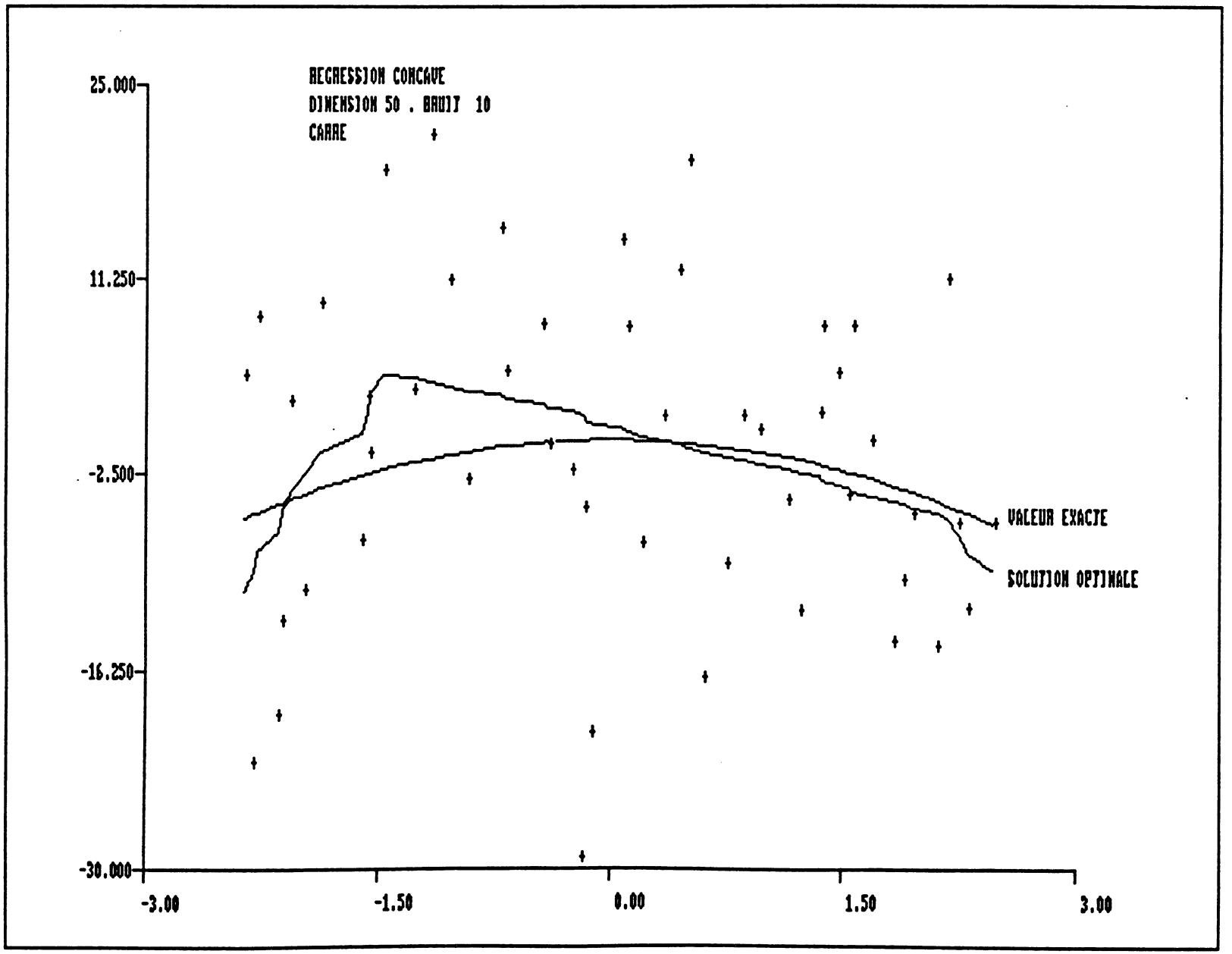
[12] Cea. J. (1971)
Optimisation : Théories et algorithmes, Dunod.

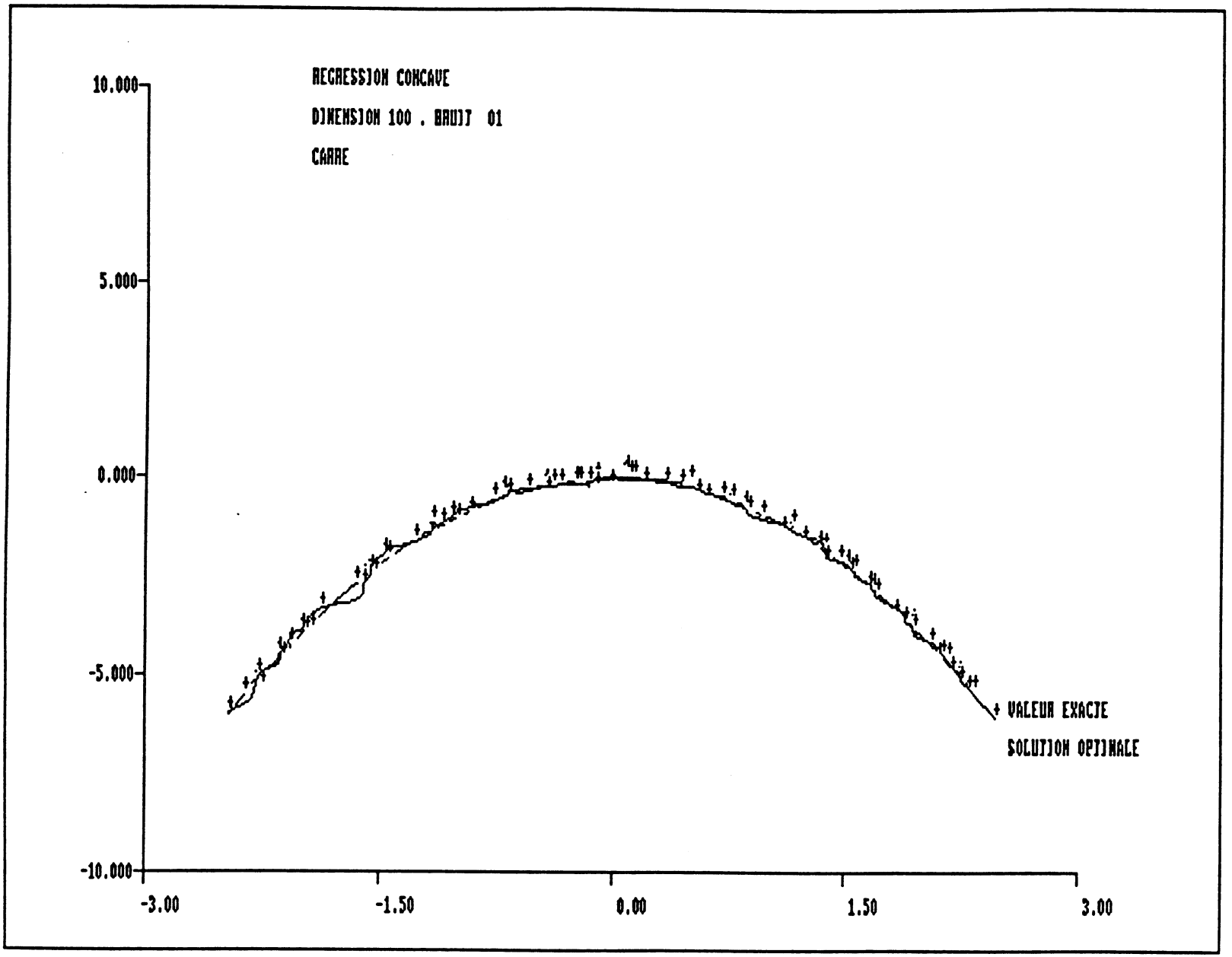
[13] Davis. Philip. (1952)
An application of doubly orthogonal functions to a problem of approximation in two regions, Trans. Amer. Math. Soc., 72, pp. 104-137.
[14] Ekeland. I. & Temam. R. (1974)
Analyse Convexe et problèmes variationnels. Dunod, Gauthier-Villars.
[15] Fletcher. R. (1980)
Practical Methods of Optimization, vol 1, John Wiley, New York.
[16] Gastinel. N. (1966)
Analyse numérique linéaire, Hermann, Paris.
[17] Gill. P. E., Murray. W. & Wright. M. H. (1981)
Practical Optimization, Academic Press.
[18] Hebden. M. D. (1973)
An algorithm for minimization using exact second derivatives. Atomic Energy Research Establishment report T.P. 515, Harwell, England.
[19] Kaniel. S. & Dax. A. (1979)
A modified Newton's method for unconstrained minimization. SIAM J. Num. Anal., pp. 324-331.
[20] Lancaster. P. (1969)
Theory of Matrix, Academic Press, New York and London.
[21] Luenberger. D. G. (1974)
A combined penalty function and gradient projection method for nonlinear programming, J. Opt. Th. Applics, pp. 251-263.
[22] Marquart. D. W. (1963)
An algorithm for least-squares estimation of nonlinear parameters, J. Soc. Indust. Appl. Math, 11, pp. 431-441.
[23] Clermont. J. R. and de la Lande. M. E. (1986)
A method for the simulation of plane or axisymmetric flows of incompressible fluids based on the concept of the stream function. Engineering computations, vol 3, 4, 339-347.
[24] Clermont. J. R. and de la Lande. M. E. (1986)
A computational method for flows of incompressible fluids based on a non-conformal transformation of the physical domain, Mech. Res. Comm. Vol 13, 5, 239-246.
[25] Clermont. J. R., de la Lande. M. E., Pham Dinh Tao, Yassine. A. (1989)
A numerical method for solving converging flows using a stream tube analysis and Trust Region technique. To appear in Proceedings of 6th International Conference on Numerical Methods in Laminar and Turbulent Flow. Swansea, U.K.
[26] Drouet D'aubigny. G., Pham Dinh Tao and Yassine. A. (1988)
A conditional conjugate gradient method for solving the isotonic and concave regression problems. to appear in Computational Statistics & Data Analysis.

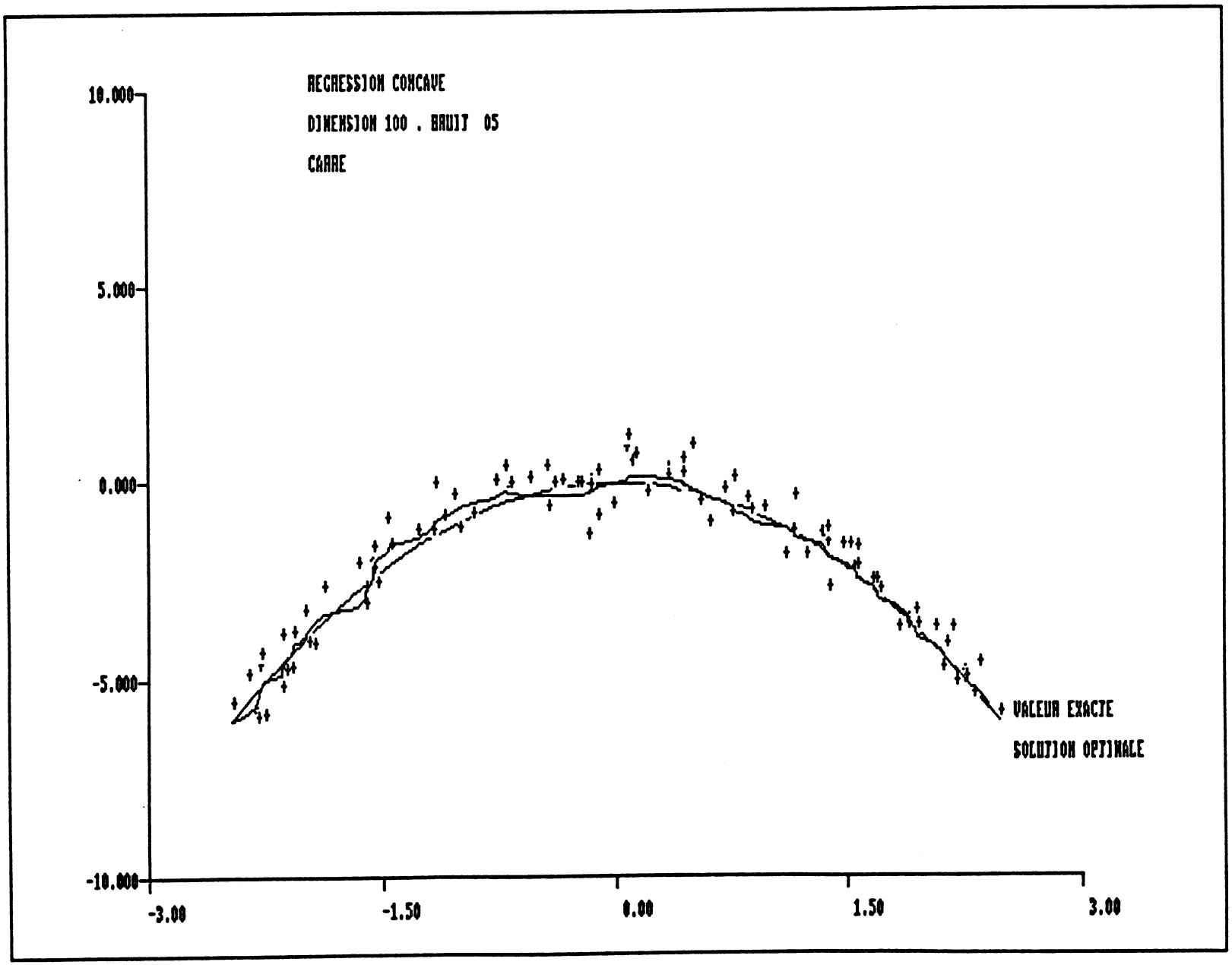

REGRESSION CONCAVE
DIMENSION 50 . BRUIT 05
CARRE
10.000
5.000
0.000
-5.000
-10.000
-3.00
-1.50
0.00
1.50
3.00
VALEUR EXACTE
SOLUTION OPTIMALE

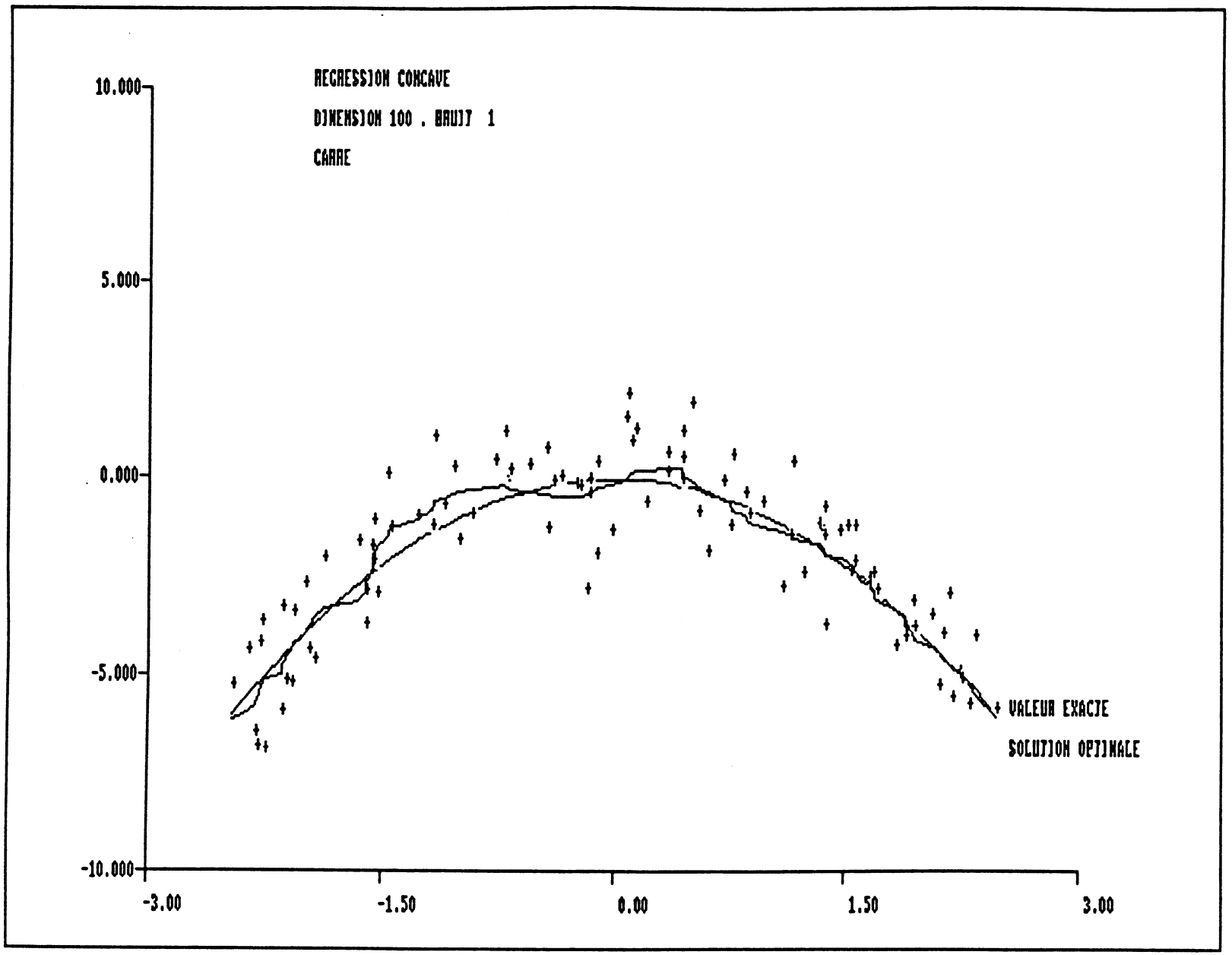

REGRESSION CONCAVE
DIMENSION 50 . BRUIT 01
CARRE
10.000
5.000
0.000
-5.000
-10.000
-3.00
-1.50
0.00
1.50
3.00
VALEUR EXACTE
SOLUTION OPTIMALE

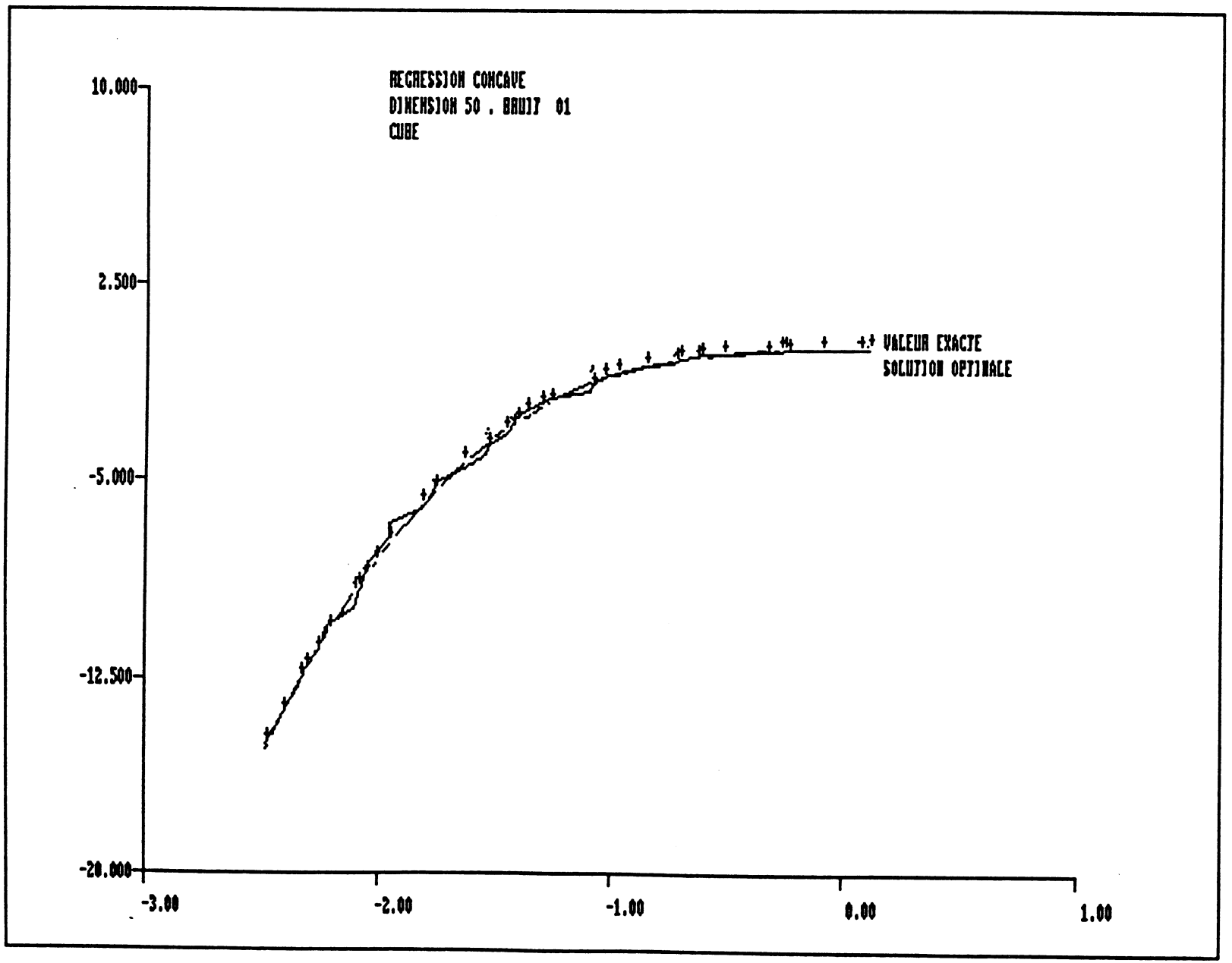

REGRESSION CONCAVE
DIMENSION 50 . BRUIT 1
CARRE
10.000
5.000
0.000
-5.000
-10.000
-3.00
-1.50
0.00
1.50
3.00
VALEUR EXACTE
SOLUTION OPTIMALE

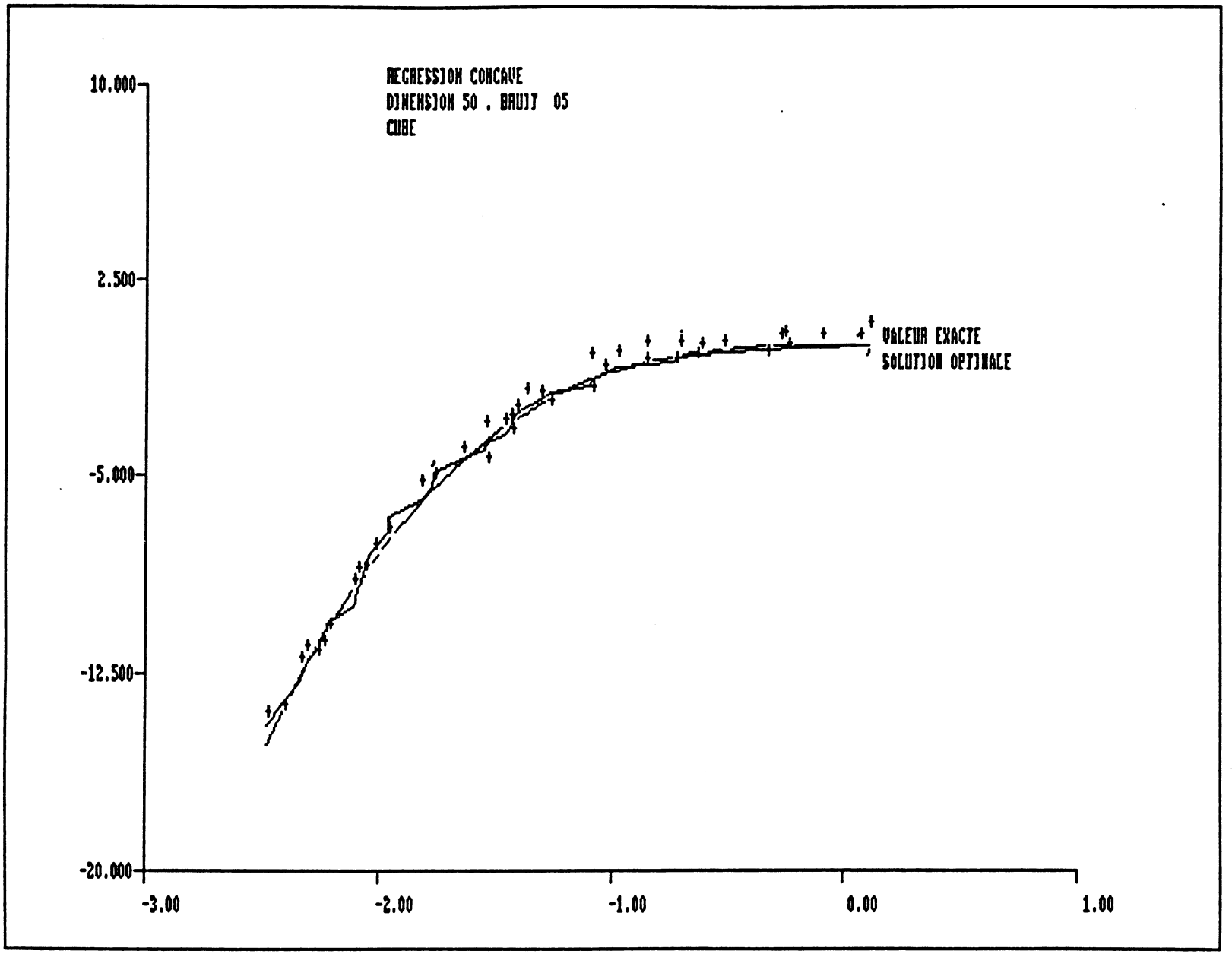

REGRESSION CONCAVE
DIMENSION 50 . HAUT 10
CARRE
25.000
11.250
-2.500
-16.250
-30.000
-3.00
-1.50
0.00
1.50
3.00
VALEUR EXACTE
SOLUTION OPTIMALE

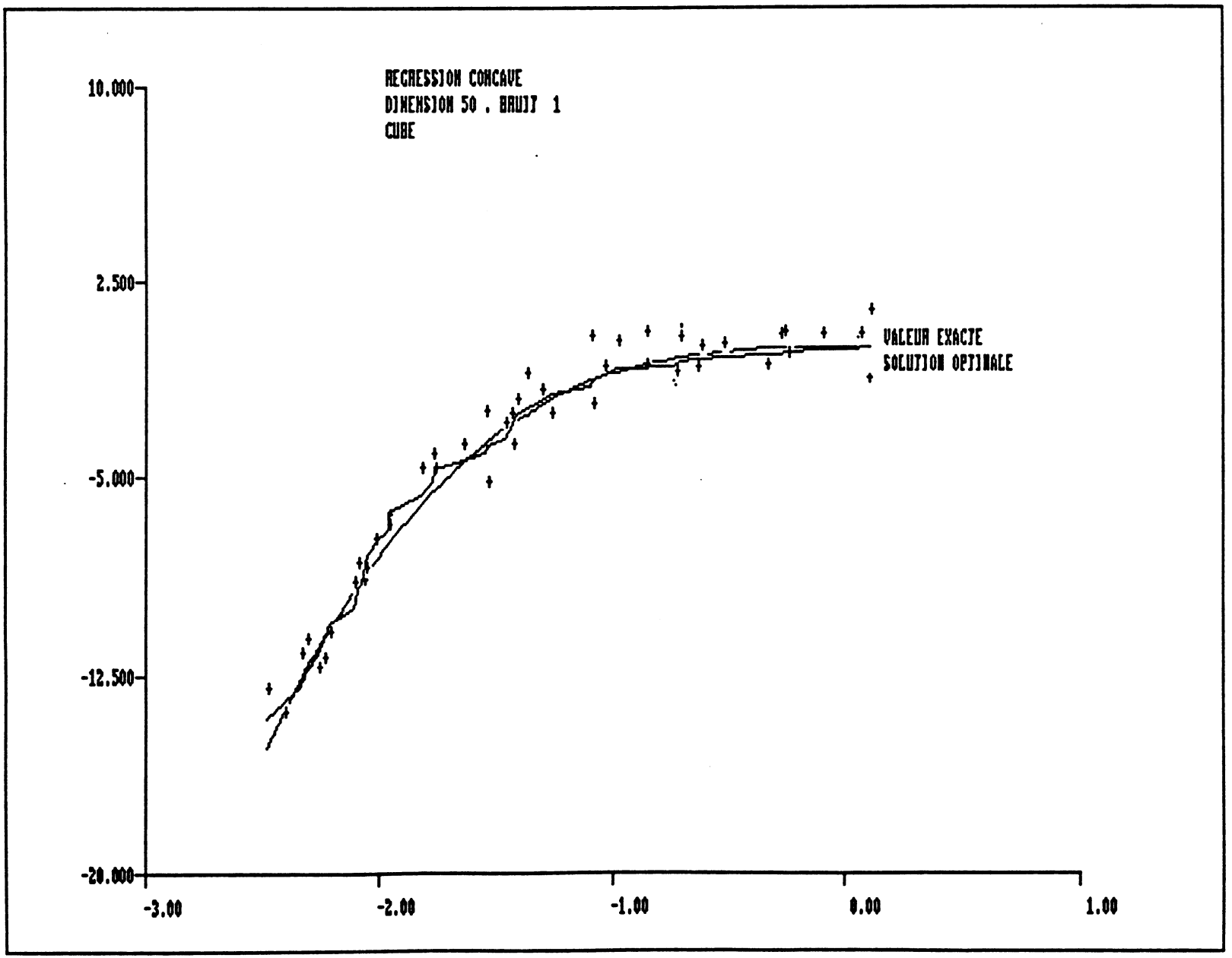

REGRESSION CONCAVE
DIMENSION 100 . BRUIT 01
CARRE
10.000
5.000
0.000
-5.000
-10.000
-3.00
-1.50
0.00
1.50
3.00
VALEUR EXACTE
SOLUTION OPTIMALE

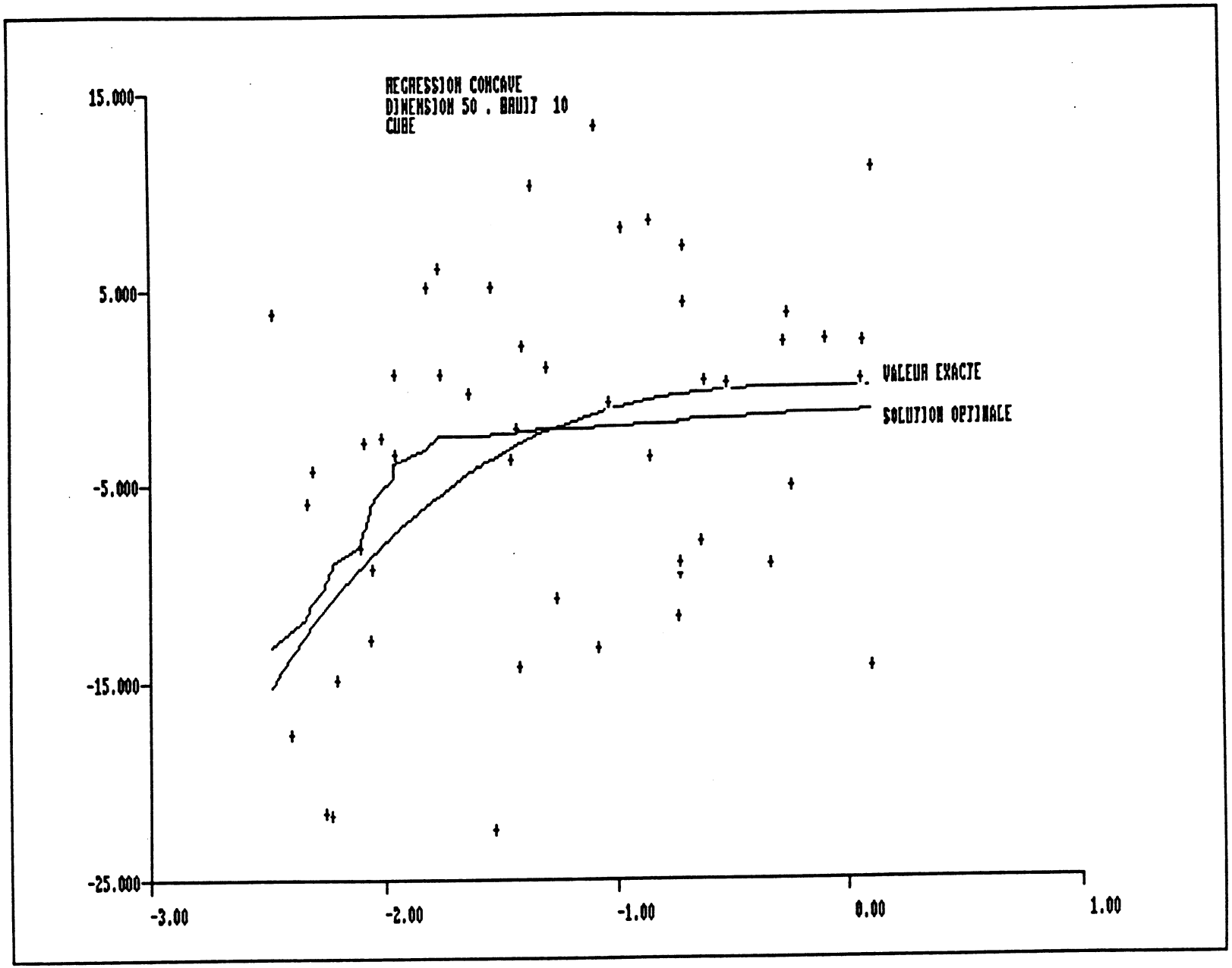

REGRESSION CONCAVE
DIMENSION 100 . BRUIT 05
CARRE
10.000
5.000
0.000
-5.000
-10.000
-3.00
-1.50
0.00
1.50
3.00
VALEUR EXACTE
SOLUTION OPTIMALE

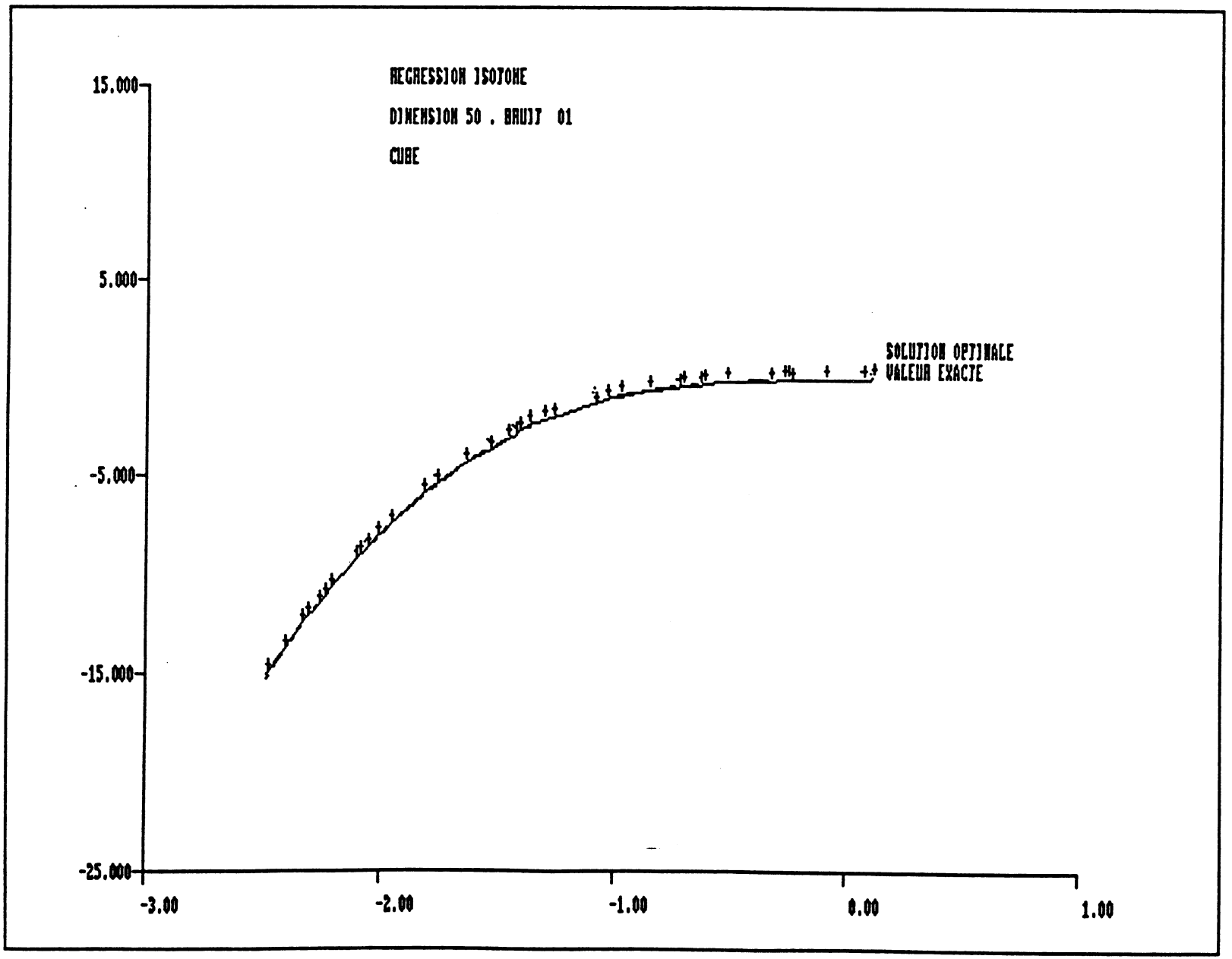

REGRESSION CONCAVE
DIMENSION 100 . BRUIT 1
CARRE
10.000
5.000
0.000
-5.000
-10.000
-3.00
-1.50
0.00
1.50
3.00
VALEUR EXACTE
SOLUTION OPTIMALE

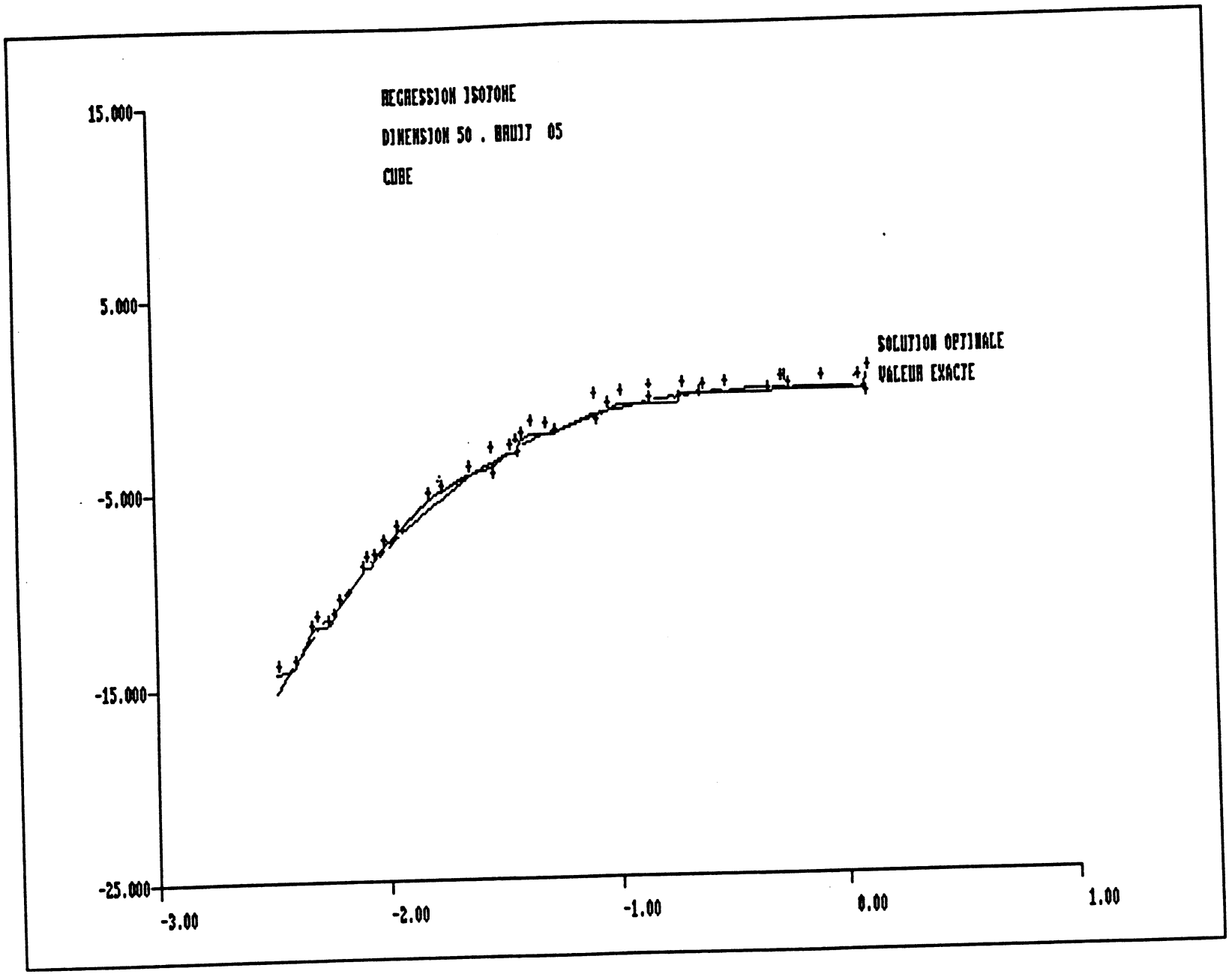

REGRESSION CONCAVE
DIMENSION 50 . BRUIT 01
CUBE
10.000
2.500
-5.000
-12.500
-20.000
-3.00
-2.00
-1.00
0.00
1.00
VALEUR EXACTE
SOLUTION OPTIMALE

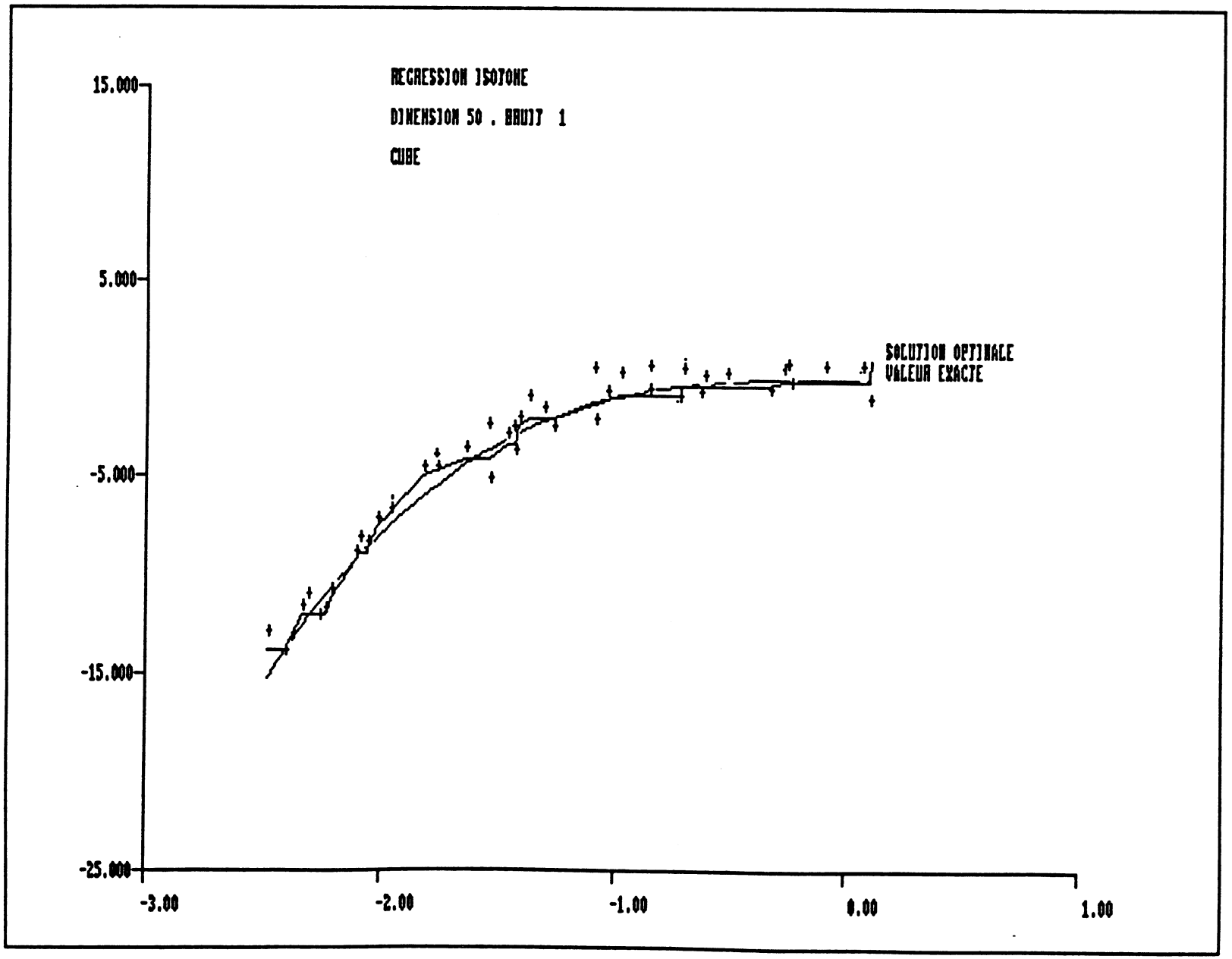

REGRESSION CONCAVE
DIMENSION 50 . BRUIT 05
CUBE
10.000
2.500
-5.000
-12.500
-20.000
-3.00
-2.00
-1.00
0.00
1.00
VALEUR EXACTE
SOLUTION OPTIMALE

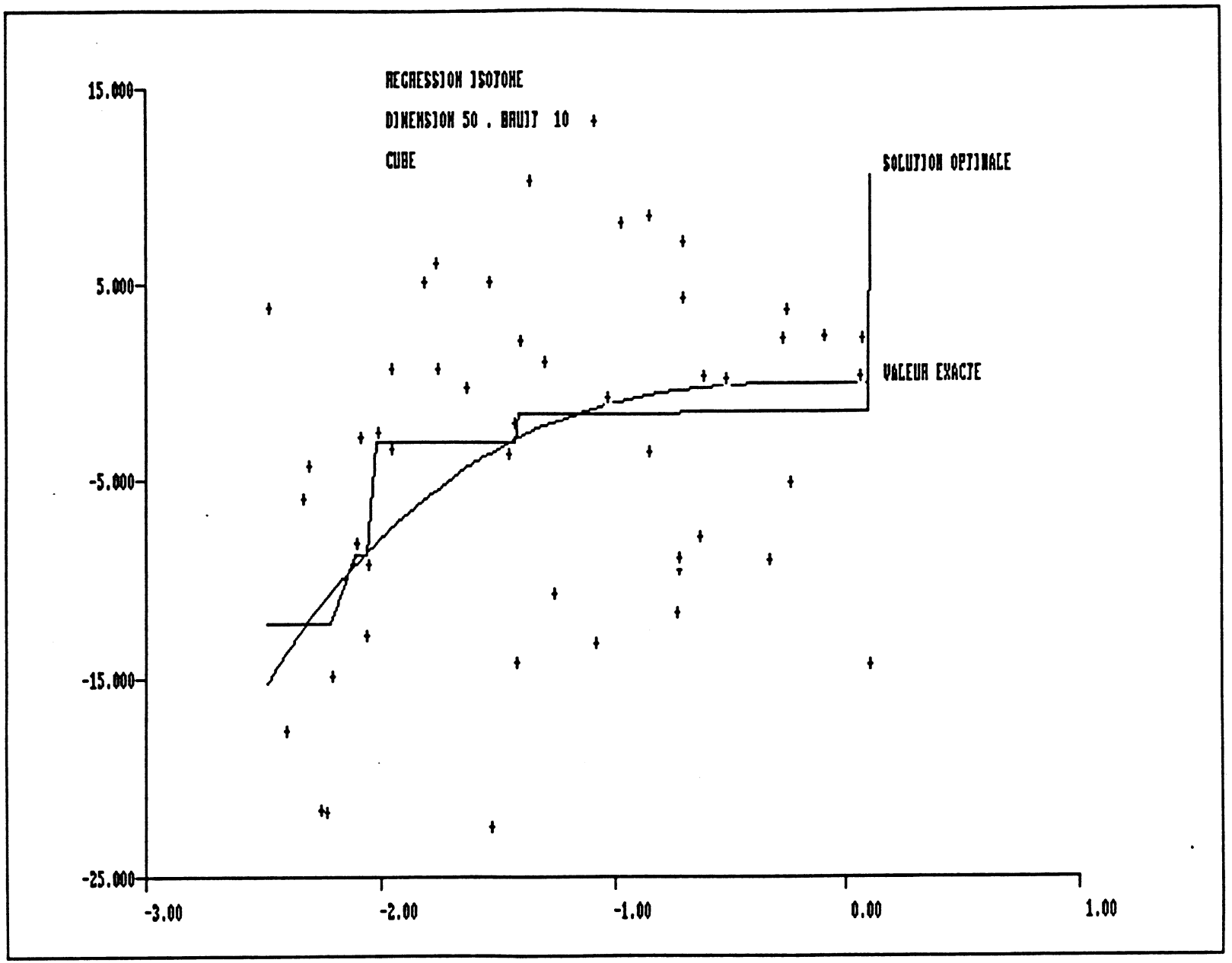

REGRESSION CONCAVE
DIMENSION 50 . BRUIT 1
CUBE
10.000
2.500
-5.000
-12.500
-20.000
-3.00
-2.00
-1.00
0.00
1.00
VALEUR EXACTE
SOLUTION OPTIMALE

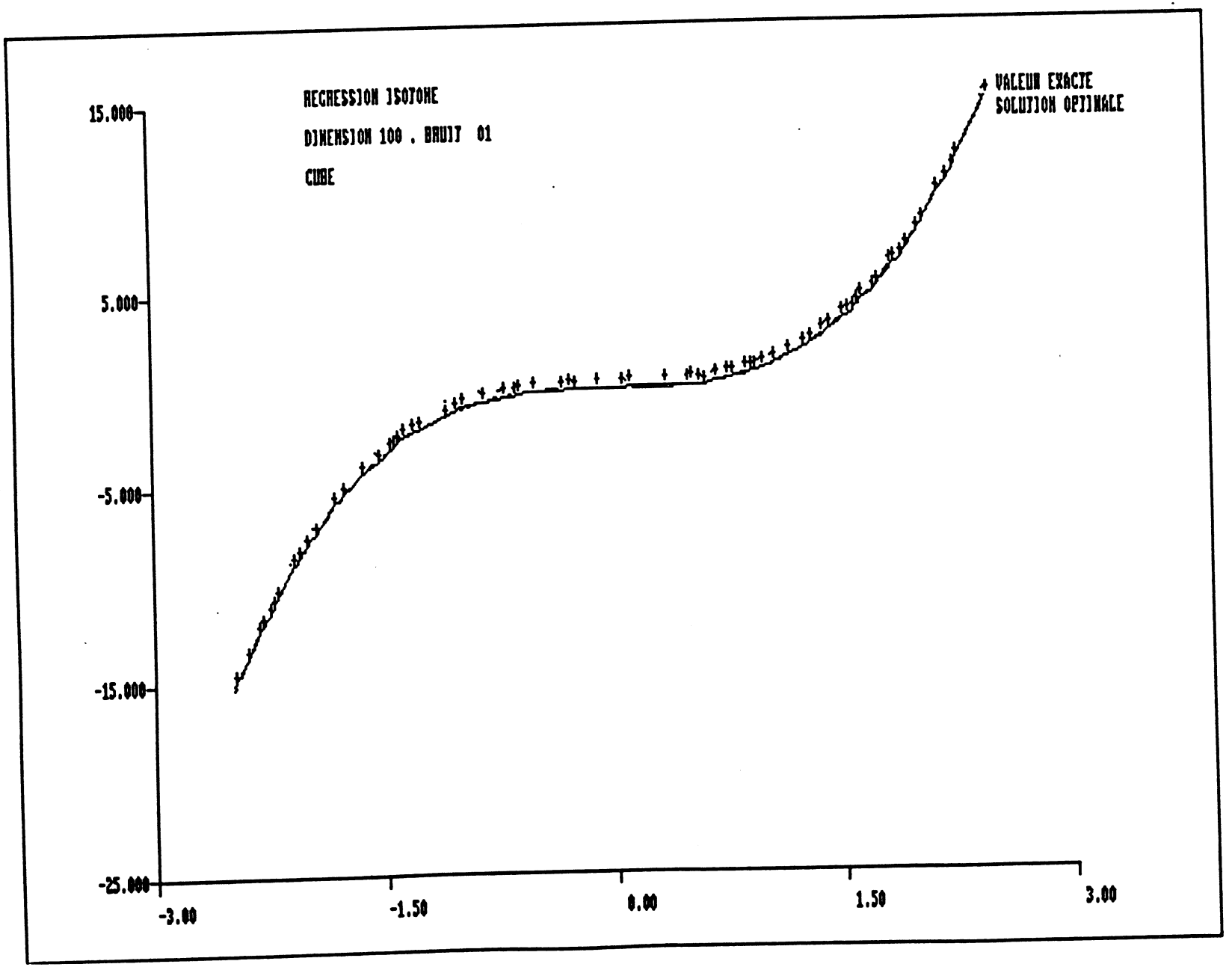

REGRESSION CONCAVE
DIMENSION 50 . BRUIT 10
CUBE
15.000
5.000
-5.000
-15.000
-25.000
-3.00
-2.00
-1.00
0.00
1.00
VALEUR EXACTE
SOLUTION OPTIMALE

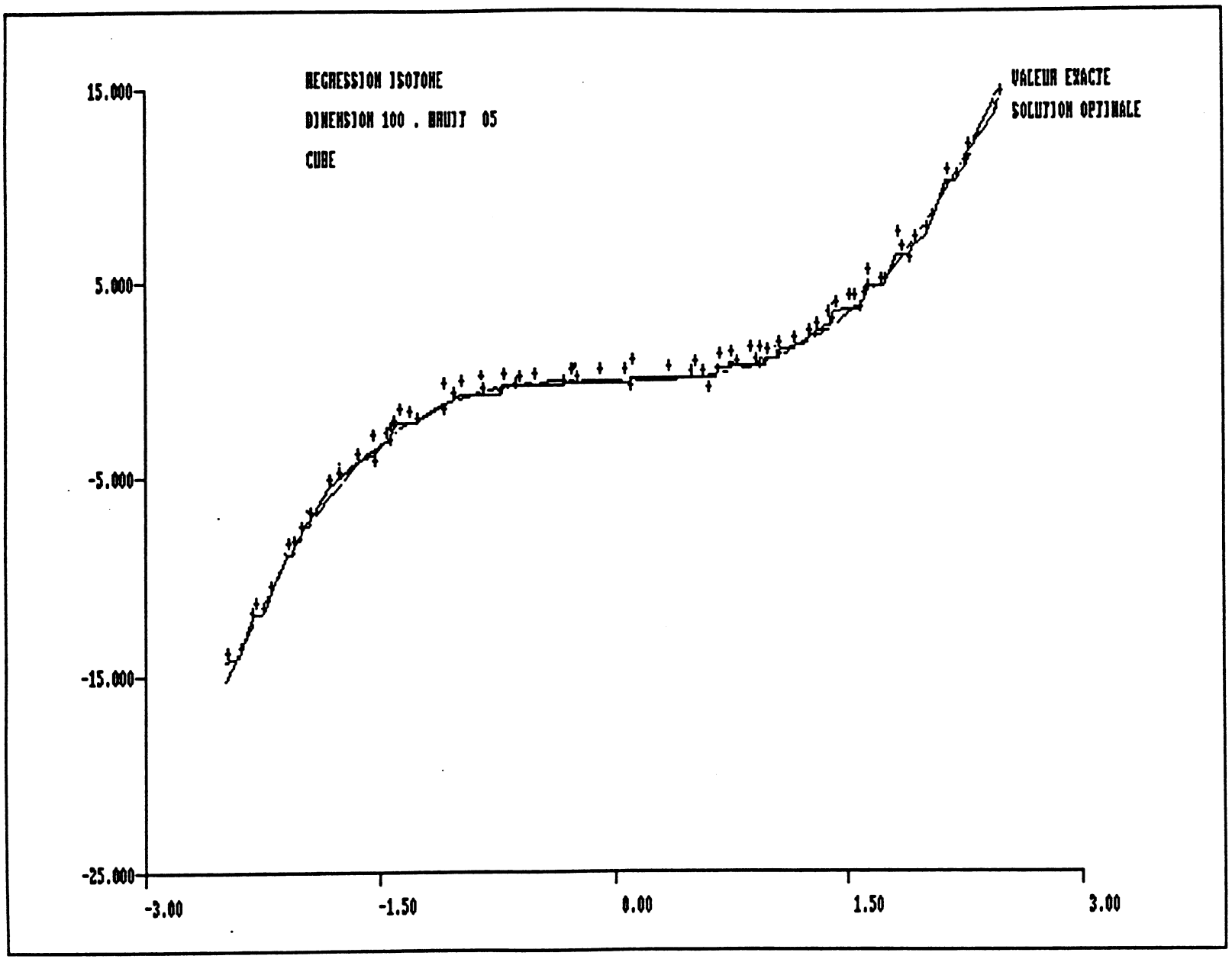

REGRESSION ISOTONE
DIMENSION 50 . BRUIT 01
CUBE
SOLUTION OPTIMALE
VALEUR EXACTE
15.000
5.000
-5.000
-15.000
-25.000
-3.00
-2.00
-1.00
0.00
1.00

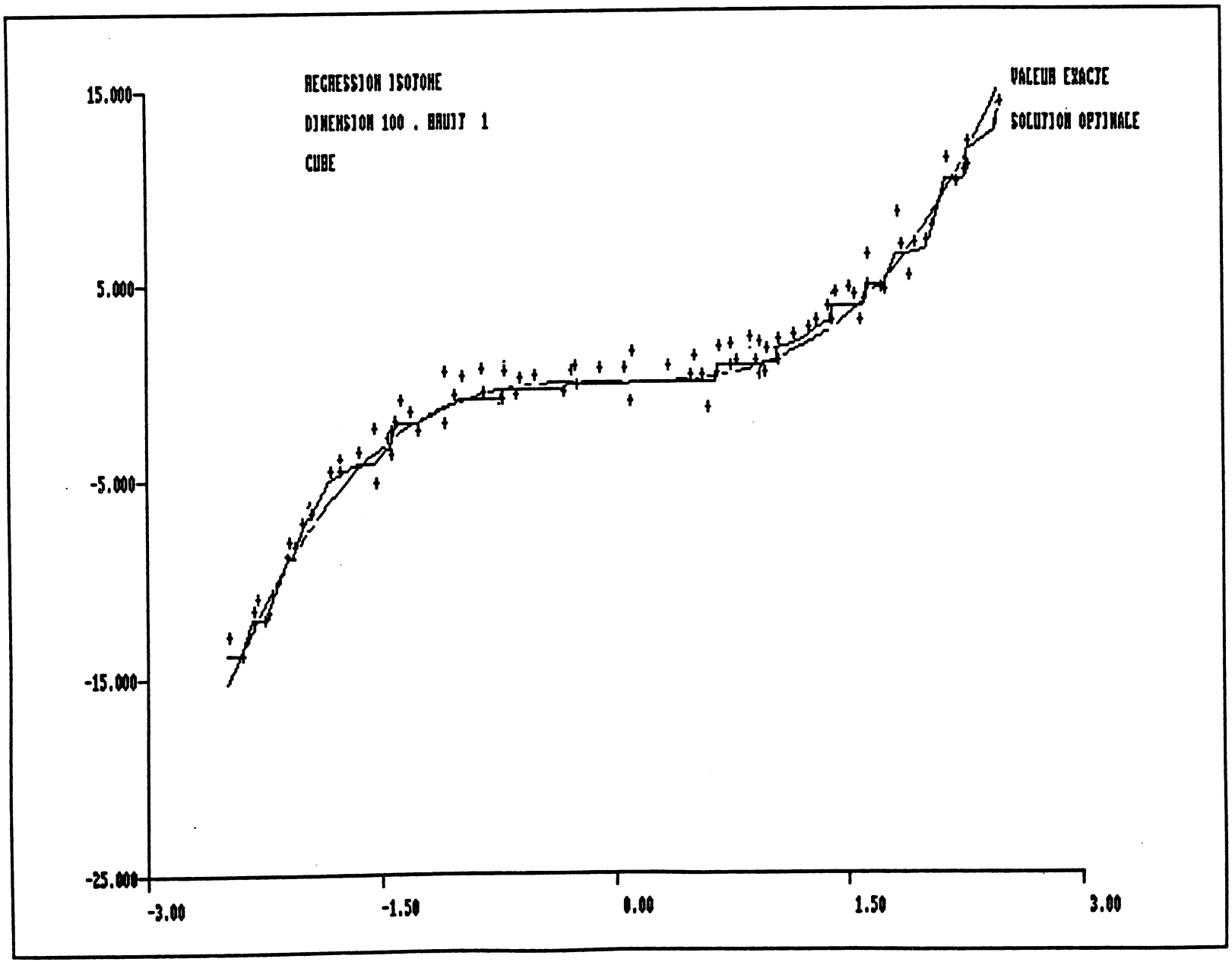

REGRESSION ISOTONE
DIMENSION 50 . BRUIT 05
CUBE
SOLUTION OPTIMALE
VALEUR EXACTE
15.000
5.000
-5.000
-15.000
-25.000
-3.00
-2.00
-1.00
0.00
1.00

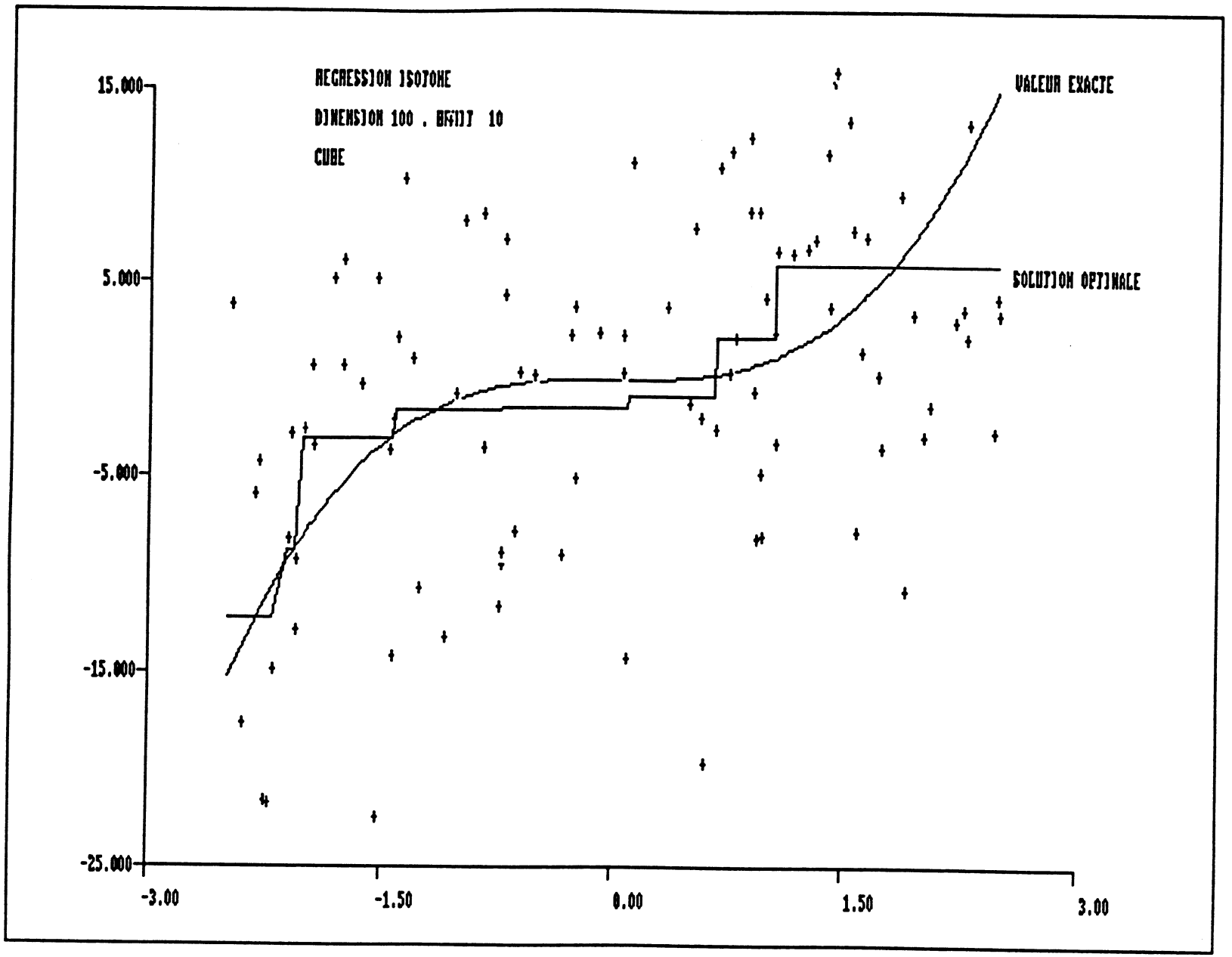

REGRESSION ISOTONE
DIMENSION 50 . BRUIT 1
CUBE
SOLUTION OPTIMALE
VALEUR EXACTE
15.000
5.000
-5.000
-15.000
-25.000
-3.00
-2.00
-1.00
0.00
1.00


REGRESSION ISOTONE
DIMENSION 50 . BRUIT 10
CUBE
SOLUTION OPTIMALE
VALEUR EXACTE
15.000
5.000
-5.000
-15.000
-25.000
-3.00
-2.00
-1.00
0.00
1.00


REGRESSION ISOTONE
DIMENSION 100 . BRUIT 01
CUBE
VALEUR EXACTE
SOLUTION OPTIMALE
15.000
5.000
-5.000
-15.000
-25.000
-3.00
-1.50
0.00
1.50
3.00


REGRESSION ISOTONE
DIMENSION 100 . BRUIT 05
CUBE
VALEUR EXACTE
SOLUTION OPTIMALE
15.000
5.000
-5.000
-15.000
-25.000
-3.00
-1.50
0.00
1.50
3.00


REGRESSION ISOTONE
DIMENSION 100 . BRUIT 1
CUBE
VALEUR EXACTE
SOLUTION OPTIMALE
15.000
5.000
-5.000
-15.000
-25.000
-3.00
-1.50
0.00
1.50
3.00


REGRESSION ISOTONE
DIMENSION 100 . BRUIT 10
CUBE
VALEUR EXACTE
SOLUTION OPTIMALE
15.000
5.000
-5.000
-15.000
-25.000
-3.00
-1.50
0.00
1.50
3.00

AUTORISATION DE SOUTENANCE

DOCTORAT 3ème CYCLE, DOCTORAT INGENIEUR,
DOCTORAT DE L'UNIVERSITE JOSEPH FOURIER - GRENOBLE 1

Vu les dispositions de l'Arrêté du **16 avril 1974,**

Vu les dispositions de l'Arrêté du **5 juillet 1984,**

Vu les rapports de M^r^ ..NGUYEN..VAN HIEN.....

M^r^ ....Robert JANIN.....

M^r^ ....Adnan....YASSINE.....est autorisé(e)
à présenter une thèse en vue de l'obtention du .DOCTORAT..de.....
...l'Université...Joseph FOURIER.....

Grenoble, le 14 JUIN 1989

Le Président de l'Université
Joseph Fourier - Grenoble 1

A. NEMOZ

Résumé :

Cette thèse est consacrée aux études adaptatives et comparatives de certains algorithmes en optimisation, à leurs implémentations effectives et leurs applications concrètes à la résolution de différents problèmes importants de Mathématiques Appliquées :
1) L'algorithme S.G.G.P pour la résolution d'un programme linéaire général.
2) La méthode de pivotage de Lemke, La méthode du gradient conjugué conditionnel, La méthode de l'inverse partiel pour la résolution des programmes quadratiques convexes.
3) Les méthodes d'approximation extérieure et les méthodes de coupes planes et les méthodes de région de confiance pour l'optimisation non convexe (minimisation d'une fonction concave sur un convexe, problèmes de moindres carrés non linéaires).
Les modélisations mathématiques adéquates et les expérimentations numériques comparatives ont été réalisées sur les problèmes suivants:
1) Les problèmes de régression en analyse de données.
2) Minimisation d'une fonction concave sur un convexe, problème non linéaire complémentaire.
3) Les réseaux neuronaux
4) L'écoulement bi et tridimensionnels à géométrie simplement connexe.
5) Les problèmes d'analyse multidimensionnelle des données de dissimilarité (MDS).
6) Minimisation d'une forme quadratique hermitienne sur la sphère euclidienne unité de C^n.

Mots-clés :

programmation linéaire, algorithme S.G.G.P, programmation linéaire et non linéaire complémentaire, programmation quadratique, méthode de pivotage de Lemke, gradient conjugué conditionnel, inverse partiel, régression isotone, régression concave, minimisation d'une fonction concave, méthodes de région de confiance, réseaux neuronaux, écoulements de fluides incompressibles, codage multidimensionnel.