

THESIS / THÈSE

MASTER EN SCIENCES MATHÉMATIQUES

Une approche duale d'un problème à plusieurs niveaux en optimisation de structure

Fontaine, Frédéric; Poisseroux, Jacqueline

Award date:
1987

Awarding institution:
Universite de Namur

[Link to publication](#)

General rights

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

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

Take down policy

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

**Une approche duale d'un problème
à plusieurs niveaux en
optimisation de structure**

Promoteurs :

Professeur NGUYEN VAN HIEN

Docteur JEAN-JACQUES STRODIOT

FONTAINE FREDERIC
POISSEROUX JACQUELINE

Facultés Universitaires Notre-Dame de la Paix

Faculté des Sciences

rue de Bruxelles 61, B-5000 NAMUR

Tel. 081-22.90.61 Telex 59222 facnam-b Telefax 081-23.03.91

Une approche duale d'un
problème à deux niveaux en
optimisation de structures.

FONTAINE FREDERIC

POISSEROUX JACQUELINE

Résumé

Dans ce mémoire, nous développons deux méthodes de linéarisation convexe pour résoudre un problème d'optimisation de structures. Elles consistent à remplacer le problème originel par une suite de sous-problèmes approximatifs. Dans la première méthode la solution de chaque sous-problème est obtenue par décomposition tandis que dans la seconde elle est recherchée par dualité. La seconde méthode a été implémentée et testée sur un certain nombre d'exemples.

Abstract

In this work two convex linearization methods have been developed for solving a problem of structural optimization. They consist in replacing the main problem by a sequence of approximating subproblems. The solution of each subproblem is obtained by decomposition in the former and by using duality in the latter. The second method has been implemented and tested on several test-problems.

Mémoire de licence en Sciences Mathématiques

Juin 1987

Unité d'Optimisation

Promoteurs: Prof. NGUYEN V.H.

Dr. J. J. STRODIOT

*Je dédie ce mémoire à ma
maman*

J. Poisseroux

Nous remercions nos parents de nous avoir permis de mener à bien nos études .

Nous tenons également à remercier nos promoteurs messieurs Nguyen Van Hien et Jean-Jacques Strodiot pour leur disponibilité et leurs conseils .

Ce mémoire fut élaboré grâce aux articles et travaux du docteur J.F.M.Barthelemy du Langley Research Center (NASA) qu'il a eu la gentillesse de nous envoyer .

Merci à E. Hallet et J. Engels pour leurs encouragements et leur aide dans la parite informatique .

PLAN

CHAPITRE 1 : INTRODUCTION

CHAPITRE 2 : PRINCIPE D'OPTIMISATION A PLUSIEURS NIVEAUX

- A) Introduction..... 2.2
- B) Optimisation à plusieurs niveaux
et décomposition..... 2.3

CHAPITRE 3 : POSITION DU PROBLEME

CHAPITRE 4 : LA LINEARISATION CONVEXE

- A) Introduction..... 4.2
- B) Différentes linéarisations..... 4.3
- C) Propriétés de la linéarisation
convexe..... 4.6
- D) Scaling..... 4.8
- E) Introduction à la relaxation..... 4.10
- F) Normalisation d'un sous-problème 4.10

CHAPITRE 5 : OPTIMISATION A UN NIVEAU RESOLUTION PAR DUALITE

- A) Algorithme..... 5.2
- B) Implémentation..... 5.21

CHAPITRE 6 : UNE AUTRE METHODE DE RESOLUTION OPTIMISATION A DEUX NIVEAUX

- A) Principe général..... 6.2
- B) Résolution de $(SP)^{(k)}$ 6.3
- C) Algorithme..... 6.6
- D) Modification de l'algorithme..... 6.8

CHAPITRE 7 : CONCLUSION

CHAPITRE 1 :

INTRODUCTION

Le problème traité dans ce mémoire [Ref.6] consiste à minimiser une fonction sous contraintes d'inégalité et contraintes de bornes sur les variables, toutes les fonctions étant une fois continûment différentiables . La caractéristique de ce problème réside dans le fait que la fonction objectif ne dépend pas d'un sous-ensemble des variables du problème, ce qui confère aux contraintes des statuts différents à savoir *local* si elles dépendent de ce sous-ensemble de variables et *global* si elles n'en dépendent pas . Cette forme particulière découle de la modélisation de problèmes d'ingénieurs comme le " 3-bar truss " .

Nous avons étudié l'article "*An improved multilevel optimisation approach for the design of complex engineering systems* " de J.F. Barthelemy dans lequel le problème originel est remplacé par une suite de sous-problèmes approximants, créés par la linéarisation convexe de la fonction objectif et des contraintes . Chaque sous-problème y est alors résolu par décomposition en sous-sous-problèmes .

Sur base de cet article, nous avons tenté une nouvelle approche . Nous avons conservé le principe des approximations successives mais avons résolu chaque sous-problème par méthode duale . Ce type de résolution s'est avéré aisé grâce à la propriété de séparabilité dont chaque sous-problème bénéficie . Nous avons implémenté l'algorithme ainsi conçu . Nous avons obtenu des résultats de convergence pour les exemples testés .

Voici l'agencement des chapitres de ce mémoire :

Dans le chapitre 2, nous exposons le principe général d'optimisation à plusieurs niveaux ainsi que les simplifications imposées dans l'article de J.F.Barthelemy .

Dans le chapitre 3, nous présentons le problème sous sa forme mathématique ainsi que les notations qui seront utilisées par la suite .

Le chapitre 4 est consacré à l'étude de la linéarisation convexe et de ses principales propriétés . Nous y introduisons le principe de relaxation ainsi que la mise à l'échelle des variables .

Nous réservons le chapitre 5 à l'élaboration de l'algorithme de résolution par dualité . Les résultats numériques que nous avons obtenus clotent ce chapitre .

Le chapitre 6 développe les principaux points de l'article de J.F. Barthelemy et explicite plus particulièrement l'algorithme qui y est proposé .

CHAPITRE 2 :

PRINCIPE D'OPTIMISATION

A

PLUSIEURS NIVEAUX

A)INTRODUCTION

Des systèmes complexes d'ingénieurs comme les avions, les voitures, les bateaux, les centrales électriques sont habituellement décomposables de telle sorte que le tout soit traité comme un assemblage de parties plus petites .

Les ingénieurs ont employé cette technique pour séparer leurs grands modèles en sous-modèles exécutés indépendamment. Récemment , cette approche a été grossie par des méthodes formelles numériques.

De manière schématique, la décomposition peut être présentée comme une pyramide de modules reliés hiérarchiquement, chacun correspondant à un sous-problème du modèle . Ici la procédure globale prend la forme d'une optimisation à plusieurs niveaux dont le but est de satisfaire les contraintes et d'améliorer la performance de tout le système .

B) OPTIMISATION A PLUSIEURS NIVEAUX ET DECOMPOSITION.

L'optimisation à plusieurs niveaux repose sur la décomposition d'un système pour diviser la tâche d'optimisation de ce système en un ensemble de tâches de sous-optimisation et d'une tâche de coordination qui donne la dépendance entre les sous-tâches
[Ref 8].

Nous allons l'expliquer en comparant avec l'optimisation sans décomposition.

1) L'optimisation conventionnelle définit pour tout le système un vecteur de variables du modèle : Z et un ensemble de contraintes d'inégalité : $G(z)$. On n'y fait aucune distinction parmi les éléments de Z et on n'y distingue pas non plus les contraintes G pouvant appartenir à différents sous-systèmes.

En choisissant une mesure de performance du système comme fonction objectif : $F(z)$, on résout donc le problème d'optimisation

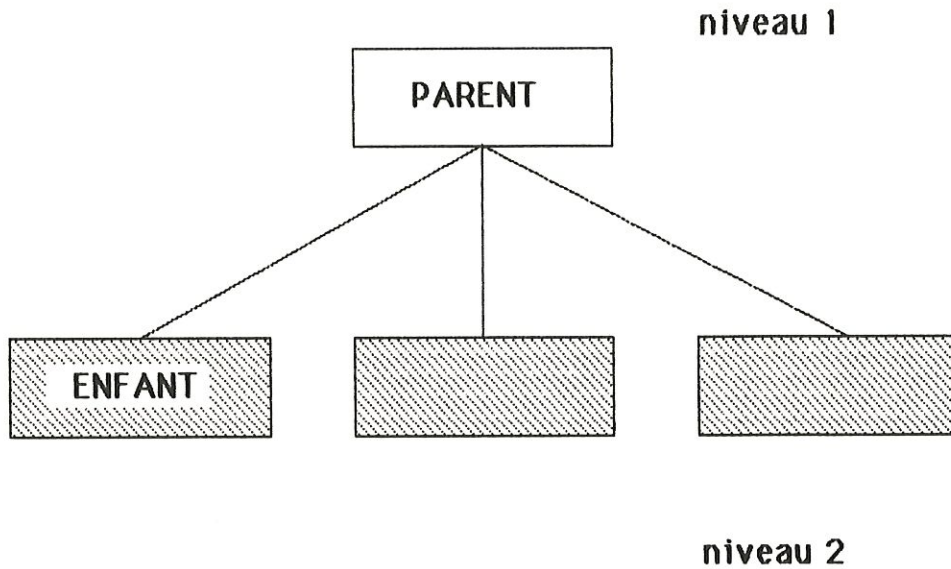
$$\begin{aligned} & \text{MIN } F(z) \\ & \text{sous } G(z) \leq 0 \\ & \quad z^l \leq z \leq z^u \end{aligned}$$

en utilisant une procédure de programmation non linéaire avec un point de départ : z^0

L'information numérique à propos des valeurs de F, G et de leur gradient qui est nécessaire à la méthode de programmation provient d'analyses de modules du système et d'analyses du système assemblé.

Ceci implique que le système peut être décomposé pour l'analyse mais pas dans le but d'optimisation.

2) Dans une optimisation à plusieurs niveaux, le système est décomposé pour l'analyse et l'optimisation .



L'output du parent devient l'input de l'enfant (de tous les enfants du niveau suivant) . Chaque enfant peut à son tour être parent d'autres enfants à niveau plus bas...

Pour chaque sous système, on définit

- > X un sous-ensemble de Z qui représente les variables locales .
- > g un sous-ensemble de G
- > C une fonction objectif locale .

Pour le problème assemblé on définit aussi

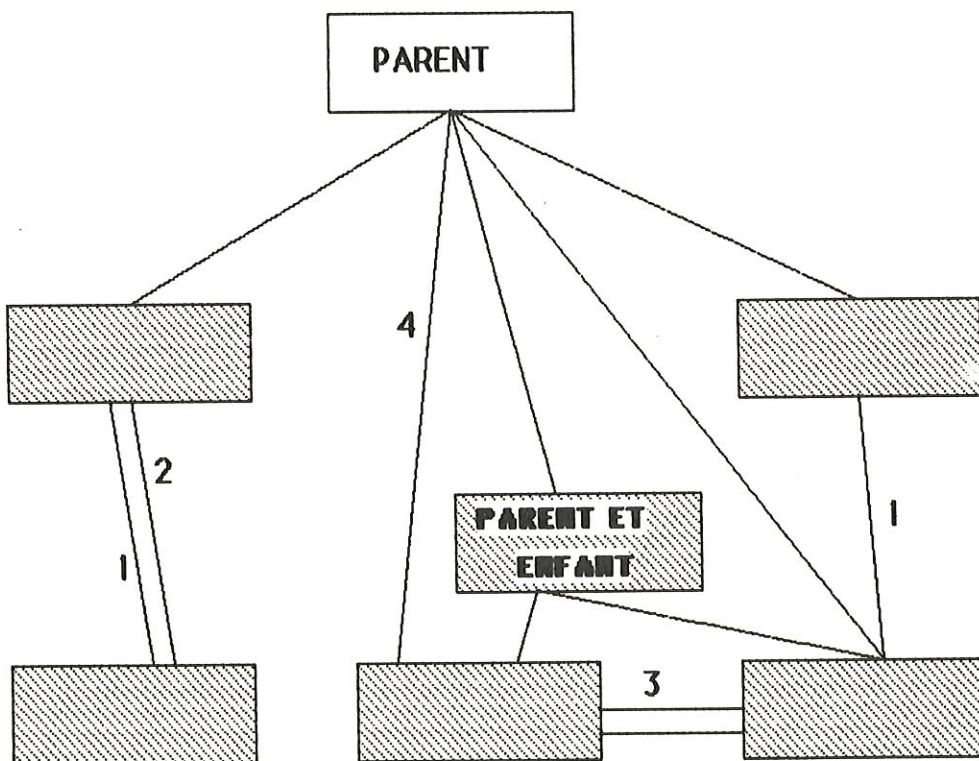
- > $F(Y)$ une fonction objectif
- > $h(Y)$ un sous-ensemble de contraintes où Y représente les variables du modèle encore appelées variables globales .

Il existe de nombreuses méthodes pour optimiser un système décomposé. Nous allons expliciter l'algorithme exposé par

J-F BARTHELEMY

1. Initialisation des variables
2. Analyse des sous-problèmes (parents, enfants)
3. Optimisation de chaque sous-problème
 - a) les sous-systèmes
 - b) le problème assemblé.

Cet algorithme (développé dans le chapitre 6) présuppose que le flot de données dans le système d'analyse soit à sens unique et que chaque enfant ait un seul parent. Ceci est une simplification car les flots peuvent être beaucoup plus compliqués :



On peut donc rencontrer les caractéristiques suivantes :

- 1) Parents multiples pour un enfant
- 2) Output d'un enfant nécessaire comme input à l'analyse du parent
- 3) Output d'un enfant nécessaire comme input à un autre enfant
- 4) Des canaux input/output de plus de un niveau au dessus ou ou en dessous .

CHAPITRE 3 :

POSITION DU PROBLEME

Notations :

ncy : le nombre de composantes du vecteur Y

$ncxi(i)$: le nombre de composantes de X_i

nch : le nombre de contraintes globales

i.e. le nombre de contraintes dépendant uniquement de Y

ncg : le nombre de contraintes locales

i.e. le nombre de contraintes dépendant de X

ou encore le nombre de composantes de X

$ncgi(i)$: le nombre de contraintes locales dépendant de X_i

Hypothèses :

$$\text{Soient } F : \mathbb{R} \longrightarrow \mathbb{R}$$

$$Y \dashrightarrow F(Y)$$

$$\text{où } Y = (y_1 \ y_2 \ \dots \ y_{ncy})^t$$

$$h_j : \mathbb{R}^{ncy} \longrightarrow \mathbb{R}$$

$$Y \dashrightarrow h_j(Y)$$

$$h = (h_1 \ h_2 \ \dots \ h_{nch})^t$$

$$g_{ij} : \mathbb{R}^{ncy} \times \mathbb{R}^{ncxi(i)} \longrightarrow \mathbb{R}$$

$$(Y, X_i) \dashrightarrow g_{ij}(Y, X_i)$$

$$\text{où } X_i = (x_1 \ x_2 \ \dots \ x_{ncxi(i)})^t$$

$$g_i = (g_{i1} \ g_{i2} \ \dots \ g_{incgi(i)})^t$$

$$g = (g_1 \ g_2 \ \dots \ g_{ncg})^t$$

$$\text{où } F, g, h \in C^1$$

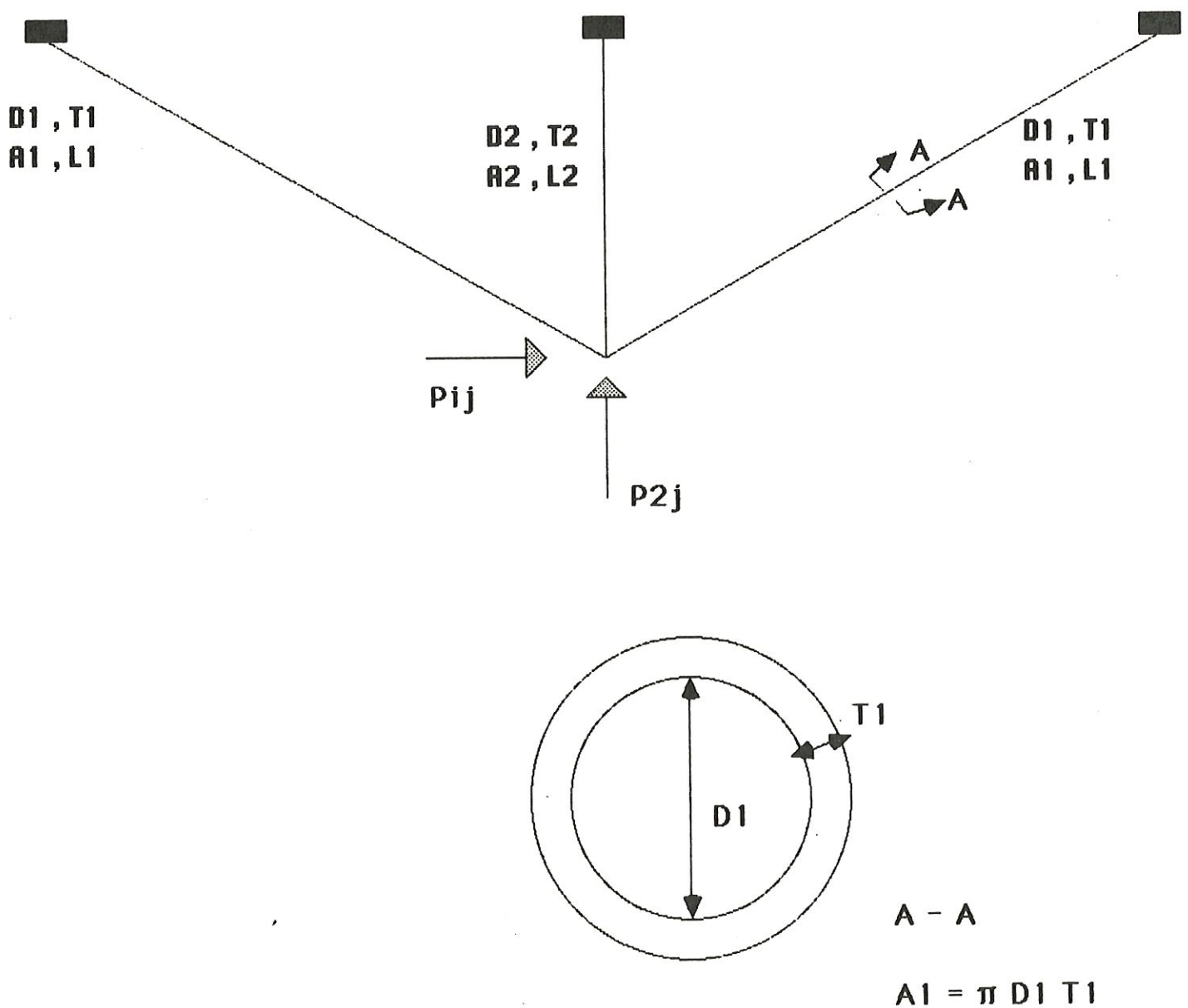
i.e. ces fonctions sont au moins une fois continûment différentiables.

Le caractère particulier du problème, à savoir la différenciation entre les contraintes (et les variables) globales et locales, n'est pas une pure invention gratuite de l'esprit d'un mathématicien. En effet, cela correspond à des problèmes bien réels d'ingénieur.

Nous allons vous donner un exemple type : le " 3-bar truss " ou treillis à trois barres. [Ref 6]

Toutefois, notre but n'étant pas d'expliquer un problème d'ingénieur, nous avons choisi un problème avec des données simplifiées :

Formulation d'un problème de treillis à trois barres avec variables locales et contraintes de comportement



données et notations générales

Longueur des membres 1 et 3 $L_1 = 4. \text{ m}$

Longueur du membre 2 $L_2 = 2. \text{ m}$

Charges $P_{1j} = 2. \times 10^5 \text{ N}$, $P_{2j} = 2. \times 10^5 \text{ N}$ (cas1)

$P_{1j} = 0.$ N , $P_{2j} = 3. \times 10^5 \text{ N}$ (cas2)

Module d'élasticité $E = 2.2 \times 10^{11} \text{ N/m}^2$

Constante de crippling $K = .2$

Rapport maximal épaisseur-diamètre $D = .2$

Diamètre et épaisseur minimaux $d_{\min} = t_{\min} = 1. \times 10^{-3} \text{ m}$

Diamètre et épaisseur maximaux $d_{\max} = t_{\max} = 1. \text{ m}$

Pression maximale $\sigma_{\max} = 1. \times 10^9 \text{ N/m}^2$

position du problème

Trouver la combinaison des variables indépendantes A_1, d_1 , A_2, d_2 qui minimise le volume du treillis tout en satisfaisant aux contraintes de pression, buckling et crippling sous les deux cas de charges et aux contraintes géométriques sur le rapport épaisseur-diamètre et les bornes supérieures et inférieures des variables .

Note : Le problème peut être posé en termes de t_1, d_1 , t_2, d_2 , cependant il ne serait pas décomposable .

volume du treillis

$$V = 2 L_1 A_1 + L_2 A_2 \quad (\text{m}^3)$$

contraintes de pression (au nombre de 12)

forme générale $g = |\sigma_{ij}| / \sigma_{\max} - 1 \leq 0$

où σ_{ij} est la pression sur la barre i ($i=1,2$ ou 3) sous le cas de charge j ($j=1$ ou 2). On peut montrer que :

$$\sigma_{1j} = \frac{P_{1j}}{\sqrt{3} A_1} - \frac{P_{2j}}{A_1 + 4 A_2} \quad (\text{N/m}^2)$$

$$\sigma_{2j} = - \frac{4 P_{2j}}{A_1 + 4 A_2} \quad (\text{N/m}^2)$$

$$\sigma_{3j} = - \frac{P_{1j}}{\sqrt{3} A_1} - \frac{P_{2j}}{A_1 + 4 A_2} \quad (\text{N/m}^2)$$

remarque : g n'étant pas différentiable, il convient donc de le remplacer par deux contraintes différentiables:

$$\sigma_{ij} - \sigma_{\max} \leq 0$$

$$-\sigma_{ij} - \sigma_{\max} \leq 0$$

contraintes de buckling (au nombre de 6)

forme générale $g = -\sigma_{ij} / \sigma_{bi} - 1 \leq 0$

où σ_{ij} sont les pressions du point précédent et σ_{bi} est la pression de buckling sur la barre i et $\pi = 3.14...$

$$\sigma_{bi} = \frac{\pi^2 E d_i^2}{8 L_i^2} \quad \text{N/m}^2$$

contraintes de crippling (au nombre de 6)

forme générale $g = -\sigma_{ij} / \sigma_{ci} - 1 \leq 0$

où σ_{ij} sont les pressions du point précédent et σ_{ci} est la pression de crippling sur la barre i

$$\sigma_{ci} = \frac{K_i E_i A_i}{\pi d_i^2} \quad \text{N/m}^2$$

contraintes du rapport épaisseur-diamètre (au nombre de 2)

$$g = \frac{A_i}{\pi d_i^2 t_{\max}} - 1 \leq 0$$

bornes supérieures sur l'épaisseur (au nombre de 2)

$$g = \frac{A_i}{\pi d_i t_{\max}} - 1 \leq 0$$

bornes inférieures sur l'épaisseur (au nombre de 2)

$$g = 1 - \frac{A_i}{\pi d_i t_{\min}} \leq 0$$

bornes supérieures sur le diamètre (au nombre de 2)

$$g = d_i / d_{\max} - 1 \leq 0$$

bornes inférieures sur le diamètre (au nombre de 2)

$$g = 1 - d_i / d_{\min} \leq 0$$

notes

1) Ce problème contient 20 contraintes d'ingénieur et 8 bornes sur les variables .

2) Les gradients peuvent être facilement évalués analytiquement .

3) Le niveau *global* implique

variables : A_1, A_2
fonction objectif : V
contraintes : pressions

4) Le niveau *local* implique

variables : d_1, d_2
contraintes: buckling,crippling,bornes sur les variables,
rapport épaisseur-diamètre

CHAPITRE 4 :

LA LINEARISATION CONVEXE

A) INTRODUCTION

Pour résoudre des problèmes d'optimisation non-linéaires, deux approches sont possibles .

La première est basée sur des méthodes générales de programmation non-linéaire (méthodes de pénalisation, de directions admissibles...) . Ces méthodes donnent d'excellents résultats de convergence mais le coût de leur application augmente très fort avec la taille des problèmes .

Dans une seconde approche, le problème d'optimisation est transformé en une suite de sous-problèmes approximants ayant une structure algébrique simple . Les contraintes de bornes étant faciles à traiter restent inchangées . La convergence de ces méthodes dépend de la qualité des approximations utilisées . Si l'approximation est mauvaise, le processus peut diverger ou converger après de grandes oscillations . **[Ref 5]**

B) DIFFERENTES LINEARISATIONS

Linéarisation directe

Il s'agit de linéariser la fonction objectif et les contraintes en fonction des variables directes.

Les sous-problèmes obtenus sont strictement linéaires et le "simplex" peut alors s'appliquer. Cependant, la solution du sous-problème se trouve toujours sur un sommet du domaine admissible. Le processus peut soit diverger, soit encore osciller entre deux sommets non optimaux.

Linéarisation inverse

Il s'agit de linéariser les contraintes en fonction des variables inverses ($1/y$) et la fonction objectif en fonction des variables directes (y).

Pour que cette opération soit applicable il faut que les variables (y) soient strictement positives, une simple translation peut l'assurer. Ce problème, après changement de variable $z=1/y$, n'est pas nécessairement convexe et la résolution du dual peut poser des difficultés, c'est pourquoi il a semblé nécessaire d'obtenir des sous-problèmes convexes.

Linéarisation mixte

Il s'agit de combiner les deux méthodes précitées en linéarisant les fonctions par rapport aux variables directes pour un certain groupe de variables et par rapport aux variables inverses pour le reste.

Ce qui donne pour une fonction $h : \mathbb{R}^n \longrightarrow \mathbb{R}$
 $x \longmapsto h(x)$
où $x=(x_1, x_2, \dots, x_n)^t$

linéarisée en x° :

$$h \sim (x; x^\circ) = \sum_{J_1} \frac{\partial h}{\partial x_{ij} x^\circ} (x_i - x_i^\circ) + \sum_{J_2} \frac{\partial h}{\partial x_{ij} x^\circ} (x_i - x_i^\circ) \frac{x_i^\circ}{x_i} + h(x^\circ)$$

où J_1 (J_2) est l'ensemble des indices des variables pour lesquelles la linéarisation directe (inverse) est utilisée.

Pour une fonction $g : \mathbb{R}^n \times \mathbb{R}^m \longrightarrow \mathbb{R}$
 $(y, x) \longmapsto g(y, x)$

où

$$y = (y_1, y_2, \dots, y_n)^t$$

$$x = (x_1, x_2, \dots, x_m)^t$$

linéarisée en (y°, x°) :

$$g^{\sim}(y, x ; y^\circ, x^\circ) = \sum_{J1} \frac{\partial g}{\partial x_i | y^\circ, x^\circ} (x_i - x_i^\circ) + \sum_{J2} \frac{\partial g}{\partial x_i | y^\circ, x^\circ} (x_i - x_i^\circ) \frac{x_i^\circ}{x_i}$$

$$+ \sum_{K1} \frac{\partial g}{\partial y_i | y^\circ, x^\circ} (y_i - y_i^\circ) + \sum_{K2} \frac{\partial g}{\partial y_i | y^\circ, x^\circ} (y_i - y_i^\circ) \frac{y_i^\circ}{y_i}$$

où J1 (J2) est l'ensemble des indices des variables x (y) pour lesquelles la linéarisation directe (inverse) est utilisée et où K1 (K2) est l'ensemble des indices des variables x (y) pour lesquelles la linéarisation directe (inverse) est utilisée.

Les calculs effectués pour obtenir ces linéarisations se trouvent en ANNEXE A4.1.

En choisissant pour la fonction h :

$$J1 = \left\{ i \mid \frac{\partial h}{\partial x_i | x^\circ} \geq 0 \right\}$$

$$J2 = \left\{ i \mid \frac{\partial h}{\partial x_i | x^\circ} \leq 0 \right\}$$

Et en choisissant pour la fonction g :

$$J1 = \left\{ i \mid \frac{\partial g}{\partial x_i} \Big|_{y^0, x^0} \geq 0 \right\}$$

$$J2 = \left\{ i \mid \frac{\partial g}{\partial x_i} \Big|_{y^0, x^0} \leq 0 \right\}$$

$$K1 = \left\{ i \mid \frac{\partial g}{\partial y_i} \Big|_{y^0, x^0} \geq 0 \right\}$$

$$K2 = \left\{ i \mid \frac{\partial g}{\partial y_i} \Big|_{y^0, x^0} \leq 0 \right\}$$

on obtient ainsi $\tilde{h}(x, x^0)$ et $\tilde{g}(y, x, y^0, x^0)$ deux fonctions convexes (ANNEXE A4.2) . Cette linéarisation mixte particulière sera appelée **linéarisation convexe** .

ILLUSTRATION DE LA LINEARISATION CONVEXE

Soit la fonction $g(x) = x^2 - 2x$.

Dans la figure **[FIG A]** nous avons représenté la linéarisation de cette fonction au point $x=2$. La fonction approximante est de la forme

$$\tilde{g}(x; 2) = 2x - 4.$$

La figure **[FIG B]** représente la linéarisation convexe de $g(x)$ au point $x=0.5$. La fonction approximante est alors de la forme

$$\tilde{g}(x; 0.5) = .25/x - 5/4$$

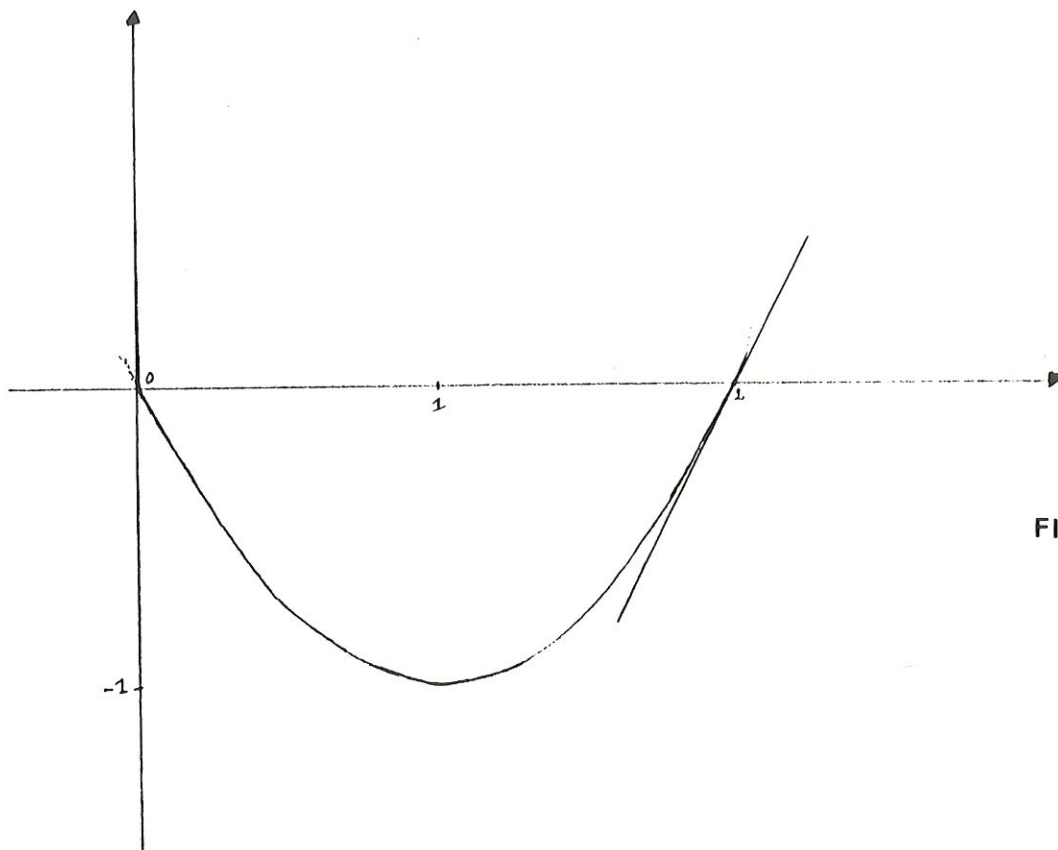


FIG A

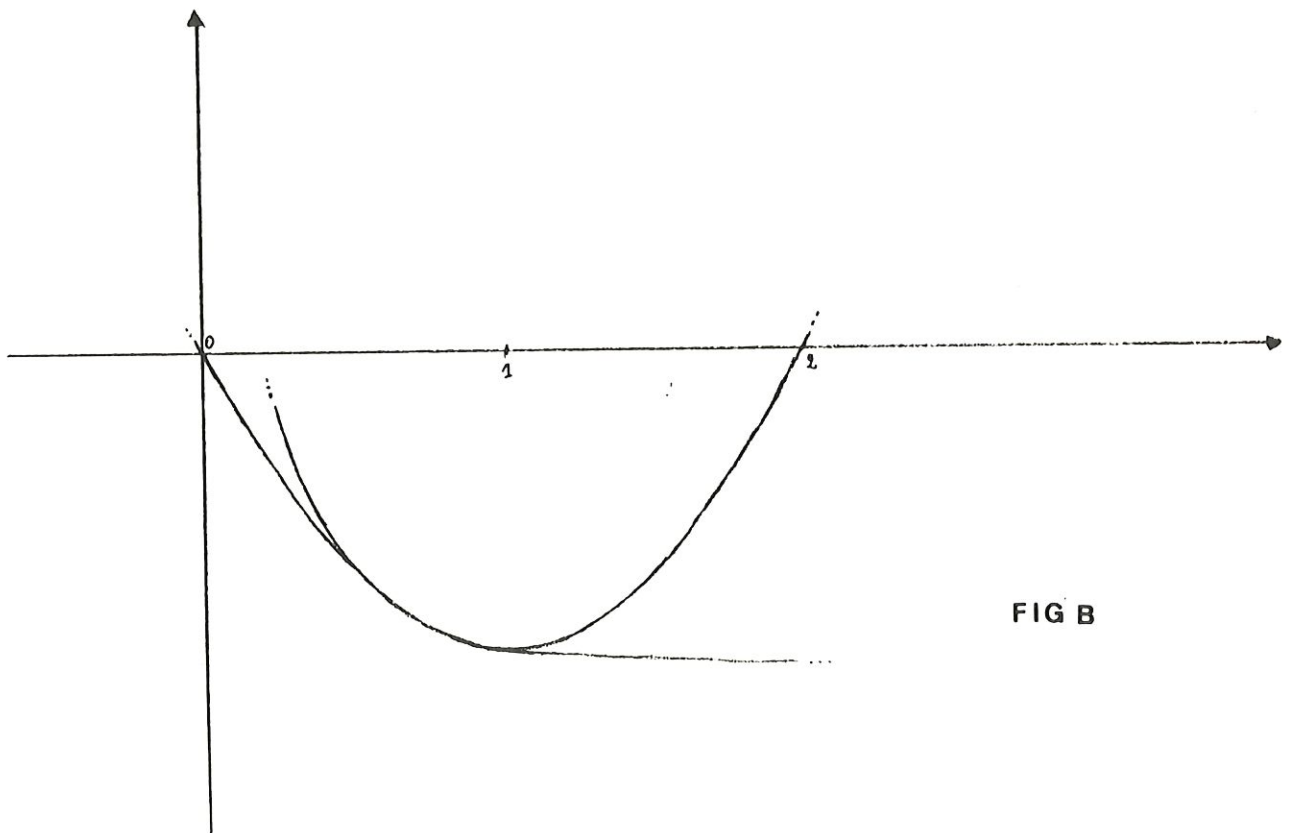


FIG B

C) PROPRIETES DE LA LINEARISATION CONVEXE

Un sous-problème linéarisé en (y°, x°) , noté $(SP^\sim)^\circ$, aura donc la forme explicite suivante :

$$\text{Min } \sum_{i+} \frac{\partial f}{\partial y_i | x^\circ} (y_i - y_i^\circ) + \sum_{i-} \frac{\partial f}{\partial y_i | x^\circ} (y_i - y_i^\circ) \frac{y_i^\circ}{y_i} + f(y^\circ)$$

s.c

$$\sum_{i+} \frac{\partial h_j}{\partial y_i | x^\circ} (y_i - y_i^\circ) + \sum_{i-} \frac{\partial h_j}{\partial y_i | x^\circ} (y_i - y_i^\circ) \frac{y_i^\circ}{y_i} + h_j(y^\circ) \leq 0$$

(j=1..nch)

$$\sum_{k+} \frac{\partial g_{ij}}{\partial x_{ik} | y^\circ, x^\circ} (x_{ik} - x_{ik}^\circ) + \sum_{k-} \frac{\partial g_{ij}}{\partial x_{ik} | y^\circ, x^\circ} (x_{ik} - x_{ik}^\circ) \frac{x_{ik}^\circ}{x_{ik}}$$

$$+ \sum_{t+} \frac{\partial g_{ij}}{\partial y_t | y^\circ, x^\circ} (y_t - y_t^\circ) + \sum_{t-} \frac{\partial g_{ij}}{\partial y_t | y^\circ, x^\circ} (y_t - y_t^\circ) \frac{y_t^\circ}{y_t}$$

$$+ g(y^\circ, x^\circ) \leq 0$$

$$(i=1..ncg, j=1..ncgi(i))$$

$$0 < y_t^l \leq y_t \leq y_t^u$$

$$(t=1..ncy)$$

$$0 < x_{ik}^l \leq x_{ik} \leq x_{ik}^u$$

$$(i=1..ncg, k=1..ncxi(i))$$

où

pour chaque fonction f, h_j

la somme indiquée $i+$ (resp $i-$)

est la somme sur les indices i tels que la dérivée par rapport à la variable y_i au point y° est ≥ 0 (resp ≤ 0)

pour chaque fonction g_{ij}

la somme indicée t^+ (resp t^-)

est la somme sur les indices t tels que la dérivée par rapport à la variable y_t au point y° est ≥ 0 (resp ≤ 0)

tandis que la somme indicée k^+ (resp k^-)

est la somme sur les indices k tels que la dérivée par rapport à la variable x_{ik} au point x° est ≥ 0 (resp ≤ 0).

Ces groupes d'indices intervenant dans les sommes indicées + (resp -) peuvent donc être différents pour chaque fonction. Ces notations seront employées lors de chaque linéarisation.

Propriété 1 : Convexité

Comme démontré en ANNEXE A4.2, chaque sous-problème linéarisé reste convexe. Cette propriété est intéressante car tout point de Kuhn et Tucker est minimum global de ce sous-problème et réciproquement.

Propriété 2 : Séparabilité

Cette propriété s'avèrera essentielle lors de la résolution du problème dual.

Définition : Un problème du type

$$\min f(x)$$

$$\text{sc } h(x) \leq 0$$

où f, h sont des fonctions de \mathbb{R}^N à valeurs réelles

est dit *séparable* s'il peut se mettre sous la forme suivante:

$$\min \sum_i f_i(x_i)$$

$$\text{sc } \sum_i h_i(x_i) \leq 0$$

où f_i, h_i sont des fonctions de \mathbb{R} dans \mathbb{R} et

$$x = (x_1, \dots, x_N)^t$$

Ce facteur est un contrôle de la qualité de l'approximation.

Parmi toutes les linéarisations mixtes possibles, la linéarisation convexe est la plus conservative. Ceci signifie qu'elle rarefie le fait que le domaine approximant soit trop grand par rapport au domaine initial et donc elle rarefie le fait que le point optimal du sous-problème n'est plus admissible pour le domaine de départ.

Il faut toutefois remarquer qu'il existe encore des cas où le domaine obtenu par la linéarisation convexe n'est pas totalement inclus dans le domaine initial.

D) SCALING

En vue d'améliorer l'efficacité et la fiabilité du procédé d'optimisation numérique, nous allons mettre les variables à l'échelle. Une formulation pratique d'un problème est telle que toutes les composantes des fonctions après scaling aient le même ordre de grandeur. **[Ref 1]**

Une forme simple de scaling est de normaliser chaque variable du problème. Dans ce cas la matrice de scaling sera

$$D = \begin{bmatrix} \frac{1}{|x_1|} & 0 \dots & 0 \\ 0 & \frac{1}{|x_2|} & 0 \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 \dots & & & \frac{1}{|x_n|} \end{bmatrix}$$

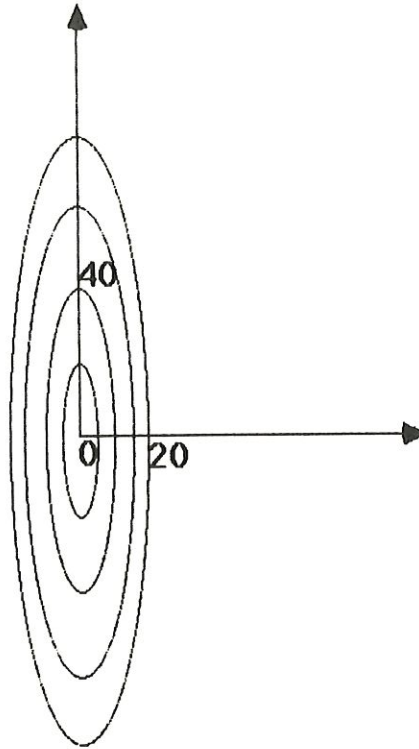
Cette matrice a pour effet de mettre chaque variable sur la même base. Notons que toutes les matrices de scaling sont des matrices diagonales. Remarquons aussi que pour notre problème, les valeurs absolues peuvent être omises car toutes les variables sont strictement positives.

Voici un exemple d'effet du scaling :

Soit la fonction $F(x_1, x_2) = (x_1/5)^2 + (x_2/20)^2$

Nous avons tracé en (Fig 1) les courbes de niveau de cette fonction qui sont une série d'ellipses .

[fig 1]



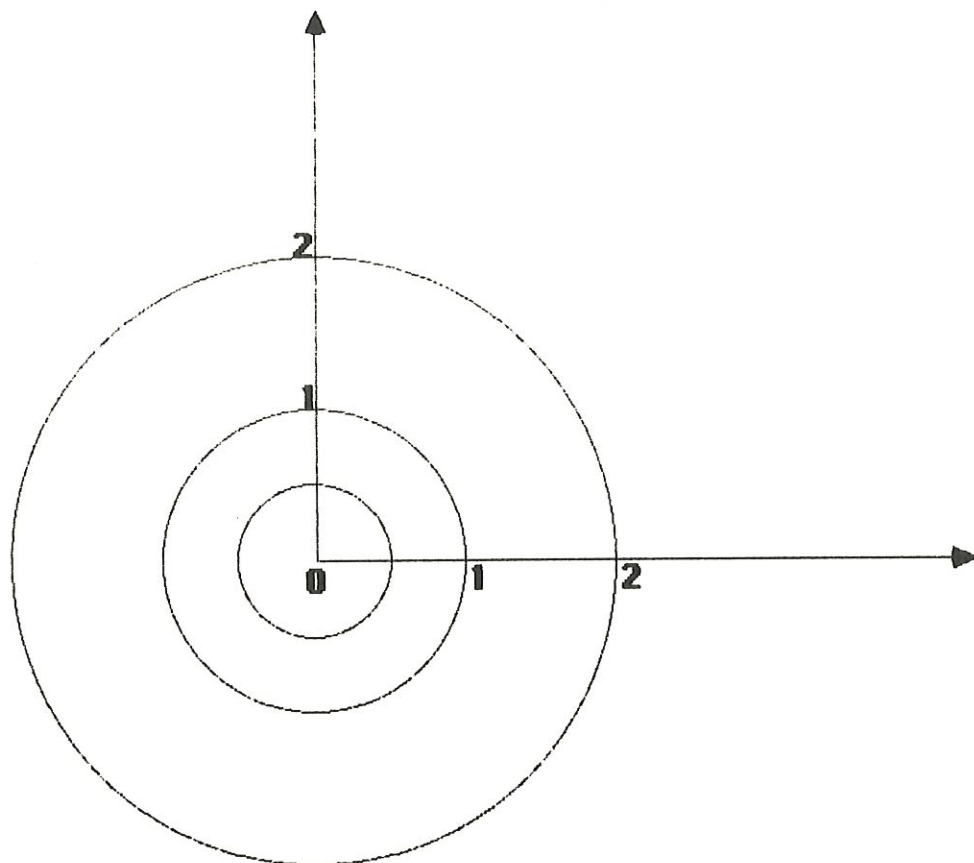
Courbes de niveau de $F(x_1, x_2) = (x_1/5)^2 + (x_2/20)^2$

Posons maintenant $x_1 = 5 x_1' / \sqrt{2}$
 $x_2 = 20 x_2' / \sqrt{2}$

La fonction après scaling devient $F(x_1', x_2') = (0.5)(x_1'^2 + x_2'^2)$

La figure (Fig 2) nous montre que les courbes de niveau sont maintenant des cercles . Pour obtenir le minimum x^* du problème original, on n'a plus qu'à refaire le changement d'échelle inverse et la valeur de la fonction reste la même avec ou sans scaling

$$F(x_1, x_2) = F(x_1', x_2')$$



[fig 2]

Courbes de niveau de $F(x_1', x_2')$

E) INTRODUCTION A LA RELAXATION

Remarquons qu'il peut être nécessaire de démarrer la méthode en un point non admissible. Puisque l'approximation utilisée est conservative, partir d'un tel point peut réduire la taille du domaine admissible et il est possible que l'approximation initiale n'ait pas de solution admissible si elle est générée en un point éloigné du domaine admissible. Dans ce cas, il est nécessaire d'employer une technique de relaxation.

F) NORMALISATION D'UN SOUS-PROBLEME

Dans le but de simplifier les calculs, nous allons normaliser le sous-problème (SP^{*}) [(y^{*}, x^{*})

Pour cela, effectuons le changement de variables

$$y_i' = \frac{y_i}{y_i^*} \quad (i=1..ncy)$$

$$x_{it}' = \frac{x_{it}}{x_{it}^*} \quad (i=1..ncg, j=1..ncxi(i))$$

Par la formule de dérivation composée nous obtenons

$$\frac{\partial f}{\partial y_i'} = \frac{\partial f}{\partial y_i} \frac{\partial y_i}{\partial y_i'} = \frac{\partial f}{\partial y_i} y_i^*$$

$$\frac{\partial h_j}{\partial y_i'} = \frac{\partial h_j}{\partial y_i} \frac{\partial y_i}{\partial y_i'} = \frac{\partial h_j}{\partial y_i} y_i^*$$

$$\frac{\partial g_{ij}}{\partial y_i'} = \frac{\partial g_{ij}}{\partial y_i} \frac{\partial y_i}{\partial y_i'} = \frac{\partial g_{ij}}{\partial y_i} y_i^*$$

$$\frac{\partial g_{ij}}{\partial x_{it}'} = \frac{\partial g_{ij}}{\partial x_{it}} \frac{\partial x_{it}}{\partial x_{it}'} = \frac{\partial g_{ij}}{\partial x_{it}} x_{it}^*$$

d'où

$$\frac{\partial f}{\partial y_i} \Big|_{(y_i=1)_{i=1..ncy}} = \frac{\partial f}{\partial y_i} \Big|_{(y_i=y_i^\circ)_{i=1..ncy}} \quad y_i^\circ$$

que l'on notera

$$\frac{\partial f}{\partial y_i} \Big|_1 = \frac{\partial f}{\partial y_i} \Big|_{y_i^\circ} \quad y_i^\circ$$

Note : Ce changement de variable est utilisé pour obtenir le sous-problème (linéarisé convexe) explicite du chapitre 5 .

CHAPITRE 5 :

OPTIMISATION A UN NIVEAU

RESOLUTION PAR DUALITE

5.1 ALGORITHME

5.1.0 Principe de l'algorithme

1) INITIALISATION

$$X = X^{\circ}$$

$$Y = Y^{\circ}$$

$$\text{iter} = 0$$

2) CRITERE D'ARRET

Si point de Kuhn et Tucker : STOP

Sinon : aller en 3)

3) ETAPE PRINCIPALE

3.1) Création d'un sous-problème linéarisé convexe en X,Y

3.2) Résolution de ce sous-problème par méthode duale
Solution : X^*, Y^*

3.3) $X = X^*$
 $Y = Y^*$
 $\text{iter} = \text{iter} + 1$

3.4) Retour en 2)

L'ORGANIGRAMME se trouve en page 5.26

5.1.1 Routine demandée à l'utilisateur

Nous demandons une routine qui fournit les valeurs de la fonction objectif et des contraintes en un point fixé ainsi que les gradients évalués en ce point . [cfr page 5.22]

5.1.2 Création d'un sous-problème linéarisé convexe

Le sous-problème linéarisé convexe s'écrit :

$$\begin{aligned} \text{Minimiser } f(Y^{(k)}) &+ \sum_{p^-} \frac{\partial f}{\partial y'_p | y'=1} - \sum_{p^+} \frac{\partial f}{\partial y'_p | y'=1} \\ &+ \sum_{p^+} \frac{\partial f}{\partial y'_p | y'=1} y'_p - \sum_{p^-} \frac{\partial f}{\partial y'_p | y'=1} \frac{1}{y'_p} \end{aligned}$$

$$\begin{aligned} \text{Sous } h_i(Y^{(k)}) &+ \sum_{p^-} \frac{\partial h_i}{\partial y'_p | y'=1} - \sum_{p^+} \frac{\partial h_i}{\partial y'_p | y'=1} \\ &+ \sum_{p^+} \frac{\partial h_i}{\partial y'_p | y'=1} y'_p - \sum_{p^-} \frac{\partial h_i}{\partial y'_p | y'=1} \frac{1}{y'_p} \leq 0 \\ &\quad (i=1..nch) \end{aligned}$$

$$\begin{aligned} g_{ij}(Y^{(k)}) &+ \sum_{p^-} \frac{\partial g_{ij}}{\partial y'_p | y'=1} - \sum_{p^+} \frac{\partial g_{ij}}{\partial y'_p | y'=1} \\ &+ \sum_{p^+} \frac{\partial g_{ij}}{\partial y'_p | y'=1} y'_p - \sum_{p^-} \frac{\partial g_{ij}}{\partial y'_p | y'=1} \frac{1}{y'_p} \\ &+ \sum_{p^-} \frac{\partial g_{ij}}{\partial x'_{ip} | x'_i=1} - \sum_{p^+} \frac{\partial g_{ij}}{\partial x'_{ip} | x'_i=1} \\ &+ \sum_{p^+} \frac{\partial g_{ij}}{\partial x'_{ip} | x'_i=1} x'_{ip} - \sum_{p^-} \frac{\partial g_{ij}}{\partial x'_{ip} | x'_i=1} \frac{1}{x'_{ip}} \leq 0 \end{aligned}$$

$$(i = 1..ncg, j = 1..ncgi(i))$$

$$Y_p^l \leq Y_p \leq Y_p^u \quad (p = 1..ncy)$$

$$X'_{ip} \leq X'_{ip} \leq X'^u_{ip} \quad (i = 1..ncg, p = 1..ncxi(i))$$

où

$$y'_i = \frac{y_i}{y_i^{(k)}}$$

$$Y' = (y'_1 \dots y'_{ncy})^t$$

$$Y' = 1 \Leftrightarrow y'_i = 1 \quad (i=1..ncy)$$

$$x'_{ij} = \frac{x_{ij}}{x_{ij}^{(k)}}$$

$$x'_i = (x'_{i1} \dots x'_{incxi(i)})^t$$

$$X' = (x'_1 \dots x'_{ncg})^t$$

$$x'_i = 1 \Leftrightarrow x'_{ij} = 1 \quad (j=1..ncxi(i))$$

Remarque : Dorénavant, pour des raisons de facilité de notation nous omettrons les " ' " en suscrit et nous travaillerons toujours avec ces notations .

Pour l'implémentation de ce sous-problème, nous n'avons besoin que des valeurs fournies par l'utilisateur :

$$\begin{aligned} 1) & f(Y^{(k)}) \\ & h_i(Y^{(k)}) \quad (i=1..nch) \\ & g_{ij}(Y^{(k)}, X_i^{(k)}) \quad (i=1..ncg \quad j=1..ncgi(i)) \end{aligned}$$

$$\begin{aligned} 2) & \nabla_Y f(Y^{(k)}) \\ & \nabla_Y h_i(Y^{(k)}) \quad (i=1..nch) \\ & \nabla_Y g_{ij}(Y^{(k)}, X_i^{(k)}) \\ & \nabla_{X_i} g_{ij}(Y^{(k)}, X_i^{(k)}) \quad ((i=1..ncg \quad j=1..ncgi(i)) \end{aligned}$$

En effet, par le changement de variable expliqué au chapitre 4 ("scaling"), nous avons établi que :

$$\frac{\partial f}{\partial y'_p | y'=1} = \frac{\partial f}{\partial y_p | y=y^{(k)}} y_p^{(k)}$$

Ainsi nous calculons les termes constants et les coefficients de la linéarisation .

3) les bornes des variables X,Y .

Pour trouver les bornes des variables X',Y' , il suffit d'appliquer le changement de variables du chapitre 4 ("scaling") aux bornes de X,Y .

5.1.3 Relaxation d'un sous-problème linéarisé en $Y^{(k)}, X_i^{(k)}$

Lorsque le domaine admissible d'un sous-problème linéarisé est vide, pour obtenir à l'itération suivante un point "le plus admissible possible" , chaque problème est modifié par l'introduction de variables artificielles - une par contrainte primale - de telle sorte que le sous-problème devient :

(SP~)(k)

$$\text{Min } f^{\sim}(Y; Y^{(k)}) + \sum_{i=1..nch} (d_i z_i + d_i z_i^2) \\ + \sum_{i=1..ncg} \sum_{j=1..ncgi(i)} (d_{ij} z_{ij} + d_{ij} z_{ij}^2)$$

$$\text{Sous } h_j^{\sim}(Y; Y^{(k)}) - z_j \leq 0 \quad (j=1..nch) \\ g_{ij}^{\sim}(Y, X_j; Y^{(k)}, X_j^{(k)}) - z_{ij} \leq 0 \quad (i=1..ncg \quad j=1..ncgi(i))$$

$$Y^l \leq Y \leq Y^u$$

$$X^l \leq X \leq X^u$$

$$z_i \geq 0 \quad (i=1..nch)$$

$$z_{ij} \geq 0 \quad (i=1..ncg \quad j=1..ncgi(i))$$

avec $d_{i(j)} > 0$ tq chaque $d_{i(j)}$ doit être un nombre relativement grand .

De toute évidence, ce problème admet toujours une solution admissible (pour tout Y, X il est possible de choisir z_i et z_{ij} de telle sorte que les contraintes deviennent satisfaites).

Il est évident que si les coefficients d_i et d_{ij} sont suffisamment grands, alors toutes les variables artificielles z_i et z_{ij} deviendront automatiquement nulles dans la solution optimale de $(SP^{\sim})^{(k)}$, pourvu que le problème $(SP)^{(k)}$ soit admissible.

D'autre part, si le domaine de $(SP)^{(k)}$ est vide, alors certains z_i et/ou z_{ij} seront strictement positifs dans la solution optimale de $(SP^{\sim})^{(k)}$.

Cependant, à cause du choix de "grands" d_i et d_{ij} les variables z_i et z_{ij} ne prendront jamais des valeurs excessives ; c'est dans ce sens que l'on dit que la solution X', Y' sera la plus admissible possible.

Ceci n'entrave en aucun cas la méthode de résolution par dualité.

5.1.4 Utilisation de E04KBF (library NAG) pour la minimisation de la fonction duale

5.1.4.1 La routine E04KBF

A. BUT

Il s'agit d'un algorithme Quasi-Newton pour trouver le minimum d'une fonction de plusieurs variables sous contraintes de bornes inférieures.

Les dérivées premières sont requises.

La routine est conçue pour des fonctions deux fois continûment différentiables.

B. EMPLOI

Nous utiliserons cette routine pour trouver le maximum de la fonction duale. Nous allons donc chercher le minimum de (- la fonction duale). Pour ce faire, nous passerons à la routine le gradient de $(-1) \times$ la fonction duale et la valeur de $(-1) \times$ la fonction duale.

C. NOTRE CHOIX DES PARAMETRES D'ENTREE

E04KBF (N,Funct,Monit,Iprint,Locsch,Intype,Minlin,Maxcal,Eta,
Xtol,Stepmx,Fest,Ibound,BI,Bu,X,Hesl,Lh,Hesd,Istate,F,
G,Iw,Liw,W,Lw,Ifail)

- N : nombre de contraintes primales .
- Funct : routine qui calcule les valeurs de (-1) *fonction duale et son gradient (cfr D).
- Monit : routine d'impression des principaux résultats (solution , valeur , gradients , itération)
- Iprint : laissé au choix de l'utilisateur
0 pour n'imprimer que la dernière itération
1 pour imprimer à chaque itération
- Locsch : true. pour que l'on effectue une recherche linéaire .
- Intype : 0 car on ne fournit à la routine aucune information sur le hessien de la fonction duale .
- Minlin : E04LBS car nous avons choisi d'effectuer une recherche linéaire qui utilise à la fois la valeur de la fonction et ses dérivées premières .
- Maxcal : $200*N$ comme conseillé .
- Eta : 0.9 (recommandé lors de l'emploi de la routine E04LBS) .
- Xtol : précision de la solution duale
Xtol est fournie par l'utilisateur .
- Stepmx: 100000. comme suggéré par la documentation de E04KBF .
- Fest : sous-estimation du minimum de (-1) *la fonction duale (choix de Fest en ANNEXE A.5.3) .
- Ibound : 2 car les seules bornes sont la non-négativité des variables duales .
- BI,Bu : ne sont pas initialisés car Ibound=2 .
- X : valeur initiale des multiplicateurs de Lagrange .
- Hesl,Hesd: ne sont pas initialisés car Intype=0 .
- Lh : $(N*(N-1)/2) = \max (N*(N-1)/2, 1)$ car étant donnée la structure du problème initial : il existe toujours deux contraintes au moins et donc au moins deux multiplicateurs de Lagrange .
Dans ce cas : $(N*(N-1)/2) \geq 1 \quad \forall N \geq 1$
- Istate,F,G: ne sont pas initialisés car Intype=0 .

l_w, W : ne sont pas initialisés car ce sont des tableaux de travail de la routine .
 l_{iw}, L_w : leur longueur respective doit néanmoins être initialisée :
 $l_{iw}=2$
 $L_w=9*N$
 $ifail$: 0 comme suggéré .

D. GESTION DU PARAMETRE D'ERREUR DE LA ROUTINE

IFAIL = 2

Il y a eu plus de maxcal évaluations de fonction . On considère que le nouveau point dual atteint est quand même un minimum de $(-1)*$ la fonction duale . On passe alors à l'étape suivante de notre algorithme .

IFAIL = 3 , 5

Deux hypothèses sont possibles : soit la recherche linéaire a échoué, soit toutes les conditions requises pour un minimum ne sont pas toutes atteintes .

On considère néanmoins que le nouveau point dual atteint est un minimum de $(-1)*$ la fonction duale . On passe alors à l'étape suivante de notre algorithme .

IFAIL = 4

Il y a eu un "overflow" lors du calcul des facteurs de Cholesky dans E04KBF . Nous recommençons alors un nouvel appel à E04KBF.

5.1.4.2 La fonction duale

Soit le problème linéarisé relaxé en $Y^{(k)}, X^{(k)}$

(SP \sim)^(k)

$$\left\{ \begin{array}{ll}
 \text{Min} & \tilde{f}(Y; Y^{(k)}) \\
 \text{Sous} & h_i \tilde{(Y; Y^{(k)})} - z_i \leq 0 \quad (i=1..nch) \\
 & g_{ij} \tilde{(Y, X_i; Y^{(k)}, X_i^{(k)})} - z_{ij} \leq 0 \quad (i=1..ncg, j=1..ncgi(i)) \\
 & Y^l \leq Y \leq Y^u \\
 & X_i^l \leq X_i \leq X_i^u \quad (i=1..ncg)
 \end{array} \right.$$

Le problème dual qui lui est associé est de la forme suivante :

$$\left\{ \begin{array}{l} \text{Maximiser } \theta(u) \\ \text{Sous } u_i \geq 0 \quad (i=1..nch) \\ \quad u_{ij} \geq 0 \quad (i=1..ncg, j=1..ncgi(i)) \end{array} \right.$$

Le caractère séparable du problème $(SP^{\sim})^{(k)}$ va nous permettre d'exprimer la fonction duale de manière explicite :

$$\begin{aligned} \theta(u) = \text{Min}_x \{ & \sum_{i+} \frac{\partial f}{\partial y_i} y_i - \sum_{i-} \frac{\partial f}{\partial y_i} \frac{1}{y_i} + \\ & \sum_{i-} \frac{\partial f}{\partial y_i} - \sum_{i+} \frac{\partial f}{\partial y_i} + f(y^{(k)}) + \\ & \sum_{i=1..nch} d_i (z_i + z_i^2) + \sum_{i=1..ncg} \sum_{j=1..ncgi(i)} d_{ij} (z_{ij} + z_{ij}^2) \\ & + \sum_{j=1..nch} u_j \left(\sum_{i+} \frac{\partial h_j}{\partial y_i} y_i - \sum_{i-} \frac{\partial h_j}{\partial y_i} \frac{1}{y_i} + \right. \\ & \left. \sum_{i-} \frac{\partial h_j}{\partial y_i} - \sum_{i+} \frac{\partial h_j}{\partial y_i} + h_j(y^{(k)}) - z_j \right) \\ & + \sum_{i=1..ncg} \sum_{j=1..ncgi(i)} u_{ij} \left(\sum_{k+} \frac{\partial g_{ij}}{\partial y_k} y_k - \right. \end{aligned}$$

$$\sum_{k^-} \frac{\partial g_{ij}}{\partial y_k} \frac{1}{y_k} + \sum_{k^+} \frac{\partial g_{ij}}{\partial x_{ik}} x_{ik}$$

$$\sum_{k^-} \frac{\partial g_{ij}}{\partial x_{ik}} \frac{1}{x_{ik}} - \sum_{k^+} \frac{\partial g_{ij}}{\partial x_{ik}}$$

$$\sum_{k^-} \frac{\partial g_{ij}}{\partial x_{ik}} - \sum_{k^+} \frac{\partial g_{ij}}{\partial y_k} + \sum_{k^-} \frac{\partial g_{ij}}{\partial y_k}$$

$$g_{ij}(Y^{(k)}, X_i^{(k)}) - z_{ij} \}}}$$

où $\mathbf{X} = \{Y, X, Z \text{ tq } Y^l \leq Y \leq Y^u \text{ et } X_i^l \leq X_i \leq X_i^u (i=1..ncg) \text{ et } Z \geq 0\}$

$$\theta(u) = \text{Min}_{Y^l \leq Y \leq Y^u} \left\{ \sum_{i^+} \frac{\partial f}{\partial y_i} y_i - \sum_{i^-} \frac{\partial f}{\partial y_i} \frac{1}{y_i} + \right.$$

$$\left. + \sum_{j=1..nch} u_j \left(\sum_{i^+} \frac{\partial h_j}{\partial y_i} y_i - \sum_{i^-} \frac{\partial h_j}{\partial y_i} \frac{1}{y_i} \right) \right.$$

$$\left. + \sum_{i=1..ncg} \sum_{j=1..ncgi(i)} u_{ij} \left(\sum_{k^+} \frac{\partial g_{ij}}{\partial y_k} y_k - \right. \right.$$

$$\left. \sum_{k^-} \frac{\partial g_{ij}}{\partial y_k} \frac{1}{y_k} \right) \} +$$

$$\text{Min}_{X^l, X^s, X^u} \left\{ \sum_{i=1..ncg} \sum_{j=1..ncgi(i)} u_{ij} \left(\sum_{k+} \frac{\partial g_{ij}}{\partial x_{ik}} x_{ik} - \right. \right.$$

$$\left. \sum_{k-} \frac{\partial g_{ij}}{\partial x_{ik}} x_{ik} \right) \} +$$

$$\text{Min}_{z_i \geq 0} \left\{ \sum_{i=1..nch} d_i (z_i + z_i^2) \right\} +$$

$$\text{Min}_{z_{ij} \geq 0} \left\{ \sum_{i=1..ncg} \sum_{j=1..ncgi(i)} d_{ij} (z_{ij} + z_{ij}^2) \right\} +$$

$$\sum_{i-} \frac{\partial f}{\partial y_i} - \sum_{i+} \frac{\partial f}{\partial y_i} + f(y^{(k)}) +$$

$$\sum_{j=1..nch} u_j \left(\sum_{i-} \frac{\partial h_j}{\partial y_i} - \sum_{i+} \frac{\partial h_j}{\partial y_i} + h_j(y^{(k)}) \right) +$$

$$\sum_{i=1..ncg} \sum_{j=1..ncgi(i)} u_{ij} \left(\sum_{k-} \frac{\partial g_{ij}}{\partial y_k} - \sum_{k+} \frac{\partial g_{ij}}{\partial y_k} + \right.$$

$$\left. \sum_{k-} \frac{\partial g_{ij}}{\partial x_{ik}} - \sum_{k+} \frac{\partial g_{ij}}{\partial x_{ik}} + g_{ij}(y^{(k)}, x_i^{(k)}) \right)$$

En utilisant le caractère séparable on peut encore décomposer ce problème en petits problèmes à une variable :

$$\theta(u) = \sum_{k=1..ncy} \text{Min}_{y_k^l \leq y_k \leq y_k^u} (L_k(y_k, u)) +$$

$$\sum_{i=1..ncg} \sum_{j=1..ncxi(i)} \text{Min}_{x_{ij}^l \leq x_{ij} \leq x_{ij}^u} (K_{ij}(x_{ij}, u)) +$$

$$\sum_{i=1..nch} \text{Min}_{z_i \geq 0} (d_i z_i + d_i z_i^2 - u_i z_i) +$$

$$\sum_{i=1..ncg} \sum_{j=1..ncgi(i)} \text{Min}_{z_{ij} \geq 0} (d_{ij} z_{ij} + d_{ij} z_{ij}^2 - u_{ij} z_{ij}) + \text{Cte}$$

où

a)

$$\text{si } \frac{\partial f}{\partial y_k} \geq 0$$

$$L_k(y_k, u) = \frac{\partial f}{\partial y_k} y_k + \sum_{i+} \frac{\partial h_i}{\partial y_k} y_k u_i - \sum_{i-} \frac{\partial h_i}{\partial y_k} \frac{1}{y_k} u_i$$

$$+ \sum_{i=1..ncg} \sum_{j+} \frac{\partial g_{ij}}{\partial y_k} y_k u_{ij} -$$

$$\sum_{i=1..ncg} \sum_{j-} \frac{\partial g_{ij}}{\partial y_k} \frac{1}{y_k} u_{ij}$$

b)

$$s_1 \frac{\partial f}{\partial y_k} \leq 0$$

$$\begin{aligned} L_k(y_k, u) = & - \frac{\partial f}{\partial y_k} y_k + \sum_{i+} \frac{\partial h_i}{\partial y_k} y_k u_i - \sum_{i-} \frac{\partial h_i}{\partial y_k} \frac{1}{y_k} u_i \\ & + \sum_{i=1..ncg} \sum_{j+} \frac{\partial g_{ij}}{\partial y_k} y_k u_{ij} - \\ & \sum_{i=1..ncg} \sum_{j-} \frac{\partial g_{ij}}{\partial y_k} \frac{1}{y_k} u_{ij} \end{aligned}$$

c)

$$K_{ij}(x_{ij}, u) = \sum_{k+} u_{ik} \frac{\partial g_{ik}}{\partial x_{ij}} x_{ij} - \sum_{k+} u_{ik} \frac{\partial g_{ik}}{\partial x_{ij}} \frac{1}{x_{ij}}$$

d)

$$\begin{aligned}
 \text{Cte} = & \sum_{i-} \frac{\partial f}{\partial y_i} \Big|_1 - \sum_{i+} \frac{\partial f}{\partial y_i} \Big|_1 + f(y^{(k)}) + \\
 & + \sum_{j=1..nch} u_j \left(\sum_{i-} \frac{\partial h_j}{\partial y_i} \Big|_1 - \sum_{i+} \frac{\partial h_j}{\partial y_i} \Big|_1 + h_j(y^{(k)}) \right) \\
 & + \sum_{i=1..ncg} \sum_{j=1..ncgi(i)} u_{ij} \left(\sum_{k-} \frac{\partial g_{ij}}{\partial y_k} \Big|_1 - \sum_{k+} \frac{\partial g_{ij}}{\partial y_k} \Big|_1 + \right. \\
 & \left. \sum_{k-} \frac{\partial g_{ij}}{\partial x_{ik}} \Big|_1 - \sum_{k+} \frac{\partial g_{ij}}{\partial x_{ik}} \Big|_1 + g_{ij}(y^{(k)}, x_i^{(k)}) \right)
 \end{aligned}$$

Comme nous faisons appel à la routine E04KBF, nous avons besoin de connaître le gradient de la fonction duale. Heureusement, la $i^{\text{ème}}$ composante de ce gradient est la $i^{\text{ème}}$ contrainte du problème primal correspondant (cfr ANNEXE A 5.1).

5.1.4.3 Evaluation des variables primales en fonction des multiplicateurs de Lagrange

En notant

$$C_k = \sum_{i+} \frac{\partial h_i}{\partial y_k} \Big|_1 u_i + \sum_{i=1..ncg} \sum_{j+} \frac{\partial g_{ij}}{\partial y_k} \Big|_1 u_{ij} \geq 0$$

$$D_k = \sum_{i-} \frac{\partial h_i}{\partial y_k} \Big|_1 u_i + \sum_{i=1..ncg} \sum_{j-} \frac{\partial g_{ij}}{\partial y_k} \Big|_1 u_{ij} \leq 0$$

$$W_{ij} = \sum_{k+} u_{ik} \frac{\partial g_{ik}}{\partial x_{ij}} \geq 0$$

$$V_{ij} = \sum_{k-} u_{ik} \frac{\partial g_{ik}}{\partial x_{ij}} \leq 0$$

Nous obtenons :

a) si $\frac{\partial f}{\partial y_k} \geq 0$

$$L_k(y_k, u) = \frac{\partial f}{\partial y_k} y_k + C_k y_k - D_k \frac{1}{y_k}$$

b) si $\frac{\partial f}{\partial y_k} \leq 0$

$$L_k(y_k, u) = - \frac{\partial f}{\partial y_k} \frac{1}{y_k} + C_k y_k - D_k \frac{1}{y_k}$$

c)

$$K_{ij}(x_{ij}, u) = W_{ij} x_{ij} - V_{ij} \frac{1}{x_{ij}}$$

A) La recherche du minimum de $L_k(y_k, u)$ sur le domaine $[y_k^l, y_k^u]$ se trouve en Annexe 2 et permet d'exprimer y_k en fonction de u :

$\frac{\partial f}{\partial y_k|_1} > 0$

$$y_k(u) = \left\{ \frac{-D_k}{(\partial f / \partial Y_k|_1) + C_k} \right\}^{1/2} \quad \text{si } (y_k^l)^2 \leq \frac{-D_k}{(\partial f / \partial Y_k|_1) + C_k} \leq (y_k^u)^2$$

$$= y_k^u \quad \text{si } \frac{-D_k}{(\partial f / \partial Y_k|_1) + C_k} \geq (y_k^u)^2$$

$$= y_k^l \quad \text{si } \frac{-D_k}{(\partial f / \partial Y_k|_1) + C_k} \leq (y_k^l)^2$$

$\frac{\partial f}{\partial y_k|_1} < 0$

$$y_k(u) = \left\{ -\frac{(\partial f / \partial Y_k|_1) + D_k}{C_k} \right\}^{1/2} \quad \text{si } (y_k^l)^2 \leq -\frac{(\partial f / \partial Y_k|_1) + D_k}{C_k} \leq (y_k^u)^2$$

$$= y_k^u \quad \text{si } -\frac{(\partial f / \partial Y_k|_1) + D_k}{C_k} \geq (y_k^u)^2$$

$$= y_k^l \quad \text{si } -\frac{(\partial f / \partial Y_k|_1) + D_k}{C_k} \leq (y_k^l)^2$$

B) La recherche du minimum de $k_{ij}(x_{ij}, u)$ sur le domaine $[x_{ij}^l, x_{ij}^u]$ se trouve en ANNEXE A 5.3 et permet d'exprimer x_{ij} en fonction de u :

$$x_{ij}(u) = \begin{cases} -\frac{V_{ij}}{W_{ij}} & \text{si } (x_{ij}^l)^2 \leq -\frac{V_{ij}}{W_{ij}} \leq (x_{ij}^u)^2 \\ x_{ij}^u & \text{si } -\frac{V_{ij}}{W_{ij}} \geq (x_{ij}^u)^2 \\ x_{ij}^l & \text{si } -\frac{V_{ij}}{W_{ij}} \leq (x_{ij}^l)^2 \end{cases}$$

C) La recherche de $\text{Min}_{z_i \geq 0} \{ d_i z_i + d_i z_i^2 - u_i z_i \} = \text{Min}_{z_i \geq 0} L(z_i)$ donne

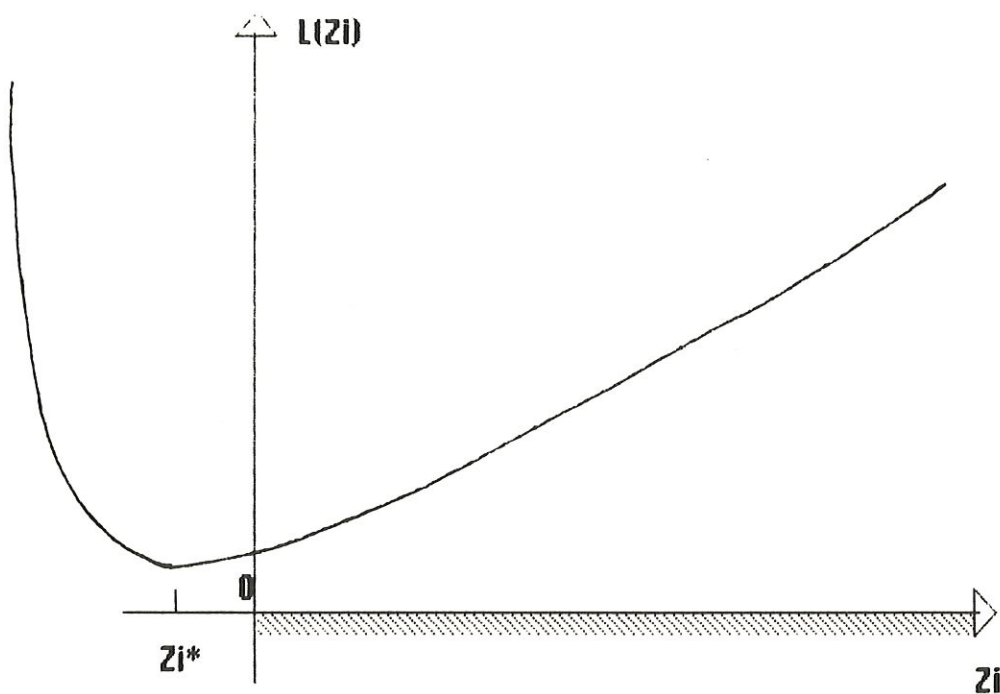
comme solution :

$$\frac{\partial L(z_i)}{\partial z_i} = d_i + 2d_i z_i - u_i$$

$$\frac{\partial L(z_i)}{\partial z_i} = 0 \Leftrightarrow z_i = \frac{u_i - d_i}{2d_i}$$

$$\frac{\partial^2 L(z_i)}{\partial z_i^2} = 2d_i > 0 \quad \text{donc } L(z_i) \text{ est une fonction convexe}$$

donc si $z_i \leq 0$ nous avons la situation suivante :



Le minimum de $L(z_i)$ sur $[0, \infty[$ est donc 0.

En résumé :

$$z_i = \frac{u_i}{2d_i} - 0.5 \quad \text{si } \frac{u_i}{2d_i} \geq 0.5$$

$$z_i = 0 \quad \text{si } \frac{u_i}{2d_i} < 0.5$$

D) La recherche de $\text{Min}_{z_{ij} \geq 0} \{ d_{ij} z_{ij} + d_{ij} z_{ij}^2 - u_{ij} z_{ij} \}$

Par analogie avec C) :

$$z_{ij} = \frac{u_{ij}}{2d_{ij}} - 0.5 \quad \text{si } \frac{u_{ij}}{2d_{ij}} \geq 0.5$$

$$z_{ij} = 0 \quad \text{si } \frac{u_{ij}}{2d_{ij}} < 0.5$$

5.1.5 Critère d'arrêt de l'algorithme

L'algorithme s'arrête quand :

(Y, X_1, \dots, X_{ncg}) et U (vecteur des multiplicateurs de Lagrange)

vérifient les conditions de Kuhn et Tucker

et

(Y, X_1, \dots, X_{ncg}) est admissible .

Les conditions de Kuhn et Tucker s'écrivent :

$$\begin{aligned} .) \quad & \nabla f(Y) + \sum_{i=1..nch} u_i \nabla h_i(Y) + \sum_{i=1..ncg} \sum_{j=1..ncxi(i)} u_{ij} \nabla g_{ij}(Y, X_i) \\ & + \sum_{i=1..ncy} e_i (v_i - v_i') + \sum_{i=1..ncg} \sum_{j=1..ncxi(i)} e_{ij} (w_{ij} - w_{ij}') = 0 \end{aligned} \quad (1)$$

où v_i (w_{ij}) est le multiplicateur de Lagrange associé à la
contrainte $y_i \leq y_i^u$ ($x_{ij} \leq x_{ij}^u$)

v_i' (w_{ij}') est le multiplicateur de Lagrange associé à la
contrainte $y_i' \leq y_i$ ($x_{ij}' \leq x_{ij}$)

$$\begin{aligned} \text{où } e_i &= \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \leftarrow i\text{ème ligne} & e_{ij} &= \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ - \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \begin{array}{l} \text{ncy lignes} \\ \leftarrow (ncy + \sum_{k=1..i-1} ncxi(k) \\ + j)\text{ème ligne} \end{array} \end{aligned}$$

$$\begin{aligned}
 &.) u_i h_i(Y) = 0 \quad (i=1..nch) \\
 &u_{ij} g_{ij}(Y, X_i) = 0 \quad (i=1..ncg \quad j=1..ncgi(i))
 \end{aligned}$$

(2)

(Conditions de Complémentarité)

$$\begin{aligned}
 &.) u_i \geq 0 \quad (i=1..nch) \\
 &u_{ij} \geq 0 \quad (i=1..ncg \quad j=1..ncgi(i)) \\
 &v_i, v'_i \geq 0 \quad (i=1..ncy) \\
 &w_{ij}, w'_{ij} \geq 0 \quad (i=1..ncg \quad j=1..ncxi(i))
 \end{aligned}$$

Si y_i atteint sa borne supérieure (inférieure), la condition (1) qui lui est associée se réécrit :

$$\begin{aligned}
 &\nabla_i f(Y) + \sum_{k=1..nch} u_k \nabla_i h_k(Y) + \sum_{k=1..ncg} \sum_{j=1..ncgi(k)} u_{kj} \nabla_i g_{kj}(Y, X_k) \\
 &= -v_i \leq 0 \quad (= v'_i \geq 0)
 \end{aligned}$$

Si x_{ij} atteint sa borne supérieure (inférieure), la condition (1) qui lui correspond se réécrit :

$$\sum_{k=1..ncg} \sum_{l=1..ncgi(k)} u_{kl} \nabla_{ij} g_{kl}(Y, X_k) = -w_{ij} \leq 0 \quad (w'_{ij} \geq 0)$$

Afin de limiter le coût-calcul de l'ordinateur si la convergence n'est pas rapide, nous avons ajouté une condition d'arrêt :

Le nombre d'itérations ne peut dépasser un nombre maximal fixé par l'utilisateur .

5.2 IMPLEMENTATION

5.2.1 Commons du programme

```
c
c Parametres du common IIM
c *****
c
c   NCZ   : Nombre total de variables primales .
c   NCY   : Nombre de variables Y .
c   MNCX  : Maximum du nombre de composantes de Xi .
c   NCXI  : Nombre de composantes de chaque Xi .
c   NCH   : Nombre de contraintes H .
c   NCG   : Nombre de Xi .
c   NCGI  : Nombre de composantes de chaque Gi .
c   MNCG  : Maximum du nombre de composantes de Gi .
c   MNCGH : Maximum de MNCG et NCH .
c   MNCXY : Maximum de NCY et MNCX .
c
c   integer ncz,ncy,mncx,ncxi(ncgmax),nch,ncg,ncgi(ncgmax),
c           1 mncg,mncgh,mncxy
c   common/dim/ncz,ncy,mncx,ncxi,nch,ncg,ncgi,mncg,mncgh,mncxy
c
c Parametres du common PURN
c *****
c
c   UBURNZ : bornes superieures des variables primales .
c   LBURNZ : bornes inferieures des variables primales .
c
c   double precision ubornz(ncuzmax),lbornz(ncuzmax)
c   common/bozn/ubornz,lbornz
c
c Parametres du common REL
c *****
c
c   D : Constantes de relaxation .
c
c   double precision d(ncumax)
c   common/rel/d
c
c Parametres du common VAP
c *****
c
c   VAR : Variables courantes mais avec scaling .
c
c   double precision var(ncuzmax)
c   common/var/var
c
c Parametres du common F
c *****
c   "common de la fonction objectif"
c
c   DCFIY : Constante de linearisation convexe .
c   DFCIY1 : Coefficients de la linearisation convexe .
c   FYK   : Valeur de la fonction .
c
c   double precision dfcty,dfcty1(ncymax),fyk
c   common/f/dfcty,dfcty1,fyk
```

```

c
c Parametres du common H
c *****
c "common des contraintes H"
c
c   DHCTY : Constantes de linearisation convexe .
c
c
c   DHCTY1 : Coefficients des linearisations convexes .
c   HYK    : Valeurs des fonctions .
c
c   double precision dhcty(nchmax),dhcty1(ncymax,nchrax),
c   1                hyk(ncgmax)
c   common/h/dhcty,dhcty1,hyk
c
c Parametres du common G
c *****
c "common des contraintes G"
c
c   DGCTXY : Constantes de linearisation convexe .
c   DGCTY1 : Coefficients en Y des linearisations convexes .
c   DGCTX1 : Coefficients en X1 des linearisations convexes .
c   GXYK   : valeurs des fonctions .
c
c   double precision dgctxy(mncgmax,ncgmax),
c   1                dgcty1(ncymax,mncgmax,ncgmax),
c   2                dgctx1(mncxmax,mncgmax,ncgmax),
c   3                gxyk(mncgmax,ncgmax)
c   common/g/dgctxy,dgcty1,dgctx1,gxyk
c
c Parametres du common SORTIE
c *****
c
c   IOUT : Niveau d'impression .
c   US   : Unite de sortie .
c   ITER : nombre d'iterations primales .
c   FCT  : nombre d'iterations duales .
c
c   integer iout,us,iter,fct
c   common/sortie/iout,us,iter,fct

```

5.2.2 User's guide

1) Nous demandons à l'utilisateur un fichier contenant une

version compilée de la routine FGH (zk,gradf,gradh,gradgx,gradgy)
(spécifications cfr ultra) et de la linker avec memoire,
routines, naglibrary, utillibrary .

.) Commons à employer dans la routine :

DIM,F,G,H

.) Paramètre d'entrée :

ZK (1..NCZ) : double precision
vecteur contenant le point courant de
linéarisation, point auquel sont évaluées les
fonctions et leur gradient

.) Paramètres de sortie :

GRADF (1..NCY) : double precision
vecteur dont le $i^{\text{ème}}$ élément contient
$$\frac{\partial f}{\partial y_i | z_k}$$

GRADH (1..NCYMAX, 1..NCH) : double precision
matrice dont le $(i,j)^{\text{ème}}$ élément contient
$$\frac{\partial h_j}{\partial y_i | z_k}$$

GRADGX (1..MNCXMAX, 1..MNCGMAX, 1..NCG) : double precision
matrice dont le $(i,j,l)^{\text{ème}}$ élément contient
$$\frac{\partial g_{lj}}{\partial x_{li} | z_k}$$

GRADGY (1..NCYMAX, 1..MNCGMAX, 1..NCG) : double precision
matrice dont le $(i,j,l)^{\text{ème}}$ élément contient
$$\frac{\partial g_{lj}}{\partial y_i | z_k}$$

) Certaines sorties de la routine se feront par l'intermédiaire des communs .

FYK : double précision (dans le commun F)
la valeur de la fonction objectif en ZK

HYK (1..NCH) : double précision (dans le commun H)

vecteur dont le $i^{\text{ème}}$ élément contient la valeur de la contrainte h_i en ZK

GXYK (1..MNCGMAX, 1..NCG) : double précision (dans le commun G)

matrice dont le $(i, j)^{\text{ème}}$ élément contient la valeur de la contrainte g_{ij} en ZK

Remarque Il est interdit de changer la valeur d'autres variables des communs .

2) Possibilités d'entrée/sortie

La lecture des données se fait soit par l'écran soit par fichier .

Dans le premier cas, les explications seront données lors de l'interaction avec l'utilisateur qui ne devra fournir que les renseignements demandés .

Dans le second cas, avant l'utilisation du programme, l'utilisateur devra créer deux fichiers (.DAT) qui devront contenir dans l'ordre :

pour le premier:

1. le nombre de contraintes globales (ou de composantes de h) :
 nch
2. le nombre de vecteurs de variables locales : ncg
3. pour i de 1 à ncg , le nombre de composantes de g_i
4. le nombre de variables globales : ncy
5. pour i de 1 à ncg , le nombre de composantes de x_i : $ncxi(i)$
6. pour i de 1 à ncy , la borne inférieure (y_i^l) et la borne supérieure (y_i^u) de y_i
7. pour i de 1 à ncg , pour j de 1 à $ncgi(i)$, la borne inférieure (x_{ij}^l) et la borne supérieure (x_{ij}^u) de x_{ij}
8. le nombre maximal d'itérations primales

pour le second :

1. pour i de 1 à ncy , le point de départ en ses composantes y_i
2. pour i de 1 à ncg , pour j de 1 à $ncg(i)$, le point de départ en ses composantes x_{ij}
3. pour chaque contrainte, un choix de constantes de relaxation . Il est conseillé de les choisir au moins dix fois plus grandes que les coefficients de la fonction objectif .

Remarque : Il est également possible de rentrer les données par un des deux fichiers et les autres à l'écran .

De toute façon, lors de l'interaction, l'utilisateur devra fournir par le terminal la précision du test de Kuhn et Tucker et la précision duale .

Remarques :

1) Il est fortement conseillé de prendre une précision duale (i.e. une précision pour E04KBF) plus grande que celle pour Kuhn et Tucker .

2) La précision du test de Kuhn-Tucker n'est pas équivalente à la précision des résultats . En effet, une précision demandée de 10^{-3} donnera des réponses exactes à 10^{-8} ou 10^{-9} , parfois .

Pour ce qui est des sorties, là encore, il y a possibilité d'avoir les résultats dans un fichier (.DAT) du choix de l'utilisateur ou de les avoir à l'écran . Dans ce cas, nous sortons néanmoins les données du problème et le résultat final dans un fichier nommé "RESULT.OPT" . Dans tous les cas, nous demandons à l'utilisateur le niveau d'impression souhaité :

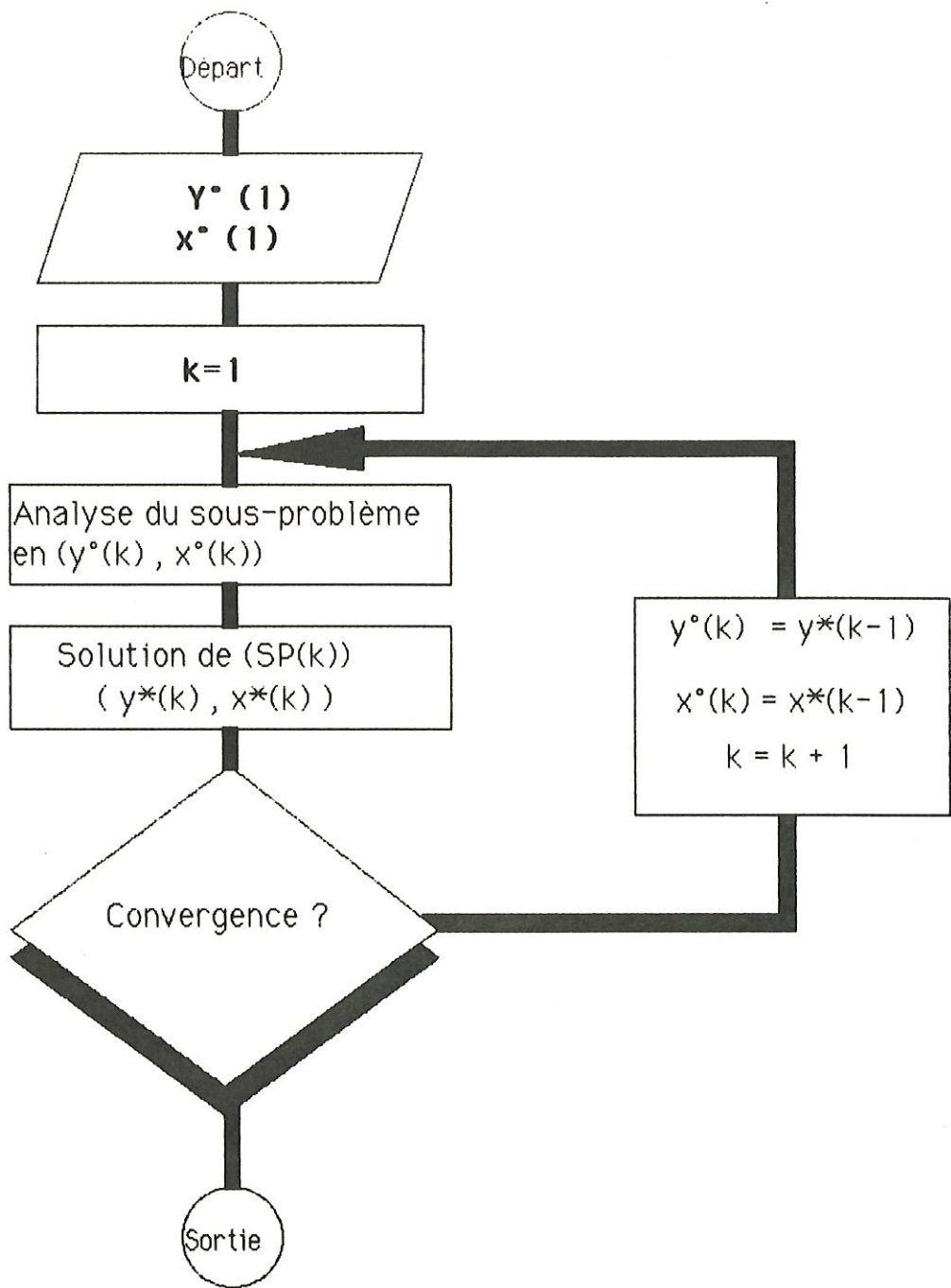
- 0 pour avoir seulement les données et le résultat final
- 1 > 0 pour avoir les résultats toutes les l itérations .

Nous avons également utilisé un niveau d'impression pour les itérations duales :

- 0 pour n'avoir aucune impression
- 1 pour avoir les résultats seulement à la dernière itération
- 2 pour avoir les résultats à toutes les itérations

3) Choix des variables duales

Arbitrairement, nous avons décidé qu'à chaque appel de E04KBF nous donnerions à nos variables duales la valeur 1. Ceci est une pratique courante.



OPTIMISATION A UN NIVEAU AVEC APPROXIMATIONS

PROGRAM RESOLUTION

 Ce programme resout le probleme d'optimisation expose dans menu
 par linearisations convexes successives.
 Chaque sous probleme ainsi cree est resolu par methode duale
 Remarque : Nous avons insere des 'include' dans les declarations des routine
 ----- alors que ce n'est pas du fortran 77 standard .
 Pour rendre ce programme standard il suffit de recopier le common
 correspondant au nom du include a la place de celui-ci

DECLARATIONS

=====

```

parameter nczmax=35,nchmax=10,ncsmax=5,mncsmax=5,mncshmax=10,
1          mncxmax=5,mncymax=10,ncumax=35,ncuzmax=70
  
```

Parametres du programme

Les entiers

```

-----
I,J      : Indices d'iteration .
NCU      : Nombre total de contraintes .
NCUZ     : NCU + le nombre total de variables .
UE1,UE2  : Numeros des fichiers d'entree .
ITERMAX  : Nombre maximal d'iterations primales .
IT       : Temps CPU .
ISECONDIFF : Fonction donnant le temps CPU .
  
```

```

integer i,j,ncu,ncuz,ue1,ue2,itermax,it,isecondiff,iflag
  
```

Les "double precision"

```

-----
ZK       : Le point courant .
GRADF   : Le gradient de la fonction objectif F .
GRADH   : Le gradient des contraintes H .
GRADGY  : Le gradient des contraintes G par rapport a Y .
GRADGX  : Le gradient des contraintes G par rapport a X .
U       : Les multiplicateurs de Lagrange .
EPS1    : La precision du test de Kuhn-Tucker .
THEMAX  : Fonction donnant une borne superieure de la fonction duale
MAX     : Borne superieure de la fonction duale .
  
```

```

double precision zk(nczmax),gradf(ncymax),gradh(ncymax,nchmax),
1               gradgy(ncymax,mncsmax,ncsmax),
2               gradgx(mncxmax,mncsmax,ncsmax),
3               u(ncumax),eps1,themax,max,theta
  
```

Les caracteres

```

-----
NAME1,NAME2 : noms des fichiers de donnees .
NAME3       : nom du fichier de sortie .
  
```

```

character*30 name1,name2,name3
  
```

Le boolean

```

-----
ARRET : Fonction d'arret de l'algorithme .
  
```

```

logical arret
  
```


Parametres de E04KBF

Cette routine est utilisee pour maximiser la fonction duale .

Les entiers

- IPRINT : Niveau d'impression .
- INTYPE : Niveau d'entree des donnees .
- MAXCAL : Nombre maximal d'appels a FUNCT .
- IBOUND : Nature des bornes .
- LH : Dimension de la matrice hessienne triangulaire inferieure .
- ISTATE : Position des variables par rapport a leurs bornes .
- IW : Tableau de travail .
- LIW : Dimension de IW .
- LW : Dimension du tableau de travail W .
- IFAIL : Niveau d'erreur .

integer iprint,intype,maxcal,ibound,lh,istate(ncumax),iw(10),
1 liw,lw,ifail

Les "double precision"

- ETA : Precision de la recherche lineaire .
- XTOL : Precision de la solution .
- STPEMX : Estimation de la distance du point de depart a la solution .
- FEST : Estimation de la valeur de la fonction objectif .
- BL,BU : Bornes superieures et inferieures des variables
- HESL : Triangulaire inferieure de la factorisation de Cholesky de la matrice hessienne .
- HESD : Elements diagonaux de la matrice diagonale de la factorisation de cholesky de la matrice hessienne .
- FC : Valeur de la fonction objectif .
- GC : Valeur du sradient de la fonction objectif .
- W : Tableau de travail .

double precision eta,xtol,stepmx,fest,bl(ncumax),bu(ncumax),
1 hesl(1000),hesd(ncumax),fc,gc(ncumax),w(1000)

Le booleen

- LOCSCH : Choix d'une recherche lineaire ou non .

logical locsch

Les routines externes

- FUNCT : Calcul de la fonction duale et de ses sradients .
- MONIT : Impression .
- E04LBS : Recherche lineaire .

external e04lbs,funct,monit

```

c COMMONS
c =====
c
c Parametres du common DIM
c *****
c
c   NCZ   : Nombre total de variables primales .
c   NCY   : Nombre de variables Y .
c   MNCX  : Maximum du nombre de composantes de Xi .
c   NCXI  : Nombre de composantes de chaque Xi .
c   NCH   : Nombre de contraintes H .
c   NCG   : Nombre de Xi et de contraintes globales .
c   NCGI  : Nombre de composantes de chaque Gi .
c   MNCG  : Maximum du nombre de composantes de Gi .
c   MNCGH : Maximum de MNCG et NCH .
c   MNCXY : Maximum de NCY et MNCX .
c
c       integer ncz,ncy,mncx,ncxi(ncsmax),nch,ncs,ncsi(ncsmax),
c           1      mncs,mncsh,mncxy
c       common/dim/ncz,ncy,mncx,ncxi,nch,ncs,ncsi,mncs,mncsh,mncxy
c
c Parametres du common BORN
c *****
c
c   UBORNZ : bornes superieures des variables primales .
c   LBORNZ : bornes inferieures des variables primales .
c
c       double precision ubornz(ncuzmax),lbornz(ncuzmax)
c       common/born/ubornz,lbornz
c
c Parametres du common REL
c *****
c
c   D : Constantes de relaxation .
c
c       double precision d(ncumax)
c       common/rel/d
c
c Parametres du common VAR
c *****
c
c   VAR : Variables courantes mais avec scaling .
c
c       double precision var(ncuzmax)
c       common/var/var
c
c Parametres du common F
c *****
c   "common de la fonction objectif"
c
c   DFCTY  : Constante de linearisation convexe .
c   DFCTY1 : Coefficients de la linearisation convexe .
c   FYK    : Valeur de la fonction .
c
c       double precision dfcty,dfcty1(ncymax),fyk
c       common/f/dfcty,dfcty1,fyk
c
c Parametres du common H
c *****
c   "common des contraintes H"
c
c   DHCTY  : Constantes de linearisation convexe .
c   DHCTY1 : Coefficients des linearisations convexes .
c   HYK    : Valeurs des fonctions .
c
c       double precision dhcty(nchmax),dhcty1(ncymax,nchmax),
c           1      hyk(nchmax)
c       common/h/dhcty,dhcty1,hyk

```



```

c Parametres du common G
c *****
c "common des contraintes G"
c
c DGCTXY : Constantes de linearisation convexe .
c DGCTY1 : Coefficients en Y des linearisations convexes .
c DGCTX1 : Coefficients en Xi des linearisations convexes .
c GXK    : Valeurs des fonctions .
c
c      double precision dsctxy(mncsmax,ncsmax),
c      1                dsctyl(ncsmax,mncsmax,ncsmax),
c      2                dsctx1(mncxmax,mncsmax,ncsmax),
c      3                sxxk(mncsmax,ncsmax)
c      common/s/dsctxy,dsctyl,dsctx1,sxxk
c
c Parametres du common SORTIE
c *****
c
c IOUT : Niveau d'impression .
c US   : Unite de sortie .
c ITER : nombre d'iterations primales .
c FCT  : nombre d'iterations duales .
c
c      integer iout,jout,us,iter,fct
c      common/sortie/iout,jout,us,iter,fct
c
c ROUTINES UTILISEES
c =====
c
c INIT      : Initialisation dans la library UTIL .
c ISECOND   : Initialisation du temps CPU .
c MENU     : Interaction avec l'utilisateur et choix du mode
c           d'entree/sortie .
c INITIAL  : Lecture des donnees .
c CALNCU   : Calcul de NCU et NCUZ .
c DEPART   : Lecture du point de depart .
c PBME     : Impression des donnees du probleme .
c FGH      : Calcul de la valeur et des gradients de la fonction
c           objectif et des contraintes .
c DFOBJ1   : Calcul de la constante et des coefficients de linearisation
c           convexe de la fonction objectif .
c DHCON1   : Calcul de la constante et des coefficients de linearisation
c           convexe des contraintes H .
c DGCON1   : Calcul de la constante et des coefficients de linearisation
c           convexe des contraintes G .
c SCALE    : Scalins sur les bornes des variables .
c SETPRI   : Calcul des variables primales en fonction des multiplicateurs
c           de Lagrange .
c ECHELLE  : Remise a l'echelle de depart des bornes et des variables .
c SAUVE    : Sauvetage du point courant .
c FINALE   : Impression des resultats .
c
c
c

```

c
c INSTRUCTIONS
c =====

c
c Initialisation du calcul temps CPU .
c

CALL INIT
CALL ISECOND (it)

c
c Initialisation des variables caracteristiques du probleme .
c

NAME1 = 'NAME1'
NAME2 = 'NAME2'
NAME3 = 'NAME3'
CALL MENU (name1,name2,name3,ue1,ue2,eps1,xtol)
IF (US.NE.5) THEN
 OPEN (UNIT=US,FILE=NAME3,STATUS='NEW')
ELSE
 OPEN (UNIT=45,FILE='RESULT.OPT',STATUS='NEW')
 WRITE (5,4000)
ENDIF
CALL INITIAL (ue1,name1,itermax)
CALL CALNCU (ncu,ncuz)
FCT = 0
ITER = 0

c
c Assignation des parametres constants de "e04kbf" .
c

IFLAG = 2
MAXCAL = 100*NCU
LIW = 10
LW = 9*NCU
LH = NCU*(NCU-1)/2
INTYPE = 0
IF (JOUT.EQ.2) THEN
 IPRINT = 1
ELSE
 IPRINT = 0
ENDIF
LOCSCH = .TRUE.
ETA = 0.5
STEPMX = 100000.
IBOUND = 2

c
c Lecture des points de depart primaux et assignation des
c points de depart duaux .
c

CALL DEPART (ue2,name2,ncu,zk)
DO 11 I=1,NCU
 U(I) = 1.
11 CONTINUE

c
c Impression des donnees .
c

CALL FBME (itermax,maxcal,eps1,xtol,zk)

c
c Evaluation des fonctions et de leur gradient .
c

CALL FGH (zk,gradf,gradh,gradsx,gradsy)

c
10 CONTINUE

```

c
c
c      Est-on a l'optimum ?
c
c
c      Test d'arret : zk est-il point de Kuhn-Tucker ?
c
c      IF (ARRET (eesl,gradf,gradh,gradsx,gradsy,u,ncu,itermax,zk))
1      GOTO 302
c
c
c      On n'est pas encore a l'optimum :
c      Linearisation convexe (normalisee relaxee) en zk .
c
c      CALL DFOBJ1 (zk,gradf)
c      CALL DHCON1 (zk,gradh)
c      CALL DGCON1 (zk,gradsx,gradsy)
c      CALL SCALE (zk)
c
c      Fin de la linearisation convexe .
c
c
c      Resolution du sous-probleme spk .
c
c      Initialisation des parametres variables de E04KBF .
c
c      DO 12 I=1,NCU
          U(I) = 1.
12     CONTINUE
c
c      MAX = THEMAX()
c      FEST = -MAX
c
c      Debut d'impression du resultat intermediaire .
c
c      IF (IOUT.NE.0) THEN
          IF (JMOD(iter,iout).EQ.0) THEN
              WRITE (us,5000) ITER
          ENDIF
      ENDIF
c
c      CONTINUE
50     IFAIL = 1
c
c      CALL E04KBF (ncu,funct,monit,iprint,loesch,intype,e04lbs,
1          maxcal,eta,xtol,stermx,fest,ibound,bl,blu,u,hesl,
2          lh,hesd,vistate,fc,sc,iw,liw,w,liw,
3          ifail)
c
c      Gestion des parametres d'erreur de la routine .
c
c
c      IF (IFAIL.EQ.4) GOTO 50
c
c      Resultats primaux intermediaires .
c
c      CALL SETPRI (u,ncu)
c      CALL ECHELLE (zk)
c      CALL SAUVE (zk)
c      CALL FGH (zk,gradf,gradh,gradsx,gradsy)
c      CALL FINALE (zk,ncuz,ncu,u)
c
c      Passage a l'iteration suivante .
c
c      ITER = ITER+1
c      GOTO 10

```

```

c
c      Impression des resultats finaux .
c
302  IOUT=-1
      WRITE (us,6000)
      WRITE (us,1000) FCT
      WRITE (us,2000) ITER
      IF (US.EQ.5) THEN
          WRITE (45,*) ' '
          WRITE (45,*) 'resultat final'
          WRITE (45,*) '*****'
          WRITE (45,*) ' '
          WRITE (45,1000) FCT
          WRITE (45,2000) ITER
      ENDIF
      CALL FINALE (zk,ncuz,ncu,u)
500  CONTINUE
c
c      Calcul du temps CPU .
c
      IT = ISECONDIFF (it)
      WRITE (us,3000) IT
      IF (US.EQ.5) THEN
          WRITE (45,3000) IT
      ENDIF
c
c FORMATS
c =====
c
1000  FORMAT (1X,'nombre d'evaluations de la fonction duale ',I5)
2000  FORMAT (1X,'nombre d'evaluations de la fonction objectif',/,
1      1X,'et des contraintes primales ',I2)
3000  FORMAT (1X,'temps CPU ',I4)
4000  FORMAT (1X,/,1X,'LE RESUME DES RESULTATS DE L'EXECUTION SE
1 TROUVE DANS RESULT.OPT .',/)
5000  FORMAT (/,1X,'resultat intermediaire,iteration',I5,/,
1      1X,'-----')
6000  FORMAT (/,1X,'resultat final',/,1X,'-----')
c
      STOP
      END
c
c I.D>

```


c
c
c
c

```
*****  
* VOICI LES FICHIERS CONTENANT LES COMMON$  
* IL FAUT DONC REMPLACER LES "INCLUDE" NON-STANDARDS PAR CES  
* FICHIERS CORRESPONDANTS  
*****
```

```
parameter      ncsmax=5,nchmax=10,ncwmax=10,  
1              nezmax=35,mncxmax=5,mncshmax=10,ncumax=35,  
2              ncuzmax=70,mncsmax=5  
integer        nez,ncw,mncx,ncs,nch,mncs,mncsh,  
1              mncx1(ncsmax),ncs1(ncsmax)  
common/din/nez,ncw,mncx,ncs1,nch,ncs,ncs1,mncs,mncsh,mncx1
```

```
      f,for  
double precision dfcty,dfctyl(ncwmax),fuk  
common/f/dfcty,dfctyl,fuk
```

```
      h,for  
double precision dhcty(nchmax),dhctyl(ncwmax,nchmax),  
1              hsk(nchmax)  
common/h/dhcty,dhctyl,hsk
```

```
      g,for  
double precision dsctys(mncsmax,ncsmax),dsctyl(ncwmax,  
1              mncsmax,ncsmax),dsctxl(mncxmax,mncsmax,  
2              ncsmax),gsuk(mncsmax,ncsmax)  
common/g/dsctys,dsctyl,dsctxl,gsuk
```

```
      born,for  
double precision ubornz(ncuzmax),lbornz(ncuzmax)  
common/born/ubornz,lbornz
```

```
      rel,for  
double precision d(ncumax)  
common/rel/d
```

```
      vari,for  
double precision var(ncuzmax)  
common/var/var
```

```

      routines.for
      SUBROUTINE FBME (itermax,maxcal,eps,xtol,zk)
      *****
c
c
c Cette routine imprime les donnees du probleme dans le(s) fichier(s) de
c sortie .
c
c Commons
c -----
      include 'dim.for'
      integer iout,iou,us,iter
      common/sortie/iou,iou,us,iter
c
c Specifications
c -----
      integer          itermax,maxcal
      double precision eps,xtol,zk(ncz)
c
c Declarations
c -----
c I      : Indice d'iteration .
c NCGT  : nombre de contraintes locales .
c NCXT  : nombre de variables locales .
c
      integer i,ncgt,ncxt
c
c Instructions
c -----
c
c Calcul des nombres de contraintes locales et de variables locales .
c
      NCGT = 0
      NCXT = 0
      DO 10 I=1,NCG
          NCGT = NCGT + NCGI(I)
          NCXT = NCXT + NCXI(I)
10 CONTINUE
c
c Impression .
c
      WRITE (US,50)
      WRITE (US,100) NCGT,NCXT,NCH,NCGT
      WRITE (US,200) ITERMAX,MAXCAL,EPS
      WRITE (US,300) XTOL
      WRITE (US,400)
      WRITE (US,500) ZK
      IF (US.EQ.5) THEN
          WRITE (45,50)
          WRITE (45,100) NCGT,NCXT,NCH,NCGT
          WRITE (45,200) ITERMAX,MAXCAL,EPS
          WRITE (45,300) XTOL
          WRITE (45,400)
          WRITE (45,500) ZK
      ENDIF

```


c
c Formats
c -----

c
50 FORMAT (1X,'DONNEES DU PROBLEME :',/,/
 1 1X,'*****',/,/
100 FORMAT (1X,'NOMBRE DE VARIABLES GLOBALES :',I2,/,/
 2 1X,'NOMBRE DE VARIABLES LOCALES :',I2,/,/
 4 1X,'NOMBRE DE CONTRAINTES GLOBALES :',I2,/,/
 6 1X,'NOMBRE DE CONTRAINTES LOCALES :',I2)
200 FORMAT (1X,'NOMBRE MAXIMAL D'ITERATIONS PRIMALES:',I2,/,/
 1 1X,'NOMBRE MAXIMAL D'ITERATIONS DUALES :',I5,/,/
 3 1X,'PRECISION DESIREE POUR LE TEST DE KUHN-
 ITUCKER :',D10,3)
300 FORMAT (1X,'PRECISION DUALE :',D10,3)
400 FORMAT (/,1X,'POINT DE DEPART PRIMAL :',/,/
 1 1X,'*****'
500 FORMAT (1X,D15,5)
c
 RETURN
 END

c

```

SUBROUTINE MENU(name1,name2,name3,ue1,ue2,eps,xtol)
c *****
c Cette routine presente le programme et sort suivant le choix
c de l'utilisateur :
c     -name1 : nom du fichier de donnees contenant les informations
c             generales du probleme .
c     -name2 : nom du fichier de donnees contenant
c             les valeurs primales de depart .
c     -name3 : nom du fichier devant contenir les resultats .
c Elle permet donc le choix du mode d'entree/sortie .
c
c Common
c -----
c     integer iout,jout,us,iter
c     common/sortie/iout,jout,us,iter
c
c Specifications
c -----
c     character*30      name1,name2,name3
c     integer          ue1,ue2
c     double precision eps,xtol
c
c Instructions
c -----
c
c     Impression du probleme .
c
c     WRITE (5,100)
c     WRITE (5,200)
c     WRITE (5,10)
c     READ (5,20) NAME1
c
c     Lecture du nom du fichier contenant les donnees generales .
c
c     WRITE (5,299)
c     WRITE (5,300)
c     WRITE (5,301)
c     READ (5,*) UE1
c     IF (UE1.NE.5) THEN
c         WRITE (5,400)
c         READ (5,'(A)') NAME1
c     ENDIF
c
c     Lecture du nom de fichier de donnees primales .
c
c     WRITE (5,450)
c     READ (5,*) UE2
c     IF (UE2.NE.5) THEN
c         WRITE (5,460)
c         READ (5,'(A)') NAME2
c     ENDIF
c
c     Lecture du nom du fichier contenant les resultats .
c
c     WRITE (5,500)
c     READ (5,*) US
c     IF (US.NE.5) THEN
c         WRITE (5,600)
c         READ (5,'(A)') NAME3
c     ENDIF

```

```

c
c   Lecture de la precision du test de Kuhn-Tucker, de la precision
c   duale et du niveau d'impression .
c
WRITE (5,700)
READ (5,*) EPS
WRITE (5,800)
READ (5,*) XTOL
WRITE (5,900)
READ (5,*) IOUT
WRITE(5,950)
READ(5,*) JOUT

c
c formats
c -----
c
100  FORMAT(1X,'MEMOIRE')
200  FORMAT(1X,'CE PROGRAMME RESOUT LE PROBLEME'/Y/Y/
1 1X,'      MIN F (g1,...,gP)          'Y/Y/Y/Y/
2 1X,'      SOUS Hk (g1,g2,...,gP) <= 0    'Y/Y/Y/
3 1X,'      GiJ (g1,...,gP,x1i,...,xim(i)) <= 0 'Y/Y/Y/
4 1X,'      0 < Lgh <= gh <= Ugh          'Y/Y/Y/
5 1X,'      0 < Lxil <= xil <= Uxil       'Y/Y/Y/
6 1X,'      OU      1 <= i <= s          'Y/Y/Y/
7 1X,'      1 <= j <= n(i)              'Y/Y/Y/
8 1X,'      1 <= h <= t                  'Y/Y/Y/
9 1X,'      1 <= l <= m(i)              'Y/)
10   FORMAT(1X,'PRESSEZ UNE TOUCHE POUR CONTINUER')
20   FORMAT(A1)
299  FORMAT('1')
300  FORMAT('1')
301  FORMAT(1X,'LA LECTURE DES DONNEES DU PROBLEME SE FAIT PAR FICHER ',
1     1X,'OU PAR ECRAN .',
2     /Y1X,'TAPEZ LE NUMERO DU FICHER (5 POUR ECRAN) :')
c
450  FORMAT(1X,'LA LECTURE DES VARIABLES PRIMALES SE FAIT PAR FICHER ',
1     1X,'OU PAR ECRAN .',
2     /Y1X,'TAPEZ LE NUMERO DU FICHER (5 POUR ECRAN) :')
c
400  FORMAT(1X,'DONNEZ LE NOM DU FICHER CONTENANT LES DONNEES :')
c
460  FORMAT(1X,'DONNEZ LE NOM DU FICHER CONTENANT LES VAR PRIMALES
1:')
c
500  FORMAT(1X,'L'IMPRESSON DES RESULTATS SE FAIT PAR FICHER ',
1     1X,'OU PAR ECRAN .',
2     /Y1X,'TAPEZ LE NUMERO DU FICHER (5 POUR ECRAN) :')
c
600  FORMAT(1X,'DONNEZ LE NOM DU FICHER DEVANT CONTENIR LES RESULTA
1TS :')
700  FORMAT(1X,'DONNEZ LA PRECISION DU TEST DE KHUN-TUCKER :')
800  FORMAT (1X,'DONNEZ LA PRECISION DUALE :')
900  FORMAT (1X,'DONNEZ LE NIVEAU D'IMPRESSON PRIMAL:'Y/Y/
1     1X,'0 ---> RESULTAT FINAL'Y/Y/
2     1X,'1 ---> TOUTES LES ITERATIONS + RESULTAT FINAL'Y/Y/
3     1X,'I ---> TOUTES LES I ITERATIONS + RESULTAT FINAL')
950  FORMAT (1X,'DONNEZ LE NIVEAU D'IMPRESSON DUAL:'Y/Y/
1     1X,'0 ---> AUCUNE IMPRESSON'Y/Y/
2     1X,'1 ---> RESULTAT FINAL A CHAQUE ITERATION PRIMALE',
3     /Y/
3     1X,'2 ---> TOUTES LES ITERATIONS ')

RETURN
END
c

```

SUBROUTINE FINALE(zk,ncuz,ncu,u)

```

*****

```

```

c Cette routine imprime les resultats .

```

```

c Commons

```

```

c -----
include      'dim.for'
include      'f.for'
include      'vari.for'
integer      iout,jout,us,iter
common/sortie/iout,jout,us,iter

```

```

c Specifications

```

```

c -----
integer      ncuz,ncu
double precision zk(ncz),u(ncu)

```

```

c Declarations

```

```

c -----
c I,J      : Indices d'iteration .
c TOT      : Nombre intermediaire de contraintes .
c JMOD      : Fonction calculant le reste d'une division entiere .
integer      i,j,tot,jmod

```

```

c Instructions

```

```

c -----
c Si IOUT=0 pas d'impression dans US .

```

```

c IF (IOUT.EQ.0) GOTO 1000

```

```

c Verification qu'on est a une iteration imprimable .

```

```

c IF (JMOD(iter,iout).EQ.0.OR.IOUT.LT.0) THEN

```

```

c Impression de la valeur de la fonction objectif .

```

```

WRITE (US,*)
WRITE (US,*) 'LA VALEUR DE LA FONCTION OBJECTIF EST:'
WRITE (US,100) FYK

```

```

c Impression des points-solutions .

```

```

DO 200 I=1,NCY
  WRITE (US,300) I , ZK(I)
200 CONTINUE
TOT=NCY
DO 500 I=1,NCG
  DO 400 J=1,NCXI(I)
    WRITE (US,600) I,J,ZK(TOT+J)
400 CONTINUE
TOT=TOT+NCXI(I)
500 CONTINUE

```

```

c Impression des variables artificielles .

```

```

WRITE (US,*)
WRITE (US,*) 'LA VALEUR DES VARIABLES ARTIFICIELLES EST:'
J=1
DO 10 I=TOT+1,NCUZ
  WRITE (US,700) J,VAR(I)
  J=J+1
10 CONTINUE

```


Impression des multiplicateurs de Lagrange .

```
WRITE (US,*)  
WRITE (US,*) 'LES MULTIPLICATEURS DE LAGRANGE VALENT:'  
DO 800 I=1,NCU  
  WRITE (US,900) I,U(I)  
CONTINUE  
WRITE (US,*)
```

Impression du resultat final dans result.out .

```
IF (US.EQ.5.AND.IOUT.LT.0) THEN
```

Impression de la valeur de la fonction objectif .

```
WRITE (45,*)  
WRITE (45,*) 'LA VALEUR DE LA FONCTION OBJECTIF EST:'  
WRITE (45,100) FYK
```

Impression des points-solutions .

```
DO 201 I=1,NCY  
  WRITE (45,300) I , ZK(I)  
CONTINUE  
TOT=NCY  
DO 501 I=1,NCG  
  DO 401 J=1,NCXI(I)  
    WRITE (45,600) I,J,ZK(TOT+J)  
  CONTINUE  
  TOT=TOT+NCXI(I)  
CONTINUE
```

Impression des variables artificielles .

```
WRITE (45,*)  
WRITE (45,*) 'LA VALEUR DES VARIABLES ARTIFICIELLES EST:'  
J=1  
DO 11 I=TOT+1,NCUZ  
  WRITE (45,700) J,VAR(I)  
  J=J+1  
CONTINUE
```

Impression des multiplicateurs de Lagrange .

```
WRITE (45,*)  
WRITE (45,*) 'LES MULTIPLICATEURS DE LAGRANGE VALENT:'  
DO 801 I=1,NCU  
  WRITE (45,900) I,U(I)  
CONTINUE  
WRITE (45,*)
```

```
  ENDIF  
ENDIF
```

Formats

```
100  FORMAT (1X,D20.10)  
300  FORMAT (1X,'Y(',I2,')=',D20.10)  
600  FORMAT (1X,'X(',I2,')=',D20.10)  
700  FORMAT (1X,'Z(',I2,')=',D20.10)  
900  FORMAT (1X,'U(',I2,')=',D20.10)  
1000 CONTINUE  
      RETURN  
      END
```

```

SUBROUTINE FUNCT( iflag, ncu, u, theta, sc, iw, liw, w, lw)
*****
c
c Cette routine calcule la valeur et les gradients de
c -(moins) la fonction duale . Elle est appelee a l'interieur
c de E04KBF .
c
c Commons
c -----
      include 'dim.for'
      include 'vari.for'
      include 'f.for'
      include 'h.for'
      include 's.for'
      include 'rel.for'
      include 'born.for'
      integer iout, jout, us, iter, fct
      common /sortie/ iout, jout, us, iter, fct
c
c Specifications
c -----
      integer iflag, ncu, liw, iw( liw ), lw
      double precision u( ncu ), theta, sc( ncu ), w( lw )
c
c Declarations
c -----
c I, J, K : Indices d'iterations .
c TOT : Nombre intermediaire de contraintes .
c NCTOT : Nombre intermediaire de variables .
      integer i, j, k, tot, nctot
c
c Instructions
c -----
c
c Incrementation du nombre d'appels a FUNCT .
c
c FCT = FCT + 1
c
c Evaluation des variables primales .
c
c CALL SETPRI( u, ncu )
c
c Evaluation des gradients de la fonction duale :
c
c Evaluation de la valeur des contraintes H .
c
      DO 10 I=1, NCH
         GC( I ) = DHCTY( I ) - VAR( NCZ + I )
         DO 20 J=1, NCY
            IF ( DHCTY1( J, I ) .GT. 0 ) THEN
               GC( I ) = GC( I ) + DHCTY1( J, I ) * VAR( J )
            ELSE
               GC( I ) = GC( I ) - DHCTY1( J, I ) / VAR( J )
            ENDIF
         20 CONTINUE
      10 CONTINUE

```


Evaluation des valeurs des contraintes G .

```

NCTOT=NCY
TOT=NCH
DO 30 I=1,NCG
  DO 40 J=1,NCGI(I)
    GC(TOT+J)=DGCTXY(J,I)-VAR(NCZ+TOT+J)
    DO 50 K=1,NCY
      IF (DGCTY1(K,J,I).GT.0) THEN
        GC(TOT+J)=GC(TOT+J)+DGCTY1(K,J,I)*VAR(K)
      ELSE
        GC(TOT+J)=GC(TOT+J)-DGCTY1(K,J,I)/VAR(K)
      ENDIF
    CONTINUE
  DO 60 K=1,NCXI(I)
    IF (DGCTX1(K,J,I).GT.0) THEN
      GC(TOT+J)=GC(TOT+J)+DGCTX1(K,J,I)*VAR(NCTOT+K)
    ELSE
      GC(TOT+J)=GC(TOT+J)-DGCTX1(K,J,I)/VAR(NCTOT+K)
    ENDIF
  CONTINUE
CONTINUE
NCTOT=NCTOT+NCXI(I)
TOT=TOT+NCGI(I)
CONTINUE

```

Fin des calculs des gradients de la fonction duale .

Gradients de -la fonction duale .

```

DO 5 I=1,NCU
  GC(I)=-GC(I)
CONTINUE

```

Calcul de -la fonction duale

```

THETA=-DFCTY
DO 70 J=1,NCY
  IF (DFCTY1(J).GT.0) THEN
    THETA=THETA-DFCTY1(J)*VAR(J)
  ELSE
    THETA=THETA+DFCTY1(J)/VAR(J)
  ENDIF
CONTINUE
TOT=NCZ
DO 80 I=1,NCU
  THETA=THETA+U(I)*GC(I)-D(I)*VAR(TOT+I)-D(I)*(VAR(TOT+I)**2)
CONTINUE
RETURN
END

```

```
SUBROUTINE MONIT(n,xc,fc,gc,istate,SP,Jrnm,cond,Posdef,niter,
1 nf,liw,liw,liw)
```

```
*****
```

```
c Cette routine effectue les impressions a l'interieur de E04KBF .
```

```
c Commons
```

```
c -----
```

```
include 'dim.for'
include 'vari.for'
integer iout,jout,us,iter,fc
common/sortie/iout,jout,us,iter,fc
```

```
c Specifications
```

```
c -----
```

```
logical Posdef
integer n,istate(n),niter,nf,liw,iw(liw),lw
double precision xc(n),fc,gc(n),SP,Jrnm,cond,w(liw)
```

```
c Instructions
```

```
c -----
```

```
c Impression du nombre d'iterations duales .
```

```
IF (IOUT.EQ.0) GOTO 50
IF (JBOD(iter,iout).EQ.0) THEN
  IF (JOUT.NE.0) THEN
    IF (JOUT.EQ.2) THEN
      WRITE (US,1101) NITER
    ELSE
      WRITE (US,1100) NITER
    ENDIF
    WRITE (US,1200) XC
    WRITE (US,1300) FC
    WRITE (US,1400) GC
  ELSE
    WRITE (US,1000) NITER
  ENDIF
ENDIF
```

```
c Format
```

```
c -----
```

```
1000 FORMAT (1X,'NOMBRE D'ITERATIONS DUALES :',I5)
1100 FORMAT (1X,'nombre d'iterations duales ',I5,/,
1 ' /-----' /)
1101 FORMAT (1X,'iteration duale',I5,/,
1 ' /-----' /)
1200 FORMAT (1X,'variables duales',/(1X,d20.6))
1300 FORMAT (1X,'fonction duale',/1X,d20.6)
1400 FORMAT (1X,'gradients duaux',/(1X,d20.6))
```

```
50 CONTINUE
```

```
c RETURN
c END
```

```

SUBROUTINE INITIAL(ue1,name1,itermax)
c *****
c
c Cette routine lit les donnees du probleme .
c
c Commons
c -----
c      include 'dim.for'
c      include 'born.for'
c Specifications
c -----
c      character*30      name1
c      integer          ue1,itermax
c
c Declarations
c -----
c      I,K      : Indices d'iteration .
c      NCTOT   : nombre intermediaire de variables .
c      integer i,k,nctot
c
c Instructions
c -----
c
c      Ouverture du fichier de donnees .
c
c      IF (UE1.NE.5) OPEN(UNIT=UE1,FILE=NAME1,STATUS='OLD')
c
c      Lecture de NCH et NCG .
c
c      IF (UE1.EQ.5) THEN
c          WRITE (5,100)
c      ENDIF
c      READ (UE1,*) NCH,NCG
c
c      Lecture de NCGI .
c
c      IF (UE1.EQ.5) THEN
c          WRITE (5,200)
c      ENDIF
c      DO 10 I=1,NCG
c          READ(UE1,*)NCGI(I)
10      CONTINUE
c
c      Lecture de NCY et NCXI .
c
c      IF (UE1.EQ.5) THEN
c          WRITE (5,300)
c      ENDIF
c      READ (UE1,*) NCY
c      DO 20 I=1,NCG
c          READ (UE1,*) NCXI(I)
20      CONTINUE
c
c      Calcul de NCZ .
c
c      NCZ = NCY
c      DO 30 I=1,NCG
c          NCZ = NCZ+NCXI(I)
30      CONTINUE
c

```

```

c      Lecture de LBORNZ et UBORNZ .
c
      IF (UE1.EQ.5) THEN
        WRITE (5,400)
      ENDIF
      DO 40 I=1,NCY
        READ (UE1,*) LBORNZ(I),UBORNZ(I)
40     CONTINUE
      NCTOT = NCY
      DO 60 I=1,NCG
        DO 50 K=1,NCXI(I)
          READ(UE1,*) LBORNZ(NCTOT+K),UBORNZ(NCTOT+K)
50     CONTINUE
        NCTOT = NCTOT+NCXI(I)
60     CONTINUE
c
c      Lecture de ITERMAX .
c
      IF (UE1.EQ.5) THEN
        WRITE (5,500)
      ENDIF
      READ(UE1,*) ITERMAX
c
c      Calcul de MNCG et MNCX .
c
      MNCG = 0
      MNCX = 0
      DO 70 I=1,NCG
        MNCG = JMAX0(mncg,ncxi(i))
        MNCX = JMAX0(mncx,ncxi(i))
70     CONTINUE
c
c      Calcul de MNCXY .
c
      MNCXY = JMAX0(ncg,mncx)
c
c      Calcul de MNCGH .
c
      MNCGH = JMAX0(ncg,mncg)
c
c      Formats
c      -----
c
100     FORMAT (1X,'DONNEZ LE NOMBRE DE CONTRAINTES GLOBALES ','/',
1       1X,'DONNEZ LE NOMBRE DE CONTRAINTES LOCALES .')
200     FORMAT (1X,'DONNEZ LE NOMBRE DE COMPOSANTES DE CHAQUE VECTEUR G
1i, i=1,s,')
300     FORMAT (1X,'DONNEZ LE NOMBRE DE VARIABLES GLOBALES ','/',
1       1X,'DONNEZ LE NOMBRE DE COMPOSANTES DE CHAQUE VECTEUR X
1i, i=1,s,')
400     FORMAT (1X,'DONNEZ LES BORNES INFERIEURE ET SUPERIEURE DE CHAQU
1E VARIABLE',/,1X,'EN COMMENCANT PAR LES VARIABLES GLOBALES PUI
2S LES LOCALES .')
500     FORMAT (1X,'DONNEZ LE NOMBRE MAXIMAL D''ITERATIONS PRIMALES .')
c
      RETURN
      END
c

```



```

SUBROUTINE DEPART(ue2,name2,ncu,zk)
*****
c
c Cette routine lit les valeurs initiales des variables primales
c ZK et les constantes associees aux variables artificielles.
c
c Commons
c -----
c         include      'dim.for'
c         include      'rel.for'
c
c Specifications
c -----
c         character*30      name2
c         double precision  zk(ncz)
c         integer           ue2,ncu
c
c Declarations
c -----
c I,J      : Indices d'iteration .
c NCTOT    : Nombre intermediaire de variables .
c           integer i,j,nctot
c
c Instructions
c -----
c
c         Ouverture du fichier de lecture .
c
c         IF (UE2.NE.5) OPEN(UNIT=UE2,FILE=NAME2,STATUS='OLD')
c
c         Lecture des variables primales de depart .
c
c         IF (UE2.EQ.5) THEN
c           WRITE (5,100)
c         ENDIF
c         DO 10 I=1,NCY
c           READ (UE2,*) ZK(I)
10        CONTINUE
c         NCTOT=NCY
c         DO 20 I=1,NCG
c           DO 30 J=1,NCXI(I)
c             READ(UE2,*) ZK(NCTOT+J)
30        CONTINUE
c           NCTOT=NCTOT+NCXI(I)
20        CONTINUE
c
c         Lecture des "choix de relaxation" D .
c
c         IF (UE2.EQ.5) THEN
c           WRITE (5,200)
c         ENDIF
c         DO 40 I=1,NCU
c           READ(UE2,*)D(I)
40        CONTINUE
c
c Formats
c -----
c
c 100      FORMAT (1X,'DONNEZ LES VALEURS DE DEPART AUX VARIABLES GLOBALES
c              1','/,1X,'ET ENSUITE AUX VARIABLES LOCALES .')
c 200      FORMAT (1X,'DONNEZ LES "CHOIX" DE RELAXATION DES CONTRAINTES .',
c              1','/,1X,'(GLOBALES PUIS LOCALES)')
c
c         RETURN
c         END

```

```

SUBROUTINE SETPRI(u,ncu)
c *****
c
c Cette routine calcule les valeurs des variables primales
c qui sont solution de l'infimum du lagrangien.
c
c Commens
c -----
c      include 'dim.for'
c      include 'rel.for'
c      include 'veri.for'
c      include 'f.for'
c      include 'g.for'
c      include 'h.for'
c      include 'born.for'
c
c Specifications
c -----
c      integer          ncu
c      double precision u(ncu)
c
c Declarations
c -----
c      I,J,K,L : Indices d'iterations
c      NCTOT   : Nombre intermediaire de contraintes ou de variables .
c      C,E     : Valeurs intermediaires pour le calcul de Y .
c      V,W     : Valeurs intermediaires pour le calcul des Xi .
c      AUX     : Variable auxiliaire .
c      POS     : Variable auxiliaire .
c      integer i,j,k,l,nctot
c      double precision c(ncymax),v(mncxmax,ncymax),
c      j                w(mncxmax,ncymax),aux,e(ncymax)
c      logical         pos
c
c Instructions
c -----
c
c      Calcul de C et E (calculs intermediaires) .
c
c      DO 100 L=1,NCY
c          C(L) = 0
c          E(L) = 0
c          DO 200 I=1,NCH
c              IF (DHCTY1(L,I).GT.0) THEN
c                  C(L) = C(L)+DHCTY1(L,I)*U(I)
c              ELSE
c                  E(L) = E(L)+DHCTY1(L,I)*U(I)
c              ENDIF
c          CONTINUE
c
c          NCTOT=NCH
c          DO 300 I=1,NCG
c              DO 400 J=1,NCGI(I)
c                  IF (DGCTY1(L,J,I).GT.0) THEN
c                      C(L) = C(L)+DGCTY1(L,J,I)*U(NCTOT+J)
c                  ELSE
c                      E(L) = E(L)+DGCTY1(L,J,I)*U(NCTOT+J)
c                  ENDIF
c              CONTINUE
c          NCTOT = NCTOT+NCGI(I)
c      CONTINUE
c
c      CONTINUE
c
c      CONTINUE
c
c      CONTINUE

```



```

c
c
c      Calcul de V et W (calculs intermediaires) .
POS = .FALSE.
NCTOT = NCH
DO 500 I=1,NCG
  DO 600 K=1,NCXI(I)
    W(K,I) = 0.
    V(K,I) = 0.
    DO 700 J=1,NCGI(I)
      IF (DGCTX1(K,J,I).GT.0) THEN
        W(K,I) = W(K,I)+DGCTX1(K,J,I)*U(NCTOT+J)
        POS=.TRUE.
      ELSE
        V(K,I) = V(K,I)+DGCTX1(K,J,I)*U(NCTOT+J)
      ENDIF
    CONTINUE
  CONTINUE
  NCTOT = NCTOT+NCGI(I)
500 CONTINUE

c
c      Calcul des Yk .
DO 1100 L=1,NCY
  IF (DFCTY1(L).GT.0) THEN
    IF (DFCTY1(L)+C(L).NE.0) THEN
      AUX = -E(L)/(DFCTY1(L)+C(L))
    ELSE
      VAR(L) = UBORNZ(L)
      GOTO 1101
    ENDIF
  ELSE
    IF (C(L).NE.0) THEN
      AUX = -(DFCTY1(L)+E(L))/C(L)
    ELSE
      VAR(L) = UBORNZ(L)
      GOTO 1101
    ENDIF
  ENDIF
  IF ((LBORNZ(L)**2.LE.AUX).AND.(AUX.LE.UBORNZ(L)**2)) THEN
    VAR(L) = DSQRT(AUX)
  ELSE
    IF (AUX.GE.UBORNZ(L)**2) THEN
      VAR(L) = UBORNZ(L)
    ELSE
      VAR(L) = LBORNZ(L)
    ENDIF
  ENDIF
1101 CONTINUE
1100 CONTINUE

```

c
c
c

Calcul des Xik .

```
NCTOT = NCY
DO 1200 I=1,NCG
  DO 1300 K=1,NCXI(I)
    IF (W(K,I).NE.0) THEN
      AUX = -V(K,I)/W(K,I)
      IF ((LBORNZ(NCTOT+K)**2.LE.AUX).AND.
1      (UBORNZ(NCTOT+K)**2.GE.AUX)) THEN
        VAR(NCTOT+K) = DSQRT(AUX)
      ELSE
        IF (AUX.GE.UBORNZ(NCTOT+K)**2) THEN
          VAR(NCTOT+K) = UBORNZ(NCTOT+K)
        ELSE
          VAR(NCTOT+K) = LBORNZ(NCTOT+K)
        ENDIF
      ENDIF
    ELSE
      IF (POS) THEN
        VAR(NCTOT+K) = LBORNZ(NCTOT+K)
      ELSE
        VAR(NCTOT+K) = UBORNZ(NCTOT+K)
      ENDIF
    ENDIF
  CONTINUE
  NCTOT = NCTOT+NCXI(I)
1200 CONTINUE
```

1300

1200

c
c
c

Calcul des variables artificielles .

```
DO 99 I=1,NCU
  VAR(NCZ+I) = U(I)/(2*D(I))-0.5
  IF (VAR(NCZ+I).LT.0) THEN
    VAR(NCZ+I) = 0.
  ENDIF
99 CONTINUE
c
RETURN
END
```

99

c

c

```

LOGICAL FUNCTION ARRET(eps,gradf,gradh,gradsw,gradsx,u,ncu,
1                    itermax,zk)
c *****
c
c Cette fonction est vraie quand le point courant est un point de
c Kuhn-Tucker du probleme initial .
c
c
c Commons
c -----
c      include 'dim.for'
c      include 's.for'
c      include 'h.for'
c      include 'horn.for'
c      integer iout,jout,us,iter
c      common/ sortie/iout,jout,us,iter
c
c Specifications
c -----
c      integer ncu,itermax
c      double precision gradsw(ncwmax,ncsmax,ncs),zk(ncz),
c      1 gradsx(ncxmax,ncsmax,ncs),u(ncu),
c      2 gradh(ncwmax,nch),gradf(ncu),eps
c
c Declarations
c -----
c      I,J,K : Indices d'iteration .
c      NCTOT : Nombre intermediaire de contraintes .
c      NC    : Nombre intermediaire de variables .
c      AUX   : Variables auxiliaires .
c      integer i,j,k,nctot,nc
c      double precision aux(nczmax)
c
c Instructions
c -----
c
c      Verification de l'admissibilite du point .
c
c      ARRET=.TRUE.
c      DO 99 I=1,NCH
c          ARRET = ARRET.AND.HYK(I).LT.EPS
c 99      CONTINUE
c      DO 98 I=1,NCG
c          DO 97 J=1,NCGI(I)
c              ARRET = ARRET.AND.GXYK(J,I).LT.EPS
c 97      CONTINUE
c 98      CONTINUE
c

```

Verification des conditions de Kuhn-Tucker .

```
IF (ARRET) THEN
  DO 10 I=1,NCY
    AUX(I) = GRADF(I)
    DO 30 J=1,NCH
      AUX(I) = AUX(I)+U(J)*GRADH(I,J)
30    CONTINUE
    NCTOT = NCH
    DO 40 J=1,NCG
      DO 50 K=1,NCGI(J)
        AUX(I) = AUX(I)+U(NCTOT+K)*GRADGY(I,K,J)
50    CONTINUE
    NCTOT = NCTOT+NCGI(J)
40    CONTINUE
10  CONTINUE
    NCTOT = NCH
    NC = NCY
    DO 11 J=1,NCG
      DO 12 I=1,NCXI(J)
        AUX(NC+I) = 0.
        DO 13 K=1,NCGI(J)
          AUX(NC+I) = AUX(NC+I)+U(NCTOT+K)*GRADGX(I,K,J)
13    CONTINUE
12    CONTINUE
    NC = NC+NCXI(J)
    NCTOT = NCTOT+NCGI(J)
11  CONTINUE
    DO 70 I=1,NCZ
      IF (LBORNZ(I),LT,ZK(I),AND,UBORNZ(I),GT,ZK(I)) THEN
        ARRET = ABS(AUX(I)),LT,EPS
      ELSE
        IF (LBORNZ(I),EQ,ZK(I)) THEN
          ARRET = AUX(I),GE,-EPS
        ELSE
          ARRET = AUX(I),LE,EPS
        ENDIF
      ENDIF
      IF (,NOT,ARRET) GOTO 1000
70  CONTINUE
```

Verification de la condition de complementarite.

```
    NCTOT = NCH
    DO 80 I=1,NCG
      DO 90 J=1,NCGI(I)
        AUX(1) = U(NCTOT+J)*GXJK(J,I)
        ARRET = ABS(AUX(1)),LT,EPS
        IF (,NOT,ARRET) GOTO 1000
90    CONTINUE
    NCTOT = NCTOT+NCGI(I)
80    CONTINUE
    DO 100 J=1,NCH
      AUX(1) = U(J)*HYK(J)
      ARRET = ABS(AUX(1)),LT,EPS
      IF (,NOT,ARRET) GOTO 1000
100   CONTINUE
1000  CONTINUE
```

ENDIF

A-t-on depasse le nombre maximal d'iterations ?

ARRET = ARRET.or,ITER.GT,ITERMAX

RETURN
END

```

SUBROUTINE CALNCU(ncu,ncuz)
*****
c
c Cette routine calcule le nombre total NCU de multiplicateurs de
c Lagrange i.e. le nombre de contraintes du probleme primal .
c NCUZ vaudra le nombre de contraintes NCU + le nombre de variables NCZ .
c
c Common
c -----
c      include 'dim.for'
c
c Specifications
c -----
c      integer ncu,ncuz
c
c Declarations
c -----
c      I : indice d'iteration
c      integer i
c
c Instructions
c -----
c
c      NCU = NCH
c      DO 10 I=1,NCG
c          NCU = NCU+NCGI(I)
10    CONTINUE
c      NCUZ = NCU+NCZ
c
c      RETURN
c      END

```

```

SUBROUTINE SCALE (zk)
*****
c
c Cette routine normalise les bornes du probleme et donc
c change les tableaux UBORNZ(NCZ),LBORNZ(NCZ).
c
c Commons
c -----
c      include 'dim.for'
c      include 'born.for'
c
c Specification
c -----
c      double precision zk(ncz)
c
c Declaration
c -----
c      I : Indice d'iteration ,
c      integer i
c
c Instructions
c -----
c
c      DO 10 I=1,NCZ
c          UBORNZ(I) = UBORNZ(I)/ZK(I)
c          LBORNZ(I) = LBORNZ(I)/ZK(I)
10    CONTINUE
c
c      RETURN
c      END

```


SUBROUTINE SAUVE (g)

c
c Cette routine met le vecteur VAR(NCZ) dans le vecteur Y(NCZ).

c
c Commons

c -----
c include 'dim.for'
c include 'vari.for'

c
c Specification

c -----
c double precision g(ncz)

c
c Declaration

c -----
c I : Indice d'iteration .
c integer i

c
c Instructions

c -----
c
c DO 10 I=1,NCZ
c Y(I) = VAR(I)
10 CONTINUE
c
c RETURN
c END

SUBROUTINE ECHELLE(zk)

c
c Cette routine remet a l'echelle les resultats intermediaires VAR(NCZ)
c ainsi que les bornes inferieures et superieures .

c
c Commons

c -----
c include 'dim.for'
c include 'vari.for'
c include 'born.for'

c
c Specification

c -----
c double precision zk(ncz)

c
c Declaration

c -----
c I : Indice d'iteration .
c integer i

c
c Instructions

c -----
c
c DO 10 I=1,NCZ
c VAR(I) = VAR(I)*ZK(I)
c UBORNZ(I) = UBORNZ(I)*ZK(I)
c LBORNZ(I) = LBORNZ(I)*ZK(I)
10 CONTINUE
c
c RETURN
c END

DOUBLE PRECISION FUNCTION THEMAX

c Cette fonction sort la valeur maximale de la fonction duale .
 c On la calcule a partir de la fonction objectif (DFCTY1,DFCTY,FYK)
 c et des bornes des variables primales (LBORNZ et UBORNZ).

c Commons

```

include 'dim.for'
include 'f.for'
include 'rel.for'
include 'born.for'
    
```

c Declarations

c I : Indice d'iteration .
 integer i

c Instructions

```

THEMAX = FYK
DO 100 I=1,NCY
    IF (DFCTY1(I).GT.0) THEN
        THEMAX = THEMAX+(DFCTY1(I)*UBORNZ(I))
    ELSE
        THEMAX = THEMAX-(DFCTY1(I)*(1/LBORNZ(I)))
    ENDIF
100 CONTINUE
RETURN
END
    
```

SUBROUTINE DFOBJ1 (zk,gradf)

c Cette routine calcule la constante et les facteurs de
 c linearisation convexe de la fonction objectif .

c Commons

```

include 'dim.for'
include 'f.for'
    
```

c Specifications

double precision zk(ncz),gradf(ncy)

c Declaration

c I : Indice d'iteration .
 integer i

c Instructions

```

DFCTY = FYK
DO 10 I=1,NCY
    DFCTY1(I) = GRADE(I)*ZK(I)
    DFCTY = DFCTY-DABS(DFCTY1(I))
10 CONTINUE
RETURN
END
    
```

```

SUBROUTINE DGCON1 (zk,gradsx,gradsy)
c *****
c
c Cette routine calcule la constante et les facteurs de
c linearisation convexe des contraintes GiJ.
c
c Commons
c -----
c      include 'dim.for'
c      include 's.for'
c
c Specifications
c -----
c      double precision zk(ncz),gradsx(mncxmax,mncsmax,ncs),
c      1          gradsy(ncymax,mncsmax,ncs)
c
c Declarations
c -----
c      I,J,K : Indices d'iteration .
c      NC    : Nombre intermediaire de variables .
c      integer      i,j,k,nc
c
c Instructions
c -----
c
      NC = NCY
      DO 10 I=1,NCG
         DO 20 J=1,NCGI(I)
            DGCTXY(J,I) = GXYK(J,I)
            DO 30 K=1,NCY
               DGCTY1(K,J,I) = GRADGY(K,J,I)*ZK(K)
               DGCTXY(J,I) = DGCTXY(J,I)-DABS(DGCTY1(K,J,I))
30          CONTINUE
            DO 40 K=1,NCXI(I)
               DGCTX1(K,J,I) = GRADGX(K,J,I)*ZK(NC+K)
               DGCTXY(J,I) = DGCTXY(J,I)-DABS(DGCTX1(K,J,I))
40          CONTINUE
20          CONTINUE
         NC = NC+NCXI(I)
10          CONTINUE
c
      RETURN
      END
EIJ>

```

SUBROUTINE DHCON1 (zk,gradh)

c
c
c Cette routine calcule la constante et les facteurs de
c linearisation convexe des contraintes hi .

c
c Commons

c -----

include 'dim.for'

include 'h.for'

c
c Specifications

c -----

double precision zk(ncz),gradh(ncymax,nch)

c
c Declarations

c -----

c I,J : Indices d'iteration .
c integer i,j

c
c Instructions

c -----

DO 10 I=1,NCH

DHCTY(I) = HYK(I)

DO 20 J=1,NCY

DHCTY1(J,I) = GRADH(J,I)*ZK(J)

DHCTY(I) = DHCTY(I)-DABS(DHCTY1(J,I))

20 CONTINUE

10 CONTINUE

c
c RETURN

END

RESULTATS NUMERIQUES

La plupart de nos exemples sont tirés ou du moins inspirés de "*Test Examples for nonlinear programming codes*" (W. HOCK et K. Schittkowski) - Springer-Verlag Berlin Heidelberg New York 1981 .

Dans l'exemple 34 , nous avons remarqué que lors du calcul de la variable primale x en fonction des variables duales, nous pouvions obtenir une division $0/0$. Dans ce problème nous avons fixé x à sa borne inférieure ce qui nous a amenés à la solution .Par contre,cette stratégie appliquée au dernier exemple entraîne une non-convergence de la méthode alors que fixer x à sa borne supérieure mène à la solution .

res1.dat
ENONCE DU PROBLEME :

MINIMISER $Y1 + 4 Y2$
SOUS $Y1 + Y2 + X1 - 6 \leq 0$
 $Y1 - Y2 \leq 0$
 $-3 Y1 + 2 Y2 + 1 \leq 0$
 $.5 \leq Y1 \leq 4$
 $.5 \leq Y2 \leq 4$
 $.5 \leq X1 \leq 2$

DONNEES DU PROBLEME :

NOMBRE DE VARIABLES GLOBALES : 2
NOMBRE DE VARIABLES LOCALES : 1
NOMBRE DE CONTRAINTES GLOBALES : 2
NOMBRE DE CONTRAINTES LOCALES : 1
NOMBRE MAXIMAL D'ITERATIONS PRIMALES:98
NOMBRE MAXIMAL D'ITERATIONS DUALES : 300
PRECISION DESIREE POUR LE TEST DE KUHN-TUCKER : 0.100D-03
PRECISION DUALE : 0.100D-07

POINT DE DEPART PRIMAL :

0.20000D+01
0.20000D+01
0.20000D+01

resultat final

nombre d'évaluations de la fonction duale 226
nombre d'évaluations de la fonction objectif
et des contraintes primales 8

LA VALEUR DE LA FONCTION OBJECTIF EST:
0.5000000000D+01
Y(1)= 0.1000000000D+01
Y(2)= 0.9999999999D+00
X(1) = 0.5000000000D+00

LA VALEUR DES VARIABLES ARTIFICIELLES EST:
Z(1)= 0.0000000000D+00
Z(2)= 0.0000000000D+00
Z(3)= 0.0000000000D+00

LES MULTIPLICATEURS DE LAGRANGE VALENT:
U(1)= 0.1399990196D+02
U(2)= 0.4999961397D+01
U(3)= 0.0000000000D+00

temps CPU 56
EJ>

res2.dat
ENONCE DU PROBLEME :

MINIMISER -Y1

SOUS -Y1-X1+7 <= 0
 X1 - 10 <= 0
 .5 <= Y1 <= 10
 .001 <= X1 <= 4

DONNEES DU PROBLEME :

NOMBRE DE VARIABLES GLOBALES : 1
NOMBRE DE VARIABLES LOCALES : 1
NOMBRE DE CONTRAINTES GLOBALES : 1
NOMBRE DE CONTRAINTES LOCALES : 1
NOMBRE MAXIMAL D'ITERATIONS PRIMALES:20
NOMBRE MAXIMAL D'ITERATIONS DUALES : 200
PRECISION DESIREE POUR LE TEST DE KUHN-TUCKER : 0.100D-05
PRECISION DUALE : 0.100D-12

POINT DE DEPART PRIMAL :

0.20000D+01

0.20000D+01

resultat final

nombre d'évaluations de la fonction duale 18
nombre d'évaluations de la fonction objectif
et des contraintes primales 2

LA VALEUR DE LA FONCTION OBJECTIF EST:

-0.1000000000D+02

Y(1)= 0.1000000000D+02

X(1) 1)= 0.4000000000D+01

LA VALEUR DES VARIABLES ARTIFICIELLES EST:

Z(1)= 0.0000000000D+00

Z(2)= 0.0000000000D+00

LES MULTIPLICATEURS DE LAGRANGE VALENT:

U(1)= 0.1000000000D+01

U(2)= 0.0000000000D+00

temps CPU 39

[[>

ENONCE DU PROBLEME : (inspire du probleme 18 p.41)

MINIMISER .01 Y1**2 + Y2**2
 SOUS 25 - Y1*Y2 <= 0
 Y1**2 - Y2**2 + X1 <= 0
 2 <= Y1 <= 50
 .5 <= Y2 <= 50
 24.9 <= X1 <= 25

DONNEES DU PROBLEME :

NOMBRE DE VARIABLES GLOBALES : 2
 NOMBRE DE VARIABLES LOCALES : 1
 NOMBRE DE CONTRAINTES GLOBALES : 1
 NOMBRE DE CONTRAINTES LOCALES : 1
 NOMBRE MAXIMAL D'ITERATIONS PRIMALES:20
 NOMBRE MAXIMAL D'ITERATIONS DUALES : 200
 PRECISION DESIREE POUR LE TEST DE KUHN-TUCKER : 0.100D-02
 PRECISION DUALE : 0.100D-06

POINT DE DEPART PRIMAL :

0.20000D+01
 0.20000D+01
 0.25000D+02

resultat final

 nombre d'evaluations de la fonction duale 432
 nombre d'evaluations de la fonction objectif
 et des contraintes primales 11

LA VALEUR DE LA FONCTION OBJECTIF EST:

0.5000000187D+01
 Y(1)= 0.1581138860D+02
 Y(2)= 0.1581138860D+01
 X(1) 1)= 0.2490000000D+02

LA VALEUR DES VARIABLES ARTIFICIELLES EST:

Z(1)= 0.0000000000D+00
 Z(2)= 0.0000000000D+00

LES MULTIPLICATEURS DE LAGRANGE VALENT:

U(1)= 0.1999551243D+00
 U(2)= 0.0000000000D+00

temps CPU 52

res19.dat
ENONCE DU PROBLEME : (inspire du probleme 19 p.42)

MINIMISER (Y1-10)**3 + (Y2-20)**3
SOUS ((Y2-5)**2) + ((Y1-6)**2) - 82.81 <= 0
-((Y1-5)**2) - ((Y2-5)**2) + X1 <= 0
13 <= Y1 <= 100
.001 <= Y2 <= 100
99.9 <= X1 <= 100

DONNEES DU PROBLEME :

NOMBRE DE VARIABLES GLOBALES : 2
NOMBRE DE VARIABLES LOCALES : 1
NOMBRE DE CONTRAINTES GLOBALES : 1
NOMBRE DE CONTRAINTES LOCALES : 1
NOMBRE MAXIMAL D'ITERATIONS PRIMALES:20
NOMBRE MAXIMAL D'ITERATIONS DUALES : 200
PRECISION DESIREE POUR LE TEST DE KUHN-TUCKER : 0.100D-03
PRECISION DUALE : 0.100D-07

POINT DE DEPART PRIMAL :

0.20100D+02
0.58400D+01
0.10000D+03

resultat final

nombre d'evaluations de la fonction duale 662
nombre d'evaluations de la fonction objectif
et des contraintes primales 15

LA VALEUR DE LA FONCTION OBJECTIF EST:
-0.7070474871D+04
Y(1)= 0.1404500000D+02
Y(2)= 0.7470039972D+00
X(1; 1)= 0.9990000000D+02

LA VALEUR DES VARIABLES ARTIFICIELLES EST:
Z(1)= 0.0000000000D+00
Z(2)= 0.0000000000D+00

LES MULTIPLICATEURS DE LAGRANGE VALENT:
U(1)= 0.1207043967D+04
U(2)= 0.1076308651D+04

temps CPU 730
EJ>


```
[[> . res22.dat
ENONCE DU PROBLEME : ( inspire du probleme 22 p.45 )
*****
MINIMISER ( Y1 - 2 )**2 + ( Y2 - 1 )**2
SOUS      Y1**2 - Y2
          Y1 + Y2 - X1
          .5 <= Y1 <= 4
          .5 <= Y2 <= 4
          1.9 <= X1 <= 2
```

```
DONNEES DU PROBLEME :
*****
```

```
NOMBRE DE VARIABLES GLOBALES : 2
NOMBRE DE VARIABLES LOCALES : 1
NOMBRE DE CONTRAINTES GLOBALES : 1
NOMBRE DE CONTRAINTES LOCALES : 1
NOMBRE MAXIMAL D'ITERATIONS PRIMALES:20
NOMBRE MAXIMAL D'ITERATIONS DUALES : 200
PRECISION DESIREE POUR LE TEST DE KUHN-TUCKER : 0.100D-03
PRECISION DUALE : 0.100D-07
```

```
POINT DE DEPART PRIMAL :
```

```
*****
      0.20000D+01
      0.20000D+01
      0.20000D+01
```

```
resultat final
```

```
-----
nombre d'evaluations de la fonction duale      73
nombre d'evaluations de la fonction objectif
et des contraintes primales      4
```

```
LA VALEUR DE LA FONCTION OBJECTIF EST:
```

```
      0.10000000001D+01
Y( 1)=      0.9999999996D+00
Y( 2)=      0.9999999995D+00
X( 1; 1)=      0.2000000000D+01
```

```
LA VALEUR DES VARIABLES ARTIFICIELLES EST:
```

```
Z( 1)=      0.0000000000D+00
Z( 2)=      0.0000000000D+00
```

```
LES MULTIPLICATEURS DE LAGRANGE VALENT:
```

```
U( 1)=      0.6666666681D+00
U( 2)=      0.6666666670D+00
```

```
temps CPU      33
```

```
[[>
```


res34.dat
ENONCE DU PROBLEME : (inspire du probleme 34 p.57)

MINIMISER -Y1

SOUS -X11 + EXP(Y1) <= 0
 -X12 + EXP(X11) <= 0
 .001 <= Y1 <= 20
 .001 <= X11 <= 20
 .001 <= X12 <= 10

DONNEES DU PROBLEME :

NOMBRE DE VARIABLES GLOBALES : 1
NOMBRE DE VARIABLES LOCALES : 2
NOMBRE DE CONTRAINTES GLOBALES : 1
NOMBRE DE CONTRAINTES LOCALES : 2
NOMBRE MAXIMAL D'ITERATIONS PRIMALES:20
NOMBRE MAXIMAL D'ITERATIONS DUALES : 300
PRECISION DESIREE POUR LE TEST DE KUHN-TUCKER : 0.100D-02
PRECISION DUALE : 0.100D-07

POINT DE DEPART FRIMAL :

0.10000D+01
0.10500D+01
0.29000D+01

resultat final

nombre d'evaluations de la fonction duale 113
nombre d'evaluations de la fonction objectif
et des contraintes primales 5

LA VALEUR DE LA FONCTION OBJECTIF EST:

-0.8340324455D+00
Y(1)= 0.8340324455D+00
X(1; 1)= 0.2302585095D+01
X(1; 2)= 0.1000000000D+02

LA VALEUR DES VARIABLES ARTIFICIELLES EST:

Z(1)= 0.0000000000D+00
Z(2)= 0.0000000000D+00
Z(3)= 0.0000000000D+00

LES MULTIPLICATEURS DE LAGRANGE VALENT:

U(1)= 0.0000000000D+00
U(2)= 0.4342979539D+00
U(3)= 0.4342937457D-01

temps CPU 32

EJ>

res3/.dat
DONNEES DU PROBLEME : (inspire du probleme 37 p.60)

NOMBRE DE VARIABLES GLOBALES : 3
NOMBRE DE VARIABLES LOCALES : 1
NOMBRE DE CONTRAINTES GLOBALES : 1
NOMBRE DE CONTRAINTES LOCALES : 1
NOMBRE MAXIMAL D'ITERATIONS PRIMALES:20
NOMBRE MAXIMAL D'ITERATIONS DUALES : 200
PRECISION DESIREE POUR LE TEST DE KUHN-TUCKER : 0.1000-02
PRECISION DUALE : 0.1000-05

POINT DE DEPART PRIMAL :

0.100000D+02
0.100000D+02
0.100000D+02
0.200000D+04

resultat final

nombre d'evaluations de la fonction duale 581
nombre d'evaluations de la fonction objectif
et des contraintes primales 18

LA VALEUR DE LA FONCTION OBJECTIF EST:

-0.3470419982D+04
Y(1)= 0.2403324856D+02
Y(2)= 0.1201668781D+02
Y(3)= 0.1201668781D+02
X(1; 1)= 0.7210000000D+02

LA VALEUR DES VARIABLES ARTIFICIELLES EST:

Z(1)= 0.0000000000D+00
Z(2)= 0.0000000000D+00

LES MULTIPLICATEURS DE LAGRANGE VALENT:

U(1)= 0.0000000000D+00
U(2)= 0.1444002785D+03

temps CPU 75
EJ>

ENONCE DU ^{resultat} PROBLEME :

MINIMISER 1/Y1

SOUS $-1/Y1 \leq 0$

$-1/Y1 - 1/X1 + 3 \leq 0$

$.25 \leq Y1 \leq .5$

$.25 \leq X1 \leq 2$

DONNEES DU PROBLEME :

NOMBRE DE VARIABLES GLOBALES : 1

NOMBRE DE VARIABLES LOCALES : 1

NOMBRE DE CONTRAINTES GLOBALES : 1

NOMBRE DE CONTRAINTES LOCALES : 1

NOMBRE MAXIMAL D'ITERATIONS PRIMALES:20

NOMBRE MAXIMAL D'ITERATIONS DUALES : 200

PRECISION DESIREE POUR LE TEST DE KUHN-TUCKER : 0.100D-03

PRECISION DUALE : 0.100D-07

POINT DE DEPART PRIMAL :

0.50000D+00

0.50000D+00

resultat final

nombre d'évaluations de la fonction duale 3

nombre d'évaluations de la fonction objectif

et des contraintes primales 1

LA VALEUR DE LA FONCTION OBJECTIF EST:

0.2000000000D+01

Y(1)= 0.5000000000D+00

X(1)= 0.2500000000D+00

LA VALEUR DES VARIABLES ARTIFICIELLES EST:

Z(1)= 0.0000000000D+00

Z(2)= 0.0000000000D+00

LES MULTIPLICATEURS DE LAGRANGE VALENT:

U(1)= 0.0000000000D+00

U(2)= 0.0000000000D+00

temps CPU 18

[]>

CHAPITRE 6 :

UNE AUTRE METHODE DE RESOLUTION

OPTIMISATION A 2 NIVEAUX

A) PRINCIPE GENERAL

La méthode proposée dans l'article de J-F Barthélémy [Ref 6] comporte deux phases :

La première consiste à générer une suite de sous-problèmes linéarisés convexes $(SP)^{(k)}$ - comme dans le chapitre 5 - .

La seconde phase consiste à résoudre chaque sous-problème $(SP)^{(k)}$ par décomposition . C'est ici que se situe la différence avec les méthodes à un seul niveau .

Ce chapitre est donc consacré au développement de cette seconde phase ; nous le concluerons par l'algorithme proposé par J.F. Barthelemy .

B) RESOLUTION DE (SP)^(k)

1) DECOMPOSITION

Chaque sous-problème linéarisé convexe est de la forme :

$$(SP)^{(k)} \left\{ \begin{array}{l} \text{Min } \tilde{f}(y; y^{(k)}) \\ \text{sous } h_j \tilde{(y; y^{(k)})} \leq 0 \quad (j=1..nch) \\ g_{ij} \tilde{(y, x; y^{(k)}, x^{(k)})} \leq 0 \\ \quad \quad \quad \quad \quad \quad \quad \quad (i=1..ncg, j=1..ncgi(i)) \\ 0 < y^l \leq y \leq y^u \\ 0 < x_i^l \leq x_i \leq x_i^u \quad (i=1..ncg) \end{array} \right.$$

(SP)^(k) va être décomposé en sous-problèmes dits
"de deux niveaux":

--->ncg sous-problèmes de **bas niveau**

((SN1)^(k))⁽ⁱ⁾ i=1..ncg dont les variables sont
x_i : i-èmes **variables locales** du problème initial .

--->1 sous-problème de **haut niveau**

(SN2)^(k) dont les variables sont
y : **variables globales** du problème initial .

2) LA FONCTION DE KREISSELMEIER - STEINHAUSER

La première idée de cette méthode est de remplacer les vecteurs de contraintes locales

$$g_i \sim(Y, X_i; Y^{(k)}, X_i^{(k)}) = \begin{bmatrix} g_{i1} \sim(Y, X_i; Y^{(k)}, X_i^{(k)}) \\ \vdots \\ g_{incg(i)} \sim(Y, X_i; Y^{(k)}, X_i^{(k)}) \end{bmatrix}$$

par une fonction scalaire C_i mesurant la violation de ces contraintes :

$$C_i(Y, X_i) = (1/\rho) \cdot \ln \left\{ \sum_{j=1..ncg(i)} \rho g_{ij} \sim(Y, X_i; Y^{(k)}, X_i^{(k)}) \right\}$$

Cette fonction est dite de Kreisselmeier - Steinhauser **[Ref.7]**.
Le choix du paramètre ρ est discuté en ANNEXE A 6.2.

Propriétés de la fonction :

- 1) $C_i(Y, X_i)$ est conservative par rapport aux contraintes $g_{ij} \sim$
- 2) $C_i(Y, X_i)$ préserve la convexité des sous-problèmes

(Démonstrations en ANNEXE A 6.1)

3) LES SOUS-PROBLEMES DE BAS NIVEAU

Pour la décomposition du $k^{\text{ème}}$ sous-problème, on a ncg sous-sous-problèmes non contraints en fixant $Y = Y^{(k)}(j)$ où (j) représente l'itération courante (encore appelée cycle) de la résolution du problème $(SP)^{(k)}$.

Lors du $j^{\text{ème}}$ cycle $(SP)^{(k)}$ devient :

$$\text{Min } F \sim(Y^{(k)}(j); Y^{(k)})$$

$$\text{Sous } \begin{cases} h \sim(Y^{(k)}(j); Y^{(k)}) \leq 0 \\ g_i \sim(Y^{(k)}(j), X_i; Y^{(k)}, X_i^{(k)}) \leq 0 \quad (i=1..ncg) \end{cases}$$

$$\begin{cases} Y^l \leq Y^{(k)} \leq Y^u \\ X_i^l \leq X_i^{(k)} \leq X_i^u \quad (i=1..ncg) \end{cases}$$

Ce problème se résume aux systèmes d'inéquations suivants :

$$(1) \quad \begin{cases} g_i^{\sim}(Y^{(k)}(j), X_i; Y^{(k)}, X_i^{(k)}) \leq 0 \\ X_i^l \leq X_i^{(k)} \leq X_i^u \end{cases} \quad (i=1..ncg)$$

remplacés par :

$$((SN1)^{(k)})^{(i)} \quad \begin{cases} \text{Min } C_i(Y^{(k)}(j), X_i) \\ \text{Sous } X_i^l \leq X_i^{(k)} \leq X_i^u \end{cases} \quad (i=1..ncg)$$

où $C_i(Y^{(k)}(j), X_i)$ est la fonction de Kreisselmeier-Steinhauser.

La solution du problème d'optimisation $((SN1)^{(k)})^{(i)}$ est une \mathbf{E} -solution du système (1) (cfr démonstration en ANNEXE A 6.4).

4) LE SOUS-PROBLEME DE HAUT NIVEAU

Soient $X_i^{*(k)(j)}$ les solutions des ncg sous-problèmes de bas niveau.

En fixant, dans $(SP)^{(k)}$, les variables locales à ces valeurs (et en gardant Y comme variables), $(SP)^{(k)}$ devient :

$$(2) \quad \begin{cases} \text{Min } F^{\sim}(Y; Y^{(k)}) \\ \text{Sous } \begin{cases} h^{\sim}(Y; Y^{(k)}) \\ g_i^{\sim}(Y, X_i^{*(k)}(j); Y^{(k)}, X_i^{(k)}) \leq 0 \quad (i=1..ncg) \end{cases} \\ Y^l \leq Y \leq Y^u \\ X_i^l \leq X_i^{*(k)}(j) \leq X_i^u \quad (i=1..ncg) \end{cases}$$

Ce sous-problème est remplacé par :

((SN2) (k))

$$\left\{ \begin{array}{l} \text{Min} \quad F^{\sim}(Y; Y^{(k)}) \\ \text{Sous} \quad h^{\sim}(Y; Y^{(k)}) \leq 0 \\ \quad \quad C_i^{\sim}(Y, X_i^{*(k)}(j); Y^{(k)}, X_i^{(k)}) \leq 0 \quad (i=1..ncg) \\ \\ \quad \quad Y^l \leq Y \leq Y^u \\ \quad \quad (1-\beta) Y^{(k)} \leq Y \leq (1+\beta) Y^{(k)} \end{array} \right.$$

où β est un paramètre qui permet de rester au voisinage du point $Y^{(k)}(j)$.

Nous avons démontré dans l'ANNEXE A 6.4 que la solution du problème (SN2)^(k) est une solution du problème (2).

C) ALGORITHME

1) INITIALISATION

1.1) poser $k = 0$ (compteur d'itérations)
poser $j = 0$ (compteur de cycles)

1.2) choisir un point de départ
 $Y^{(k)}, X_1^{(k)}, \dots, X_{ncg}^{(k)}$

2) ETAPE PRINCIPALE

1) écrire le sous-problème (SN2)^(k) et les sous-problèmes (SN1)^{(k)(i)} pour $i=1..ncg$
(ce qui nécessite l'évaluation de la fonction objectif et des contraintes ainsi que de leur gradient au point courant)

- 2.1) poser $\gamma^{(k)}(j) = \gamma^{(k)}$
- 2.2) résolution de (SN1)^{(k)(i)} $i=1..ncg$
les solutions optimales seront notées $X_i^{*(k)}(j)$
- 2.3) étude de sensibilité
(cfr ANNEXE A 6.3)
- 2.4) poser $X_i^{(k)} = X_i^{*(k)}(j)$
- 2.5) résoudre (SN2)^(k)
la solution optimale sera notée $\gamma^{*(k)}(j)$
- 2.6) test de convergence des (SN1)^{(k)(i)} et (SN2)^(k) vers la solution du (SP)^(k)
si convergence : poser $\gamma^{*(k)} = \gamma^{*(k)}(j)$
 $X_i^{*(k)} = X_i^{*(k)}(j)$ ($i=1..ncg$)
aller en 3.
sinon : aller en 2.7
- 2.7) poser $\gamma^{(k)}(j+1) = \gamma^{*(k)}(j)$
 $X_i^{(k)}(j+1) = X_i^{*(k)}(j)$ ($i=1..ncg$)
 $j = j+1$
retour en 2.2
- 3.) test de convergence des (SP)^(k) vers une solution du problème initial
si convergence : poser $\gamma^* = \gamma^{*(k)}$
 $X_i^* = X_i^{*(k)}$ ($i=1..ncg$)
STOP
sinon : aller en 4.
- 4.) poser $\gamma^{(k+1)} = \gamma^{*(k)}$
 $X_i^{(k+1)} = X_i^{*(k)}$ ($i=1..ncg$)
 $k = k+1$
 $j = j+1$
retour à l'étape principale

D) MODIFICATION DE L'ALGORITHME

Pour réduire le coût des calculs, on ajoute une possibilité de disconnection dans la résolution de chaque sous-problème .

Après le premier cycle dans chaque itération, les sous-problèmes dont les contraintes sont restées loin d'être violées sont ignorées pour la suite de l'itération . Ceci permet d'éviter une optimisation et une analyse de sensibilité qui ne sont pas nécessaires .

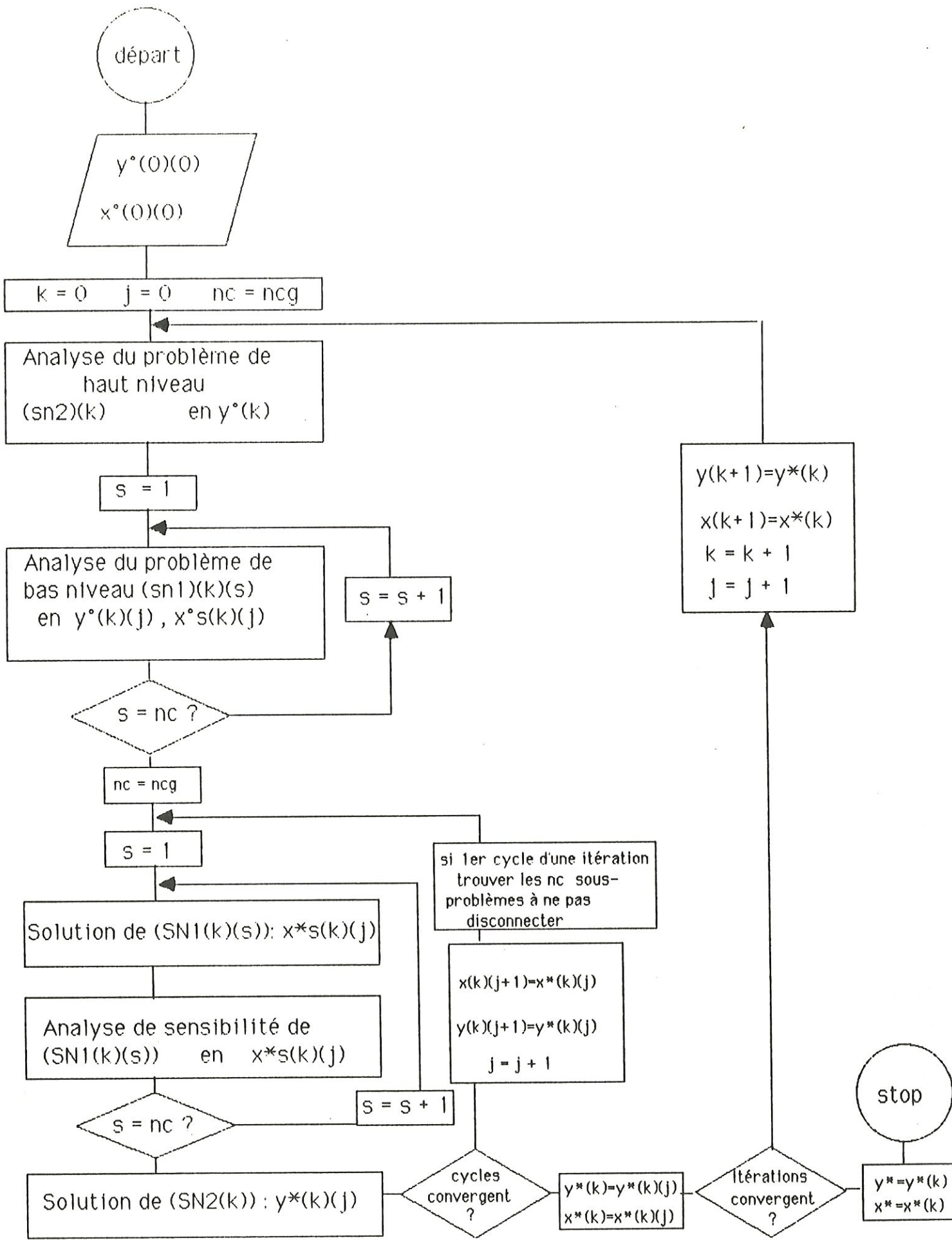
Pour appliquer cette disconnection, il suffirait d'ajouter l'étape suivante au point 2.7 de l'algorithme :

Si on est au premier cycle d'une itération :

déterminer les sous-problèmes $(SN1)^{(k)(i)}$ qui doivent être disconnectés (i.e. les sous-problèmes dont les contraintes sont loin d'être violées)

A noter qu'en plus de cette étape supplémentaire, il faudrait changer les "ncg" de l'étape principale par des "nc" (i.e. le nombre de sous-problèmes de bas niveau restant connectés) . En effet, on ne travaillerait plus que sur les $(SN1)^{(k)(i)}$ connectés . Il est sous-entendu qu'on rangerait les sous-problèmes de telle sorte que les nc premiers soient les "toujours connectés" (i.e. $i=1..nc$) .

L'organigramme de l'algorithme modifié se trouve en page 6.9 .



CHAPITRE 7

CONCLUSION

L'algorithme que nous avons implémenté permet un petit nombre d'évaluations de la fonction objectif, des contraintes et de leurs gradients respectifs. Ceci réduit donc le coût-calcul. Les temps CPU pour les exemples que nous avons testés varient entre 18 et 730 centièmes de seconde. Ces avantages nous invitent à conseiller l'emploi de cette méthode, pourtant...

Comme nous l'avons souligné dans les commentaires des résultats numériques du chapitre 5, une étude théorique nous semble nécessaire pour éviter toute indétermination lors de l'évaluation des variables primales. Ceci permettrait d'élaborer un algorithme général (c'est-à-dire indépendant des problèmes à résoudre). Toutefois nous obtenons déjà de très bons résultats de convergence sur bon nombre de problèmes testés.

ANNEXES DU CHAPITRE 4

ANNEXE 4.1 : La linearisation mixte

$$\text{Soit } h: \mathbb{R}^n \longrightarrow \mathbb{R}$$

$$x \longmapsto h(x)$$

Nous allons linéariser h par rapport aux variables x_i pour $i \in J_1$ et par rapport aux variables $1/x_i$ pour $i \in J_2$.

$$\text{En notant } x = (x_1, x_2, \dots, x_n)^t$$

$$\tilde{x} = (x_i)_{i \in J_1}$$

$$\tilde{y} = (y_i)_{i \in J_2} \text{ où } y_i = 1/x_i$$

nous pouvons réécrire $h(x) = \tilde{h}(\tilde{x}, \tilde{y})$ et nous linéarisons \tilde{h} au point $(\tilde{x}^{\circ}, \tilde{y}^{\circ})$

$$\tilde{h}(\tilde{x}, \tilde{y}) \cong \tilde{h}(\tilde{x}^{\circ}, \tilde{y}^{\circ}) + \left(\begin{array}{c} \frac{\partial \tilde{h}}{\partial x_i} \Big|_{i \in J_1} \\ \frac{\partial \tilde{h}}{\partial y_i} \Big|_{i \in J_2} \end{array} \right)_{(\tilde{x}^{\circ}, \tilde{y}^{\circ})} \begin{pmatrix} \tilde{x} - \tilde{x}^{\circ} \\ \tilde{y} - \tilde{y}^{\circ} \end{pmatrix}$$

$$\cong \tilde{h}(\tilde{x}^{\circ}, \tilde{y}^{\circ}) + \sum_{J_1} \frac{\partial \tilde{h}}{\partial x_i} \Big|_{\tilde{x}^{\circ}} (x_i - x_i^{\circ}) + \sum_{J_2} \frac{\partial \tilde{h}}{\partial y_i} \Big|_{\tilde{y}^{\circ}} (y_i - y_i^{\circ})$$

En utilisant la formule de dérivation de fonction composée

$$\frac{\partial \tilde{h}}{\partial y_i} = \frac{\partial \tilde{h}}{\partial x_i} \frac{\partial x_i}{\partial y_i} = \frac{\partial \tilde{h}}{\partial x_i} (-1/y_i^2)$$

nous obtenons

$$\tilde{h}(\tilde{x}, \tilde{y}) = \tilde{h}(\tilde{x}^{\circ}, \tilde{y}^{\circ}) + \sum_{J_1} \frac{\partial \tilde{h}}{\partial x_i} \Big|_{\tilde{x}^{\circ}} (x_i - x_i^{\circ})$$

$$- \sum_{J_2} \frac{\partial \tilde{h}}{\partial x_i} \Big|_{\tilde{x}^{\circ}} (y_i - y_i^{\circ}) (1/y_i^{\circ})^2$$

$$h(x) \equiv \tilde{h}(x; x^\circ) = h(x^\circ) + \sum_{J_1} \frac{\partial h}{\partial x_i} \Big|_{x^\circ} (x_i - x_i^\circ) - \sum_{J_2} \frac{\partial h}{\partial x_i} \Big|_{x^\circ} (1/x_i - 1/x_i^\circ) (x_i^\circ)^2$$

Par analogie pour $g : \mathbb{R}^n \times \mathbb{R}^m \longrightarrow \mathbb{R}$
 $(y, x) \longmapsto g(y, x)$

En notant $x = (x_1, x_2, \dots, x_n)^t$
 $x1^\sim = (x_i)_{i \in J_1}$
 $x2^\sim = (w_i)_{i \in J_2}$ où $w_i = 1/x_i$
 $y = (y_1, y_2, \dots, y_m)^t$
 $y1^\sim = (y_i)_{i \in J_1}$
 $y2^\sim = (z_i)_{i \in J_2}$ où $z_i = 1/x_i$

nous pouvons réécrire $g(y, x) = \tilde{g}(y1^\sim, y2^\sim, x1^\sim, x2^\sim)$, nous linéarisons \tilde{g} au point $(y1^\sim^\circ, y2^\sim^\circ, x1^\sim^\circ, x2^\sim^\circ)$ et en utilisant la formule de dérivation composée pour revenir aux variables initiales

$$g(y, x; y^\circ, x^\circ) \equiv \tilde{g}(y1^\sim, y2^\sim, x1^\sim, x2^\sim) = g(y^\circ, x^\circ) + \sum_{J_1} \frac{\partial g}{\partial y_i} \Big|_{y^\circ, x^\circ} (y_i - y_i^\circ) - \sum_{J_2} \frac{\partial g}{\partial y_i} \Big|_{y^\circ, x^\circ} (1/y_i - 1/y_i^\circ) (y_i^\circ)^2 + \sum_{K_1} \frac{\partial g}{\partial x_i} \Big|_{y^\circ, x^\circ} (x_i - x_i^\circ) - \sum_{K_2} \frac{\partial g}{\partial x_i} \Big|_{y^\circ, x^\circ} (1/x_i - 1/x_i^\circ) (x_i^\circ)^2$$

Il s'agit de montrer que le problème linéarisé suivant est convexe .

$$\begin{aligned} \text{Min } & \tilde{f}(y ; y^0) \\ \text{sous } & h_j \tilde{(y ; y^0)} \leq 0 \quad (j=1..nch) \\ & g_{ij} \tilde{(y, x ; y^0, x^0)} \leq 0 \quad (i=1..ncg, j=1..ncgi(i)) \\ & 0 < y_i^l \leq y_i \leq y_i^u \quad (i=1..ncy) \\ & 0 < x_{ik}^l \leq x_{ik} \leq x_{ik}^u \quad (i=1..ncg, k=1..ncxi(i)) \end{aligned}$$

Pour cela il suffit de montrer que $\tilde{f}(y ; y^0)$, $h_j \tilde{(y ; y^0)}$, et $g_{ij} \tilde{(y, x ; y^0, x^0)}$ sont des fonctions convexes . Voyons que les matrices hessiennes de ces fonctions sont demi-définies positives .

$$\begin{bmatrix} d1 & 0 & \dots & 0 \\ 0 & d2 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & dn \end{bmatrix}$$

où $d_i = -2 \frac{(y_i^0)^2}{(y_i^0)^3} \frac{\partial f}{\partial y_i | y^0}$ si $\frac{\partial f}{\partial y_i | y^0} < 0$
 0 sinon

Cette matrice est semi-définie positive car $d_i \geq 0$ pour chaque indice $i=1..ncy$. Nous pouvons raisonner de même pour les fonctions h_j .

La matrice hessienne de $g_{ij} \tilde{}$ est égale à

$$\begin{bmatrix} p1 & 0 & \dots & 0 \\ 0 & p2 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & pn & 0 & \dots & 0 \\ 0 & \dots & 0 & q1 & 0 & \dots & 0 \\ 0 & \dots & \dots & \dots & q2 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & \dots & \dots & qm \end{bmatrix}$$

$$\text{où } p_k = -2 \frac{(y_k)^2}{(y_k)^3} \frac{\partial g_{ij}}{\partial y_k | y^\circ, x^\circ} \quad \text{si } \frac{\partial g_{ij}}{\partial y_k | y^\circ, x^\circ} < 0$$

$$0 \quad \text{sinon} \quad (k=1..ncy)$$

$$\text{et } q_k = -2 \frac{(x_{ik}^\circ)^2}{(x_{ik}^\circ)^3} \frac{\partial g_{ij}}{\partial x_{ik} | y^\circ, x^\circ} \quad \text{si } \frac{\partial g_{ij}}{\partial x_{ik} | y^\circ, x^\circ} < 0$$

$$0 \quad \text{sinon} \quad (k=1..ncxi(i))$$

Cette matrice est donc semi-définie positive car p_k et q_k sont positifs pour chaque indice k .

cqfd.

ANNEXES DU CHAPITRE 5

ANNEXE 5.1 : Calcul du gradient de la fonction duale

Soit

$$\theta(u) = \min \{ f(x) + \sum_{i=1..m} u_i g_i(x) \}$$

$$\text{où } f, g_i : \mathbb{R}^n \longrightarrow \mathbb{R}$$

$$f, g_i \in C^1$$

$$x = (x_1, x_2, \dots, x_n)^t$$

Nous pouvons encore écrire

$$\theta(u) = f(x^*) + \sum_{i=1..m} u_i g_i(x^*) \quad (1)$$

où $x^* = x^*(u)$ réalise le minimum .

Calculons le gradient de $\theta(u)$:

$$\nabla \theta(u) = \left(\frac{\partial \theta}{\partial u_i} \right)_{i=1..m}$$

$$\frac{\partial \theta}{\partial u_k} = \sum_{i=1..m} \frac{\partial f}{\partial x^*_i} \frac{\partial x^*_i}{\partial u_k} + \sum_{i=1..n} \sum_{j=1..m} \frac{\partial g_j}{\partial x^*_i} \frac{\partial x^*_i}{\partial u_k} + g_k$$

$$\frac{\partial \theta}{\partial u_k} = \sum_{i=1..m} \left\{ \frac{\partial f}{\partial x^*_i} + \sum_{j=1..m} \frac{\partial g_j}{\partial x^*_i} \right\} \frac{\partial x^*_i}{\partial u_k} + g_k$$

Si x^*_1 atteint une de ses bornes alors

$$\frac{\partial x^*_1}{\partial u_k} = 0 \text{ d'où } \frac{\partial \theta(u)}{\partial u_k} = g_k$$

Si x^*_1 est strictement compris entre ses bornes alors

$$\frac{\partial f}{\partial x^*_1} \frac{\partial x^*_1}{\partial u_k} + \sum_{i=1,m} \frac{\partial g_i}{\partial x^*_1} = 0 \text{ par les conditions}$$

d'optimalité du premier ordre car x^* réalise
le minimum de **(1)**

$$\text{d'où } \frac{\partial \theta(u)}{\partial u_k} = g_k$$

En conclusion

$$\nabla \theta(u) = g(x^*)$$

cqfd.

ANNEXE 5.2 : Trouver y'_j en fonction de u

Nous allons pour cela minimiser $Z_j(y'_j, u)$ sur $[y'_j{}^l, y'_j{}^u]$

où $y'_j > 0$

avec

$$Z_j(y'_j, u) = \frac{\partial f}{\partial y'_j} y'_j + C_j y'_j - D_j (1/y'_j) \quad \text{si } \frac{\partial f}{\partial y'_j} > 0$$

$$= -\frac{\partial f}{\partial y'_j} y'_j + C_j y'_j - D_j (1/y'_j) \quad \text{si } \frac{\partial f}{\partial y'_j} < 0$$

$$\text{si } \frac{\partial f}{\partial y'_j} > 0$$

$$\frac{\partial Z_j}{\partial y'_j} = \frac{\partial f}{\partial y'_j} + C_j + \frac{D_j}{(y'_j)^2}$$

$$\text{si } D_j = 0 \quad \rightarrow \quad \frac{\partial Z_j}{\partial y'_j} = \frac{\partial f}{\partial y'_j} + C_j \geq 0$$

\rightarrow La fonction Z_j est croissante en la variable y'_j , son minimum est donc atteint en y'_j , la borne inférieure de l'intervalle considéré.

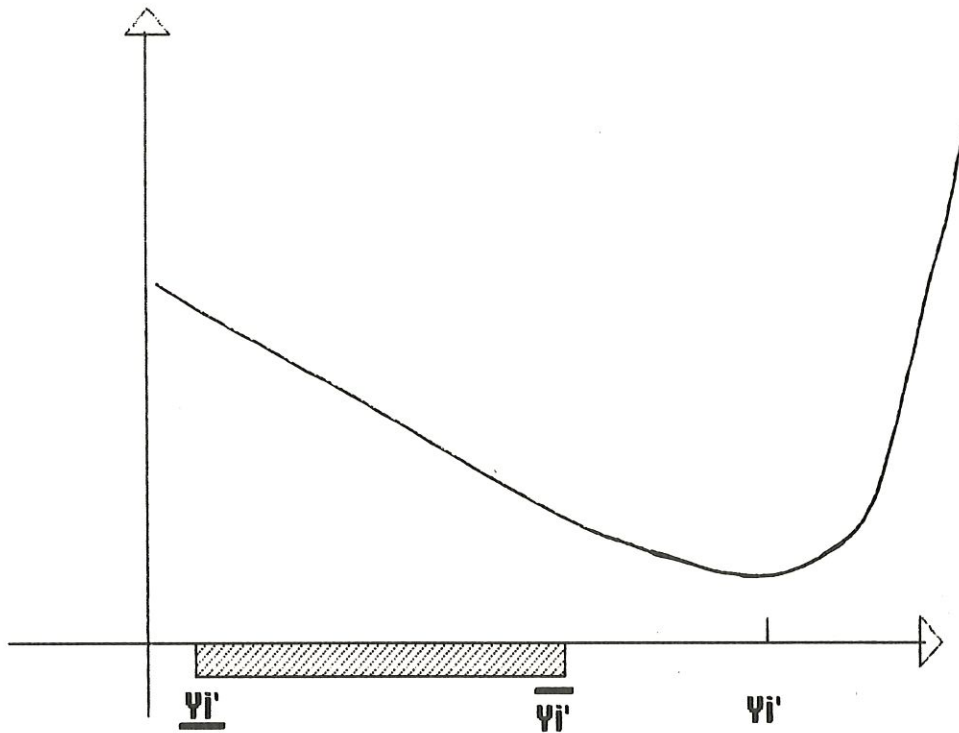
si $D_j \neq 0$ \rightarrow Annuler la dérivée de Z_j par rapport à y'_j donne :

$$y'_j{}^2 = -\frac{D_j}{(\partial f / \partial y'_j) + C_j}$$

Z_j est une fonction convexe pour $y'_j > 0$ car

$$\frac{\partial^2 Z_j}{\partial y'_j{}^2} = -2 \frac{D_j}{y'_j{}^3} > 0$$

Si $y'_j \geq y'_j{}^u$, nous avons la situation suivante :

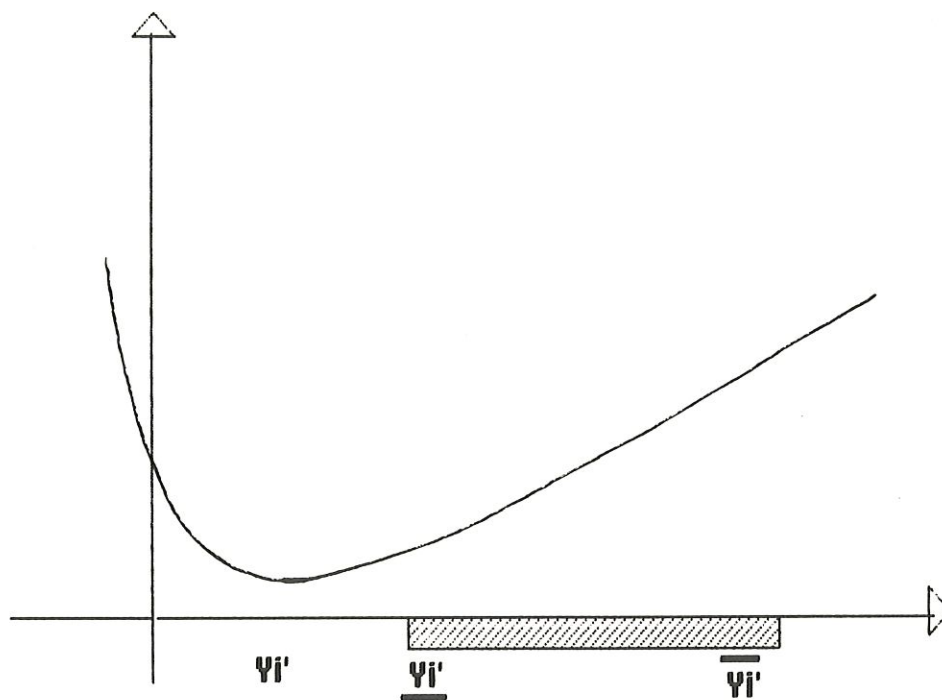


[fig. 1]

Graphe de $Z_j (y'_j \geq y'_j{}^u)$

Le minimum de Z_j sur $[y'_j{}^l, y'_j{}^u]$ est donc en $y'_j{}^u$.

Si $y'_j \leq y'_j{}^l$, nous avons la situation suivante :



[fig. 2]
 Graphe de $L_j (y'_j \leq y'_j{}^l)$

Le minimum de L_j sur $[y'_j{}^l, y'_j{}^u]$ est donc en $y'_j{}^u$.

$$\text{si } \frac{\partial f}{\partial y'_j} < 0$$

$$\frac{\partial L_j}{\partial y'_j} = \frac{\partial f}{\partial y'_j} \frac{1}{(y'_j)^2} + C_j + \frac{D_j}{(y'_j)^2}$$

$$\text{si } C_j = 0 \quad \rightarrow \quad \frac{\partial^2 L_j}{\partial y_j'^2} = \left(\frac{\partial f}{\partial y_j'} + D_j \right) \frac{1}{(y_j')^2} < 0$$

\rightarrow La fonction L_j est décroissante en la variable y_j' , son minimum est donc atteint en $y_j'^u$, la borne supérieure de l'intervalle considéré.

si $C_j \neq 0$ \rightarrow Annuler la dérivée de L_j par rapport à y_j' donne :

$$y_j'^2 = - \frac{(\partial f / \partial y_j' | 1) + D_j}{C_j}$$

L_j est une fonction convexe pour $y_j' > 0$ car

$$\frac{\partial^2 L_j}{\partial y_j'^2} = -2 \frac{(\partial f / \partial y_j' | 1) + D_j}{y_j'^3} > 0$$

Si $y_j' \geq y_j'^u$ comme dans la figure le minimum de L_j sur $[y_j^l, y_j^u]$ est donc en $y_j'^u$

Si $y_j' \leq y_j^l$ comme dans la figure le minimum de L_j sur $[y_j^l, y_j^u]$ est donc en y_j^l .

Ceci nous fournit bien les résultats annoncés en page 5.16.

Trouver x'_{ij} en fonction de u

Pour cela nous allons minimiser $k_{jj}(x'_{ij}, u)$ sur $[x'_{ij}^l, x'_{ij}^u]$ où x'_{ij} est un réel strictement positif et avec

$$k_{jj}(x'_{ij}, u) = W_{ij} x'_{ij} - V_{ij} (1/x'_{ij})$$

$$\frac{\partial k_{jj}}{\partial x'_{ij}} = W_{ij} + \frac{V_{ij}}{(x'_{ij})^2}$$

Annuler la dérivée de $k_{jj}(x'_{ij}, u)$ par rapport à x'_{ij} donne

$$(x'_{ij})^2 = - \frac{V_{ij}}{W_{ij}}$$

Si $V_{ij} \neq 0$ k_{jj} est une fonction convexe pour $x'_{ij} > 0$
car

$$\frac{\partial^2 k_{jj}}{\partial x'_{ij}{}^2} = -2 \frac{V_{ij}}{(x'_{ij})^3} > 0$$

Si $x'_{ij} \geq x'_{ij}^u$ (par analogie avec la figure 1), le minimum de k_{jj} sur $[x'_{ij}^l, x'_{ij}^u]$ est donc en x'_{ij}^u .

Si $x'_{ij} \leq x'_{ij}^l$ (par analogie avec la figure 2), le minimum de k_{jj} sur $[x'_{ij}^l, x'_{ij}^u]$ est donc en x'_{ij}^l .

Si $V_{ij} = 0$ k_{jj} est une fonction croissante en x'_{ij} car

$$\frac{\partial k_{jj}}{\partial x'_{ij}} = W_{ij} \geq 0$$

d'où son minimum est atteint en x'_{ij}^l . Ceci nous fournit les résultats annoncés en page 5.17.

Annexe 5.3 : Estimation de Fest

Dans les spécifications de E04KBF , Fest doit être une estimation ou une sous-estimation du minimum de la fonction à minimiser . Nous avons donc implémenté une routine THEMEX qui calcule une sur-estimation de la fonction duale...qui sera donc a fortiori une sur-estimation du maximum de cette fonction .

Pour calculer cette estimation nous avons majoré chaque terme de la fonction duale :

chaque variable a été remplacée

par sa borne supérieure au dénominateur (1)

par sa borne inférieure au numérateur ,(2)

les termes constants (négatifs) ont été omis (3) ainsi que

les facteurs des multiplicateurs de lagrange (4) .

$$\begin{aligned} & \sum_{i+} \frac{\partial f}{\partial y_i} \frac{1}{|y_i|} - \sum_{i-} \frac{\partial f}{\partial y_i} \frac{1}{|y_i|} + \\ & \sum_{i-} \frac{\partial f}{\partial y_i} \frac{1}{|y_i|} - \sum_{i+} \frac{\partial f}{\partial y_i} \frac{1}{|y_i|} + f(Y^{(k)}) + \\ & + \sum_{j=1..nch} u_j \left(\sum_{i+} \frac{\partial h_j}{\partial y_i} \frac{1}{|y_i|} - \sum_{i-} \frac{\partial h_j}{\partial y_i} \frac{1}{|y_i|} + \right. \\ & \left. \sum_{i-} \frac{\partial h_j}{\partial y_i} \frac{1}{|y_i|} - \sum_{i+} \frac{\partial h_j}{\partial y_i} \frac{1}{|y_i|} + h_j(Y^{(k)}) - z_j \right) \\ & + \sum_{l=1..ncg} \sum_{j=1..ncgl(1)} u_{lj} \left(\sum_{k+} \frac{\partial g_{lj}}{\partial y_k} \frac{1}{|y_k|} - \right. \end{aligned}$$

$$\sum_{k^-} \frac{\partial g_{ij}}{\partial y_k} \frac{1}{y_k} + \sum_{k^+} \frac{\partial g_{ij}}{\partial x_{ik}} x_{ik} -$$

$$\sum_{k^-} \frac{\partial g_{ij}}{\partial x_{ik}} \frac{1}{x_{ik}} - \sum_{k^+} \frac{\partial g_{ij}}{\partial x_{ik}} +$$

$$\sum_{k^-} \frac{\partial g_{ij}}{\partial x_{ik}} - \sum_{k^+} \frac{\partial g_{ij}}{\partial y_k} + \sum_{k^-} \frac{\partial g_{ij}}{\partial y_k} +$$

$$g_{ij}(Y^{(k)}, X_i^{(k)}) - z_{ij})$$

Ceci devient donc

$$fest = - \left(\sum_{i^+} \frac{\partial f}{\partial y_i} y_i^u - \sum_{i^-} \frac{\partial f}{\partial y_i} \frac{1}{y_i^l} + f(Y^{(k)}) \right)$$

ANNEXES DU CHAPITRE 6

ANNEXE 6.1 : propriétés de la fonction de Kreisselmeier-Steinhauser

Soient $g_i(X)$ ($i=1..m$) m contraintes dépendant du vecteur X .

$$X = (x_1, \dots, x_m):$$

$$g_i(X) \leq 0 \quad (i=1..m)$$

Une mesure cumulative de la satisfaction ou la violation des contraintes est donnée par :

$$C(X) = (1/\rho) \cdot \ln \{ \sum_{i=1..m} \exp \{ \rho \cdot g_i(X) \} \}$$

où ρ est un nombre positif choisi par l'utilisateur.

propriété 1 *la fonction de Kreisselmeier-Steinhauser est une enveloppe conservatrice de la contrainte maximale*
$$g_{\max} \leq C(X) \leq g_{\max} + (1/\rho) \cdot \ln(m)$$

démonstration :

a) Posons $g_{\max} = \max_{i=1..m} g_i$

On peut alors écrire

$$C(X) = g_{\max} + (1/\rho) \cdot \ln \{ \sum_{i=1..m} \exp \{ \rho \cdot (g_i(X) - g_{\max}(X)) \} \}$$

Soit j^* l'indice de la contrainte réalisant le maximum :

$$g_{\max}(X) = g_{j^*}(X)$$

Alors

$$C(X) = g_{\max} + (1/\rho) \cdot \ln \{ \sum_{j \neq j^*} \exp \{ \rho \cdot (g_j(X) - g_{\max}(X)) \} + \exp \{ \rho \cdot (g_{j^*}(X) - g_{\max}(X)) \} \} \quad (1)$$

$$C(X) = g_{\max} + (1/\rho) \cdot \ln \{ \sum_{j \neq j^*} (\geq 0) + 1 \}$$

d'où $C(X) \geq g_{\max}(X)$

b) Dans (1) on majore chaque terme de la somme par 1 :

$$C(X) \leq g_{\max} + (1/\rho) \cdot \ln \{ \sum_{j=1..m} 1 \}$$

\Leftrightarrow

$$C(X) \leq g_{\max} + (1/\rho) \cdot \ln(m)$$

c) Conclusion

$$g_{\max} \leq C(X) \leq g_{\max} + (1/\rho) \ln(m) \quad \forall X \in \mathbb{R}^n$$

□

propriété 2 $C(X)$ est convexe si $g_i(X)$ sont convexes ($i=1..m$)

démonstration :

a) Calcul de $\nabla C(X)$:

$$C(X) = (1/\rho) \ln \left\{ \sum_{i=1..m} \exp \left[\rho \cdot (g_i(X)) \right] \right\}$$

$$\nabla C(X) = (1/\rho) \cdot \frac{\sum_{i=1..m} \exp \left[\rho \cdot (g_i(X)) \right] \cdot \rho \cdot \nabla g_i(X)}{\sum_{i=1..m} \exp \left[\rho \cdot (g_i(X)) \right]}$$

que l'on peut encore écrire :

$$\nabla C(X) = \sum_{i=1..m} \exp \left[\rho \cdot (g_i(X) - C(X)) \right] \cdot \nabla g_i(X)$$

b) Calcul de $\nabla^2 C(X)$:

$$\nabla^2 C(X) = \sum_{i=1..m} \exp \left[\rho \cdot (g_i(X) - C(X)) \right] \cdot \nabla (g_i(X) - C(X)) \cdot \nabla g_i(X)^t + \sum_{i=1..m} \exp \left[\rho \cdot (g_i(X) - C(X)) \right] \cdot \nabla^2 (g_i(X))$$

$$\nabla^2 C(X) = \sum_{i=1..m} \exp \left[\rho \cdot (g_i(X) - C(X)) \right] \cdot \left\{ \nabla g_i(X) \cdot \nabla g_i(X)^t \cdot \rho + \nabla^2 (g_i(X)) \right\} - \rho \cdot \nabla C(X) \cdot \nabla C(X)^t$$

En utilisant la propriété 1 on peut encore écrire

$$\nabla^2 C(X) = \sum_{i=1..m} \left(\exp \left[\rho \cdot (g_i(X) - C(X)) \right] \cdot \nabla^2 (g_i(X)) \right) + \sum_{i=1..m} \sum_{j=1..m} \left(\exp \left[\rho \cdot (g_i(X) - C(X)) \right] \cdot \exp \left[\rho \cdot (g_j(X) - C(X)) \right] \cdot \left\{ \nabla g_i(X) - \nabla g_j(X) \right\} \cdot \left\{ \nabla g_i(X) - \nabla g_j(X) \right\}^t \right)$$

c) Si les contraintes g_i sont des fonctions convexes, alors leur matrice hessienne $\nabla^2 g_i$ est semi-définie positive .

De plus, $\{\nabla g_i(X) - \nabla g_j(X)\} \cdot \{\nabla g_i(X) - \nabla g_j(X)\}^t$ est semi-définie positive en tant que matrice produit d'un vecteur par lui-même .

Finalement, tous les coefficients de $\nabla^2 C$ sont positifs .

Donc :

.) $\nabla^2 C (X)$ est semi-définie positive car somme de matrices semi-définies positives .

c'est-à-dire $C(X)$ est convexe si les $g_i(X)$ le sont .

□

ANNEXE 6.2 Choix du paramètre ρ

.) $\nabla C(X)$ peut s'écrire :

$$\nabla C(X) = \left\{ \sum_{i \neq j^*} \exp \left\{ \rho \cdot (g_i(X) - C(X)) \right\} \cdot \nabla g_i(X) \right\} \\ + \exp \left\{ \rho \cdot (g_{\max}(X) - C(X)) \right\} \cdot \nabla g_{\max}(X)$$

où j^* est tq $g_{\max} = g_{j^*}$

or $C(X) \geq g_j(X)$

car $\forall X$ par la propriété 1

$$g_{\max}(X) \approx C(X) \text{ si } \rho \rightarrow +\infty$$

d'où :

$$\nabla C(X) \leq \nabla g_{\max}(X) \text{ si } \rho \rightarrow +\infty$$

.) Par la propriété 1 on remarque donc que plus ρ est grand et plus C colle à g_{\max} . [fig. 1]

Il peut alors y avoir des problèmes quand g_{\max} est une fonction discontinue.

Le choix du paramètre ρ doit donc être un compromis entre le désir de coller le plus possible à g_{\max} et la crainte de la discontinuité des gradients.

Un choix conseillé par J.F.Barthelemy est :

$$\rho = \frac{\ln(m)}{\epsilon}$$

où ϵ est la tolérance du programme d'optimisation.

Un avantage de ce choix est que :

Si $C(X) = 0$, alors, par la propriété 1, on obtiendra $|g_{\max}(X)| \leq \epsilon$
c'est-à-dire g_{\max} sera une contrainte active.

Illustration du choix du paramètre ρ

Solent les deux fonctions

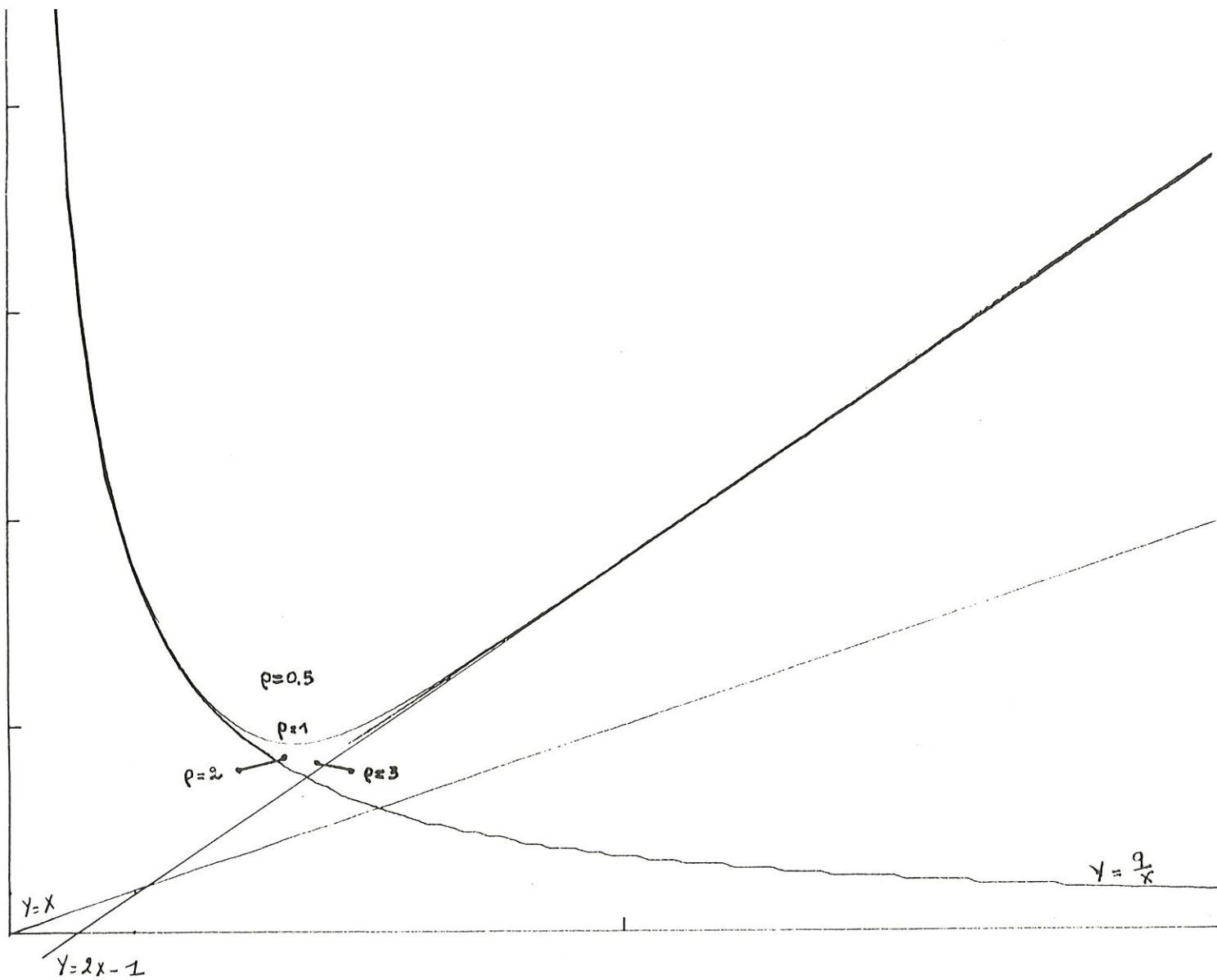
$$g_1(x) = 2x - 1$$

$$g_2(x) = 9/x$$

d'où

$$C(x) = (1/\rho) \ln \{ \exp(2\rho x - \rho) + \exp(9\rho/x) \}$$

La figure suivante présente ces fonctions pour différentes valeurs de ρ :



ANNEXE 6.3 Etude de sensibilité

La sensibilité d'un sous-problème

$$\begin{aligned} \text{Min } & C(Y^0, X) \\ \text{Sous } & X^l \leq X \leq X^u \end{aligned}$$

par rapport à des petites variations de Y peut être estimée au voisinage de Y^0 par le développement suivant :

$$\begin{aligned} C^{\sim}(Y; Y^0) = & C(Y^0, X^*) + \sum_{i+} \frac{\partial C(Y, X^*)}{\partial y_i} \Big|_{Y^0} (y_i - y_i^0) \\ & + \sum_{i-} \frac{\partial C(Y, X^*)}{\partial y_i} \Big|_{Y^0} (y_i - y_i^0) \frac{y_i^0}{y_i} \end{aligned}$$

où $\frac{\partial C(Y, X^*)}{\partial y_i} \Big|_{Y^0}$ est obtenu par analyse de sensibilité

d'un problème sans contrainte **[Ref.9]**

ANNEXE 6.4 Relation entre (SN1)^{(k)(i)} et (1)

Soit X_i^* solution de $\text{Min } C_i(Y^{(k)(j)}, X_i)$
 $X_i^l \leq X_i \leq X_i^u$

Si $C_i(Y^{(k)(j)}, X_i^*) = C_i^*$

alors par la propriété 1 de la fonction de Kreisselmeier-Steinhauser

$$g_{\max}(Y^{(k)(j)}, X_i^* ; Y^{(k)}, X_i^{(k)}) \leq C_i^*$$

d'où X_i^* sera C_i^* -admissible pour le problème (1).

cqfd

Relation entre (SN2)^(k) et (2)

Comme $C_i^{\sim}(Y, X_i^{*(k)(j)} ; Y^{(k)}, X_i^{(k)}) \approx C_i(Y, X_i^{*(k)(j)})$
pour $Y^{(k)(1-\beta)} \leq Y \leq (1+\beta)Y^{(k)}$

Et que $C_i(Y, X_i^{*(k)(j)}) \geq g_{i \max}(Y, X_i^{*(k)(j)})$

bù $g_{i \max} = \max_j g_{ij}$ par la propriété (1) de la fonction de
Kreisselmeier-Steinhauser

on peut immédiatement en déduire que

$$C_i^{\sim}(Y, X_i^{*(k)(j)} ; Y^{(k)}, X_i^{(k)}) \leq 0$$

d'où $g_{ij}(Y, X_i^{*(k)(j)}) \leq 0 \quad (j=1..ncgi(i))$

cqfd

REFERENCES :

- [REF 1]** Large scale systems Y. Y HAIMES
Studies in management science and systems 7
North Holland (1982)
- [REF2]** Elements of structural optimisation
R. T HAFTKA
M. P KAMAT
Mechanics of structural systems
M. Nijhoff (1985)
- [REF3]** Numerical optimisation techniques for engineering
design with applications G. N VANDERPLAATS
Mac Graw-Hill (1984)
- [REF4]** These de doctorat V. BRAIBANT
Collection des publications n° 102
Université de Liège (1986)
- [REF5]** Structural optimisation : A new dual method using mixed
variables C. FLEURY
V. BRAIBANT
International journal for numerical methods in
engineering, vol 23,409/428 (1986)
- [REF6]** An improved multilevel optimisation approach for the
design of complex engineering systems
J-F. M BARTHELEMY
AIAA paper 86-0950-CP (1986)
- [REF7]** Systematic control design by optimising a vector
performance index G. KREISSELMEIER
R. STEINHAUSER
Proc of IFAC symposium on computer aided design of
control systems, Zurich, Switzerland (1979)
113-117
- [REF8]** A computer simulator for development of engineering
system design methodologies
S. L. PADULA
J. SOBISZCZANSKI -
SOBIESKI
(à paraître)

[REF9] Introduction to sensitivity and stability analysis in
nonlinear programming A.V FIACCO
Mathematics in science and engineering,
volume 165
Academis press, New York. (1983)