UNIVERSITÉ DE NANCY I U.E.R. SCIENCES MATHÉMATIQUES

PROGRAMMES D'OPTIMISATION

APPLICATION AU CONTROLE OPTIMAL AVEC CONTRAINTE

THESE

pour l'obtention du

DOCTORAT DE SPÉCIALITÉ DE MATHÉMATIQUES APPLIQUÉES

Soutenue le 28 Novembre 1975

par

Marc ROCHER

MEMBRES DU JURY :

Président : M. J. LEGRAS
Examinateurs : MM. J.P. HATON
R. HUSSON

UNIVERSITÉ DE NANCY I U.E.R. SCIENCES MATHÉMATIQUES

PROGRAMMES D'OPTIMISATION

APPLICATION AU CONTROLE OPTIMAL AVEC CONTRAINTE

THESE

pour l'obtention du

DOCTORAT DE SPÉCIALITÉ DE MATHÉMATIQUES APPLIQUÉES

Soutenue le 28 Novembre 1975

par

Marc ROCHER

MEMBRES DU JURY :

Président : M. J. LEGRAS
Examinateurs : MM. J.P. HATON
R. HUSSON

Je remercie Monsieur le Professeur LEGRAS dont les conseils et la bienveillance m'ont toujours été très précieux.
Qu'il trouve ici, l'expression de ma profonde reconnaissance.

Messieurs POTDEVIN et SANCHEZ du Laboratoire d'Informatique pour leurs précieux conseils et pour le temps qu'ils ont bien voulu me consacrer.

Messieurs les Professeurs HATON et HUSSON qui ont accepté d'être membres du Jury.

SOMMAIRE

INTRODUCTION

Dans une première partie, nous développons des techniques d'optimisation en variables libres utilisant certaines améliorations de la méthode du gradient.

Nous verrons ensuite comment adapter ces méthodes dans le cas d'une optimisation en variables liées et en particulier, nous examinerons la différence entre un problème à contraintes mixtes et un problème à contraintes bilatérales.

La troisième partie consiste en une application de ces méthodes à des problèmes de contrôle optimal à critères et contraintes terminaux ou pouvant se ramener à ce cas.

Nous verrons qu'une fois déterminé le mode de représentation de la fonction de commande nous nous ramenons sans difficulté à un problème d'optimisation simple.

Les méthodes exposées seront testées sur des exemples dont la solution est connue.

CHAPITRE I

OPTIMISATION EN VARIABLES LIBRES

I - 1 TECHNIQUE ELEMENTAIRE DU GRADIENT

Soit un vecteur $X^r (x_1^r, x_2^r, \ldots, x_n^r)$ et h positif donnés. On se propose de trouver une direction de montée D telle que

$$\Omega = f(X^r + h\,D) - f(X^r) \text{ soit maximum}$$

Cette étude n'est possible pour h quelconque que dans quelques cas particuliers et dans le cas général nous nous limiterons à une étude locale et nous supposerons h assez petit pour que nous puissions approcher $f(X^r + h\,D)$ par son développement linéaire.

Dans ces conditions nous savons que la direction D de norme fixée qui maximise D grad f est :

$$D = k \text{ grad } f$$

on formera donc la suite itérative

$$X^{r+1} = X^r + h_r (\text{grad } f)_{X^r}$$

et pour h_r assez petit et $(\text{grad } f)_{X^r} \neq 0$ on aura

$$f(X^{r+1}) > f(X^r)$$

on arrêtera le mécanisme en X^* lorsque

$$\|\text{grad } f\|_{X^*} < \varepsilon \text{ donné.}$$

Avantages et inconvénients

Au crédit de la méthode nous pouvons mettre :

- une grande simplicité de mise en œuvre

- une efficacité certaine en particulier loin de l'optimum.

Les inconvénients sont :

- la difficulté de choix du pas h. Trop petit il augmente inutilement le nombre des itérations, trop grand il peut empêcher la convergence

- la méthode converge très lentement au voisinage de l'optimum. Ce fait est sans importance si l'on cherche la valeur de ce maximum, mais si l'on s'intéresse aux coordonnées de l'optimum, ce défaut peut se révéler très gênant.

- La convergence sera aussi très lente dans le cas d'une matrice associée au hessien de f mal conditionnée.

Ces inconvénients nous amènent à considérer diverses améliorations de cette technique élémentaire du gradient.

I - 2 TECHNIQUE DU GRADIENT AVEC DETERMINATION AUTOMATIQUE DU PAS

On se propose de déterminer h de façon que

$$\Omega (h) = f (X^r + h (\text{grad } f)_{X^r}) - f (X^r) \text{ soit maximum}$$

Pour cela, nous approcherons $f (X^r + h D)$, fonction de la seule variable h par g (h) polynôme du second degré en h obtenu par interpolation sur le support $[0, \ell , 2\ell]$ où ℓ est arbitraire.

Posons $\begin{cases} f_0 = f (X^r) \\ f_1 = f (X^r + h D) \\ f_2 = f (X^r + 2 h D) \end{cases}$

Le polynôme s'écrit

$$g (h) = f_0 + \frac{h}{\ell} \left[- \frac{3}{2} f_0 + 2 f_1 - \frac{1}{2} f_2 \right] + \frac{h^2}{2\ell^2} \left[f_0 - 2 f_1 + f_2 \right]$$

Ce polynôme possède un maximum si $f_0 - 2 f_1 + f_2 < 0$ et dans ces conditions, ce maximum est obtenu pour

$$h^* = - \frac{\ell}{2} \frac{- 3 f_0 + 4 f_1 + f_2}{f_0 - 2 f_1 + f_2}$$

Nous pouvons remarquer que h^* est la solution stricte lorsque f (X) est un polynôme du second degré en $x_1, x_2, \ldots, x_n$.

Mais dans le cas d'une fonction f quelconque, h^* peut donner des résultats aberrants (cas ou g (h) possède un minimum par exemple). Par conséquent, la mise en œuvre pratique de la méthode demandera la mise en place de sécurités qui seront développées dans la partie consacrée aux programmes.

Notons aussi que le fait que la direction de montée soit celle du gradient n'a aucune importance, le résultat reste applicable pour une direction quelconque.

I - 3 TECHNIQUE D'ACCELERATION DE CONVERGENCE

Nous allons d'abord considérer le cas ou f est un polynôme du

second degré en $x_1, x_2, \ldots, x_n$, alors :

$$f(X) = f(0) + B \cdot X + \frac{1}{2}\, {}^tX \cdot A \cdot X$$

ou B est le tableau dont les éléments sont $\left(\frac{\partial f}{\partial x_i}\right)$ composantes de (grad f)

et A le hessien, matrice carrée de terme général $\left(\frac{\partial^2 f}{\partial x_i \, \partial x_j}\right)_0$.

On a alors : $f'(X) = B + A \cdot X$

Les coordonnées de l'optimum sont solutions de

$$A \cdot X^* + B = 0$$

Appliquons n + 1 fois la méthode du gradient avec des pas $h_0, h_1, \ldots, h_n$ quelconques. Nous formons ainsi la suite

$$X^0, X^1, \ldots, X^r, X^{r+1}, \ldots, X^{n+1}$$

ou $$X^{r+1} = X^r + h_r \, (\text{grad } f)_{X^r}$$

soit $$X^{r+1} - X^r = h_r \, A \, [X^r - X^*]$$

Mais, les n + 1 tableaux $X^1 - X^0, \ldots, X^{n+1} - X^n$ étant formés chacun de n éléments ne sont pas indépendants. Donc il existe des nombres $\alpha_0, \alpha_1, \ldots, \alpha_n$ non tous nuls tels que

$$\alpha_0 (X^1 - X^0) + \alpha_1 (X^2 - X^1) + \ldots + \alpha_n (X^{n+1} - X^n) \equiv 0$$

Cette relation matricielle permet de calculer les nombres $\alpha_0, \alpha_1, \ldots, \alpha_n$ en fonction de l'un d'entre eux.

Or $X^{r+1} - X^r = h_r \, A \, (X^r - X^*)$ permet d'écrire

$$A \, [\, \alpha_0 h_0 (X^0 - X^*) + \alpha_1 h_1 (X^1 - X^*) + \ldots + \alpha_n h_n (X^n - X^*) \,] \equiv 0$$

et, A étant supposée inversible

$$\Longrightarrow \quad \alpha_0 h_0 [X^0 - X^*] + \ldots + \alpha_n h_n [X^n - X^*] = 0$$

donc, si

$$\sum_{i=0}^{n} \alpha_i h_i \neq 0$$

$$\boxed{X^* = \frac{\sum_{i=0}^{n} \alpha_i h_i X^i}{\sum_{i=0}^{n} \alpha_i h_i}}$$

Donc, dans le cas d'une forme quadratique, il est possible à partir de n + 1 itérés consécutifs calculés par la méthode du gradient de calculer X^* solution de

$$A X^* + B = 0$$

Cette méthode s'étend sans difficulté à une fonction quelconque. Il suffit d'introduire les sécurités habituelles pour le cas où le tableau X^* n'améliorerait pas le dernier vecteur obtenu.

CHAPITRE II

OPTIMISATION EN VARIABLES LIEES

Nous supposerons le problème mis sous la forme

$$\max f(X) \quad \text{ou} \quad X = (x_1, x_2, \ldots, x_n)$$

avec
$$\begin{cases} \varphi_\ell (X) = 0 & \forall \ell \in L \\ \varphi_m (X) \geq 0 & \forall m \in M \end{cases}$$

L : ensemble des contraintes bilatérales au nombre de p

M : ensemble des contraintes unilatérales.

En un point X^r, nous noterons $\varphi_j (X^r) = 0$ ou $j \in J^r$ (noté fréquemment J) une contrainte unilatérale saturée en X^r et $\varphi_k (X^r) > 0$ ou $k \in K^r$ (noté K) une contrainte unilatérale vérifiée et non saturée en X^r.

Il est évident que J^r et K^r pourront évoluer au cours du traitement.

II - 1 OPTIMISATION AVEC CONTRAINTES BILATERALES

Dans ce cas, on se limite à :

$$\max f(X) \text{ avec}$$

$$\varphi_\ell (X) = 0 \qquad \forall \ell \in L$$

n : nombre de variables

p : nombre de contraintes, strictement inférieur à n.

II - 1 - 1 Technique d'élimination numérique

Les p contraintes peuvent être considérées comme p

équations permettant de définir p variables en fonction des n - p autres variables. Nous devons choisir parmi les n variables

| p variables de base (ou variables liées)

| n - p variables indépendantes (ou variables libres ou hors base)

Soit X_B le tableau des p variables de base et X_H le tableau des n - p variables indépendantes.

Le système des contraintes s'écrit alors :

$$\varphi_i (X_B, X_H) = 0 \qquad i \in L$$

Sous certaines conditions : existence des solutions, non nullité du Jacobien, ce système permet d'expliciter les variables de base en fonction des variables indépendantes. La fonction f (X) peut alors s'écrire f (X_H, X_B) et après avoir remplacé X_B par K (X_L) on obtient une fonction qui ne dépend que des seules variables indépendantes

$$F (X_H) = f (X_H, K (X_H))$$

Cependant, dans la plupart des cas cette élimination formelle est impossible aussi, nous remplacerons cette élimination formelle par un algorithme qui à partir du système $\varphi_i (X_B, X_H) = 0$ permet le calcul des valeurs numériques des variables de X_B et de F (X_H) en fonction des valeurs numériques des variables indépendantes. C'est la technique d'élimination numérique.

L'essentiel du mécanisme consiste dans la résolution par rapport aux variables de base du système d'équations $\varphi_i (X_B, X_H) = 0$. La méthode choisie est la méthode de linéarisation (ou de Newton).

Les détails de cette mise en œuvre sont donnés dans la partie des programmes.

II - 1 - 2 Gradient réduit

La fonction F (X_H) est une fonction des seules variables indépendantes X_H, aussi, pour pouvoir lui appliquer des techniques liées à la méthode du gradient nous avons besoin des dérivées de F (X_H) par rapport aux variables libres.

Pour les distinguer du gradient de f : $\frac{\partial f}{\partial x_i} (x_1, \ldots, x_n)$ nous le désignerons par gradient réduit et noterons F' (X) le tableau $\frac{\partial F (X_H)}{\partial x_h}$ où x_h est une variable libre.

Introduisons les matrices suivantes :

$$[\, f' (X) \,] = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} \qquad [\, f'_H \,] = \begin{bmatrix} \frac{\partial f}{\partial x_{h_1}} \\ \vdots \\ \frac{\partial f}{\partial x_{h_{n-p}}} \end{bmatrix} \qquad [\, f'_B \,] = \begin{bmatrix} \frac{\partial f}{\partial x_{b_1}} \\ \vdots \\ \frac{\partial f}{\partial x_{b_p}} \end{bmatrix}$$

où $x_{h_1}, \ldots, x_{h_{n-p}}$ sont les n - p variables indépendantes

$x_{b_1}, \ldots, x_{b_p}$ étant les p variables de base.

De même nous introduisons pour les contraintes les matrices Φ', Φ'_H et Φ'_B définies par

$$[\ \Phi'\ ] = \begin{bmatrix} \frac{\partial \varphi_1}{\partial x_1} & \cdots & \frac{\partial \varphi_1}{\partial x_n} \\ \vdots & & \\ \frac{\partial \varphi_p}{\partial x_1} & \cdots & \frac{\partial \varphi_p}{\partial x_n} \end{bmatrix} \qquad [\ \Phi'_H\ ]^{p}_{n-p} = \begin{bmatrix} \frac{\partial \varphi_1}{\partial x_{h_1}} & \cdots & \frac{\partial \varphi_1}{\partial x_{h_{n-p}}} \\ \vdots & & \\ \frac{\partial \varphi_p}{\partial x_{h_1}} & \cdots & \frac{\partial \varphi_p}{\partial x_{h_{n-p}}} \end{bmatrix}$$

$$[\ \Phi'_B\ ]^{p}_{p} = \begin{bmatrix} \frac{\partial \varphi_1}{\partial x_{b_1}} & \cdots & \frac{\partial \varphi_1}{\partial x_{b_p}} \\ \vdots & & \\ \frac{\partial \varphi_p}{\partial x_{b_1}} & \cdots & \frac{\partial \varphi_p}{\partial x_{b_p}} \end{bmatrix}$$

Soit x_h une variable libre, en dérivant $F(X_H) \equiv f(X_B, X_H) \equiv f(X_H, K(X_H))$ nous obtenons

$$\frac{\partial F}{\partial x_h} = \frac{\partial f}{\partial x_h} + \sum_{i \in B} \frac{\partial f}{\partial x_b} \frac{\partial x_b}{\partial x_h} \qquad (1)$$

de même à partir de $\varphi_i(X_H, X_B) = 0$ nous obtenons

$$\frac{\partial \varphi_i}{\partial x_h} + \sum_{i \in B} \frac{\partial \varphi_i}{\partial x_b} \frac{\partial x_b}{\partial x_h} = 0 \qquad (2)$$

Soit

$$[\ \Delta\ ]^{n-p}_{p} = \begin{bmatrix} \frac{\partial x_{b_1}}{\partial x_{h_1}} & \cdots & \frac{\partial x_{b_p}}{\partial x_{h_1}} \\ \vdots & & \\ \frac{\partial x_{b_1}}{\partial x_{h_{n-p}}} & \cdots & \frac{\partial x_{b_p}}{\partial x_{h_{n-p}}} \end{bmatrix}$$

Les relations (1) s'écrivent $[\ F'\ ] = [\ f'_H\ ] + \Delta\ .\ [\ f'_B\ ]$

et les relations (2) deviennent $[\ \Phi'_H\ ] + [\ \Phi'_B\ ]\ .\ {}^t\Delta = 0$

d'où nous tirons

$$\Delta = -\ {}^t[\ \Phi'_H\ ]\ .\ {}^t[\ \Phi'_B\ ]^{-1}$$

et finalement $$\boxed{[\ F'\ ] = [\ f'_H\ ] - {}^t[\ \Phi'_H\ ]\ .\ {}^t[\ \Phi'_B\ ]^{-1}[\ f'_B\ ]}$$

D'un point de vue numérique, il est intéressant d'introduire la matrice colonne $[\ \Lambda\ ]$ définie par

$$[\ \Lambda\ ] = {}^t[\ \Phi'_B\ ]^{-1}[\ f'_B\ ]$$

$[\ \Lambda\ ]$ est donc solution du système ${}^t[\ \Phi'_B\ ]\ .\ [\ \Lambda\ ] = [\ f'_B\ ]$

et $$[\ F'\ ] = [\ f'_H\ ] - [\ \Phi'_H\ ]\ .\ [\ \Lambda\ ] \qquad (3)$$

Expression dont l'écriture habituelle est

$$\frac{\partial F}{\partial x_h} = \frac{\partial f'}{\partial x_h} - \sum_i \lambda_i \frac{\partial \varphi_i}{\partial x_h}$$

on constate donc que le calcul du gradient réduit nécessite

a) la résolution d'un système linéaire pour déterminer les coefficients de Lagrange $\lambda_1, \lambda_2, \ldots, \lambda_p$.

b) L'application des formules (3) soit la multiplication d'une matrice par un vecteur colonne.

Cette méthode est beaucoup plus rapide que la mise en œuvre de la formule obtenue initialement.

II - 2 OPTIMISATION EN CONTRAINTES MIXTES

II 2 - 1 Caractérisation de l'optimum. Conditions de Kuhn et Tucker

II.2.1.1 Théorème de Farkas - Minkowski

Dans un espace de dimension n, étant donné un premier groupe de p formes linéaires $U_\ell(X)$ et un second groupe de q formes $V_j(X)$, ces p + q formes étant supposées indépendantes. Soit W (X) une forme linéaire donnée.

Pour que W (X) soit négative ou nulle pour tout tableau X vérifiant les conditions $\begin{cases} U_\ell(X) = 0 & \ell \in L \\ V_j(X) \geq 0 & j \in J \end{cases}$

il faut et il suffit qu'il existe des nombres λ_ℓ et μ_j tels que :

$$\begin{cases} W(X) = \sum_{\ell \in L} \lambda_\ell U_\ell(X) + \sum_{j \in J} \lambda_j V_j(X) \\ \text{avec} \quad \mu_j \leq 0 \quad \forall j \end{cases}$$

II.2.1.2 Etude d'un maximum local

Soit X^* le tableau des coordonnées d'un maximum local, J et K les ensembles des indices des contraintes unilatérales respectivement saturées et libres en X^*.

X^* doit vérifier

$$\begin{cases} f(X^*) \geq f(X) \\ \forall X, \quad \begin{cases} \varphi_\ell(X) = 0 \\ \varphi_j(X) \geq 0 \\ \varphi_k(X) \geq 0 \end{cases} \end{cases} \qquad (4)$$

sachant que $\begin{cases} \varphi_\ell(X^*) = 0 \\ \varphi_j(X^*) = 0 \\ \varphi_k(X^*) > 0 \end{cases}$

Le fait de se limiter à une étude locale revient à accepter les simplifications suivantes obtenues par linéarisation en $X - X^*$:

$$\begin{cases} f(X) = f(X^*) + f'(X^*) \cdot [X - X^*] \\ \varphi_i(X) = \varphi_i(X^*) + \varphi_i'(X^*) \cdot [X - X^*] \\ \forall i \in \{L \cup J \cup K\} \end{cases}$$

Les équations précédentes peuvent s'expliciter en :

$$f'(X^*) \cdot [X - X^*] = \sum_{s=1}^{n} \frac{\partial f}{\partial x_s}(X^*) \cdot (x_s - x_s^*)$$

$$\varphi_i'(X^*) \cdot [X - X^*] = \sum_{s=1}^{n} \frac{\partial \varphi_i}{\partial x_s}(X^*)\,(x_s - x_s^*)$$

Dans ces conditions les relations (4) deviennent :

$$\begin{cases} f'(X^*) \cdot [X - X^*] \leq 0 \\ \varphi_\ell'(X^*) \cdot [X - X^*] = 0 \qquad \ell \in L \end{cases} \qquad (5)$$

$$\begin{cases} \varphi'_j(X^*) \cdot [X - X^*] \geq 0 & j \in J \\ \varphi_k(X^*) + \varphi'_k(X^*) \cdot [X - X^*] > 0 & k \in K. \end{cases} \qquad (6)$$

Mais $\varphi_k(X^*) > 0$ entraîne que la dernière inégalité sera toujours vérifiée si l'on prend $X - X^*$ assez petit, donc : les contraintes unilatérales vérifiées mais non saturées n'interviennent pas dans l'étude d'un maximum local.

Or, les premiers membres des relations (5) sont p formes linéaires qui jouent le rôle des U (X) du théorème de Farkas - Minkowski les premiers membres des inégalités (6) remplaçant les formes V (X) et W (X) étant ici $f'(X^*) \cdot [X - X^*]$.

Dans ces conditions, si les p + q formes linéaires sont indépendantes (les contraintes sont alors dites régulières), nous pouvons appliquer le théorème de Farkas - Minkowski et :

$$\begin{cases} \exists \lambda_\ell, \lambda_j \quad f'(X^*)[X - X^*] = \sum_{\ell \in L} \lambda_\ell \varphi'_\ell(X^*)(X - X^*) \\ \qquad\qquad + \sum_{j \in J} \lambda_j \varphi'_j(X^*)(X - X^*) \\ \lambda_j \leq 0 \quad \forall j \in J. \end{cases}$$

Il faut que cette relation soit vérifiée pour tout X appartenant au domaine admissible.

Ce sera le cas si

$$f'(X^*) = \sum_{\ell \in L} \lambda_\ell \varphi'_\ell(X^*) + \sum_{j \in J} \lambda_j \varphi'_j(X^*)$$

Soit $[\varphi']$ la matrice définie par

$$[\varphi'] = \begin{bmatrix} \frac{\partial \varphi_1}{\partial x_1} & \cdots & \frac{\partial \varphi_{p+q}}{\partial x_1} \\ \vdots & & \\ \frac{\partial \varphi_1}{\partial x_n} & \cdots & \frac{\partial \varphi_{p+q}}{\partial x_n} \end{bmatrix} \quad \text{et} \quad [\Lambda] = \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_{p+q} \end{bmatrix}$$

Les équations précédentes peuvent s'écrire

$$[f'(X^*)] = [\varphi'(X^*)] \cdot [\Lambda]$$

Les λ_ℓ sont les multiplicateurs de Lagrange et les λ_j sont les multiplicateurs de Kuhn et Tucker.

Les équations caractérisant un maximum local sont :

- les n relations

$$\frac{\partial f}{\partial x_s}(X^*) - \sum_i \lambda_i \frac{\partial \varphi_i}{\partial x_s}(X^*) = 0$$

- les p + q contraintes

$$\varphi_i(X^*) = 0$$

Ce système dépend donc des contraintes que nous supposons saturées en X^*. Nous pouvons faire un choix arbitraire et tester par la suite si notre choix est bon en vérifiant que les multiplicateurs de Kuhn et Tucker sont bien tous négatifs ou nuls. Dans le cas contraire il faut essayer d'autres choix.

Cette méthode est envisageable, en particulier dans le cas ou le nombre des contraintes unilatérales est faible mais il semble préférable

de recourir à des méthodes de montée qui permettent d'obtenir des valeurs croissantes de f (X) en nous indiquant les contraintes à saturer.

II. 2. 2 Méthode simple de montée

II. 2. 2. 1 Recherche de la meilleure direction de montée locale associée à un ensemble de contraintes

Soit un tableau X^r donné. On se propose de chercher une direction D admissible de norme δ donnée telle que pour h positif donné

$$f(X^r + h\,D) \text{ soit maximum}$$

Ce qui peut s'écrire

$$\begin{cases} \max f(X^r + h\,D) \text{ si} \\ \Sigma\, d_i^2 = \delta^2 \;;\; \varphi_\ell(X^r) = 0 \;;\; \varphi_j(X^r) = 0 \\ \varphi_\ell(X^r + h\,D) = 0 \qquad \ell \in L \\ \varphi_j(X^r + h\,D) \geq 0 \qquad j \in J \end{cases}$$

Dans le cas général, le problème ainsi posé est insoluble formellement. Nous le simplifions en cherchant une meilleure direction locale, nous acceptons donc de linéariser la fonction à maximiser et les contraintes et le problème devient :

$$\begin{cases} \max D\,f'(X^r) = \Omega \\ \psi_0 = \sum_1^n d_s^2 - \delta^2 = 0 \\ \psi_\ell = D \cdot \varphi'_\ell(X^r) = 0 \qquad \ell \in L \\ \psi_j = D \cdot \varphi'_j(X^r) \geq 0 \qquad j \in J \end{cases}$$

Soit avec des notations classiques

$$\begin{cases} D \cdot f'(X^r) = \sum_{s=1}^n d_s \dfrac{\partial f}{\partial x_s}(X^r) \\ D\,\varphi'_i(X^r) = \sum_{s=1}^n d_s \dfrac{\partial \varphi_i}{\partial x_s}(X^r) \end{cases}$$

Même ainsi simplifié le problème est difficile à traiter, aussi nous nous contenterons :

a) de vérifier que la direction de meilleure montée obtenue en gardant saturée pendant l'étape d'origine X^r les contraintes saturées en X^r est satisfaisante, c'est-à-dire que tous les coefficients de Kuhn et Tucker sont négatifs ou nuls et, dans le cas contraire :

b) nous chercherons une direction de montée admissible pour laquelle nous ne libérerons qu'une seule des contraintes saturées en X^r.

Nous maintenons saturées toutes les contraintes saturées en X^r. Les équations générales du paragraphe précédent deviennent

$$\begin{cases} \Omega' = \lambda_o \psi'_o + \Sigma_\ell \lambda_\ell \psi'_\ell + \Sigma_j \lambda_j \psi'_j \qquad (7) \\ \psi_o = 0 \\ \psi_\ell = 0 \\ \psi_j = 0 \end{cases}$$

Mais Ω, ψ_ℓ et ψ_j étant linéaires par rapport aux d_s, nous obtenons

$$\begin{cases} \dfrac{\partial \Omega}{\partial d_s} = \dfrac{\partial f}{\partial x_s}\ (X^r) \qquad \text{soit} \quad \Omega' = f'\ (X^r) \\ \\ s = 1, \ \ldots, \ n \end{cases}$$

de même

$$\begin{cases} \dfrac{\partial \psi_i}{\partial d_s} = \dfrac{\partial \varphi_i}{\partial x_s}\ (X^r) \quad \text{soit} \quad \psi_i' = \varphi_i'\ (X^r) \\ \\ i \in \{L \cup J\} \end{cases}$$

et $\psi_o' = 2\ D$

La relation (7) devient alors

$$f'\ (X^r) = 2\ \lambda_o\ D + \sum_\ell \lambda_\ell\ \varphi_\ell'\ (X^r) + \sum_j \lambda_j\ \varphi_j'\ (X^r)$$

et en posant $2\ \lambda_o = 1$ nous obtenons la direction de montée D par

$$\boxed{\begin{array}{l} D = f'\ (X^r) - \sum_i \lambda_i\ \varphi_i'\ (X^r) \\ \\ i \in \{L \cup J\} \end{array}}$$

Les λ étant calculés en résolvant le système des contraintes

$$\boxed{D\ \varphi_i' = 0 \qquad i \in \{L \cup J\}}$$

Introduisons alors la matrice carrée symétrique B d'élément générique

$$b_{ij} = \varphi_i'\ (X^r)\ \varphi_j'\ (X^r) = \sum_{s=1}^{n} \frac{\partial \varphi_i}{\partial x_s}\ (X^r)\ \frac{\partial \varphi_j}{\partial x_s}\ (X^r)$$

et la matrice E d'élément

$$e_i = f'\ (X^r)\ .\ \varphi_i'\ (X^r) = \sum_{x=1}^{n} \frac{\partial f}{\partial x_s}\ \frac{\partial \varphi_i}{\partial x_s}\ (X^r)$$

Soit Λ la matrice colonne d'élément λ_i.

Le système définissant les λ_i devient

$$[\ B\ ]\ .\ [\ \Lambda\ ] = [\ E\ ]$$

et la résolution de ce système définit les λ_i qui dans le cas général nous donne une direction D.

$$D = f'\ (X^r) - \sum_i \lambda_i\ \varphi_i'\ (X^r)$$

Cette direction est la direction locale de meilleure montée, si nous maintenons saturées toutes les contraintes saturées en X^r.

Si tous les coefficients de Kuhn et Tucker sont négatifs ou nuls, nous savons de plus que cette direction reste direction de meilleure montée même si nous libérons toutes les contraintes unilatérales. Donc, nous avons intérêt pendant cette étape à maintenir saturées les contraintes qui étaient saturées en son origine X^r.

Si au contraire un au moins des coefficients de Kuhn et Tucker est positif, cela montre qu'il y a au voisinage de D des directions de montée plus satisfaisantes que D. On a alors intérêt à libérer une ou plusieurs contraintes.

Nous nous contenterons dans ce cas de chercher une direction de

montée admissible en ne libérant qu'une contrainte correspondant à un coefficient de Kuhn et Tucker positif.

Cependant, avant d'effectuer cette recherche, nous allons donner la méthode pratique de calcul de la direction de montée D définie précédemment.

Introduisons à priori une direction D' de la forme

$$D' = f'(X^r) - \sum_i \lambda'_i \varphi'_i(X^r) \qquad i \in \{L \cup J\}$$

où les inconnus λ'_i devront minimiser

$$\|D'\| = \sqrt{\sum_i d_i^2}$$

Introduisons les matrices

$$[\Phi'] = \begin{bmatrix} \frac{\partial \varphi_1}{\partial x_1} & \cdots & \frac{\partial \varphi_{p+q}}{\partial x_1} \\ \vdots & & \vdots \\ \frac{\partial \varphi_1}{\partial x_n} & \cdots & \frac{\partial \varphi_{p+q}}{\partial x_n} \end{bmatrix} \qquad [\mathcal{A}] = \begin{bmatrix} \lambda'_r \\ \vdots \\ \lambda'_{p+q} \end{bmatrix}$$

on a alors $D' = f'(X^r) - \Phi' \cdot \mathcal{A}'$

Un procédé classique pour trouver les λ' qui minimisent $\|D\|$ nous est donné par la méthode de Gauss qui conduit à :

$${}^t\Phi' \cdot \Phi' \cdot \mathcal{A}' = {}^t\Phi' \cdot f'$$

et il est clair que ${}^t\Phi' \cdot \Phi' = B$

$${}^t\Phi' \cdot f' = E$$

donc, les tableaux $\mathcal{A}$ et $\mathcal{A}'$ sont identiques.

En fait ce n'est pas tout à fait cette méthode qui a été programmée mais une méthode de triangulisation qui permet la recherche directe des coefficients λ_i (sous-programme MCADIR).

Remarquons enfin, que $B = {}^t\Phi' \cdot \Phi'$ montre que B est une matrice définie positive. Ce fait nous sera utile par la suite.

II.2.2.2 Cas ou une contrainte est libérée - Recherche d'une direction de montée admissible

Supposons donc qu'un au moins des coefficients de Kuhn et Tucker soit positif et soit λ_{j_1} ce coefficient.

Désignons par J' l'ensemble des indices de J autre que j_1. Nous allons libérer la seule contrainte d'indice j_1 de sorte que toute contrainte d'indice $j' \in J'$ restera saturée pendant l'étape d'origine X^r.

Nous allons montrer que la direction D' de meilleure montée pour l'ensemble des contraintes d'indices appartenant à L ou J' est une direction de montée admissible compatible avec la liaison libérée.

Nous savons que D' satisfait à :

$$\begin{cases} D' = f'(X^r) - \sum_i \lambda'_i \varphi'_i(X^r) & i \in \{L \cup J'\} \\ D' \varphi'_i(X^r) = 0 \end{cases}$$

Dire que D' est compatible avec la liaison libérée, c'est dire que

$$\varphi_{j_1}(X^r + h D') \geq 0$$

condition linéarisée en

$$\varphi_{j_1}(X^r) + h\, D' \varphi'_{j_1}(X^r) \geq 0$$

or $\begin{cases} \varphi_{j_1}(X^r) = 0 \implies D' \varphi'_{j_j}(X^r) \geq 0 \\ h > 0 \end{cases}$

Introduisons le coefficient de Kuhn et Tucker λ'_{j_1} nul par hypothèse et soit

$$z_{j_1} = D' \varphi'_{j_1}(X^r)$$

Les relations précédentes peuvent s'écrire

$$\begin{cases} D' = f'(X^r) - \sum_i \lambda'_i \varphi'_i(X^r) & i \in \{L \cup J\} \quad (8) \\ \mathfrak{Z} = D' \varphi'_i(X^r) & (9) \end{cases}$$

ou $\mathfrak{Z}$ est un tableau dont tous les éléments sont nuls à l'exception de z_{j_1}.

Les relations (8) nous permettent de calculer D' qui, remplacée dans (9) donne :

$$\left(\sum_j \lambda'_j \varphi'_j(X^r)\right) \varphi'_i(X^r) = f'(X^r) \varphi'_i(X^r) - \mathfrak{Z} \qquad i \in \{L \cup J\}.$$

Si nous introduisons alors les matrices B et E déjà rencontrées, nous obtenons le système matriciel :

$$B \,.\, \mathcal{\lambda}' = E - \mathfrak{Z}$$

Les coefficients λ relatifs à D étaient solutions de

$$B \,.\, \mathcal{\lambda} = E$$

Ce qui entraîne :

$$B \,.\, (\mathcal{\lambda} - \mathcal{\lambda}') = \mathfrak{Z}$$

et $${}^t(\mathcal{\lambda} - \mathcal{\lambda}')\, B\, (\mathcal{\lambda} - \mathcal{\lambda}') = {}^t(\mathcal{\lambda} - \mathcal{\lambda}')\, \mathfrak{Z}.$$

Or, nous savons que B est une matrice définie positive donc :

$${}^t(\mathcal{\lambda} - \mathcal{\lambda}') \,.\, \mathfrak{Z} \geq 0$$

or, $\mathfrak{Z}$ possède un seul élément non nul : z_{j_1} donc

$$(\lambda_{j_1} - \lambda'_{j_1}) \,.\, \mathfrak{Z}_{j_1} \geq 0$$

mais $\lambda'_{j_1} = 0$ par hypothèse, d'où finalement

$$\boxed{\lambda_{j_1}\, \mathfrak{Z}_{j_1} \geq 0}$$

Nous voyons donc que, si nous libérons une contrainte correspondant à un coefficient de Kuhn et Tucker strictement positif, nous obtenons une direction de montée D' compatible avec la liaison libérée.

Si tous les coefficients λ'_j ou $j \in J'$ sont négatifs ou nuls, ce sera une direction de meilleure montée, sinon, nous savons que c'est une direction de montée certaine et nous nous en contenterons.

Remarque : Si n = 3 et si les contraintes sont au nombre de 1 ou 2, la direction D est la projection orthogonale du gradient de f(X) soit sur le plan tangent à la surface $\varphi_1 = 0$. soit sur la tangente à la courbe d'intersection de $\varphi_1 = 0$ et $\varphi_2 = 0$ d'où le nom de "gradient projeté" donné à cette direction.

II - 3 METHODE DE MONTEE AVEC ACCELERATION DE CONVERGENCE

Si nous utilisons le gradient projeté comme direction de montée, nous ne pouvons pas mettre en œuvre l'accélération de convergence car cette technique ne s'applique qu'au gradient réduit.

Mais si nous utilisons la direction du gradient réduit comme direction de montée nous ne savons plus ce qui se passe lorsque nous libérons une contrainte. En effet la direction du gradient réduit n'a aucune raison d'être compatible avec la liaison libérée.

Pour pouvoir utiliser l'accélération de convergence dans une optimisation avec contraintes unilatérales, il est donc nécessaire dans le même calcul d'utiliser tantôt l'une tantôt l'autre des directions de montée.

Nous avons programmé la méthode suivante : on commence par une première itération avec comme direction de montée le gradient projeté et on calcule les coefficients de Kuhn et Tucker.

- S'ils sont tous négatifs, on entame un cycle de n + 1 itérations pendant lequel la direction de montée sera celle du gradient réduit. Il y a sortie du cycle et retour au gradient projeté dès qu'il y a modification des contraintes saturées (c'est-à-dire si l'on rencontre une nouvelle contrainte) ou bien s'il y a un incident pendant le traitement. A la fin du cycle on essaie l'accélération de convergence.

- Si l'un au moins des coefficients de Kuhn et Tucker est positif, on garde la direction du gradient projeté comme direction de montée.

CHAPITRE III

OPTIMISATION EN ESPACE INFINI

- CONTROLE OPTIMAL

III - 1 PROBLEME A CRITERE TERMINAL SANS CONTRAINTE

III.1.1 Position du problème

Soit le système différentiel

$$\begin{cases} \dfrac{dX}{dt} = F(X, U, t) \\ X(0) = X_0 \text{ donné} \end{cases} \qquad 0 \leq t \leq T$$

ou $F = \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}$ et soit $\Omega = g(x_1(T), \ldots, x_n(T))$

ou Ω est un critère ne dépendant que des valeurs finales du vecteur d'état X.

On se propose d'optimiser Ω,

on remarque en fait que Ω dépend du vecteur de commande u. Nous écrirons $\Omega = \Omega[u(t)]$.

Dans certains cas simples Ω est une fonction explicite de u et il est possible de trouver une solution formelle au problème, mais, dans le cas général, nous n'aurons à notre disposition qu'un procédé numérique qui nous permettra de calculer le nombre Ω connaissant les fonctions u (t).

Nous ne pourrons alors caractériser chaque fonction que par un nombre fini d'informations permettant de l'approcher d'où une double difficulté : sur le plan théorique d'abord car en agissant ainsi, on abime le

problème et on ne sait pas trop à quel niveau sur le plan numérique ensuite car un choix se pose sur le mode de représentation à utiliser.

En fait, il est impossible de définir un mode de représentation utilisable dans tous les cas, nous ne pouvons que donner deux familles de représentations :

- une représentation par une écriture formelle approchée :

un polynôme, une fraction rationnelle, etc...

de type $a_1 e^{-a_2 t} + a_3$, $\Sigma a_i u_i(t)$ etc...

nous noterons ce mode $u = u(a_1, a_2, \ldots, a_s)$

- une représentation par une table de n valeurs complétée par une règle d'interpolation.

Une fois fixée la règle d'interpolation, u n'est fonction que des valeurs tabulées $u_1, u_2, \ldots, u_i = u(t_i), \ldots, u_n$,

nous noterons $u = u(u_1, u_2, \ldots, u_n)$.

Nous verrons que dans le cas d'une table dense nous n'aurons plus besoin de règle d'interpolation, seules les valeurs tabulées de u seront utilisées.

Que nous choisissions l'une ou l'autre de ces méthodes pour représenter u (t), nous avons remplacé le problème d'optimisation de la fonctionnelle par celui de l'optimisation de la fonction $\Omega(a_1, \ldots, a_s)$ ou $\Omega(u_1, \ldots, u_n)$.

Moyennant cette approximation, nous nous sommes ramenés aux

problèmes traités dans les chapitres précédents et nous pouvons donc utiliser les méthodes exposées. La seule difficulté pratique est le calcul du gradient mais ceci peut être résolu par l'utilisation d'une formule d'approximation n'utilisant que les valeurs de la fonction Ω.

Cependant, dans le cas d'une fonctionnelle dépendant uniquement des variables terminales (ce qui est notre cas), nous allons voir que nous pouvons calculer strictement son gradient. C'est-à-dire que nous pourrons remplacer les formules approchées précédentes par un calcul strict ce qui est toujours préférable et un exemple simple nous le montrera aisément. (on verra de plus qu'avec le calcul strict le temps de calcul est diminué).

III.1.2 Calcul du gradient d'une fonctionnelle des valeurs terminales X (T)

Notations :

Soit une fonctionnelle G (y) dépendant de l'argument y (t) et $a \leq t \leq b$.

Donnons à y (t) un accroissement $\delta y = \varepsilon \theta(t)$

Lorsque $y(t) \longrightarrow y(t) + \varepsilon \theta(t)$, Ω subit un accroissement

$$\Delta\Omega = \Omega[\, y(t) + \varepsilon\, \theta(t)\,] - \Omega(y(A))$$

on appelle différentielle de la fonctionnelle Ω et on note Ω la limite si elle existe de $\frac{\Delta\Omega}{\varepsilon}$ lorsque $\varepsilon \longrightarrow 0$

$$d\Omega = \lim_{\varepsilon \longrightarrow 0} \frac{\delta \Omega}{\varepsilon}$$

$d\Omega$ dépend de $y(t)$ et de $\theta(t)$.

Considérons maintenant la classe restreinte de fonctions $\omega_\tau(t)$ telle que :

$$\omega_\tau(t) = 0 \qquad \text{si } t \notin [\tau - \frac{h}{2}, \tau + \frac{h}{2}]$$

et $\omega_\tau(t)$ de signe constant sur $[\tau - \frac{h}{2}, \tau + \frac{h}{2}]$

Soit
$$\begin{cases} \mu = \max |\omega_\tau(t)| & a \leq t \leq b \\ \sigma = \displaystyle\int_{\tau - \frac{h}{2}}^{\tau + \frac{h}{2}} \omega_\tau(t)\, dt \end{cases}$$

Formons le rapport $\dfrac{\Omega[y + \omega_\tau(t)] - \Omega(y)]}{\sigma}$

On appelle dérivée fonctionnelle de $\Omega(y)$ en τ notée $\Omega'_\tau(y)$ la limite si elle existe du rapport précédent lorsque h et μ tendent simultanément vers 0. $\Omega'_\tau(y)$ est une fonction de τ.

Différentielle et dérivée fonctionnelle sont liées par la relation fondamentale suivante :

$$\boxed{\delta \Omega(y) = \int_a^b \Omega'_\tau [y(t)]\ \delta y(t)\, dt}$$

Dans toute la suite nous nous limiterons à une seule fonction de commande et ceci afin de simplifier les notations car tout ce qui suit s'étend facilement

au cas d'un vecteur de commande.

Soit donc le système différentiel

$$\begin{cases} \dfrac{dX}{dt} = F(X, U, t) \\ X_0 \text{ donné,} \end{cases}$$

A une variation δu de u correspond une variation $\delta X(t)$ de X, et

$$\frac{d}{dt}[X + \delta X] = F(X,U,t) + F'_X(X,U,t)\,\delta X + F'_U(X,U,t)\,\delta u$$

d'où :

$$\begin{cases} \dfrac{d}{dt}[\delta X(t)] = F'_X(X,U,t)\,\delta X + F'_U(X,U,t)\,\delta u \\ \delta X(0) = 0 \end{cases}$$

Remarquons que nous travaillons sur des tableaux et que par exemple :

$$F'_X\ \delta X = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \cdots & \dfrac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \dfrac{\partial f_n}{\partial x_1} & \cdots & \dfrac{\partial f_n}{\partial x_n} \end{bmatrix} \quad \begin{bmatrix} \delta x_1 \\ \vdots \\ \delta x_n \end{bmatrix}$$

Afin de progresser, limitons nous à des accroissements $\delta u(t)$ de la classe ω_τ c'est-à-dire que :

$$\delta u(t) = \begin{cases} 0 & \text{si } t \notin [\tau - \frac{h}{2}, \tau + \frac{h}{2}] \\ \varepsilon & \text{si } t \in [\tau - \frac{h}{2}, \tau + \frac{h}{2}] \end{cases}$$

avec ces hypothèses, sur $t \notin [\tau - \frac{h}{2}, \tau + \frac{h}{2}]$:

$$\begin{cases} \dfrac{d}{dt}[\delta X(t)] = F'_X(X,U,t)\,\delta X(t) \\ \delta X_0 = 0 \end{cases}$$

or, le système étant linéaire et homogène, la condition $\delta X_0 = 0$ impose que :

$$\delta X(t) \equiv 0 \qquad t \notin [\tau - \frac{h}{2}, \tau + \frac{h}{2}]$$

Pour $\tau - \frac{h}{2} \leq t \leq \tau + \frac{h}{2}$, le système peut s'écrire :

$$\frac{d}{dt}[\delta X(t)] = F'_X(X,U,t)\,\delta X(t) + \varepsilon F'_U(X,U,t)$$

Intégrons le système entre $\tau - \frac{h}{2}$ et $\tau + \frac{h}{2}$ en tenant compte du fait que $\delta X(\tau - \frac{h}{2}) = 0$, nous obtenons :

$$\delta X(\tau + \frac{h}{2}) = \int_{\tau - \frac{h}{2}}^{\tau + \frac{h}{2}} F'_X(X,U,t)\,\delta X(t)\,dt + \varepsilon \int_{\tau - \frac{h}{2}}^{\tau + \frac{h}{2}} F'_U(X,U,t)\,dt.$$

Lorsque h et ε tendent simultanément vers 0, $\delta X(\tau + \frac{h}{2})$ est un infiniment petit du même ordre que εh, donc le premier terme du second membre devient négligeable devant le second et

$$\delta X(\tau_+) = \varepsilon h\, F'_U[X(\tau), u(\tau), \tau]$$

Introduisons le système

$$\frac{dP}{dt} = -\,t\,F'_X \cdot P(t) \text{ adjoint de } \frac{d\delta X(t)}{dt} = F'_X \cdot \delta X.$$

La variation de la fonction $g[X(T)]$ est :

$$\Delta g = g[X - \delta X] - g[X]$$

Supposons δX infiniment petit, nous avons :

$$\Delta g = \frac{\partial g}{\partial x_1}[X(T)]\,\delta x_1 + \ldots + \frac{\partial g}{\partial x_n}[X(T)]\,\delta x_n$$

Fixons alors comme conditions "initiales" du système adjoint :

$$P(T) = (\text{grad } g)_{t=T}$$

soit, si $P = \begin{bmatrix} P_1 \\ \vdots \\ P_n \end{bmatrix}$ $\begin{cases} P_1(T) = \left(\frac{\partial g}{\partial x_1}\right)_{t=T} \\ \vdots \\ P_n(T) = \left(\frac{\partial g}{\partial x_n}\right)_{t=T} \end{cases}$

Les propriétés des systèmes adjoints nous permettent d'écrire l'égalité des produits scalaires :

$$\langle \delta X(T),\ P(T) \rangle = \langle \delta X(\tau_+),\ P(\tau) \rangle$$

soit

$$\Delta g = p_1(\tau_+)\,\delta x_1 + \ldots + p_n(\tau_+)\,\delta x_n(\tau_+)$$

or, par définition, la dérivée fonctionnelle de g est :

$$g'_\tau = \lim_{\sigma \to 0} \frac{\Delta g}{\sigma} \quad \text{ou} \quad \sigma = \int_{\tau - \frac{h}{2}}^{\tau + \frac{h}{2}} \omega_\tau(t)\,dt$$

ici $\begin{cases} \delta X(\tau_+) = \varepsilon h\, F'_u \\ \tau = \varepsilon h \end{cases}$

donc $g'_\tau = \langle P(\tau),\ F'_u \rangle$

Pour retrouver les notations classiques, introduisons l'Hamiltonien H défini par :

$$H = f_1(X,U,t)\,p_1 + \ldots\ldots + f_n(X,U,t)\,p_n(t)$$

quantité fonction des variables indépendantes X, U, P, t.

On peut alors écrire :

$$\boxed{g'_{\tau} = \left(\frac{\partial H}{\partial u}\right)_{\tau}}$$

En résumé, pour calculer g'_{τ} , dans l'expression arithmétique $\frac{\partial H}{\partial u}$ qui dépend de $X(\tau)$, $P(\tau)$, $u(\tau)$ et τ, nous exécutons les opérations suivantes :

$u(t)$ étant donné, les $X(\tau)$ s'obtiennent par intégration du système

$$\begin{cases} \frac{d}{dt}\,[\,X(t)\,] = F(X,U,t) \\ X_0 \text{ donné} \end{cases}$$

$P(\tau)$ se calcule par intégration du système adjoint

$$\begin{cases} \frac{d}{dt}\,[\,P(t)\,] = -{}^{t}F'_{X} \,.\, P(t) \\ P(T) = (\text{grad } g)_{t=T} \end{cases}$$

Nous pouvons maintenant résoudre le problème posé initialement est qui était de calculer $\frac{\partial \Omega}{\partial a_i}$ ou $\frac{\partial \Omega}{\partial u_i}$ expressions dans lesquelles

$$\Omega = g\,[\,x_1(T), \ldots, x_n(T)\,]$$

et $u = u(a_1, \ldots, a_s)$ ou $u = u(u_i, \ldots, u_n)$ selon le mode de représentation choisi.

Supposons que $u = u(a_1, \ldots, a_s)$. Donnons à a_i un accroissement δa_i

$$\delta u(t) = \frac{\partial u}{\partial a_i}\,\delta a_i$$

or, la relation fondamentale liant différentielle et dérivée fonctionnelle montre que

$$\delta \Omega = \int_0^T g'_{\tau}(t)\,\delta u(t)\,dt$$

mais $g'_{\tau}(t) = \frac{\partial H}{\partial u}$ donc

$$\delta \Omega = \int_0^T \frac{\partial H}{\partial u}(X,U,P,t)\,\frac{\partial u}{\partial a_i}\,\delta a_i\,dt$$

divisons les deux membres par δa_i et faisons tendre δa_i vers 0 :

$$\boxed{\frac{\partial \Omega}{\partial a_i} = \int_0^T \frac{\partial H}{\partial u}(X,U,P,t)\,\frac{\partial u}{\partial a_i}\,dt}$$

Un raisonnement strictement semblable montre que, pour une représentation avec table nous aurons de même :

$$\frac{\partial \Omega}{\partial u_i} = \int_0^T \frac{\partial H}{\partial u}\,\frac{\partial u}{\partial u_i}\,dt.$$

Dans ce deuxième cas, une simplification va cependant se produire dans le cas d'une table dense.

En effet, $\delta \Omega = \int_0^T \frac{\partial H}{\partial u}\,\delta u\,dt$ est calculé numériquement par

$$\delta \Omega = \sum_j e_j \left(\frac{\partial H}{\partial u}\,\delta u\right)_{tj}$$

ou e_j et t_j dépendent de la méthode d'intégration utilisée.

Si toutes les valeurs $u_j = u(t_j)$ appartiennent à la table de définition de $u(t)$ (cas d'une table dense), en ne modifiant qu'une seule valeur u_j nous obtenons

$$\delta \Omega = e_j \left(\frac{\partial H}{\partial u}\right)_{tj} \delta u_j$$

et

$$\frac{\partial \Omega}{\partial u_j} = e_j \left(\frac{\partial H}{\partial u}\right)_{tj}$$

III - 2 EXTENSIONS DES METHODES PRECEDENTES

III.2.1 Cas d'un critère intégral

Les résultats précédents ne sont valables que pour un critère terminal cependant un critère intégral se ramène sans difficulté à ce cas :

soit le système

$$\begin{cases} \frac{dX}{dt} = F(X, U, t) \\ X_0 \text{ donné} \end{cases}$$

et soit le critère

$$\Omega = \int_0^T q\ [\ X, U, t\ ]\ dt$$

$(x_1, \ldots, x_n)$ étant le vecteur d'état du système, introduisons la variable supplémentaire x_{n+1} définie par

$$\begin{cases} \frac{d\,x_{n+1}}{dt} = q\ [\ X, U, t\ ] \\ x_{n+1}\ (0) = 0 \end{cases}$$

on a $\int_0^T \frac{dx_{n+1}}{dt}\ dt = \int_0^T q\ [\ X, U, t\ ]\ dt = x_{n+1}\ (T) = \Omega$

et nous sommes ramenés au cas d'un critère terminal.

III.2.2 Cas d'un problème avec contraintes

Dans un problème d'optimisation avec contraintes, nous avons besoin des dérivées partielles des contraintes par rapport aux différentes variables. Si nous voulons appliquer la méthode précédente pour calculer ces dérivées, il nous faut supposer que ces contraintes sont terminales ou peuvent se ramener à ce cas comme les contraintes intégrales.

A partir de cette hypothèse, les méthodes d'optimisation développées s'appliquent sans difficulté.

Nous verrons d'ailleurs que les seules difficultés pratiques se trouvent dans la résolution du système différentiel et dans le calcul de l'intégrale donnant le gradient de Ω ou des contraintes.

Caractérisation de l'optimum

Pour que $\tilde{u}\ (t)$ corresponde à un optimum (en fait il s'agit d'un point stationnaire), il faut que $\delta\Omega = 0$ pour toute variation δu compatible avec les conditions imposées à la fonction de commande, donc :

$$\int_0^T \frac{\partial H}{\partial u}\ [\ \tilde{X},\ \tilde{U},\ \tilde{P},\ t\ ]\ \delta\, u\ (t)\ dt = 0$$

Si nous n'imposons aucune condition aux fonctions de commande, cette relation devient

$$\frac{\partial H}{\partial u}\ [\ \tilde{X},\ \tilde{P},\ \tilde{U},\ t\ ] = 0$$

Dans les applications numériques ou $u(t) = u(a_1, \ldots, a_s)$ ou $u = u(u_1, \ldots, u_n)$ nous testerons les relations :

$$\begin{cases} \dfrac{\partial \Omega}{\partial a_i} = \displaystyle\int_0^T \dfrac{\partial H}{\partial u} \dfrac{\partial u}{\partial a_i}\, dt \\ i = 1, \ldots s \end{cases} \quad \text{ou} \quad \begin{cases} \dfrac{\partial \Omega}{\partial u_i} = \displaystyle\int_0^T \dfrac{\partial H}{\partial u} \dfrac{\partial u}{\partial u_i}\, dt \\ i = 1, \ldots, n \end{cases}$$

CHAPITRE IV

PROGRAMMES ET NOTICES D'UTILISATION

Les programmes présentés dans cette quatrième partie ont été écrits en FORTRAN IV de base en vue de leur utilisation sur un ordinateur MITRA 15 de la C.I.I.

L'ordinateur utilisé a une mémoire centrale de 32 K octets et possède un disque de 2,5 millions d'octets. Cette faible capacité nous a obligé à faire un découpage en de nombreux sous-programmes, méthode indispensable aussi bien au niveau de la compilation qu'au niveau de l'exécution.

Le découpage a été réalisé de façon à rendre les sous-programmes indépendants les uns des autres, chacun d'eux réalisant l'une des opérations décrites dans les chapitres précédents (calcul du gradient réduit, du gradient projeté, du vecteur X^* de l'accélération de convergence, etc...) et une de ces opérations seulement.

Cette technique permet un large choix quand à l'utilisation de cette bibliothèque, un inconvénient est l'augmentation du temps de calcul en cas d'exécution sur un ordinateur de plus forte capacité de mémoire centrale pour lequel un recouvrement des divers sous-programmes n'est pas obligatoire.

Cependant, la transformation des divers éléments en un programme unique ne se conçoit que dans le cas d'un problème particulier et une modification des programmes généraux donnés devra être faite afin de mieux les adapter au problème posé.

IV - 1 ELIMINATION NUMERIQUE

Le sous-programme CALVA qui exécute cette élimination numérique calcule les variables de base par rapport aux variables indépendantes en résolvant le système des contraintes $\begin{cases} \varphi_i (X_B, X_H) = 0 \\ i \in \{L \cup J\} \end{cases}$

La technique de résolution utilisée est la méthode de Newton qui consiste à construire la suite d'itérés $X_B^1, \ldots, X_B^r, X_B^{r+1}$ définie par

$$X_B^{r+1} = X_B^r - H^r$$

ou H^r est calculé par $[D\Phi / DX] . [H^r] = [\Phi]$.

Expression dans laquelle $[D\Phi / DX]$ est la matrice carrée (p, p) dont les éléments sont les valeurs des dérivées $\frac{\partial \varphi_i}{\partial x_b}$ calculées au point X_B^r et $[\Phi]$ est la matrice colonne constituée par les valeurs des contraintes bilatérales ou unilatérales saturées en X^r.

Ce système linéaire est résolu par le sous-programme de bibliothèque MCADIR. On arrête les itérations lorsque le système des contraintes est vérifié à ε près. ε' étant un paramètre du programme. Une sécurité limite le nombre d'itérations. En cas d'incident, le contrôle est redonné au programme appelant avec positionnement d'un indicateur.

IV - 2 CAS DE CONTRAINTES BILATERALES - METHODE DU GRADIENT REDUIT AVEC DETERMINATION AUTOMATIQUE DU PAS ET ACCELERATION DE CONVERGENCE

a) Description de l'algorithme

Soit à maximiser $\Omega = f(X)$

avec les contraintes $\varphi_i = 0 \qquad i \in L$.

Donnons nous un vecteur X^r compatible avec les contraintes, le problème est de trouver X^{r+1} meilleur que X^r.

Nous commençons par calculer le gradient simple de la fonction en X^r. Puis, nous choisissons p variables de base. Nous calculons alors le gradient réduit de la fonction f, c'est-à-dire le gradient de $F(X_L)$ considérée comme fonction des seules variables libres.

Sur la direction ainsi trouvée, nous cherchons un vecteur Y admissible meilleur que X^r. Pour cela, le premier pas h essayé est le pas effectivement utilisé à l'itération précédente (seul le pas de la première itération sera arbitraire).

S'il est impossible d'améliorer X^r, nous revenons au programme appelant avec positionnement d'un indicateur d'incident, sinon, avec un pas ℓ nous avons obtenu Y, au pas 2ℓ correspond un vecteur $\mathfrak{Z}_1$ qui après élimination numérique nous donne en général un vecteur $\mathfrak{Z}$ (sinon nous prenons $X^{r+1} = Y$).

Si la condition $f(X^r) - 2 f(Y) + f(\mathfrak{Z}) < 0$ est vérifiée, nous construisons alors le vecteur U 1 correspondant au pas

$$h^* = \frac{\ell}{2} \; \frac{-3 f(X^r) + 4 f(Y) - f(\mathfrak{Z})}{f(X^r) - 2 f(Y) + f(\mathfrak{Z})}$$

U 1, après élimination numérique donne en général un vecteur U.

Si cette élimination numérique avorte ou bien si $f(X^r) - 2f(Y) + f(\mathfrak{Z}) \geq 0$, nous prendrons pour X^{r+1} le meilleur des deux tableaux Y et $\mathfrak{Z}$, sinon, nous prendrons pour X^{r+1} le meilleur des trois tableaux Y, $\mathfrak{Z}$ et U.

Si nous voulons utiliser la méthode d'accélération de convergence, soit en permanence, soit à partir d'un moment choisi par l'utilisateur par un test sur clé, soit encore par un test du programme (par exemple en vérifiant que le pas choisi à chaque itération est bien le pas h_i^* test assez sévère), il suffit après n + 1 itérations pendant lesquelles nous gardons en mémoire les vecteurs itérés ainsi que les pas successifs utilisés de construire le vecteur

$$X1^* = \frac{\sum_{i=0}^{n} \alpha_i h_i X^i}{\sum_{i=0}^{n} \alpha_i h_i}$$

vecteur qui après élimination numérique donne en géral un vecteur X^*. Il suffit ensuite de vérifier que X^* améliore bien X^r.

On peut remarquer que la méthode d'accélération de convergence demande simplement la résolution d'un système linéaire (n-p, n-p).

Il n'y a aucune raison pour que cette méthode soit efficace "loin" de l'optimum, mais comme nous ne savons pas préciser le voisinage de X^* dans lequel cette technique fonctionne, il semble bon de déclencher le calcul dans tous les cas. Le temps de calcul ainsi perdu semble largement

compensé par le gain que l'on obtient dès que cette méthode est efficace.

Le programme s'arrête lorsque le gradient réduit devient nul, ce qui revient à tester la condition

$$\max_i \left| \frac{\partial F}{\partial x_{b_i}} \right| < \varepsilon$$ ou ε est un paramètre du programme.

b) Notice d'utilisation des programmes

La mise en œuvre pratique de l'algorithme précédent demande l'utilisation de 10 sous-programmes dont 4 sous-programmes d'introduction de données qui doivent dont être réécrits dans chaque cas.

Les programmes ont été écrits pour un problème ayant au maximum 3 contraintes et portant sur 15 variables et l'écriture a été faite en double précision. Ces limites semblent difficiles à dépasser sur le MITRA 15 utilisé.

La double précision est presque indispensable si l'on veut que la technique d'accélération de convergence donne des résultats acceptables. Nous reviendrons sur les problèmes de précision des calculs lors de la discussion des résultats.

Les six sous-programmes de base sont :

- CALVA qui exécute l'élimination numérique par la méthode de Newton
- MCADIR qui résoud un système linéaire au sens des moindres carrés
- GREDUI qui calcule le gradient réduit à partir du gradient simple

- XREDUI qui cherche sur la direction de montée du gradient réduit un vecteur X^{r+1} meilleur que le précédent X^r et qui met en œuvre la méthode de détermination automatique du pas.

- CALXET qui sont les deux sous-programmes nécessaires à l'accélération de convergence. CALXET calcule le vecteur X^* et XETM exécute l'élimination numérique avant de tester s'il y a amélioration du dernier itéré.

Les conditions d'appel, les variables et les différents sous-programmes utilisés par chacun de ces sous-programmes sont donnés lors du listing général.

En plus du programme principal, nous devons réécrire pour chaque problème les 4 sous-programmes d'introduction de données F, GRADF, COSA, DEPCO. Ces écritures devront se présenter sous la forme suivante :

- F - est une fonction simple calculant la fonction objectif.

```
      DOUBLE PRECISION FUNCTION F(X)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION X(1)
        }
        } calcul de F(X)
        }
      RETURN
      END
```

X est le point où nous calculons la fonction

- GRADF - est le sous-programme qui calcule le gradient simple de la fonction objectif


```
      SUBROUTINE GRADF(X,C,N)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION X(1),C(1)
        }
        } C(I) = ∂f/∂x_i
        }
      RETURN
      END
```

X : point ou nous calculons le gradient

C : tableau dans lequel nous rangeons les composantes du gradient de f

N : entier dont le rôle n'apparait qu'en contrôle optimal.

- COSA - permet de calculer les contraintes en X.

```
      SUBROUTINE COSA(X,C)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION X(1),C(1)
        }
      C(I) = expression de la i ème contrainte
        }
      RETURN
      END
```

Le tableau C contient les valeurs des contraintes calculées en X.

- DEPCO - Ce sous-programme calcule les dérivées des contraintes par rapport aux diverses variables.

```
      SUBROUTINE DEPCO(X,C)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION X(1),C(3,15)
      DO 1 I = 1,15
      C(1,I) = 0.
      C(2,I) = 0.
    1 C(3,I) = 0.
        }
        } calcul de C(I,J)
        }
      RETURN
      END
```

X est le point où le calcul est demandé. Les dérivées sont rangées dans le tableau dans lequel $C(I, J) = \frac{\partial \varphi_i}{\partial x_j}$ où I est l'indice de la contrainte et J celui de la variable.

Remarquons que le programme s'applique à des problèmes sans contrainte où les deux sous-programmes précédents devront quand même être présents.

Le programme principal devra introduire les paramètres suivants :

NPLUSP : nombre total de variables

NC : nombre de contraintes

EPSCON : indique la précision de l'élimination numérique. Le système des contraintes devra être vérifié à EPSCON près, c'est-à-dire qu'une contrainte sera vérifiée si $|\varphi| <$ EPSCON.

La valeur à donner à ce paramètre dépend bien sûr de la difficulté du système à résoudre, il vaut mieux commencer par une trop grande précision (un indicateur nous signalera l'incident si la résolution devient impossible) que de tolérer des erreurs trop importantes qui risquent de nous dévier sensiblement de l'optimum. Il est possible d'utiliser une précision variable, la plus forte précision étant demandée lorsque l'accélération de convergence est efficace.

ITERMA : nombre maximum d'itérations de la méthode de Newton.

EPS : valeur minimum du pas de la méthode du gradient.

NOMAX : nombre maximum d'itérations pour le programme.

PASIN : valeur du pas de la méthode du gradient à la première itération.

IVL : tableau entier indiquant la nature des dérivées variables

$$\begin{cases} IVL(I) = 0 & \text{si X(I) est indépendante} \\ IVL(I) = 1 & \text{si X(I) est une variable de base.} \end{cases}$$

Il est évident que ce tableau pourrait être construit par le programme et modifié automatiquement en cas d'incident. Mais l'importance du choix des variables à prendre dans la base est telle qu'il est préférable de pouvoir les introduire comme paramètres.

X : tableau contenant en début de programme le vecteur initial. Ce vecteur doit être compatible avec les contraintes.

XMAX : Ce tableau contient le dernier itéré obtenu et donc le maximum en cas de réussite de la méthode.

FMAX = F(XMAX) contient la dernière valeur de la fonction et donc l'optimum en fin d'exécution.

Ces divers paramètres sont rangés dans trois zones COMMON étiquetées, notées :

COMMON/A/C(15), D(15), X(15), AL(15), XMAX(15), IVL(15)

COMMON/B/TE, PASIN, EPSCON, EPS, FMAX

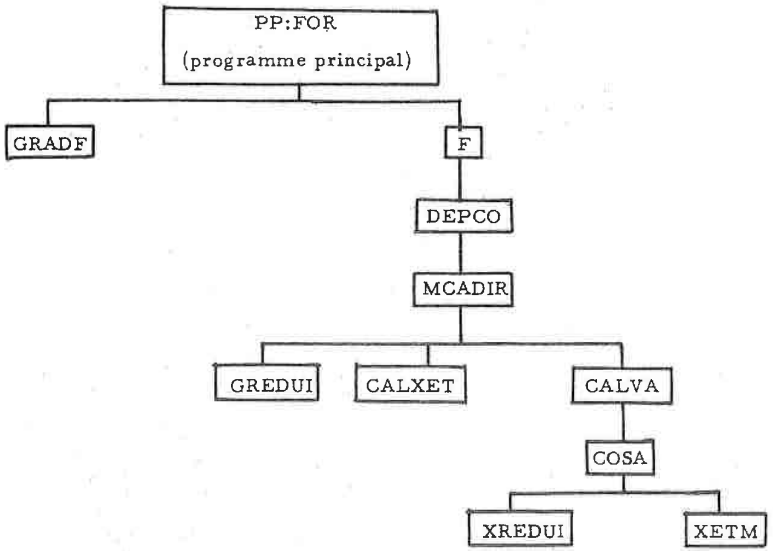
COMMON/C/INEW, NPLUSP, NC

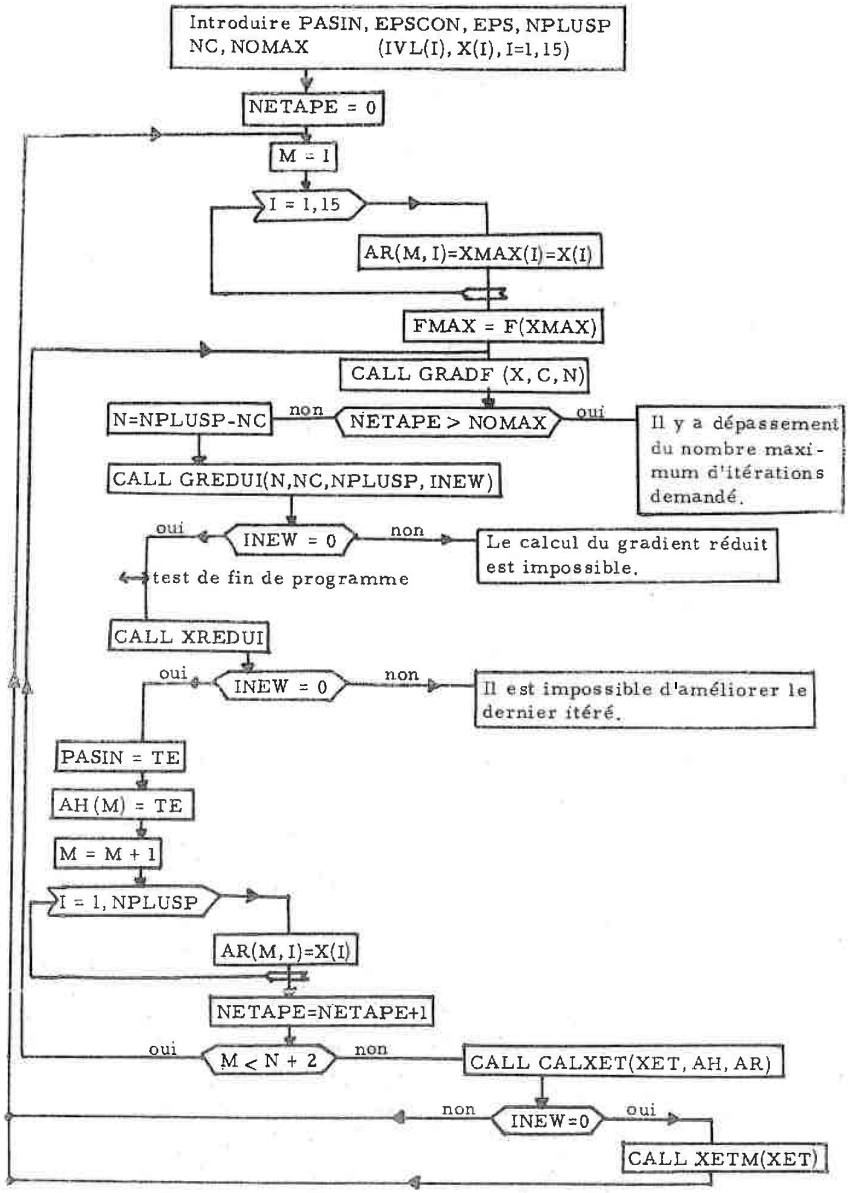
Les variables présentes dans ces COMMON mais non définies précédemment sont des paramètres dont le rôle change d'un sous-programme à l'autre.

Remarque. L'exécution de ce programme sur le MITRA 15 dont nous disposons impose le recours à une technique de recouvrement.

Selon le problème posé, il est possible de modifier l'ordre, nous donnons ici un exemple correspondant au cas figurant dans les listings joints.

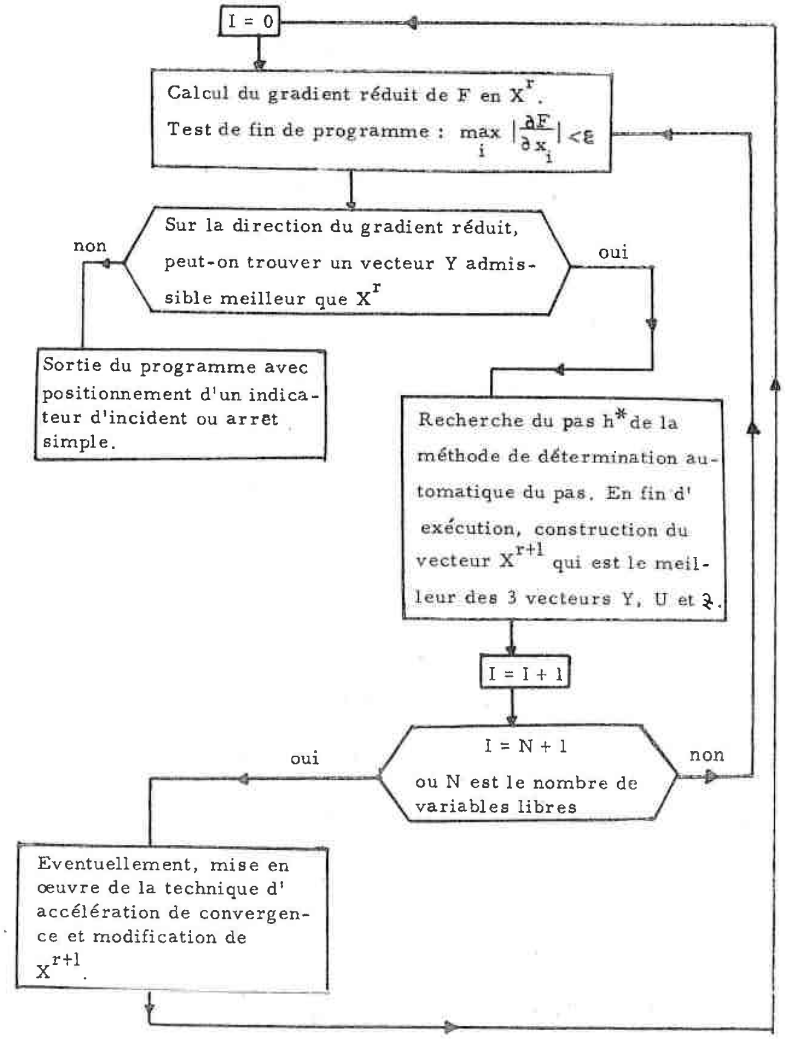
Dans cet exemple, l'arbre était :





Méthode du gradient réduit

Exemple de construction du programme principal : optimisation avec accélération de convergence.



Méthode du gradient réduit - Diagramme fonctionnel

IV - 3 CAS DE CONTRAINTES MIXTES - METHODE DU GRADIENT PROJETE

a) Description de l'algorithme

Soit à maximiser $\Omega = f(X)$

avec les contraintes $\begin{cases} \varphi_i = 0 & i \in L \\ \varphi_j \geq 0 & j \in J \end{cases}$

Prenons un vecteur X^r compatible avec les contraintes et soit K l'ensemble des indices des p contraintes bilatérales ou unilatérales saturées en X^r.

Nous commençons par calculer le gradient projeté de la fonction en X^r, ce qui, sauf incident nous donne une direction de montée D.

Deux cas peuvent se présenter :

- ou bien certains coefficients de Kuhn et Tucker sont strictements positifs, dans ce cas nous libérons la contrainte unilatérale correspondant au plus fort de ces coefficients et nous calculons la nouvelle direction de montée (notée D' dans la partie théorique),

- ou bien tous les coefficients de Kuhn et Tucker sont négatifs ou nuls et nous gardons la direction de montée D.

Sur la direction de montée obtenue, nous cherchons un vecteur X^{r+1} meilleur que X^r. En cas de succès (sinon nous quittons le programme avec positionnement d'un indicateur d'incident), deux cas peuvent se présenter :

- ou bien cette direction ne nous fait pas rencontrer de nouvelles contraintes et nous enchainons les itérations,

- ou bien cette direction nous fait quitter le domaine des contraintes. Dans ce cas, nous saturons la première contrainte rencontrée sur cette direction et nous enchaînons les itérations en prenant pour X^{r+1} le point ainsi obtenu.

Il y a arrêt du programme lorsque le gradient devient inférieur à un ε donné avec tous les coefficients de Kuhn et Tucker négatifs ou nuls. Deux méthodes ont été essayées pour saturer la nouvelle contrainte rencontrée dans le cas où le gradient projeté nous fait quitter le domaine des contraintes.

La première contrainte consiste à utiliser la technique d'élimination numérique en ajoutant simplement la nouvelle contrainte au système d'équations utilisé pour X^r. Le point initial de la méthode étant le vecteur X^r lui-même.

X^r étant solution de $\begin{cases} \varphi_i = 0 \\ i \in K \end{cases}$, soit i_1 l'indice de la nouvelle contrainte rencontrée, $K_1 = K \cup i_1$, X^{r+1} sera solution de $\begin{cases} \varphi_i = 0 \\ i \in K_1 \end{cases}$

En utilisant cette méthode, nous n'avons aucun sous-programme supplémentaire à écrire et surtout, le calcul est très rapide. Cependant, le point initial étant arbitraire, il est possible que le système des contraintes ne puisse pas être résolu de plus, même en cas de succès, cette

résolution risque de nous faire quitter sensiblement la direction de montée.

Dans certains cas où l'on est certain que la résolution du système des contraintes sera possible et si l'on accepte éventuellement de ne rien gagner lorsque le domaine des contraintes varie, on peut appliquer cette méthode.

Mais, dans le cas général, nous avons préféré programmer une méthode de dichotomie qui est plus sure mais qui nécessite beaucoup plus de calculs.

h_1 et h_2 étant deux pas encadrant le pas h qui sature la contrainte considérée (c'est-à-dire que $X^r + h_1 D$ sort du domaine admissible et $X^r + h_2 D$ vérifie toutes les contraintes de X^r), la méthode consiste à prendre $h' = \frac{h_1 + h_2}{2}$ et :

- si $X^r + h' D$ sont du domaine des contraintes $h_1 = h'$
- si $X^r + h' D$ vérifie les contraintes $h_2 = h'$

on arrête les itérations lorsque la nouvelle contrainte est vérifiée à ε près.

Nous avons considéré que la méthode du gradient projeté n'est qu'une technique de dépannage très utile dans le cas ou nous rencontrons de nouvelles contraintes mais qu'il est préférable de ne pas utiliser systématiquement car il est impossible de lui adjoindre la technique d'accélération de convergence et nous savons que le gradient simple converge lentement au voisinage de l'optimum.

Pour ces raisons, cette méthode est plus efficace si elle est utilisée en liaison avec le technique du gradient réduit comme nous le verrons par la suite.

Cependant, il est concevable de l'utiliser seule, par exemple si l'on veut une valeur approchée du maximum sans beaucoup de précision sur le vecteur X ou bien si l'on veut déterminer les contraintes utiles au problème, c'est-à-dire les contraintes qui sont saturées à l'optimum.

b) Notice d'utilisation des programmes

La mise en œuvre pratique de la méthode demande l'utilisation de 12 sous-programmes dont 7 sous-programmes de bibliothèque et 5 sous-programmes d'introduction de données qui devront donc être réécrits à chaque utilisation.

Les sous-programmes de bibliothèque sont :

- TAB - qui construit les tableaux IVL et ILU et repère les contraintes utiles.

- CORBAS - qui permet de modifier la base en y mettant des variables qui doivent nécessairement y être, c'est-à-dire celles qui interviennent seules dans certaines contraintes.

- NCADIR - sous-programme analogue à MCADIR, mais écrit en simple précision.

- GPROJ - calcule le gradient projeté de la fonction f et les coefficients de Kuhn et Tucker. C'est également ce programme qui éventuellement libère une contrainte et calcule la nouvelle direction de montée D'.

- XPROJ - ce programme recherche sur la direction donnée par GPROJ un vecteur X^{r+1} meilleur que l'itéré précédent X^r. C'est ce sous-programme qui sature s'il y a lieu une nouvelle contrainte par appel de DICHØ.

- DICHO - Sous-programme de dichotomie chargé de saturer la première contrainte rencontrée sous la direction de montée donnée par GPROJ.

- CALVAS - sous-programme d'élimination numérique exécutant le calcul des variables liées en fonction des variables libres.

Comme dans le cas du gradient réduit, cette bibliothèque a été conçue pour au maximum 3 contraintes et 15 variables. Les calculs sont effectués en simple précision.

En plus du programme principal, il faudra réécrire à chaque utilisation 5 sous-programmes d'introduction de données INI, F, GRADF, COSA, DEPCO.

- INI - sous-programme d'introduction de données qui permet de rentrer dans le programme les paramètres suivants :

- NA : paramètre de dimensionnement de tableaux. Ici NA = 15. Si l'on veut modifier ce paramètre, il faut réécrire toutes les cartes COMMON et DIMENSION dans lesquelles se trouvent des tableaux (15), (3,15) ou (15,15)

- NCB : nombre de contraintes bilatérales

- NPLUSP : nombre de variables

- EPS : valeur minimum tolérée du pas h de la méthode du gradient

- EPSCON : précision de l'élimination numérique. Une contrainte φ est vérifiée si $|\varphi| < EPSCON$

- ITERMA : nombre maximum d'itérations demandées dans la méthode de Newton

- KI : nombre total de contraintes

- PASIN : valeur initiale du pas de la méthode du gradient

- IVX : tableau entier défini par

IVX(I) = J si la $i^{ème}$ contrainte ne dépend que de x_j

IVX(I) = 0 si la $i^{ème}$ contrainte dépend de plusieurs variables

- X : valeur du vecteur initial.

Ces différents paramètres sont rangés dans 4 COMMON étiquetés ce qui fait que le programme INI a la structure suivante :

```
SUBROUTINE INI
COMMON/A/C(15),D(15), X(15), AL(15), XMAX(15), IVL(15)
COMMON/B/TE, PASIN, EPSCON, EPS, FMAX
COMMON/C/INEW, NPLUSP, NC
COMMON/D/ILU(15), IVX(15), IB(15), NA, KI, ICKT, NCB, ITERMA.
    ≀
    ≀  Introduction des paramètres précédents
    ≀
RETURN
END
```

Les variables placées dans les zones COMMON mais non définies dans la liste précédente sont locales aux divers sous-programmes et changent de signification en cours de calcul.

Les 4 autres sous-programmes à réécrire à chaque utilisation ont le même rôle et portent le même nom (F, GRADF, COSA, DEPCO) que certains sous-programmes de la méthode du gradient réduit. Il y a cependant des différences dans l'écriture et ceci est dû au fait qu'ici nous repérons les contraintes unilatérales non saturées et les variables libres. Cette forme est donc plus générale que celle décrite dans la méthode du gradient réduit (où toutes les contraintes étaient bilatérales). En conséquence, lorsque nous exécuterons un programme qui utilise les deux méthodes, il faudra

modifier les appels des sous-programmes donnés pour le gradient réduit afin de les rendre compatibles avec ce qui suit :

- F - Sous-programme d'introduction de la fonction objectif

```
FUNCTION F(X)
DIMENSION X(1)
    ≀
    ≀  calcul de F(X)
    ≀
RETURN
END
```

- GRADF - Ce sous-programme calcule le gradient simple de la fonction objectif. Le calcul n'est demandé que pour les variables x_i telles que IB(I)=1.

```
      SUBROUTINE GRADF(X, IB, C)
      DIMENSION X(1), IB(1), C(1)
      IF(IB(1). NE. 1)GO TO 1
      C (1) = ∂F/∂x1
        ≀
        ≀
    1 RETURN
      END
```

- COSA - Ce sous-programme permet de calculer les contraintes en X. Le calcul n'est demandé que pour les contraintes φ_i telles que IB(I)=1. S'il existe des contraintes bilatérales, elles doivent être placées en tête.

```
SUBROUTINE COSA(X, IB, C)
DIMENSION X(1), IB(1), C(1)
C(1) = expression de φ1
C(K) = expression de φK   } calcul des K contraintes bilatérales
IF(IB(K+1). NE. 1) GO TO 1
```

```
      C(K+1) = expression de φ_{K+1}
        }
        }
1     RETURN
      END
```

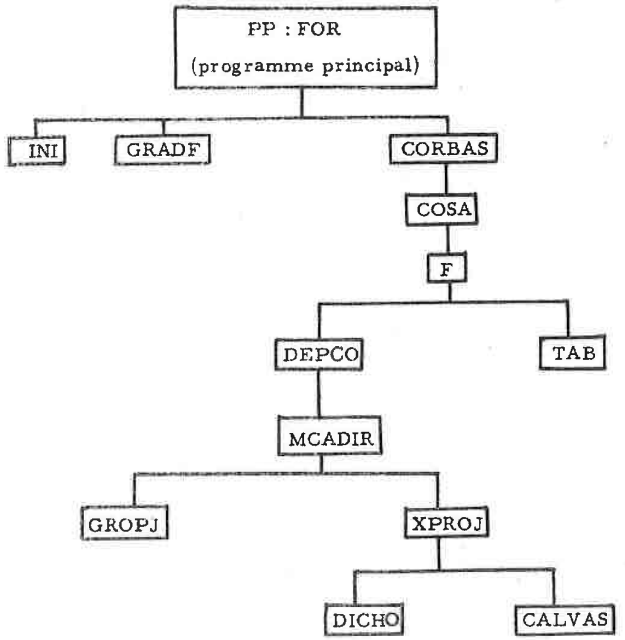
- DEPCO - Ce sous-programme calcule les dérivées des contraintes par rapport aux variables x_i. Le calcul n'est demandé que pour les contraintes φ_I telles que IB(I)=1. NA est un paramètre de dimensionnement de tableaux.

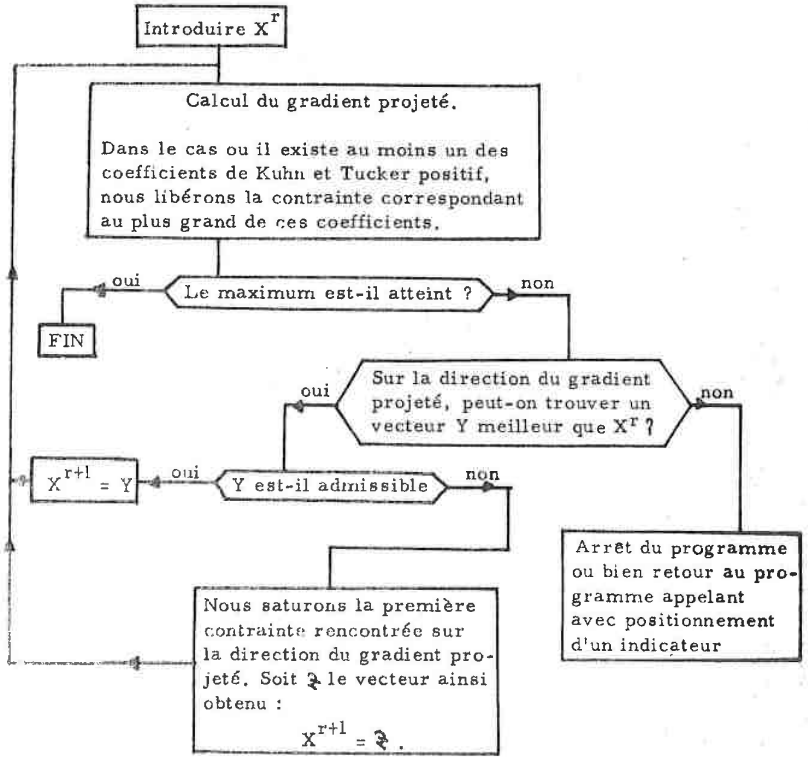
Dans le tableau C(I, J), I désigne la contrainte et J la variable

$$C(I, J) = \frac{\partial \varphi_I}{\partial x_J} .$$

```
      SUBROUTINE DEPCO (X, IB, NA, G)
      DIMENSION X(1), IB(1), G(NA, NA)
      DO 1 I=1, NA
      DO 1 J=1, NA
1     G(I, J)=0.
      IF(IB(1).NE.1) GO TO 2
        }
          calcul de C (1, I) soit ∂φ_1/∂x_i
        }
        }
2       }
      RETURN
      END
```

Remarque : Comme pour la méthode du gradient réduit, l'exécution de ces programmes sur MITRA 15 demande la mise en œuvre de recouvrements. Dans l'exemple que traite le listing donné, l'arbre de recouvrement était le suivant :





Méthode du gradient projeté - Diagramme fonctionnel

IV - 4 CAS D'UN PROBLEME DE CONTROLE OPTIMAL

Nous avons vu qu'un problème de contrôle optimal se ramène à un problème d'optimisation simple lorsque nous acceptons de remplacer la fonction de commande u (t) qui dépend d'un nombre infini de variables par la fonction u $(a_0, \ldots, a_p)$ ou u $(u_0, \ldots, u_s)$ qui dépend d'un nombre fini de paramètres.

Si de plus le problème est à critère terminal et les contraintes sont des contraintes terminales (ou peuvent s'y ramener), nous sommes capables de calculer strictement le gradient de la fonction et les dérivées des contraintes par rapport soit aux $a_o, \ldots, a_p$, soit aux $u_o, \ldots, u_s$ selon le mode de représentation choisi pour la fonction de commande u.

Du point de vue numérique, les seuls points nouveaux par rapport aux problèmes d'optimisation précédents se trouvent dans l'intégration des systèmes différentiels et dans le calcul des intégrales servant à calculer les diverses dérivées.

IV.4.1 Intégration d'un système différentiel par une méthode de Kutta-Runge explicite avec détermination du pas.

Le calcul d'un gradient, que ce soit celui de la fonction ou celui des contraintes, demande l'intégration du système direct et celle du système adjoint en des points identiques qui sont ceux du support d'intégration dans le calcul de l'intégrale donnant la dérivée cherchée.

La méthode d'intégration utilisée est une technique de Kutta-Runge explicite à adaptation contrôlée du pas développée par BOUTRIT (Thèse de 3ème cycle - Nancy 1975).

L'idée de la méthode est la suivante :

soit le système $\begin{cases} \frac{dX}{dt} = F[X, t] \\ X_0 \text{ donné,} \end{cases}$

h étant donné, une formule de Kutta-Runge d'ordre 4 permet d'écrire

$$X_{KR_4} = X(t_0 + h) = X(t_0) + R_1 K_1 + R_2 K_2 + R_3 K_3 + R_4 K_4$$

Les coefficients R et K dépendant de la formule utilisée.

Dans les mêmes conditions, c'est-à-dire avec h, t_0 et X_0 identiques, une formule de Kutta-Runge d'ordre 3 aurait donné

$$X_{KR_3} = X(t_0 + h) = X(t_0) + R'_1 K'_1 + R'_2 K'_2 + R'_3 K'_3$$

Introduisons comme norme d'un vecteur $\|v\| = \max_{i = 1, \ldots, n} |v_i|$

soit X^* la solution stricte du système en $t_0 + h$

nous savons que $\| X_{KR_4} - X^* \| = 0(h^5)$

$$\| X_{KR_3} - X^* \| = 0(h^4)$$

Nous définissons l'écart par :

$$e_p(h) = \| X_{KR_4} - X_{KR_3} \| = 0(h^4)$$

Nous pouvons supposer que $e_p(h)$ donne une bonne estimation de l'erreur commise par la méthode d'ordre 3, ce qui revient à assimiler X_{KR_4} avec X^*. Cet écart est accessible par le calcul.

D'où l'algorithme suivant : nous intégrons le système différentiel par la méthode d'ordre 4 puis nous calculons l'écart précédent, ce qui revient à intégrer une deuxième fois le système par une méthode d'ordre 3. Si cet écart est supérieur à une valeur maximum tolérée, nous diminuons le pas h et nous recommençons, sinon, nous acceptons le résultat. Cependant, si l'écart est trop faible, nous augmentons le pas à l'itération suivante.

Cette façon d'agir semble supprimer certaines instabilités constatées dans l'utilisation des méthodes de Kutta-Runge.

On voit qu'afin de réduire le temps de calcul, il est préférable d'utiliser des formules de Kutta-Runge d'ordre 3 et 4 ayant le maximum de coefficients communs, la technique programmée utilisait les formules suivantes :

$$\begin{cases} K_1 = h F(X_0, t_0) \\ K_2 = h F(X_0 + \frac{1}{2} K_1, t_0 + \frac{h}{2}) \\ K_3 = h F[X_0 - K_1 + 2 K_2, t_0 + h] \\ X_{KR_3} = X(t_0) + (K_1 + 4 K_2 + K_3)/6 \\ K_4 = h F(X_0 + \frac{1}{2} K_2, t_0 + \frac{h}{2}) \\ K_5 = h F(X_0 + K_4, t_0 + h) \\ X_{KR_4} = X_0 + (K_1 + 2 K_2 + 2 K_4 + K_5)/6 \\ e(h) = \frac{1}{6} |2 K_2 + K_3 - 2 K_4 - K_5| \end{cases}$$

Le choix de l'écart maximum toléré dépend évidemment du système à intégrer et de la nature du problème. Dans le cas où l'on s'intéresse aux valeurs des $(a_o, \ldots, a_p)$ ou $(u_o, \ldots, u_s)$ définissant la fonction de commande plus encore qu'à la valeur maximum, il est préférable de faire une étude préalable du système afin de déterminer le meilleur ε possible.

IV.4.2 Méthode d'intégration par arcs avec estimation de l'erreur de méthode

Pour les mêmes raisons, le calcul de l'intégrale demande certaines précautions. Nous avons utilisé une méthode d'intégration par arcs. En fin de calcul, nous faisons une estimation de l'erreur de méthode commise. Si cette erreur est trop forte, nous devons augmenter le nombre de points du support d'intégration (ce qui nous conduit à recommencer les intégrations des systèmes différentiels), si l'erreur est acceptable, nous enchaînons les calculs.

a) Intégration par arcs

Soit à calculer $I = \int_a^b f(x)\,dx$.

La méthode d'intégration par arcs consiste à partager le segment $[a, b]$ en n intervalles partiels égaux $[x_i, x_{i+1}]$ et à remplacer sur chacun de ces intervalles la fonction f (x) par le polynôme d'interpolation de degré 3 sur le support $[x_{i-1}, x_i, x_{i+1}, x_{i+2}]$.

Soit $\Delta_i = \int_{x_i}^{x_{i+1}} f(x)\,dx \qquad h = \frac{b-a}{n}$

Posons $x' = x - x_i$, le polynôme d'interpolation s'écrit :

$$P_3(x') = \left[\, 1 \;\; \frac{x'}{h} \;\; \frac{x'^2}{h^2} \;\; \frac{x'^3}{h^3} \,\right] \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1/3 & -1/2 & 1 & -1/6 \\ 1/2 & -1 & 1/2 & 0 \\ -1/6 & 1/2 & -1/2 & 1/6 \end{bmatrix} \begin{bmatrix} f_{i-1} \\ f_i \\ f_{i+1} \\ f_{i+2} \end{bmatrix}$$

d'ou

$$\Delta_i = \int_{x_i}^{x_{i+1}} P_3(x')\,dx' = \frac{h}{24}\left[-f_{i-1} + 13\,f_i + 13\,f_{i+1} - f_{i+2}\right]$$

Dans le cas ou f (x) est connue en dehors du segment $[a, b]$ la formule est applicable pour tous les segments Δ_i, sinon, on trouve des difficultés dans le calcul de Δ_o et Δ_{n-1}.

Dans les programmes, nous avons regroupé les deux segments extrêmes, c'est-à-dire que nous calculons I comme :

$$I = (\Delta_0 + \Delta_1) + \Delta_2 + \ldots + \Delta_{1-3} + (\Delta_{n-2} + \Delta_{n-1}).$$

Avec ce mode de calcul, nous obtenons :

$$I = h\left[\frac{8\,f_0 + 31\,f_1 + 20\,f_2 + 25\,f_3}{24} + \Sigma f_i + \frac{25\,f_{n-3} + 20\,f_{n-2} + 31\,f_{n-1} + 8\,f_n}{24}\right]$$

b) Estimation de l'erreur commise

Sur l'intervalle $[x_i, x_{i+1}]$, $\Delta_i = \int_{x_i}^{x_{i+1}} f(x)\,dx$ est calculée en

remplaçant f (x) par son polynôme d'interpolation de degré 3 sur le support $[x_{i-1}, x_i, x_{i+1}, x_{i+2}]$.

L'erreur d'interpolation est :

$$\varepsilon(t) = (t + h)\, t\, (t - h)\, (t - 2h)\, \frac{f^{(4)}(c(t))}{24}$$

L'erreur systématique sur un intervalle est donc

$$\mathcal{E}_i = \int_0^h \varepsilon(t)\, dt$$

$$= \int_0^h (t + h)\, t\, (t - h)\, (t - 2h)\, \frac{f^{(4)}(c(t))}{24}\, dt.$$

Or si $0 \leq t \leq h$, $\quad (t + h)\, t\, (t - h)\, (t - 2h) \geq 0$

Par application de la formule de la moyenne nous obtenons

$$\mathcal{E}_i = \frac{f^{(4)}(c_i)}{24} \int_0^h (t + h)\, t\, (t - h)\, (t - 2h)\, dt$$

soit $\quad \mathcal{E}_i = \frac{11\, h^5}{720}\, f^{(4)}(c_i)$

L'erreur systématique globale est donc :

$$\mathcal{E} = \frac{11\, h^5}{720} \sum_{i=0}^{n-1} f^{(4)}(c_i)$$

Mais dans notre cas, le calcul formel des dérivées successives de la fonction à intégrer est impossible. Aussi nous avons remplacé ce calcul par l'estimation suivante : nous interpolons la fonction à intégrer par un polynôme de degré 4 sur le support $[x_{i-1}, x_i, x_{i+1}, x_{i+2}, x_{i+3}]$ et nous assimilons la dérivée du polynôme à celle de la fonction.

Ce qui nous donne comme estimation des dérivées :

sur Δ_o : $\frac{1}{h^4} [f_{-1} + f_3 + 6 f_1 - 4 (f_0 + f_2)]$

sur Δ_i : $\frac{1}{h^4} [f_{i-1} + f_{i+3} + 6 f_{i+1} - 4 (f_i + f_{i+2})]$

sur Δ_{n-1} : $\frac{1}{h^4} [f_{n-2} + f_{n+2} + 6 f_n - 4 (f_{n-1} + f_{n+1})]$

Là encore, la fonction n'étant pas connue en dehors de l'intervalle d'intégration, nous avons des difficultés pour le calcul de la dérivée sur les intervalles extrêmes.

Plusieurs méthodes sont possibles, mais comme aucune méthode ne se dégage dans le cas général, nous avons simplement pris :

$$\mathcal{E} = 2\, \varepsilon_1 + \varepsilon_2 + \ldots + 3\, \varepsilon_{n-3}$$

ou ε_i représente l'erreur approchée sur chaque intervalle.

Remarques - Dans l'intégration des systèmes différentiels, le changement de pas est fréquent car, comme il est peu couteux d'une itération à la suivante, nous avons tout intérêt à calculer le pas optimum c'est-à-dire le plus grand possible compatible avec l'erreur maximum tolérée. Il est évident que pour l'intégration précédente, le calcul de l'erreur n'est considéré que comme une simple sécurité car une augmentation du nombre de points du support demande une nouvelle intégration des systèmes différentiels. Il s'agit donc moins de trouver le nombre de points optimum qu'un nombre de points acceptable.

On constate que les erreurs estimées des méthodes précédentes sont du même ordre de grandeur. Il parait difficile de dire dans le cas général laquelle "impose" un calcul superflu à l'autre. Cependant, si l'on est certain que le calcul de l'intégrale impose un pas h inférieur à

ce qu'il est nécessaire dans l'intégration des systèmes différentiels, on a tout intérêt à revenir à un calcul par une formule de Kutta-Runge simple. Dans l'autre sens, le gain est moins évident.

IV.4.3 Programmes

Un problème de contrôle optimal n'apporte aucune modification d'utilisation des programmes donnés par rapport à un problème d'optimisation.

Il y a simplement adjonction des sous-programmes qui mettent en œuvre les techniques précédentes, c'est-à-dire RESOL et CK pour l'intégration des systèmes différentiels et INTEG pour le calcul d'une intégrale simple.

Ces sous-programmes n'ont pas à être réécrits à chaque utilisation. On peut simplement signaler ici que le paramètre N qui apparaît dans

GRADF (X, C, N)

indique le nombre de points du support d'intégration lors du premier calcul du gradient.

Les listings qui sont joints correspondent à un exemple donné ci-après.

IV - 5 EXEMPLES D'UTILISATION DES PROGRAMMES

Ce premier exemple a pour objet de montrer les divers modes d'utilisation des programmes proposés.

Il s'agit de maximiser

$$\Omega = - x_1^2 - x_2^2 - 2 x_3^2 - x_4^2 + 5 x_1 + 5 x_2 + 21 x_3 - 7 x_4$$

avec les contraintes :

$$\begin{cases} \varphi_1 \equiv - x_1^2 - x_2^2 - x_3^2 - x_4^2 - x_1 + x_2 - x_3 + x_4 + 8 \geq 0 \\ \varphi_2 \equiv - x_1^2 - 2 x_2^2 - x_3^2 - 2 x_4^2 + x_1 + x_4 + 10 \geq 0 \\ \varphi_3 \equiv - 2 x_1^2 - x_2^2 - x_3^2 - 2 x_1 + x_2 + x_4 + 5 \geq 0 \end{cases}$$

La solution théorique est $\Omega = 44$ obtenue pour $X = (0, 1, 2, -1)$ qui sature les contraintes φ_1 et φ_3.

Le problème étant à contraintes mixtes, nous avons d'abord essayé la méthode du gradient projeté seule.

Les résultats donnés dans le tableau I sont satisfaisants, mais on constate qu'à partir de la 3ème itération les contraintes φ_1 et φ_3 sont saturées et ceci jusqu'à la fin du calcul.

D'où l'idée de considérer le problème comme un problème à contraintes bilatérales en prenant comme point de départ une vecteur compatible avec ces contraintes ;

$$\begin{cases} \max \Omega \\ \varphi_1 = 0 \\ \varphi_3 = 0 \end{cases}$$

Mais ici se pose le problème du choix de la base et les tableaux de résultat II et III montrent qu'avec deux bases différentes on obtient des résultats très différents en qualité. C'est cette importance d'un bon choix de la base

qui nous a fait entrer les variables de base en paramètres. L'utilisation doit pouvoir à tout moment modifier cette base au vu des résultats obtenus.

Enfin, nous avons groupé les deux méthodes du gradient projeté et du gradient réduit dans un même programme qui a été exécuté sur IRIS 80. Les résultats sont donnés dans le tableau IV et nous constatons que l'accélération de convergence nous amène en deux fois au résultat cherché.

Le problème de la précision nécessaire à l'accélération de convergence est mis en évidence par l'exemple suivant :

$$\begin{cases} \max \Omega = -(x_1^2 + x_2^2 + x_3^2 + x_4^2)\,[\,x_1^3 + 2\,x_2\,x_4 + x_3 + 1\,] \\ \qquad\qquad + 7\,x_1\,x_3\,x_4 - 8\,x_4 + x_3 \\ x_1^3 + 2\,x_2\,x_4 + x_3 = 0 \\ 7\,x_1\,x_3\,x_4 - 8\,x_4 + x_3 = 0. \end{cases}$$

Les calculs ont été effectués par la méthode du gradient réduit sur MITRA 15 puis sur IRIS 80.

Les résultats sont donnés dans les tableaux V et VI. Nous avons également donné dans ces tableaux les valeurs du gradient réduit.

Nous avons ensuite proposé deux exemples de contrôle optimal. Dans le premier, nous illustrons le gain apporté par l'utilisation de l'algorithme de recherche du gradient à la place d'une formule approchée.

Soit le système :

$$\begin{cases} \dfrac{dx_1}{dt} = -6\,x_1 + 5\,x_2 + 3\,x_3 + u \\ \dfrac{dx_2}{dt} = -8\,x_1 + 7\,x_2 + 4\,x_3 \qquad\qquad 1 \le t \le 2 \\ \dfrac{dx_3}{dt} = -2\,x_1 + x_2 + x_3 + 2\,u \end{cases}$$

$$\begin{cases} x_1(1) = 1 + 4\,e \\ x_2(1) = 6\,e \\ x_3(1) = 2 \end{cases}$$

nous avons pris $u = \dfrac{(X(1) - 1)^2 + (X(2) - 1)^2}{t}$

Le problème est de maximiser

$$\Omega = -(x_1(2) - 6\,e^2 - 1)^2 - (x_2(2) - 10\,e^2)^2 - (x_3(2) + 2\,e^2 - 2)^2$$

La solution théorique est obtenue pour $(1,1)$ $\Omega = 0$.

Partant de (18, 32), nous avons exécuté les calculs par la méthode du gradient réduit avec des formules approchées et la méthode stricte pour le calcul du gradient.

Les résultats du tableau VII correspondent à la formule d'approximation

$$\frac{\partial f}{\partial x} = \frac{f(x + h) - f(x - h)}{2h}$$

ceux du tableau VIII à la méthode stricte.

Même en prenant des formules à 3 et 5 points, nous ne sommes pas arrivés à obtenir la même rapidité de convergence que par la méthode stricte, mais le temps de calcul devenait beaucoup plus important.

L'exemple suivant montre le traitement d'un problème de contrôle optimal avec une contrainte intégrale.

Soit le système
$$\begin{cases} \frac{dx_1}{dt} = -6x_1 + 5x_2 + 3x_3 + u \\ \frac{dx_2}{dt} = -8x_1 + 7x_2 + 4x_3 + u - \frac{1}{t} \\ \frac{dx_3}{dt} = -2x_1 + x_2 + x_3 + 2u \end{cases}$$

$$1 \leq t \leq 2$$

$$\begin{cases} x_1(1) = 1 + 4e \\ x_2(1) = 6e \\ x_3(1) = 2 \end{cases}$$

Posons
$$\begin{cases} x_1^* = 1 + \log 2 + 6e^2 \\ x_2^* = 10\, e^2 \\ x_3^* = 2 + 2\log 2 - 2\, e^2 \end{cases}$$

Le problème est :
$$\begin{cases} \max \Omega = -(x_1 - x_1^*)^2 - (x_2 - x_2^*) - (x_3 - x_3^*)^2 \\ \text{avec} \quad \int_1^2 x_3(t)\, dt = -2,663975 \end{cases}$$

Nous prenons pour $u = x_1 t^2 + x_2 t + \frac{x_3}{t} + \frac{x_4}{t^2}$

nous nous ramenons à un problème à contrainte terminale en posant

$$\begin{cases} \frac{dx_4(t)}{dt} = x_3(t) + 2,663975 \\ x_4(1) = 0. \end{cases}$$

Le problème devient

$$\begin{cases} \max \Omega \\ \text{avec} \quad x_4(2) = 0. \end{cases}$$

Les résultats obtenus sont donnés dans le tableau IX.

Nombre d'itérations	Contraintes saturées	Variables utilisées	x_1	x_2	x_3	x_4	F
	010	100	0	0	0	-2	10
1	110	110	0,2953666	0,2803248	1,1773634	-1,8463981	34,180468
2	100	100	0,3561918	0,5806938	1,5813935	-1,4819783	40,6055620
3	101	110	0,3129674	0,7087672	1,7606938	-1,2845927	42,624820
4	101	110	0,2299923	0,7382767	1,8524206	-1,1960220	43,222975
5	101	110	0,1659973	0,7893081	1,9094551	-1,1331809	43,5806270
6	101	110	0,1176976	0,8405570	1,9445944	-1,0894758	43,7838320
7	101	110	0,0821235	0,8835568	1,9661081	-1,0596037	43,8925510
8	101	110	0,0565468	0,9166429	1,9792587	-1,0394651	43,947768
9	101	110	0,0385389	0,9410797	1,9873059	-1,0260212	43,9750370
10	101	110	0,0260651	0,9586725	1,9922417	-1,0171050	43,9881930
11	101	110	0,0175319	0,9711480	1,9952756	-1,0112190	43,9944440
12	101	110	0,0117447	0,9799181	1,9971406	-1,0073441	43,9974070
13	101	110	0,0078450	0,9860475	1,9982862	-1,0047990	43,9987780
14	101	110	0,0052293	0,9903157	1,9989872	-1,0031301	43,9994220
20	101	110	0,0004494	0,9989208	1,9999874	-1,0002309	43,999970
50	101	110	0,0000018	0,9999973	1,9999999	-1,0000008	43,9999820
100	101	110	0,0000011	1,0000020	+1,9999993	-1,0000008	43,9999820

Tableau I - Méthode du gradient projeté

Colonne des contraintes : | 1 indique une contrainte bilatérale ou unilatérale saturée
0 indique une contrainte unilatérale non saturée.

Colonne des variables utilisées : | 1 indique une variable de base
0 indique une variable libre.

	x_1	x_2	x_3	x_4	F
	0,0901575	0,8279479	1,9707980	-1,0618320	43,8209063
	0,0182702	1,1267779	1,9452253	-1,0360399	43,9161076
	0,0413913	1,0458851	1,9551903	-1,0430313	43,9675559
	0,0472218	1,0186462	1,9590508	-1,0442226	43,9729345
	0,0466333	1,0153623	1,9605016	-1,0432192	43,9744716
	0,0453595	1,0148652	1,9616522	-1,0420004	43,9758904
*	-0,0005145	0,9950138	2,0017842	-0,9988501	43,9998804
	-0,0017316	0,9998096	2,0012889	-0,9984901	43,9999690
	-0,0015771	0,9994324	2,0012866	-0,9985684	43,9999714
*	-0,0002766	0,9997850	2,0002589	-0,9997323	43,9999989

Tableau II - Méthode du gradient réduit

x_3 et x_4 sont les variables indépendantes

	x_1	x_2	x_3	x_4	F
	0,0901575	0,8279479	1,9707980	-1,0618320	43,8209063
	-0,0109786	0,9018005	2,0328195	-0,9779174	43,9543434
	0,0256523	0,9650643	1,9908600	-1,0175708	43,9901346
*	-0,0023230	1,0022908	2,0009995	-0,9983401	43,9999361
	0,0000531	1,0014337	1,9995518	-1,0002507	43,9999906
	-0,0003413	1,0003397	2,0001467	-0,9997560	43,9999986
*	-0,0000006	1,0000005	2,0000003	-0,9999996	44,0000000

Tableau III - Méthode du gradient réduit

x_2 et x_4 sont les variables indépendantes

Le signe * montre que l'accélération de convergence a été utilisée.

Méthode utilisée	x_1	x_2	x_3	x_4	F	Liaisons utilisées
	0	0	0	-2	10	0 1 0
Gradient projeté	0,15397013	0,3125	1,3125	-1,7917265	35,659977	1 1 0
Gradient projeté	0,27181082	0,63623024	1,8107119	-1,2453825	42,695830	1 0 0
Gradient projeté	0,090157534	0,82794788	1,9707984	-1,0618322	43,820912	1 0 1
Gradient projeté	0,39744742	0,91640300	1,9924377	-1,0241517	43,961116	1 0 1
Gradient réduit	-0,018583710	0,97635909	2,0195649	-0,98091627	43,993050	1 0 1
Gradient réduit	0,020890861	1,0000257	1,9847393	-1,0181297	43,995560	1 0 1
Gradient réduit	-0,019254870	0,9949332	2,0149070	-0,98295926	43,995981	1 0 1
Accélération de convergence	0,0021631806	0,99464863	1,9999725	-1,0010971	43,999855	1 0 1
Gradient réduit	-0,0034484308	1,0025450	2,0017258	-0,99741549	43,999874	1 0 1
Gradient réduit	0,0024057450	1,001240	1,9979222	-1,0022421	43,999927	1 0 1
Gradient réduit	-0,0024220223	0,99963135	2,0018308	-0,99787426	43,999938	1 0 1
Accélération de convergence	$-0,84960.10^{-5}$	1,0000001	2,0000006	-0,9999927	44,000000	1 0 1
Gradient réduit	$0,7439.10^{-5}$	1,0000002	1,9999941	-1,0000067	44,000000	1 0 1
Gradient réduit	$-0,70207.10^{-5}$	0,99999857	2,0000054	-0,99999378	44,000000	1 0 1

Tableau IV - Méthode mixte gradient projeté - gradient réduit.

x_1	x_2	x_3	x_4	g_1	g_2	F
1	-1	1	1	2, 714	-0, 714	-4
0,8152	-0, 9067	1,1839	0, 9516	-0, 4419	-0, 7323	-3, 7938
0,8457	-1, 006	1,1299	0,8622	0, 3286	-0,18299	-3, 7476
0,81906	-1, 007	1,1570	0,8471	-0, 06219	-0,10137	-3, 7419
0,82266	-1, 02089	1,150296	0,83605	-0, 040753	-0, 021838	-3, 74117
0,81834	-1, 02126	1,154748	0,83366	-0, 0223959	-0, 007510	-3, 741098
* 0,819446	-1, 022886	1,153226	0,832681	$0,2034.10^{-4}$	$0,1788.10^{-3}$	-3, 7410786
0,819448	-1, 022862	1,1532289	0,83270505	$-0,306.10^{-4}$	$0,11999.10^{-4}$	-3, 7410773
0,81945054	-1, 0228623	1,1532266	0,83270592	$-0,1202.10^{-2}$	$0,6992.10^{-5}$	-3, 7410773
0,81945055	-1, 0228622	1,1532267	0,83270605	$0,8196.10^{-6}$	$0,6084.10^{-5}$	-3, 7410773
0,81945055	-1, 0228622	1,1532267	0,83270605	$0.8174.10^{-6}$	$0,60776.10^{-5}$	-3, 7410773

* montre que l'accélération de convergence a été mise en œuvre avec amélioration du résultat.

Exécution sur MITRA 15

Tableau V

	x_1	x_2	x_3	x_4	g_1	g_2	F
	1	-1	1	1	2, 714	-0, 714	-4
	0, 8152	-0, 9067	1, 18387	0, 9516	-0, 4419	-0, 73226	-3, 7938
	0, 84575	-1, 006	1, 1299	0, 862246	0, 32864	-0, 182996	-3, 7476
	0, 8190618	-1, 007268	1, 1570905	0, 8471273	-0, 06219	-0, 10137	-3, 741934
	0, 8226619	-1, 0208998	1, 1502961	0, 83605225	0, 040753	-0, 021838	-3, 7411736
	0, 81834221	-1, 0212592	1, 1547484	0, 83366647	-0, 0223959	-0, 007510	-3, 741098
*	0, 81944588	-1, 0228863	1, 1532262	0, 83268134	$0, 2034.10^{-4}$	$0, 17876.10^{-3}$	-3, 7410786
	0, 81944845	-1, 0228622	1, 1532289	0, 83270505	$-0, 30577.10^{-4}$	$0, 120019.10^{-4}$	-3, 7410773
	0, 81945138	-1, 0228624	1, 1532257	0, 83270627	$0, 1391.10^{-4}$	$0, 49892.10^{-5}$	-3, 7410773
*	0, 81945071	-1, 0228613	1, 1532267	0, 83270692	$-0, 1130.10^{-7}$	$-0, 5630.10^{-7}$	-3, 7410773
	0, 81945071	-1, 0228613	1, 1532267	0, 83270692	$0, 62467.10^{-7}$	$-0, 19639.10^{-7}$	-3, 7410773

Le signe * indique que l'accélération de convergence a été mise en œuvre avec amélioration du résultat

Exécution sur IRIS 80

Tableau VI

	X1	X2	F
	18	32	$-0,3753539.10^{7}$
	3, 939601	6, 354584	$-0,334454.10^{4}$
	2, 910872	4, 480710	$-0,597182.10^{3}$
	2, 071973	2, 952646	-59, 14614
	1, 180750	1, 329255	$-0,4781327.10^{-1}$
	1, 117666	1, 214342	$-0,8587118.10^{-2}$
	1, 065996	1, 120220	$-0,8498144.10^{-3}$
	1, 035271	1, 064251	$-0,6933081.10^{-4}$
	1, 020142	1, 036693	$-0,7374948.10^{-5}$
	1, 011344	1, 020666	$-0,7420256.10^{-6}$
	1, 007582	1, 013813	$-0,1480854.10^{-6}$
	1, 004086	1, 007444	$-0,1248685.10^{-7}$
	1, 002029	1, 003696	$-0,758688.10^{-9}$
	1, 002015	1, 003672	$-0,739255.10^{-9}$
	1, 001372	1, 002501	$-0,1592909.10^{-9}$
	1, 000568	1, 001035	$-0,4665193.10^{-11}$
*	1, 000292	1, 000535	$-0,336225.10^{-12}$
	1, 000291	1, 000533	$-0,3306686.10^{-12}$
	1, 000289	1, 000530	$-0,3231372.10^{-12}$
	1, 000285	1, 000525	$-0,3079839.10^{-12}$
*	1, 000284	1, 000522	$-0,30486299.10^{-12}$

Tableau VII - Calcul du gradient par une formule approchée

X1	X2	F
18	32	$-0,37535392.10^{7}$
3,825296	6,152023	$-0,2863548.10^{4}$
1,315797	1,575864	-0,4469654
1,018851	1,034375	$-0,5675233.10^{-5}$
1,000801	1,001461	$-0,1852394.10^{-10}$
1,000051	1,000094	$-0,9359858.10^{-15}$

Tableau VIII

Calcul du gradient par la méthode stricte.

	x_1	x_2	x_3	x_4
	0,031004640	0,01	0,9	0,01
	-0,010687257	0,031738192	0,93480322	0,04505279
	-0,010687273	0,031739839	0,93480234	0,045050328
	-0,010692908	0,031746669	0,93480491	0,045049178
*	-0,010828933	0,03009433	0,93481317	0,044873353
	-0,010829074	0,032009474	0,93481330	0,044873522

	g_1	g_2	g_3	g_4
	0,062401	0,09990587	0,10062228	$-0,418038.10^{-2}$
	$0,236376.10^{-5}$	$-0,12697.10^{-5}$	$-0,35354.10^{-5}$	$-0,55158.10^{-9}$
	$0,409077.10^{-5}$	$0,1537719.10^{-5}$	$-0,68888.10^{-6}$	$-0,54789.10^{-9}$
	$-0,21975.10^{-6}$	$0,526456.10^{-5}$	$-0,749578.10^{-5}$	$-0,52307.10^{-9}$
*	$0,75639.10^{-7}$	$0,249688.10^{-6}$	$0,30889.10^{-6}$	$-0,65300110^{-11}$
	$-0,15919.10^{-6}$	$-0,128067.10^{-6}$	$-0,723675.10^{-7}$	$-0,564907.10^{-11}$

Le signe * montre que l'accélération de convergence a été mise en œuvre avec amélioration du résultat.

Exécution sur IRIS 80

Tableau IX - Exemple de contrôle optimal avec contrainte intégrale.

CONCLUSION

L'utilisation de techniques d'optimisation pour la résolution de problèmes de contrôle optimal ne pose pas de difficultés théoriques une fois donné le mode de représentation de la fonction de commande.

Dans un problème concret où nous avons une idée sur la forme de la fonction solution, il semble préférable d'utiliser toutes les fois où c'est possible une représentation formelle qui a l'avantage d'alléger les calculs.

Dans le cas où notre représentation de la solution est imprécise, il faudra utiliser une représentation par table. Ceci nous conduit généralement à un nombre élevé d'inconnues ce qui alourdit les calculs. De plus, si nous désirons des calculs précis ce qui sera le cas si l'on s'intéresse à la fonction de commande, nous risquons d'être gêné au niveau de l'accélération de convergence qui nécessite avant sa mise en œuvre n+1 itérations, n étant le nombre de variables libres.

En effet, avec un nombre important de paramètres nous sommes pris entre deux conditions contraires. Si nous démarrons nos itérations loin de l'optimum, il n'y a aucune raison pour que l'accélération de convergence soit efficace et si nous partons d'un point proche de l'optimum, si n est grand, après n+1 itérations aux résultats très voisins et avec un gradient très petit, l'influence des erreurs de chute risque de fausser la méthode et de rendre l'accélération de convergence inutile.

Nous ne pouvons que signaler cette difficulté sans apporter de solutions car il semble impossible de chiffrer les erreurs d'arrondis commises.

Il faudrait examiner ce que donnent dans ces cas difficiles d'autres méthodes d'accélération de convergence comme celle du gradient conjugué afin de mieux cerner les cas d'utilisations des méthodes exposées.

PROGRAMMES

```
C  ----------------
C  !CONTROLE OPTIMAL!
C  ----------------
C
C  CES SOUS-PROGRAMMES SONT UTILISABLES POUR UN PROBLEME DE CONTROLE
C  OPTIMAL.
C  LES SOUS-PROGRAMMES F,GRADF,COSA ET DEPCO DEPENDENT DU PROBLEME ET SONT
C  REECRITS A CHAQUE UTILISATION . ICI,NOUS AVONS UTILISE LE MODELE DONNE
C  DANS LA METHODE DU GRADIENT REDUIT.
C  LES PROGRAMMES RESOL,CK ET INTEG SONT UTILISABLES DANS TOUS LES CAS.
C
C
C
C  CE SOUS-PROGRAMME CALCULE LA VALEUR DE LA FONCTION F CONNAISSANT LE
C  VECTEUR DE COMMANDE U.
C  LE TABLEAU X CONTIENT LES ELEMENTS NOTES A(I) OU U(I) DANS LA PARTIE
C  THEORIQUE. CES ELEMENTS DEFINISSENT LA FONCTION DE COMMANDE ET SONT
C  LES VERITABLES INCONNUES DU PROBLEME.
C
      DOUBLE PRECISION FUNCTION F(X)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(1),Y(4),T(4),C(4,5),XT(4,100)
      COMMON/A1/ION
      DATA A1,A2,A3/46.027484D0,73.8905610D0,-11.391818D0/
      DATA EPS,IAB,NV,T0,T1/0.1D-06,1.D0,4,1.D0,2.D0/
      EXTERNAL SYS
       A=DEXP(1.D0)
      Y(1)=4.D0*A+1.D0
      Y(2)=6.D0*A
      Y(3)=2.D0
      Y(4)=0.D0
      CALL RESOL(EPS,T1,NV,TAB,T0,Y,SYS,X,C,XT)
      IF(ION.EQ.1)STOP 7
      F=-(XT(1,2)-A1)*(XT(1,2)-A1)-(XT(2,2)-A2)*(XT(2,2)-A2)
     1-(XT(3,2)-A3)*(XT(3,2)-A3)
      RETURN
      END
```

```
C
C     CE SOUS-PROGRAMME CALCULE LES DERIVEES DE LA FONCTION F PAR RAPPORT AUX
C     X(I) QUI DEFINISSENT LE VECTEUR DE COMMANDE U.
C     N EST LE NOMBRE INITIAL DE POINTS A UTILISER DANS LE CALCUL DES
C     INTEGRALES.
C
      SUBROUTINE GRADF(X,C,N)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON/A1/ION
      DIMENSION Y(4),X(1),C(1),TR(4,5),XT(4,100),XT1(4,100),W(100)
      EXTERNAL SYS,SYSAD
      DATA EPS1,NV,T0,T1,EPS/0.1D-04,4,1.D0,2.D0,0.1D-06/
      DATA A1,A2,A3/46.027484D0,73.89056D0,-11.391818D0/
      N=40
   10 TAB=1.D0/(N-1.D0)
      A=DEXP(1.D0)
      Y(1)=4.D0*A+1.D0
      Y(2)=6.D0*A
      Y(3)=2.D0
      Y(4)=0.D0
      CALL RESOL(EPS,T1,NV,TAB,T0,Y,SYS,X,TR,XT)
      IF(ION.EQ.1)STOP 1
      TAB=-TAB
      Y(1)=-2.D0*(XT(1,N)-A1)
      Y(2)=-2.D0*(XT(2,N)-A2)
      Y(3)=-2.D0*(XT(3,N)-A3)
      Y(4)=0.D0
      CALL RESOL(EPS,T0,NV,TAB,T1,Y,SYSAD,X,TR,XT1)
      IF(ION.EQ.1)STOP 2
      DO 1 I=1,N
      J=N+1-I
    1 W(I)=XT1(1,J)+XT1(2,J)+2.D0*XT1(3,J)
      DO 2 I=1,N
    2 XT(1,I)=W(I)
      DO 3 I=1,4
      T=1.D0
      H=1.D0/(N-1.D0)
      DO 4 J=1,N
      W(J)=XT(I,J)
      IF(I.EQ.1)GOTO10
      IF(I.EQ.2)GOTO20
      IF(I.EQ.3)GOTO30
      W(J)=W(J)/T/T
      GOTO4
   10 W(J)=W(J)*T*T
      GOTO4
   20 W(J)=W(J)*T
      GOTO4
   30 W(J)=W(J)/T
    4 T=T+H
      CALL INTEG(W,N,1.D0,2.D0,EPS1,II,CC)
      C(I)=CC
      IF(II.EQ.1.AND.N.LT.100)GOTO50
      GOTO3
   50 N=N+10
      GOTO100
    3 CONTINUE
      RETURN
      END
```



```
C
C     CE SOUS-PROGRAMME CALCULE LES VALEURS DES CONTRAINTES.
C     LE TABLEAU X CONTIENT LES PARAMETRES U(I) OU A(I) QUI DEFINISSENT LA
C     FONCTION DE COMMANDE U.
C
      SUBROUTINE COSA(X,C)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(1),C(1),O(4,5),XT(4,100),Y(4),W(100)
      EXTERNAL SYS
      COMMON/A1/ION
      DATA EPS,NV,T0,T1/0.1D-06,4,1.D0,2.D0/
      TAB=1.D0
      A=DEXP(1.D0)
      Y(1)=4.D0*A+1.D0
      Y(2)=6.D0*A
      Y(3)=2.D0
      Y(4)=0.D0
      CALL RESOL(EPS,T1,NV,TAB,T0,Y,SYS,X,O,XT)
      IF(ION.EQ.1)STOP 8
      C(1)=XT(4,2)
      RETURN
      END
```

```
C
C     CE SOUS-PROGRAMME CALCULE LES DERIVEES DES CONTRAINTES PAR RAPPORT AUX
C     X(I) QUI DEFINISSENT LE VECTEUR DE COMMANDE U.
C
      SUBROUTINE DEPCO(X,C)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON/A1/ION
      DIMENSION X(1),C(3,15),Y(4),TR(4,5),XT(4,100),XT1(4,100),W(100)
      EXTERNAL SYS,SYSAD
      DATA EPS1,NV,T0,T1,EPS/0.1D-04,4,1.D0,2.D0,0.1D-06/
      DO 5 I=1,3
      DO 5 J=1,15
    5 C(I,J)=0.D0
      N=40
  100 TAB=-1.D0/(N-1.D0)
      Y(1)=0.D0
      Y(2)=0.D0
      Y(3)=0.D0
      Y(4)=1.D0
      CALL RESOL(EPS,T0,NV,TAB,T1,Y,SYSAD,X,TR,XT1)
      IF(ION.EQ.1)STOP 2
      DO 1 I=1,N
      J=N+1-I
    1 W(I)=XT1(1,J)+XT1(2,J)+2.D0*XT1(3,J)
      DO 2 I=1,N
    2 XT(1,I)=W(I)
      DO 3 I=1,4
      T=1.D0
      H=1.D0/(N-1.D0)
      DO 4 J=1,N
      W(J)=XT(1,J)
      IF(I.EQ.1)GOTO10
      IF(I.EQ.2)GOTO20
      IF(I.EQ.3)GOTO30
      W(J)=W(J)/T/T
      GOTO4
   10 W(J)=W(J)*T*T
      GOTO4
   20 W(J)=W(J)*T
      GOTO4
   30 W(J)=W(J)/T
    4 T=T+H
      CALL INTEG(W,N,1.D0,2.D0,EPS1,II,CC)
      C(1,I)=CC
      IF(II.EQ.1.AND.N.LT.100)GOTO50
      GOTO3
   50 N=N+10
      GOTO100
    3 CONTINUE
      RETURN
      END
```



```
C
C     CE SOUS-PROGRAMME CALCULE LES VALEURS DU SYSTEME DIRECT AU TEMPS T.
C
C     PARAMETRES:
C     -----------
C     X:VALEUR DU TEMPS T OU NOUS DEMANDONS LE CALCUL.
C     Y:POINT OU NOUS DEMANDONS LE CALCUL.
C     NV:DIMENSION DU SYSTEME DIFFERENTIEL.
C     AA:TABLEAU CONTENANT LE VECTEUR X VERITABLE INCONNUE DU PROBLEME.
C     LES ELEMENTS DE CE TABLEAU SONT LES U(I) OU A(I) DE LA PARTIE THEORIQUE.
C     FI:TABLEAU CONTENANT LES VALEURS DU SYSTEME.
C     U:VALEUR DE LA FONCTION DE COMMANDE.
C
C     AUCUN SOUS-PROGRAMME N'EST UTILISE.
C
      SUBROUTINE SYS(X,Y,FI,NV,AA)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION Y(1),FI(1),AA(1)
      U=AA(1)*X*X+AA(2)*X+AA(3)/X+AA(4)/X/X
      FI(1)=-6.D0*Y(1)+5.D0*Y(2)+3.D0*Y(3)+U
      FI(2)=-8.D0*Y(1)+7.D0*Y(2)+4.D0*Y(3)+U-1.D0/X
      FI(3)=-2.D0*Y(1)+Y(2)+Y(3)+2.D0*U
      FI(4)=Y(3)+2.6639749D0
      RETURN
      END
```

```
CE SOUS-PROGRAMME CALCULE LES VALEURS DU SYSTEME ADJOINT AU TEMPS T.

PARAMETRES:
-----------
X:VALEUR DU TEMPS T OU NOUS DEMANDONS LE CALCUL.
Y:POINT OU NOUS DEMANDONS LE CALCUL.
NV:DIMENSION DU SYSTEME DIFFERENTIEL.
AA:TABLEAU CONTENANT LE VECTEUR X VERITABLE INCONNUE DU PROBLEME.
LES ELEMENTS DE CE TABLEAU SONT LES U(I) OU A(I) DE LA PARTIE THEORIQUE.
FI:TABLEAU CONTENANT LES VALEURS DU SYSTEME.

AUCUN SOUS-PROGRAMME N'EST UTILISE.
      SUBROUTINE SYSAD(X,Y,FI,NV,AA)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION Y(1),FI(1),AA(1)
      FI(1)=6.D0*Y(1)+8.D0*Y(2)+2.D0*Y(3)
      FI(2)=-(5.D0*Y(1)+7.D0*Y(2)+Y(3))
      FI(3)=-(3.D0*Y(1)+4.D0*Y(2)+Y(3)+Y(4))
      FI(4)=0.D0
      RETURN
      END
```



```
                    ------------------------------
                    !INTEG(F,N,A,H,EPS,INEW,CINTEG)!
                    ------------------------------

          DONNEES:F,N,A,H,EPS
          --------

          RESULTATS:CINTEG,INEW
          ---------


     CE SOUS-PROGRAMME CALCULE UNE INTEGRALE PAR LA METHODE D INTEGRATION
     PAR ARCS ET DONNE UNE ESTIMATION DE L ERREUR DE METHODE COMMISE.


     PARAMETRES:
     -----------

     F:TABLEAU DE N ELEMENTS CONTENANT LES VALEURS DE LA FONCTION AUX POINTS
       DU SUPPORT.
     N:NOMBRE DE POINTS DU SUPPORT D INTEGRATION.
     A,B:BORNES D INTEGRATION.
     EPS:PRECISION MINIMUN DEMANDEE.
     CINTEG:VALEUR DE L INTEGRALE.
     INEW:INDICATEUR D INCIDENT.
               *INEW=0   SI LA PRECISION EST SUPERIEURE A EPS.
               *INEW=1   SI LA PRECISION EST INFERIEURE A EPS.


     AUCUN SOUS-PROGRAMME N EST UTILISE.


     SUBROUTINE INTEG(F,N,A,H,EPS,INEW,CINTEG)
     IMPLICIT DOUBLE PRECISION (A-H,O-Z)
     DIMENSION F(1)
     H=(B-A)/(N-1.D0)
     X=(8.D0*F(1)+31.D0*F(2)+20.D0*F(3)+25.D0*F(4))/24.D0
     N1=N-4
     DO 1 I=5,N1
   1 X=X+F(I)
     X=X+(25.D0*F(N-3)+20.D0*F(N-2)+31.D0*F(N-1)+8.D0*F(N))/24.D0
     CINTEG=X*H
     E=(F(1)+F(5)-4.D0*(F(2)+F(4))+6.D0*F(3))*2.D0
     DO 2 I=1,N1
   2 E=E+F(I)+F(I+4)-4.D0*(F(I+1)+F(I+3))+6.D0*F(I+2)
     E1=(F(N-4)+F(N)-4.D0*(F(N-3)+F(N-1))+6.D0*F(N-2))*2.D0
     E=(E+E1)*11.D0*H/720.D0
     U=E/CINTEG*100.D0
     INEW=0
     IF(DABS(U).LE.EPS)RETURN
     INEW=1
     RETURN
     END
```

```
C     ------------------------------------------
      !RESOL(EPS,BSII,NV,TAB,T0,Y,SP,AA,C,RESUL)!
C     ------------------------------------------

C     DONNEES:EPS,BSII,NV,TAB,T0,Y,SP,AA,C,RESUL
C     --------

C     RESULTATS:RESUL
C     ----------


C     CE SOUS-PROGRAMME INTEGRE UN SYSTEME DIFFERENTIEL PAR UNE METHODE
C     DE KUTTA-RUNGE D ORDRE 4 AVEC CONTROLE DU PAS.

C     PARAMETRES:
C     -----------
C     EPS:ERREUR TOLEREE.
C     BSII:BORNE FINALE DU SUPPORT D INTEGRATION.
C     NV:DIMENSION DU SYSTEME DIFFERENTIEL.
C     TAB:SERT A TABULER LES RESULTATS QUI SONT RANGES DANS LE TABLEAU RESUL
C         POUR LES VALEURS DE T  ALLANT DE T0 A BSII DE TAB EN TAB.
C     T0:BORNE INITIALE DU SUPPORT D INTEGRATION.
C     Y:TABLEAU CONTENANT LES VALEURS INITIALES DU SYSTEME.
C     SP:SOUS-PROGRAMME QUI PERMET DE CHOISIR ENTRE L INTEGRATION DU SYSTEME
C        DIRECT OU CELLE DU SYSTEME ADJOINT.
C     AA:TABLEAU CONTENANT LE VECTEUR X VERITABLE INCONNUE DU PROBLEME.
C     C:TABLEAU DE TRAVAIL.
C     RESUL:TABLEAU DANS LEQUEL SONT RANGES LES RESULTATS.

C     SOUS-PROGRAMME UTILISE:
C     -----------------------
C     CK


      SUBROUTINE RESOL(EPS,BSII,NV,TAB,T0,Y,SP,AA,C,RESUL)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON/A1/ION
      DIMENSION C(NV,5),RESUL(NV,100),Y(1),AA(1),YA(5)
      EXTERNAL SP
      ION=0
      DO 1 I=1,NV
    1 RESUL(I,1)=Y(I)
      IO=2
      NPAS=0
      X=T0
      XTAB=X
      T=TAB
      XTAB=XTAB+TAB
    2 NPAS=NPAS+1
      IF(NPAS.GT.500)GOTO14
      CALL CK(X,Y,T,C,NV,SP,AA)
      XA=X+T
      YN=0.00
      DO 3 I=1,NV
    3 YN=YN+Y(I)*Y(I)
      YN=DSQRT(YN)
      A=0.1D-14
```

```
      IF(YN.LE.A)YN=A
      DO 4 I=1,NV
    4 YA(I)=Y(I)+(C(I,1)+2.D0*C(I,2)+2.D0*C(I,4)+C(I,5))/6.D0
      E=0.D0
      DO 5 I=1,NV
      E4=DABS(2.D0*C(I,2)+C(I,3)-2.D0*C(I,4)-C(I,5))/6.D0
      IF(E.LT.E4)E=E4
    5 CONTINUE
      E=E/YN
      A=0.1D-29
      IF(E.GT.A)GOTO6
      T=T*(1.1D0)
      GOTO2
    6 A=0.25D0
      T=T*(EPS/E)**A
      A=1.5D0
      IF(E-EPS*A)7,7,2
    7 X=XA
      DO 8 I=1,NV
    8 Y(I)=YA(I)
      IF(DABS(X-XTAB).GT.(0.0001D0))GOTO13
      NPAS=0
      DO 9 I=1,NV
      RESUL(I,IO)=Y(I)
    9 IO=IO+1
      XTAB=XTAB+TAB
      IF(X+T-XTAB)12,11,11
   11 T=XTAB-X
      IF(DABS(X-HSII).GT.(0.0001D0))GOTO2
      RETURN
   12 IF(T.GE.(0.D0))GOTO10
      IF(X+T-XTAB)11,11,12
   10 IO=1
      RETURN
      END
```



```
C
C
C                    ----------------------
C                    !CK(X,Y,T,C,NV,SP,AA)!
C                    ----------------------
C          DONNEES:X,Y,T,NV,SP,AA
C          --------
C          RESULTATS:C
C          ----------
C
C
C   CE SOUS-PROGRAMME CALCULE LES COEFFICIENTS DES FORMULES DE KUTTA-RUNGE
C   D ORDRE 3 ET 4 QUI DANS LE PROGRAMME PRINCIPAL SERVENT A CALCULER UN
C   ECART D ORDRE 4 ET A INTEGRER LE SYSTEME DIFFERENTIEL PAR DES
C   FORMULES DE KUTTA-RUNGE D ORDRE 4.
C
C   TOUS LES PARAMETRES UTILISES SONT INTERNES A RESOL.
C
C   SOUS-PROGRAMME UTILISE:
C   ----------------------
C   SP QUI PEUT ETRE SOIT SYS SOIT SYSAD SELON QUE RESOL INTEGRE LE
C      SYSTEME DIRECT OU LE SYSTEME ADJOINT.
C
C
      SUBROUTINE CK(X,Y,T,C,NV,SP,AA)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION Y(1),AA(1),FI(5),YA(5),C(NV,5)
      EXTERNAL SP
      CALL SP(X,Y,FI,NV,AA)
      DO 1 I=1,NV
    1 C(I,1)=FI(I)*T
      XA=X+T/2.D0
      DO 2 I=1,NV
    2 YA(I)=Y(I)+C(I,1)/2.D0
      CALL SP(XA,YA,FI,NV,AA)
      DO 3 I=1,NV
    3 C(I,2)=FI(I)*T
      DO 4 I=1,NV
    4 YA(I)=Y(I)-C(I,1)+2.D0*C(I,2)
      XA=X+T
      CALL SP(XA,YA,FI,NV,AA)
      DO 5 I=1,NV
    5 C(I,3)=FI(I)*T
      DO 6 I=1,NV
    6 YA(I)=Y(I)+C(I,2)/2.D0
      XA=X+T/2.D0
      CALL SP(XA,YA,FI,NV,AA)
      DO 7 I=1,NV
    7 C(I,4)=FI(I)*T
      XA=X+T
      DO 8 I=1,NV
    8 YA(I)=Y(I)+C(I,4)
      CALL SP(XA,YA,FI,NV,AA)
      DO 9 I=1,NV
    9 C(I,5)=FI(I)*T
      RETURN
      END
```

```
%JOB  ROCHER
%C/FORTD/
C
C
C                    ---------------------------------
C                    !TECHNIQUE DU GRADIENT PROJETE.!
C                    ---------------------------------
C
C
C
C
C     LA PRORLEME POSE EST LE SUIVANT:
C          MAXIMISER: F=-X(1)*X(1)-X(2)*X(2)-2*X(3)*X(3)-X(4)*X(4)+5X(1)+5X(2)
C                     +21X(3)-7X(4)
C
C     AVEC LES CONTRAINTES:
C     -X(1)*X(1)-X(2)*X(2)-X(3)*X(3)-X(4)*X(4)-X(1)+X(2)-X(3)+X(4)+8>=0
C     -X(1)*X(1)-2*X(2)*X(2)-X(3)*X(3)-2*X(4)*X(4)+X(1)+X(4)+10>=0
C     -2*X(1)*X(1)-X(2)*X(2)-X(3)*X(3)-2*X(1)+X(2)+X(4)+5>=0
C
C     POINT INITIAL: (0.0.0.-1)  QUI SATURE LA CONTRAINTE NUMERO 2.
C     --------------
C     SOLUTION THEORIQUE: F=44       XMAX=(0.1.2.-1)
C     -------------------
C     A L'OPTIMUN LES CONTRAINTES 1 ET 3 SONT SATUREES.
C
C
C     NOUS AVONS INTRODUIT LES VALEURS SUIVANTES:
C
C     EPSCON=0.1E-04
C     PASIN=1
C     EPS=0.1E-10
C     NOMAX=50
C     ITERMA=50
C
C     PROGRAMME PRINCIPAL:
C     --------------------
C
      COMMON/A/C(15),D(15),X(15),AL(15),XMAX(15),IVL(15)
      COMMON/B/TE,PASIN,EPSCON,EPS,FMAX
      COMMON/C/INEW,NPLUSP,NC
      COMMON/D/ILU(15),IVX(15),IB(15),NA,K1,ICKT,NCB,ITERMA
      NOMAX=50
      NETAPE=0
C
C     APPEL DU PROGRAMME D'INTRODUCTION DE DONNEES.
C
      CALL INI
C
C     CONSTRUCTION DES TABLEAUX IVL ET ILU.
C
    1 CALL TAB
    2 FORMAT(1X,4(F11.7,4X),'F=',F11.7,5X,3I1,5X,3I1)
      WRITE(108,2)(XMAX(I),I=1,4),FMAX,(ILU(I),I=1,3),(IVL(I),I=1,3)
C
C     CALCUL DU GRADIENT SIMPLE DE LA FONCTION OBJECTIF.
C
      CALL GRADF(XMAX,IB,C)
C
```

```
CALCUL DU GRADIENT PROJETE DE LA FONCTION OBJECTIF.
SI INEW=1,IL Y A EU UN INCIDENT,NOUS ARRETONS LE PROGRAMME.


      CALL GPROJ(X,D,C,IVL)
      IF(INEW.EQ.1)STOP 1

RECHERCHE D'UN TABLEAU MEILLEUR QUE XMAX SUR LA DIRECTION DU GRADIENT
PROJETE. EN CAS D'ECHEC DE LA METHODE,NOUS ARRETONS LE PROGRAMME.

      CALL XPROJ(D,X,XMAX,IVL)
      IF(INEW.EQ.1)STOP
      NETAPE=NETAPE+1
      IF(NETAPE.GT.NOMAX)STOP1
      GOTO1
      END
```



```
C                    -----
C                    !INI!
C                    -----
C          DONNEES:AUCUNE.
C          --------
C
C          RESULTATS:NCB,NPLUSP,EPS,EPSCON,ITERMA,K1,PASIN,NA,IVX,X
C          ----------
C
C
C     CE SOUS-PROGRAMME EST UN SOUS-PROGRAMME D INTRODUCTION DE DONNEES ET
C     D INITIALISATION.
C
C     PARAMETRES:
C     -----------
C     NCB:NOMBRE DE CONTRAINTES BILATERALES.
C     NPLUSP:NOMBRE DE VARIABLES.
C     EPS:VALEUR MINIMUN TOLEREE DU PAS H DE LA METHODE DU GRADIENT.
C     EPSCON:PRECISION MINIMUN DEMANDEE POUR L ELIMINATION NUMERIQUE(LE SYSTEME
C            DES CONTRAINTES EST VERIFIE A EPSCON PRES.)
C     ITERMA:NOMBRE MAXIMUN D ITERATIONS DEMANDEES DANS L ELIMINATION NUMERIQUE.
C     K1:NOMBRE TOTAL DE CONTRAINTES.
C     PASIN:VALEUR INITIALE DU PAS DE LA METHODE DU GRADIENT.
C     NA:PARAMETRE DE DIMENSIONNEMENT DE TABLEAUX.DANS CET EXEMPLE,NA=15.CE QUI
C        EST LE MAXIMUN POUR POUVOIR FAIRE UNE EXECUTION SUR MITRA 15.
C        UNE MODIFICATION DE CE PARAMETRE DANS UN SENS COMME DANS L'AUTRE
C        DEMANDE LA REECRITURE DES CARTES COMMON ET DIMENSION OU FIGURENT DES
C         TABLEAUX DIMENSIONNES A 15  QU'ILS SOIENT A 1 OU 2 INDICES.
C     IVX:TABLEAU ENTIER DEFINI COMME SUIT:
C              *SI IVX(I)=0   LA CONTRAINTE NUMERO I NE DEPEND QUE DE X(J).
C              *SI IVX(I)=0   LA CONTRAINTE NUMERO I EST QUELCONQUE.
C     X:TABLEAU DU VECTEUR INITIAL.CE VECTEUR DOIT ETRE COMPATIBLE AVEC LES
C       CONTRAINTES.
C     IB:TABLEAU ENTIER DONT TOUS LES ELEMENTS SONT EGAUX A 1.
C
C     AUCUN SOUS-PROGRAMME N'EST APPELE.
C
      SUBROUTINE INI
      COMMON/A/C(15),D(15),X(15),AL(15),XMAX(15),IVL(15)
      COMMON/B/TE,PASIN,EPSCON,EPS,FMAX
      COMMON/C/INEW,NPLUSP,NC
      COMMON/D/ILU(15),IVX(15),IB(15),NA,K1,ICKT,NCB,ITERMA
      READ(105,100)NCB,NPLUSP,EPS,EPSCON,ITERMA,K1,PASIN
  100 FORMAT(I2,I2,E14.7,E14.7,I3,I2,E14.7)
      NA=15
      DO 1 I=1,NA
      IB(I)=1
    1 IVX(I)=0
      READ(105,101)(X(I),I=1,15)
  101 FORMAT(5E14.7)
      RETURN
      END
```

```
C
C     CE SOUS-PROGRAMME CALCULE LE GRADIENT SIMPLE DE LA FONCTION OBJECTIF EN X.
C     LES COMPOSANTES DU GRADIENT SONT RANGEES DANS LE TABLEAU C.
C
      SUBROUTINE GRADF(X,IB,C)
      DIMENSION X(1),IB(1),C(1)
      C(1)=-2.*X(1)+5.
      C(2)=-2.*X(2)+5.
      C(3)=-4.*X(3)+21.
      C(4)=-2.*X(4)-7.
      RETURN
      END
```



```
C
C                    ------------------------------
C                    !CORBAS(K1,IVX,ILU,IVL,NPLUSP)!
C                    ------------------------------
C
C          DONNEES:K1,IVX,ILU,NPLUSP
C          --------
C
C          RESULTATS:IVL
C          ----------
C
C
C     CE SOUS-PROGRAMME D AMELIORATION DE LA BASE CORRIGE LES EFFETS DES
C     EVENTUELLES PERMUTATIONS CIRCULAIRES EN PLACANT DANS LA BASE LES
C     VARIABLES QUI INTERVIENNENT SEULES DANS CERTAINES CONTRAINTES UTILES.
C
C     PARAMETRES:
C     ----------
C     K1:NOMBRE TOTAL DE CONTRAINTES.
C     IVX:TABLEAU ENTIER DEFINI COMME SUIT:
C               *SI IVX(I)=J   LA CONTRAINTE NUMERO I NE DEPEND QUE DE X(J).
C               *SI IVX(I)=0   LA CONTRAINTE NUMERO I EST QUELCONQUE.
C     ILU:TABLEAU ENTIER INDIQUANT LES CONTRAINTES A PRENDRE EN COMPTE.
C               *SI ILU(I)=1   LA CONTRAINTE NUMERO I EST BILATERALE OU
C                              UNILATERALE SATUREE.
C               *SI ILU(I)=0   LA CONTRAINTE NUMERO I EST UNILATERALE NON
C                              SATUREE.
C     NPLUSP:NOMBRE DE VARIABLES.
C     IVL:TABLEAU ENTIER INDIQUANT LA NATURE DES DIVERSES VARIABLES.
C               *SI IVL(I)=0   X(I) EST LIBRE.
C               *SI IVL(I)=1   X(I) EST LIEE.
C
C     AUCUN SOUS-PROGRAMME N'EST APPELE.
C
      SUBROUTINE CORBAS(K1,IVX,ILU,IVL,NPLUSP)
      DIMENSION IVX(1),ILU(1),IVL(1)
      DO 1 I=1,K1
      IF(IVX(I).EQ.0)GOTO1
      IF(ILU(I).EQ.0)GOTO1
      J=IVX(I)
      IF(IVL(J).EQ.1)GOTO1
      DO 2 J1=1,NPLUSP
      IF(IVL(J1).EQ.0)GOTO2
      DO 3 K=1,K1
      IF(IVX(K).NE.J1)GOTO3
      IF(ILU(K).EQ.0)GOTO3
      GOTO2
    3 CONTINUE
      IVL(J1)=0
      K11=IVX(I)
      IVL(K11)=1
      GOTO1
    2 CONTINUE
    1 CONTINUE
      RETURN
      END
```

```
C
C     CE SOUS-PROGRAMME CALCULE LES VALEURS DES CONTRAINTES EN X. CES VALEURS
C     SONT RANGEES DANS LE TABLEAU G.
C
      SUBROUTINE COSA(X,IB,G)
      DIMENSION X(1),IB(1),G(1)
      G(1)=-X(1)*X(1)-X(2)*X(2)-X(3)*X(3)-X(4)*X(4)-X(1)+X(2)-X(3)+X(4)
     1+8.
      G(2)=-X(1)*X(1)-2.*X(2)*X(2)-X(3)*X(3)-2.*X(4)*X(4)+X(1)+X(4)+10.
      G(3)=-2.*X(1)*X(1)-X(2)*X(2)-X(3)*X(3)-2.*X(1)+X(2)+X(4)+5.
      RETURN
      END
```



```
C
C     CE SOUS-PROGRAMME CALCULE LA FONCTION OBJECTIF EN X.
C
      FUNCTION F(X)
      DIMENSION X(1)
      F=-X(1)*X(1)-X(2)*X(2)-2.*X(3)*X(3)-X(4)*X(4)+5.*X(1)+5.*X(2)
     1+21.*X(3)-7.*X(4)
      RETURN
      END
```

```
C     -----
C     !TAB!
C     -----
C
C     DONNEES:NPLUSP,X,NA,NCB,K1,IVX,EPSCON
C     --------
C
C     RESULTATS:ILU,IVL,XMAX,FMAX,NC
C     ----------
C
C
C
C     CE SOUS-PROGRAMME CONSTRUIT LES TABLEAUX IVL,ILU ET XMAX,CALCULE LA
C     NOUVELLE VALEUR DE LA FONCTION (FMAX=F(XMAX)) ET DETERMINE LE NOMBRE
C     DE CONTRAINTES A PRENDRE EN CONSIDERATION.(CONTRAINTES BILATERALES+
C     CONTRAINTES UNILATERALES SATUREES.)
C
C     PARAMETRES:
C     -----------
C     NPLUSP:NOMBRE DE VARIABLES.
C     X:TABLEAU CONTENANT EN DEBUT DE PROGRAMME LE DERNIER ITERE TROUVE.
C     NA:PARAMETRE DE DIMENSIONNEMENT DE TABLEAUX.SI AUCUNE MODIFICATION DU
C         PROGRAMME N'A ETE FAITE NA=15. SINON NA EST EGAL AUX NOUVELLES
C     DIMENSIONS CHOISIES.
C     NCB:NOMBRE DE CONTRAINTES BILATERALES.
C     K1:NOMBRE TOTAL DE CONTRAINTES.
C     IVX:TABLEAU ENTIER DEFINI COMME SUIT:
C               *SI IVX(I)=0   LA CONTRAINTE NUMERO I NE DEPEND QUE DE X(J).
C               *SI IVX(I)=0   LA CONTRAINTE NUMERO I EST QUELCONQUE.
C     ILU:TABLEAU ENTIER INDIQUANT LES CONTRAINTES A PRENDRE EN COMPTE.
C               *SI ILU(I)=1   LA CONTRAINTE NUMERO I EST BILATERALE OU
C                              UNILATERALE SATUREE.
C               *SI ILU(I)=0   LA CONTRAINTE NUMERO I EST UNILATERALE NON
C                              SATUREE.
C     IVL:TABLEAU ENTIER INDIQUANT LA NATURE DES DIVERSES VARIABLES.
C               *SI IVL(I)=0   X(I) EST LIBRE.
C               *SI IVL(I)=1   X(I) EST LIEE.
C     XMAX:XMAX=X
C     FMAX:VALEUR DE LA FONCTION EN XMAX.
C     NC:NOMBRE DE CONTRAINTES UTILES EN X.(BILATERALES+UNILATERALES SATUREES)
C     EPSCON:UNE CONTRAINTE EST CONSIDEREE COMME SATUREE SI SA VALEUR ABSOLUE
C           EST INFERIEURE A EPSCON.
C     IB:TABLEAU ENTIER DONT TOUS LES ELEMENTS SONT EGAUX A 1.
C
C     SOUS-PROGRAMMES APPELES:
C     ------------------------
C     COSA   CORBAS
C
C
      SUBROUTINE TAB
      COMMON/A/C(15),D(15),X(15),AL(15),XMAX(15),IVL(15)
      COMMON/B/IE,PASN,EPSCON,EPS,FMAX
      COMMON/C/INEW,NPLUSP,NC
      COMMON/D/ILU(15),IVX(15),IB(15),NA,K1,ICKT,NCB,ITERMA
      DIMENSION G(15)
      NC=0
```

```
      DO 1 I=1,NPLUSP
      XMAX(I)=X(I)
      DO 2 I=1,NA
      IVL(I)=0
      ILU(I)=0
      IB(I)=1
      FMAX=F(XMAX)
      IF(K1.EQ.0)RETURN
      NC=NCB
      IF(NC.EQ.0)GOTO4
      DO 5 I=1,NC
      IVL(I)=1
      ILU(I)=1
      IF(NCB.EQ.K1)GOTO6
      CALL COSA(X,IB,G)
      NO=NCB+1
      MAO=NCB
      DO 7 I=NO,K1
      IF(ABS(G(I)).GE.EPSCON)GOTO7
      NC=NC+1
      ILU(I)=1
      MAO=MAO+1
      IVL(MAO)=1
    7 CONTINUE
      CALL CORBAS(K1,IVX,ILU,IVL,NPLUSP)
      RETURN
      END
```



```
C
C     CE SOUS-PROGRAMME CALCULE LES DERIVEES DES CONTRAINTES EN X.
C     CES DERIVEES SONT RANGEES DANS LE TABLEAU G:
C     DANS G(I,J) I DESIGNE LA CONTRAINTE ET J LA VARIABLE.
C
      SUBROUTINE DERCO(X,ILU,NA,G)
      DIMENSION X(1),ILU(1),G(NA,NA)
      DO 1 I=1,NA
      DO 1 J=1,NA
    1 G(I,J)=0.
      G(1,1)=-2.*X(1)-1.
      G(1,2)=-2.*X(2)+1.
      G(1,3)=-2.*X(3)-1.
      G(1,4)=-2.*X(4)+1.
      G(2,1)=-2.*X(1)+1.
      G(2,2)=-4.*X(2)
      G(2,3)=-2.*X(3)
      G(2,4)=-4.*X(4)+1.
      G(3,1)=-4.*X(1)-2.
      G(3,2)=-2.*X(2)+1.
      G(3,3)=-2.*X(3)
      G(3,4)=1.
      RETURN
      END
```

```
C     ------------------------------
C     !NCADIR(QR,B,MN,M,N,AL,Y,SUM,IP)!
C     ------------------------------
C
C
C     DONNEES:QR,B,MN,M,N
C     --------
C
C     RESULTATS:B,IP(1)
C     ----------
C
C
C  CE SOUS-PROGRAMME RESOUD UN SYSTEME LINEAIRE DETERMINE OU
C  SURDETERMINE EN EN CHERCHANT LA MEILLEURE APPROXIMATION AU SENS DES
C  MOINDRES CARRES.
C
C  PARAMETRES:
C  -----------
C  QR:MATRICE CONTENANT EN DEBUT DE PROGRAMME LE PREMIER MEMBRE DU SYSTEME
C  A RESOUDRE.
C  B:TABLEAU CONTENANT EN DERUT DE PROGRAMME LE SECOND MEMBRE DU SYSTEME
C  ET EN FIN D EXECUTION LE VECTEUR SOLUTION.
C  MN:PARAMETRE DE DIMENSIONNEMENT DE TABLEAUX.(MN>=SUP(M,N)).
C  M:NOMBRE D EQUATIONS.
C  N:NOMBRE D INCONNUES.
C  IP(1):EN FIN D EXECUTION ,JOUE LE ROLE D UN INDICATEUR D INCIDENT:
C            *SI IP(1)=0   LE RESULTAT EST CORRECT.
C            *SI IP(1)=I   IL Y A EU INTERRUPTION DU TRAITEMENT A LA
C                          LIGNE NUMERO I.
C
C  LES TABLEAUX IP,AL,Y,SUM SONT DES TABLEAUX LOCAUX DE DIMENSION MN.
C
C  AUCUN SOUS-PROGRAMME N EST APPELE.
C
C
      SUBROUTINE NCADIR(QR,B,MN,M,N,AL,Y,SUM,IP)
      DIMENSION QR(MN,MN),B(MN),AL(MN),Y(MN),SUM(MN),IP(MN)
      IF(M.EQ.1.AND.N.EQ.1)GOTO21
      DO 2 J=1,N
      D=0.
      DO 1 I=1,M
      QA=QR(I,J)
    1 D=D+QA*QA
      SUM(J)=D
    2 IP(J)=J
      DO 12 K=1,N
      SIG=SUM(K)
      JBAR=K
      IF(K.EQ.N)GOTO5
      KK=K+1
      DO 4 J=KK,N
      IF(SIG-SUM(J))3,4,4
    3 SIG=SUM(J)
      JBAR=J
    4 CONTINUE
    5 IF(JBAR.EQ.K)GOTO7
      I=IP(K)
      IP(K)=IP(JBAR)
      IP(JBAR)=I
```

```
      SUM(JBAR)=SUM(K)
      SUM(K)=SIG
      DO 6 I=1,M
      SIG=QR(I,K)
      QR(I,K)=QR(I,JBAR)
      QR(I,JBAR)=SIG
    6 D=0.
      DO 8 I=K,M
      QA=QR(I,K)
    8 D=D+QA*QA
      SIG=D
      IBAR=K
      IF(SIG.LE.(0.1E-20))GOTO20
      QRKK=QR(K,K)
      ALK=SQRT(SIG)
      IF(QRKK.GE.0)ALK=-ALK
      AL(K)=ALK
      BETA=1./(SIG-QRKK*ALK)
      QR(K,K)=QRKK-ALK
      IF(K.EQ.N)GOTO13
      K1=K+1
      DO 10 J=K1,N
      D=0.
      DO 9 I=K,M
    9 D=D+QR(I,K)*QR(I,J)
      Y(J)=BETA*D
      K1=K+1
      DO 12 J=K1,N
      DO 11 I=K,M
   11 QR(I,J)=QR(I,J)-QR(I,K)*Y(J)
   12 SUM(J)=SUM(J)-QR(K,J)*QR(K,J)
      DO 14 I=1,M
   14 Y(I)=B(I)
      DO 16 J=1,N
      GA=0.
      DO 15 I=J,M
   15 GA=GA+Y(I)*QR(I,J)
      GA=GA/(AL(J)*QR(J,J))
      DO 16 I=J,M
   16 Y(I)=Y(I)+GA*QR(I,J)
      SUM(N)=Y(N)/AL(N)
      N1=N-1
      DO 18 K1=1,N1
      I=N-K1
      R3=-Y(I)
      I1=I+1
      DO 17 J=I1,N
   17 R3=R3+SUM(J)*QR(I,J)
      SUM(I)=-R3/AL(I)
      DO 19 I=1,N
      IRO=IP(I)
   19 B(IRO)=SUM(I)
      IP(1)=0
      RETURN
   20 IP(1)=IBAR
      RETURN
      IF(DABS(QR(1,1)).GT.(0.1D-20))GOTO22
      IP(1)=1
      RETURN
   22 B(1)=B(1)/QR(1,1)
      IP(1)=0
      RETURN
      END
```

```
C                    --------------------
C                    !GPROJ(XA,DA,C,IVL)!
C                    --------------------
C
C          DONNEES:XA,C,IVL,NCB,NC,K1,NPLUSP,ILU,IVX,NA
C          --------
C
C          RESULTATS:DA,INEW,ICKT
C          ----------
C
C
C    CE SOUS-PROGRAMME CALCULE LE GRADIENT PROJETE DE LA FONCTION OBJECTIF
C    ET LIBERE S IL Y A LIEU LA CONTRAINTE CORRESPONDANT AU PLUS GRAND
C    DES COEFFICIENTS POSITIFS DE KUHN ET TUCKER.
C
C
C    PARAMETRES:
C    -----------
C    XA:POINT OU S EFFECTUE LE CALCUL.
C    C:TABLEAU CONTENANT EN DEBUT DE PROGRAMME LE GRADIENT SIMPLE DE LA
C      FONCTION OBJECTIF.
C    IVL:TABLEAU ENTIER INDIQUANT LA NATURE DE CHAQUE VARIABLE.
C              *SI IVL(I)=0    X(I) EST LIBRE.
C              *SI IVL(I)=1    X(I) EST LIEE.
C    NCB:NOMBRE DE CONTRAINTES BILATERALES.
C    NC:NOMBRE DE CONTRAINTES UTILES.(UNILATERALES SATUREES ET BILATERALES)
C    K1:NOMBRE TOTAL DE CONTRAINTES.
C    NPLUSP:NOMBRE DE VARIABLES.
C    ILU:TABLEAU ENTIER INDIQUANT LES CONTRAINTES A PRENDRE EN COMPTE.
C              *SI ILU(I)=1    LA CONTRAINTE NUMERO I EST BILATERALE OU
C                              UNILATERALE SATUREE.
C              *SI ILU(I)=0    LA CONTRAINTE NUMERO I EST UNILATERALE
C                              NON SATUREE.
C    NA:PARAMETRE DE DIMENSIONNEMENT DE TABLEAUX.(NA=15 DANS LE PROGRAMME)
C    IVX:TABLEAU ENTIER PERMETTANT DE PLACER DANS LA BASE DES VARIABLES
C        QUI DOIVENT NECESSAIREMENT S Y TROUVER.
C              * SI IVX(I)=J    LA CONTRAINTE NUMERO I NE DEPEND QUE DE X(J).
C              * SI IVX(I)=0    LA CONTRAINTE NUMERO I EST QUELCONQUE.
C    DA:TABLEAU RENFERMANT LE GRADIENT PROJETE EN FIN DE CALCUL.
C    INEW:INDICATEUR D INCIDENT.
C              * INEW=0 SI LE GRADIENT PROJETE EST EFFECTIVEMENT CALCULE.
C              *INEW=1 SI LA RESOLUTION DU SYSTEME LINEAIRE AVORTE.
C    ICKT:INDICATEUR PRENANT LA VALEUR 0 SI EN FIN DE CALCUL TOUS LES
C        COEFFICIENTS DE KUHN ET TUCKER SONT NEGATIFS OU NULS,ET LA VALEUR
C         1 SI AU MOINS UN DES COEFFICIENTS EST STRICTEMENT POSITIF.
C
C    SOUS-PROGRAMMES APPELES:
C    ------------------------
C    DEPCO   CORBAS   NCADIR
C
C
      SUBROUTINE GPROJ(XA,DA,C,IVL)
      DIMENSION XA(1),DA(1),T1(15,15),C(15),AL(15),Y(15),SUM(15)
      DIMENSION IP(15),T2(15,15),T3(15,15),G(15),IVL(1)
      COMMON/B/TE,PASIN,EPSCON,EPS,EMAX
      COMMON/C/INEW,NPLUSP,NC
      COMMON/D/ILU(15),IVX(15),IB(15),NA,K1,ICKT,NCB,ITERMA
      IA=0
```

```
      IF(NC.EQ.0)GOTO12
      CALL DEPCO(XA,ILU,NA,T1)
      MAO=0
      DO 2 I=1,K1
      IF(ILU(I).NE.1)GOTO2
      MAO=MAO+1
      DO 1 J=1,NPLUSP
    1 T2(MAO,J)=T1(I,J)
    2 CONTINUE
      DO 3 I=1,NA
      DO 3 J=1,NA
      T1(I,J)=T2(J,I)
    3 T3(I,J)=T1(I,J)
      DO 4 I=1,NPLUSP
    4 G(I)=C(I)
      CALL MCADIR(T3,G,NA,NPLUSP,NC,AL,Y,SUM,IP)
      IF(IP(1).EQ.0)GOTO5
      INEW=1
      RETURN
    5 AMAX=-1.
      IF(NCB.EQ.K1)GOTO7
      MAO=NCB
      NAB=NCB+1
      DO 6 I=NAB,K1
      IF(ILU(I).NE.1)GOTO6
      MAO=MAO+1
      IF(G(MAO).LT.AMAX)GOTO6
      AMAX=G(MAO)
      IIMAX=I
    6 CONTINUE
      IF(AMAX.GT.(0.1E-29))GOTO9
    7 ICKT=0
    8 INEW=0
      GOTO14
    9 IF(IA.EQ.0)GOTO10
      ICKT=1
      GOTO8
   10 IA=1
      NC=NC-1
      DO 11 I=1,NPLUSP
      IF(IVL(I).NE.1)GOTO11
      IVL(I)=0
      ILU(IIMAX)=0
      CALL CORBAS(K1,IVX,ILU,IVL,NPLUSP)
      GOTO17
   11 CONTINUE
   12 INEW=0
      ICKT=0
      DO 13 I=1,NPLUSP
   13 DA(I)=C(I)
      RETURN
   14 DO 16 I=1,NPLUSP
      DA(I)=0.
      DO 15 J=1,NC
   15 DA(I)=DA(I)+T1(I,J)*G(J)
   16 DA(I)=C(I)-DA(I)
      RETURN
      END
```



```
C                    ----------------------
C                    !XPROJ(D,X,XMAX,IVL)!
C                    ----------------------
C
C          DONNEES:D,XMAX,IVL,NPLUSP,PASIN,ILU,EPSCON,NCB,K1,NC,EPS,IVX,
C          --------ITERMA
C
C          RESULTATS:X,INEW
C          ----------
C
C
C  CE SOUS-PROGRAMME CHERCHE A AMELIORER LE DERNIER ITERE XMAX SUR LA
C  DIRECTION DE MONTEE DU GRADIENT PROJETE.
C  SI CETTE DIRECTION NOUS FAIT SORTIR DU DOMAINE DEFINI PAR LES
C  CONTRAINTES,XPROJ SATURE LA PREMIERE CONTRAINTE RENCONTREE PAR UNE
C  METHODE DE DICHOTOMIE(APPEL DE DICHO) ET RETOURNE LE CONTROLE AU PROGRAMME
C  APPELANT.
C  REMARQUES:
C  ----------
C  # DANS LE CAS OU NOUS SATURONS UNE NOUVELLE CONTRAINTE,IL EST POSSIBLE
C  QUE LE VECTEUR XMAX AINSI OBTENU NE SOIT PAS MEILLEUR QUE L ANCIEN,EN
C  EFFET LA RESOLUTION DU SYSTEME DES CONTRAINTES INTRODUIT DES MARGES
C  D ERREUR QUE NOUS NE CONTROLONS PAS ET QUI PEUVENT DEPLACER LE POINT PAR
C  RAPPORT A LA DIRECTION DE MONTEE.
C  CETTE METHODE EST ADMISSIBLE CAR LA CONNAISSANCE DES CONTRAINTES A
C  SATURER EST BEAUCOUP PLUS IMPORTANTE POUR LA SUITE DES CALCULS QUE LE
C  GAIN D UNE ITERATION.
C  #NOUS AVONS RENONCE DANS CE SOUS-PROGRAMME A UTILISER LA TECHNIQUE DE
C  DETERMINATION AUTOMATIQUE DU PAS CAR LA METHODE DU GRADIENT PROJETE
C  N EST POUR NOUS QU UNE SECURITE QUI ASSURE UNE DIRECTION DE MONTEE
C  LORSQUE LES CONTRAINTES VARIENT,PHENOMENE QUE NOUS CONSIDERERONS COMME
C  EXEPTIONNEL.
C  IL EST EVIDENT QUE DANS UN PROBLEME OU LES CONTRAINTES VARIENT BEAUCOUP,
C  CE PROGRAMME DEMANDERA A ETRE AMELIORE.
C
C  PARAMETRES:
C  -----------
C  D:TABLEAU CONTENANT LE GRADIENT PROJETE DE LA FONCTION OBJECTIF.
C  XMAX:POINT DE DEPART DE L ITERATION. LE PROGRAMME CHERCHE A AMELIORER
C  FMAX:VALEUR DE LA FONCTION EN XMAX.
C  IVL:TABLEAU ENTIER INDIQUANT LA NATURE DES DIVERSES VARIABLES.
C            *SI IVL(I)=0   X(I) EST LIBRE.
C            *SI IVL(I)=1   X(I) EST LIEE.
C  NPLUSP:NOMBRE DE VARIABLES.
C  PASIN:EN DEBUT DE PROGRAMME,VALEUR INITIALE DU PAS DE LA METHODE DU
C  GRADIENT ET EN FIN DE PROGRAMME,VALEUR DU PAS EFFECTIVEMENT UTILISE.
C  ILU:TABLEAU ENTIER INDIQUANT LES CONTRAINTES A PRENDRE EN COMPTE.
C            *SI ILU(I)=1   LA CONTRAINTE NUMERO I EST BILATERALE OU
C                           UNILATERALE SATUREE.
C            *SI ILU(I)=0   LA CONTRAINTE NUMERO I EST UNILATERALE NON
C                           SATUREE.
C  EPCON:PRECISION MINIMUN DEMANDEE DANS L ELIMINATION NUMERIQUE.
C  ITERMA:NOMBRE MAXIMUN D ITERATIONS TOLEREES DANS LA METHODE DE NEWTON.
C  NCB:NOMBRE DE CONTRAINTES BILATERALES.
C  K1:NOMBRE TOTAL DE CONTRAINTES.
C  NC:NOMBRE DE CONTRAINTES UTILES.(BILATERALES+UNILATERALES SATUREES)
C  EPS:VALEUR MINIMUN TOLEREE DU PAS.
C  IVX:TABLEAU ENTIER DEFINI COMME SUIT:
```

```
C                *SI IVX(I)=J   LA CONTRAINTE NUMERO I NE DEPEND QUE DE X(J).
C                *SI IVX(I)=0   LA CONTRAINTE NUMERO I EST QUELCONQUE.
C     X:TABLEAU CONTENANT EN FIN DE PROGRAMME (ET SI INEW=0)UN VECTEUR QUI
C       AMELIORE XMAX.
C     INEW:INDICATEUR D'INCIDENT:
C                *SI INEW=0 NOUS AVONS PU AMELIORER XMAX.
C                *SI INEW=1 NOUS N'AVONS PAS PU AMELIORER LE DERNIER ITERE.
C
C
C     SOUS/PROGRAMMES APPELES:
C     ------------------------
C     CALVAS   COSA   DICHO   CORHAS
C
      SUBROUTINE XPROJ(D,X,XMAX,IVL)
      COMMON/B/TE,PASIN,EPSCON,EPS,FMAX
      COMMON/C/INEW,NPLUSP,NC
      COMMON/D/ILU(15),IVX(15),IB(15),NA,K1,ICKT,NCB,ITERMA
      DIMENSION D(1),X(1),XMAX(1),Y(15),AL(15),IVL(1)
      INEW=0
      IAO=0
      NP=0
      TE=PASIN
      IO=0
    1 DO 2 I=1,NPLUSP
    2 X(I)=XMAX(I)+TE*D(I)
      CALL CALVAS(X)
      IF(INEW.NE.0)GOTO6
      IF(IO.EQ.1)GOTO3
      F1=F(X)
      IF(F1.LT.FMAX)GOTO6
    3 CALL COSA(X,IB,Y)
      CALL DICHO(I,Y,IO,TE,SS,ILU,NCB,K1,EPSCON)
      GOTO(8,4,3),I
    4 F1=F(X)
      DO 5 I=1,K1
      AL(I)=ABS(Y(I))
      IF(AL(I).GE.EPSCON)GOTO5
      IF(ILU(I).EQ.1)GOTO5
      NC=NC+1
      IAO=1
    5 CONTINUE
      PASIN=TE
      IF(NC.GT.NPLUSP)GOTO9
      RETURN
    6 NP=NP+1
      IF(NP.GT.NPLUSP)GOTO8
      LOU=IVL(NPLUSP)
      DO 7 I=2,NPLUSP
      J=NPLUSP+2-I
    7 IVL(J)=IVL(J-1)
      IVL(1)=LOU
      CALL CORHAS(K1,IVX,ILU,IVL,NPLUSP)
      GOTO1
    8 TE=TE/2.
      IF(TE.LT.EPS)GOTO9
      NP=0
      GOTO1
    9 INEW=1
      RETURN
      END
```



```
C     ------------------------------------
C     !DICHO(I,Y,IO,TE,SS,ILU,NCB,K1,EPSCON)!
C     ------------------------------------
C
C     DONNEES:Y,IO,TE,SS,ILU,NCB,K1,EPSCON
C     --------
C
C     RESULTATS:I,TE,SS
C     ----------
C
C
C     CE SOUS-PROGRAMME LIE A XPROJ SATURE LA PREMIERE CONTRAINTE RENCONTREE
C     SUR LA DIRECTION DE MONTEE DU GRADIENT PROJETE LORSQUE CETTE DIRECTION
C     NOUS FAIT QUITTER LE DOMAINE DEFINI PAR LES CONTRAINTES.
C     LA METHODE UTILISEE EST LA METHODE DE DICHOTOMIE EXPLICITEE DANS LA PARTIE
C     THEORIQUE.
C
C     REMARQUES:
C     ----------
C     CE SOUS-PROGRAMME A ETE CONCU UNIQUEMENT POUR LE ROLE QU'IL TIENT DANS
C     XPROJ ET NE PEUT PAS ETRE UTILISE DETACHE DE CE PROGRAMME.
C     TOUS LES PARAMETRES SONT INTERNES A XPROJ.
C     AUCUN SOUS-PROGRAMME N'EST APPELE.
C
      SUBROUTINE DICHO(I,Y,IO,TE,SS,ILU,NCH,K1,EPSCON)
      DIMENSION ILU(1)
      DIMENSION Y(1)
      MAO=0
      IF(NCH.NE.K1)GOTO2
    1 I=2
      RETURN
    2 J=NCH+1
      DO 3 I=J,K1
      IF(Y(I).GE.(-EPSCON))GOTO3
      MAO=MAO+1
    3 CONTINUE
      IF(MAO)10,4,10
    4 IF(IO)5,1,5
    5 MAO=0
      DO 6 I=J,K1
      IF(ILU(I).EQ.1)GOTO6
      IF(ABS(Y(1)).GE.EPSCON)GOTO6
      MAO=MAO+1
    6 CONTINUE
      IF(MAO-1)7,1,9
    7 TE=SS+TE
    8 I=1
      RETURN
    9 SS=TE
      GOTO8
   10 IO=1
      GOTO9
      END
```

```
                    -----------
                    !CALVAS(X)!
                    -----------

          DONNEES:NC,X,ILU,K1,NPLUSP,ITERMA,NA,IVL,EPSCON
          --------

          RESULTATS:X,INEW
          ----------


C  SOUS-PROGRAMME CALCULE LES VARIABLES LIEES A PARTIR DES VARIABLES
C  LIBRES PAR LA METHODE DE NEWTON.
C
C  PARAMETRES:
C  -----------
C
C  NC:NOMBRE DE CONTRAINTES UTILES.(BILATERALES+UNILATERALES SATUREES)
C  X:TABLEAU SUR LEQUEL S EFFECTUE L ELIMINATION.
C  ILU:TABLEAU ENTIER DONNANT LE TYPE DE CHAQUE CONTRAINTE:
C            *SI ILU(I)=1   LA CONTRAINTE NUMERO I EST BILATERALE OU
C                           UNILATERALE SATUREE.
C            *SI ILU(I)=0   LA CONTRAINTE NUMERO I EST UNILATERALE NON
C                           SATUREE.
C  K1:NOMBRE TOTAL DE CONTRAINTES.
C  NPLUSP:NOMBRE DE VARIABLES.
C  ITERMA:NOMBRE MAXIMUN D'ITERATIONS TOLEREES DANS LA METHODE DE NEWTON.
C  NA:PARAMETRE DE DIMENSIONNEMENT DE TABLEAU.ICI,NA=15.
C  IVL:TABLEAU ENTIER DEFINISSANT LE TYPE DE CHAQUE VARIABLE:
C            *SI IVL(I)=1     X(I) EST LIEE.
C            *SI IVL(I)=0     X(I) EST LIBRE.
C  EPSCON:DONNE LA PRECISION DE L ELIMINATION.LE SYSTEME DES CONTRAINTES
C  EST VERIFIE A EPSCON PRES.
C  INEW:INDICATEUR D INCIDENT:
C                       *SI INEW=0 L ELIMINATION NUMERIQUE EST CORRECTE.
C                       *SI INEW=1 IL Y A INCIDENT
C
C  SOUS-PROGRAMMES APPELES:
C  ------------------------
C
C  COSA   DEPCO   NCADIR


      SUBROUTINE CALVAS(X)
      COMMON/A/C(15),D(15),Z(15),AL(15),XMAX(15),IVL(15)
      COMMON/B/TE,PASIN,EPSCON,EPS,FMAX
      COMMON/C/INEW,NPLUSP,NC
      COMMON/D/ILU(15),IVA(15),IB(15),NA,K1,ICKT,NCB,ITERMA
      DIMENSION X(1),Y(15),SUM(15),IP(15),F(15)
      DIMENSION G(15,15),G1(15,15)
      INTEGER P
      P=NC
      IF(P.EQ.0)GOTO10
      IT=0
    6 CALL COSA(X,IB,F)
      CALL DEPCO(X,ILU,NA,G1)
      NAO=0
```

```
      DO 1 I=1,K1
      IF(ILU(I).NE.1)GOTO1
      MAO=MAO+1
      F(MAO)=F(I)
    1 CONTINUE
      MAO=0
      DO 2 I=1,K1
      IF(ILU(I).NE.1)GOTO2
      MAO=MAO+1
      MAA=0
      DO 3 J=1,NPLUSP
      IF(IVL(J).NE.1)GOTO3
      MAA=MAA+1
      G(MAO,MAA)=GI(I,J)
    3 CONTINUE
    2 CONTINUE
      IT=IT+1
      IF(IT.LE.ITERMA)GOTO7
    8 INEW=1
      RETURN
    7 CALL NCADIR(G,F,NA,P,P,AL,Y,SUM,IP)
      IF(IP(1).NE.0)GOTO8
      MAO=0
      DO 4 I=1,NPLUSP
      IF(IVL(I).NE.1)GOTO4
      MAO=MAO+1
      X(I)=X(I)-F(MAO)
    4 CONTINUE
      CALL COSA(X,ILU,Y)
      DO 5 I=1,K1
      IF(ILU(I).NE.1)GOTO5
      IF(ABS(Y(I)).LT.EPSCON)GOTO5
      GOTO6
    5 CONTINUE
   10 INEW=0
      RETURN
      END
```



```
C
C
C
C     ------------------------------------
C     !   TECHNIQUE DU GRADIENT REDUIT   !
C     ------------------------------------
C
C
C
C
C
C
C     LE PROBLEME POSE EST LE SUIVANT:
C               MAXIMISER F=-X(1)*X(1)-X(2)*X(2)-2*X(3)*X(3)-X(4)*X(4)
C                          +5X(1)+5X(2)+21X(3)-7X(4)
C     AVEC LES CONTRAINTES:
C     -X(1)*X(1)-X(2)*X(2)-X(3)*X(3)-X(4)*X(4)-X(1)+X(2)-X(3)+X(4)+8=0
C     -X(1)*X(1)-2*X(2)*X(2)-X(3)*X(3)-2*X(4)*X(4)+X(1)+X(4)+10=0
C
C     POINT INITIAL: 0.07011455
C     ------------- 0.74074510
C                    2.0014640
C                   -1.0361300
C
C     LA SOLUTION THEORIQUE EST: (0,1,2,-1)  :  F=44
C
C
C     NOUS AVONS INTRODUIT LES VALEURS SUIVANTES:
C
C     EPSCON=0.1D-08
C     PASIN=1.
C     EPS=0.1D-08
C     NOMAX=50
C     ITERMA=50
C
C
C     PROGRAMME PRINCIPAL
C     -------------------
C
C     CE PROGRAMME ENCHAINE LES DIVERSES OPERATIONS DECRITES DANS LA PARTIE
C     THEORIQUE.
C     L'ECRITURE PROPOSEE N'EST DONNEE QU'A TITRE D'EXEMPLE, EN PARTICULIER,LE
C     TRAITEMENT DES INCIDENTS DOIT ETRE REENVISAGE A CHAQUE UTILISATION.
C
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON ITERMA
      COMMON/AA/IA
      COMMON/A/C(15),D(15),X(15),AL(15),XMAX(15),IVL(15)
      COMMON/B/TE,PASIN,EPSCON,EPS,FMAX
      COMMON/C/INEW,NPLUSP,NC
      DIMENSION XET(15),AH(15),AR(15,15)
C
C     INTRODUCTION DES DONNEES.
C
      READ(105,100)PASIN,NPLUSP,NC,EPSCON,EPS,NBPTS
      READ(105,101)(IVL(I),I=1,15)
      READ(105,102)(X(I),I=1,15)
  100 FORMAT(F7.3,I1,I1,D14.7,D14.7,I3)
  101 FORMAT(15I1)
```

```
  102 FORMAT(5D14.7)
      NOMAX=50
      NETAPE=0
      ITERMA=50
    3 MAO=1
      DO 1 I=1,15
      XMAX(I)=X(I)
    1 AR(MAO,I)=XMAX(I)
      FMAX=F(XMAX)
    2 IA=0
      WRITE(108,5)(XMAX(I),I=1,4),FMAX,NETAPE
    5 FORMAT(1X,4(F11.7,4X),'F=',F11.7,6X,I6)
C
C     CALCUL DU GRADIENT SIMPLE DE LA FONCTION OBJECTIF.
C
      CALL GRADF(X,C,NBPTS)
      IF(NETAPE.GT.NOMAX)STOP
      N=NPLUSP-NC
C
C     CALCUL DU GRADIENT REDUIT DE LA FONCTION OBJECTIF.
C     SI INEW=1 IL Y A EU INCIDENT. DANS CET EXEMPLE CE CAS EST IMPOSSIBLE,NOU
C     NE LE TESTONS PAS.
C
      CALL GREDUI(N,NC,NPLUSP,INEW)
      WRITE(108,7)(D(I),I=1,2)
    7 FORMAT(50X,2(F14.7,6X))
C
C     RECHERCHE D'UN VECTEUR X(R+1) MEILLEUR QUE X(R) SUR LA DIRECTION DE
C     MONTEE DU GRADIENT REDUIT.
C     SI INEW=1,IL Y A INCIDENT. DANS CET EXEMPLE NOUS ARRETONS LE PROGRAMME.
C
      CALL XREDUI
      IF(INEW.NE.0)STOP 5
      PASIN=TE
      AH(MAO)=TE
      MAO=MAO+1
      DO 11 I=1,NPLUSP
   11 AR(MAO,I)=X(I)
      NETAPE=NETAPE+1
      IF(MAO.LT.(N+2))GOTO2
C
C     MISE EN OEUVRE DE L'ACCELERATION DE CONVERGENCE,ICI PAR UN TEST SUR CLE.
C
      IF(KEY(15).EQ.KEY(0))GOTO3
      CALL CALXET(XET,AH,AR)
C
C     A CE STADE SI INEW=1,IL Y A EU UN INCIDENT DANS LE CALCUL DE X*.SI INEW=0
C     NOUS RENDONS CE VECTEUR COMPATIBLE AVEC LES CONTRAINTES ET NOUS VERIFIONS
C     QUE LE RESULTAT OBTENU AMELIORE LE DERNIER ITERE.
C
      IF(INEW)3,4,3
    4 CALL XETM(XET)
C
C     SI INEW=0,X* EST COMPATIBLE AVEC LES CONTRAINTES ET MEILLEUR QUE XMAX.
C     SINON,INEW=1.
C
      IF(INEW.EQ.0)WRITE(108,6)
    6 FORMAT(1X,'L ACCELERATION DE CONVERGENCE A FONCTIONNE')
      GOTO3
      END
```



```
C
C     CE SOUS-PROGRAMME CALCULE LE GRADIENT SIMPLE EN X DE LA FONCTION OBJECTIF
C     ET LE RANGE DANS LE TABLEAU C. LE PARAMETRE N N'EST UTILE QU'EN CONTROLE
C     OPTIMAL.
C
      SUBROUTINE GRADF(X,C,N)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(1),C(1)
      C(1)=-2.D0*X(1)+5.D0
      C(2)=-2.D0*X(2)+5.D0
      C(3)=-4.D0*X(3)+21.D0
      C(4)=-2.D0*X(4)-7.D0
      RETURN
      END
```

```
C
C     CE SOUS-PROGRAMME CALCULE LA VALEUR DE LA FONCTION OBJECTIF AU POINT X.
C
      DOUBLE PRECISION FUNCTION F(X)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(1)
      F=-X(1)*X(1)-X(2)*X(2)-2.D0*X(3)*X(3)-X(4)*X(4)+5.D0*X(1)
     1+5.D0*X(2)+21.D0*X(3)-7.D0*X(4)
      RETURN
      END
```



```
C     CE SOUS-PROGRAMME CALCULE LES DERIVEES EN X DES CONTRAINTES PAR RAPPORT
C     AUX DIVERSES VARIABLES.
C     DANS C(I,J),I DESIGNE LA CONTRAINTE ET J LA VARIABLE.
C
      SUBROUTINE DEPCO(X,C)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(1),C(3,15)
      DO 1 I=1,15
      C(1,I)=0.D0
      C(2,I)=0.D0
    1 C(3,I)=0.D0
      C(1,1)=-2.D0*X(1)-1.D0
      C(1,2)=-2.D0*X(2)+1.D0
      C(1,3)=-2.D0*X(3)-1.D0
      C(1,4)=-2.D0*X(4)+1.D0
      C(2,1)=-4.D0*X(1)-2.D0
      C(2,2)=-2.D0*X(2)+1.D0
      C(2,3)=-2.D0*X(3)
      C(2,4)=1.D0
      RETURN
      END
```

```
C
C                    ------------------------------
C                    !MCADIR(QR,B,MN,M,N,AL,Y,SUM,IP)!
C                    ------------------------------
C
C          DONNEES:QR,B,MN,N,M
C          --------
C
C
C          RESULTATS:B,IP(1)
C          ----------
C
C
C   CE SOUS-PROGRAMME RESOUD UN SYSTEME LINEAIRE DETERMINE OU
C   SURDETERMINE EN EN CHERCHANT LA MEILLEURE APPROXIMATION AU SENS DES
C   MOINDRES CARRES.
C
C   PARAMETRES:
C   -----------
C   QR:MATRICE CONTENANT EN DEBUT DE PROGRAMME LE PREMIER MEMBRE DU SYSTEME
C   A RESOUDRE.
C   B:TABLEAU CONTENANT EN DEBUT DE PROGRAMME LE SECOND MEMBRE DU SYSTEME
C   ET EN FIN D EXECUTION LE VECTEUR SOLUTION.
C   MN:PARAMETRE DE DIMENSIONNEMENT DE TABLEAUX.(MN>=SUP(M,N)).
C   N:NOMBRE D INCONNUES.
C   M:NOMBRE D EQUATIONS.
C   IP(1):EN FIN D EXECUTION ,JOUE LE ROLE D UN INDICATEUR D INCIDENT:
C            *SI IP(1)=0   LE RESULTAT EST CORRECT.
C            *SI IP(1)=I   IL Y A EU INTERRUPTION DU TRAITEMENT A LA
C                          LIGNE NUMERO I.
C
C   LES TABLEAUX IP,AL,Y,SUM SONT DES TABLEAUX LOCAUX DE DIMENSION MN.
C
C   AUCUN SOUS-PROGRAMME N'EST APPELE.
C
C
      SUBROUTINE MCADIR(QR,B,MN,M,N,AL,Y,SUM,IP)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION QR(MN,MN),B(MN),AL(MN),Y(MN),SUM(MN),IP(MN)
      IF(M.EQ.1.AND.N.EQ.1)GOTO21
      DO 2 J=1,N
      D=0.
      DO 1 I=1,M
      QA=QR(I,J)
    1 D=D+QA*QA
      SUM(J)=D
    2 IP(J)=J
      DO 12 K=1,N
      SIG=SUM(K)
      JBAR=K
      IF(K.EQ.N)GOTO5
      KK=K+1
      DO 4 J=KK,N
      IF(SIG-SUM(J))3,4,4
    3 SIG=SUM(J)
      JBAR=J
    4 CONTINUE
    5 IF(JBAR.EQ.K)GOTO7
      I=IP(K)
      IP(K)=IP(JBAR)
      IP(JBAR)=I
      SUM(JBAR)=SUM(K)
      SUM(K)=SIG
      DO 6 I=1,M
      SIG=QR(I,K)
      QR(I,K)=QR(I,JBAR)
    6 QR(I,JBAR)=SIG
    7 D=0.
      DO 8 I=K,M
      QA=QR(I,K)
    8 D=D+QA*QA
      SIG=D
      IBAR=K
      IF(SIG.LE.(0.1D-29))GOTO20
      QRKK=QR(K,K)
      ALK=SQRT(SIG)
      IF(QRKK.GE.0)ALK=-ALK
      AL(K)=ALK
      BETA=1./(SIG-QRKK*ALK)
      QR(K,K)=QRKK-ALK
      IF(K.EQ.N)GOTO13
      K1=K+1
      DO 10 J=K1,N
      D=0.
      DO 9 I=K,M
    9 D=D+QR(I,K)*QR(I,J)
   10 Y(J)=BETA*D
      K1=K+1
      DO 12 J=K1,N
      DO 11 I=K,M
   11 QR(I,J)=QR(I,J)-QR(I,K)*Y(J)
   12 SUM(J)=SUM(J)-QR(K,J)*QR(K,J)
   13 DO 14 I=1,M
   14 Y(I)=B(I)
      DO 16 J=1,N
      QA=0.
      DO 15 I=J,M
   15 QA=QA+Y(I)*QR(I,J)
      QA=QA/(AL(J)*QR(J,J))
      DO 16 I=J,M
   16 Y(I)=Y(I)+QA*QR(I,J)
      SUM(N)=Y(N)/AL(N)
      N1=N-1
      DO 18 K1=1,N1
      I=N-K1
      R3=-Y(I)
      I1=I+1
      DO 17 J=I1,N
   17 R3=R3+SUM(J)*QR(I,J)
   18 SUM(I)=-R3/AL(I)
      DO 19 I=1,N
      IRO=IP(I)
   19 B(IRO)=SUM(I)
      IP(1)=0
      RETURN
   20 IP(1)=IBAR
      RETURN
   21 IF(DABS(QR(1,1)).GT.(0.1D-29))GOTO22
      IP(1)=1
      RETURN
   22 B(1)=B(1)/QR(1,1)
      IP(1)=0
      RETURN
      END
```

```
C
C                          -------------------------
C                          !GREDUI(N,NC,NPLUSP,INEW)!
C                          -------------------------
C
C               DONNEES:N,NC,NPLUSP,C,X,IVL
C               --------
C
C               RESULTATS:D,INEW
C               ----------
C
C
C     CE SOUS-PROGRAMME CALCULE LE GRADIENT REDUIT DE LA FONCTION A OPTIMISER.
C
C     PARAMETRES:
C     -----------
C     N:NOMBRE DE VARIABLES LIBRES.
C     NC:NOMBRE DE CONTRAINTES.
C     NPLUSP:NOMBRE TOTAL DE VARIABLES.
C     C:TABLEAU CONTENANT EN DEBUT DE PROGRAMME LE GRADIENT SIMPLE DE LA
C       FONCTION A OPTIMISER. CE TABLEAU EST DETRUIT EN COURS DE CALCUL.
C     X:TABLEAU CONTENANT LES COORDONNEES DU POINT OU L ON CALCULE LE
C     GRADIENT REDUIT.
C     IVL:TABLEAU INDIQUANT LA NATURE DES DIVERSES VARIABLES.
C               *SI IVL(I)=0   X(I) EST LIBRE.
C               *SI IVL(I)=1   X(I) EST LIEE.
C     D:TABLEAU CONTENANT LE GRADIENT EN FIN DE CALCUL SI INEW=0.
C     INEW:INDICATEUR D INCIDENT.
C               *SI INEW=0 LE GRADIENT REDUIT EST CORRECT.
C               *SI INEW=1 IL Y A INCIDENT,LE RESULTAT EST INCORRECT.
C
C
C     SOUS-PROGRAMMES APPELES:
C     ------------------------
C     DEPCO       MCADIR
C
C
C
      SUBROUTINE GREDUI(N,NC,NPLUSP,INEW)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON/A/C(15),D(15),X(15),AL(15),XMAX(15),IVL(15)
      DIMENSION T1(3,15),T2(3,3),T11(3,15)
      DIMENSION A1(3),A2(3),A3(3),IP(3),SUM(3),T3(3,3)
      INEW=0
      IF(NC.NE.0)GOTO2
      DO 1 I=1,NPLUSP
    1 D(I)=C(I)
      RETURN
    2 CALL DEPCO(X,T1)
      DO 4 I=1,NC
      M=0
      MAA=0
      DO 5 J=1,NPLUSP
      IF(IVL(J).EQ.0)GOTO3
      MAA=MAA+1
      T2(1,MAA)=T1(I,J)
      GOTO5
    3 M=M+1
      T11(I,M)=T1(I,J)
    5 CONTINUE
    4 CONTINUE
      DO 6 I=1,NC
      DO 6 J=1,NC
    6 T3(I,J)=T2(J,I)
      MAA=0
      MAU=0
      DO 8 I=1,NPLUSP
      IF(IVL(I).EQ.0)GOTO7
      MAA=MAA+1
      SUM(MAA)=C(I)
      GOTO8
    7 MAU=MAU+1
      AL(MAU)=C(I)
    8 CONTINUE
      CALL MCADIR(T3,SUM,3,NC,NC,A1,A2,A3,IP)
      IF(IP(1).EQ.0)GOTO9
      INEW=1
      RETURN
    9 DO 11 I=1,N
      C(I)=0.D0
      DO 10 J=1,NC
   10 C(I)=C(I)+T11(J,I)*SUM(J)
   11 D(I)=AL(I)-C(I)
      RETURN
      END
```

```
C                          ------------------
C                          !CALXET(XET,AH,AR)!
C                          ------------------
C
C               DONNEES:AH,AR,NPLUSP,NC,IVL
C               --------
C
C               RESULTATS:XET,INEW
C               ----------
C
C
C     CE SOUS-PROGRAMME CALCULE LE VECTEUR X* DE LA TECHNIQUE D ACCELERATION D
C     CONVERGENCE.
C
C     PARAMETRES:
C     -----------
C     AH:TABLEAU OU SONT STOCKES LES DIFFERENTS PAS UTILISES.
C     AR:TABLEAU A DEUX DIMENSIONS CONTENANT LES ITERES SUCCESSIFS.
C     DANS AR(I,J),I REPRESENTE LE NUMERO DU VECTEUR ET J LA COORDONNEE
C     CONSIDEREE.
C     NPLUSP:NOMBRE DE VARIABLES.
C     NC:NOMBRE DE CONTRAINTES.
C     IVL:TABLEAU INDIQUANT LA NATURE DES DIVERSES VARIABLES.
C               *SI IVL(I)=0      X(I) EST LIBRE.
C               *SI IVL(I)=1      X(I) EST LIEE.
C     XET:VECTEUR X*.
C     INEW:INDICATEUR D INCIDENT.
C               *SI INEW=0  LE VECTEUR X* EST EFFECTIVEMENT OBTENU.
C               *SI INEW=1  IL Y A INCIDENT,LE RESULTAT DE X* EST ERRONE.
C
C     SOUS-PROGRAMME APPELE:
C     ----------------------
C     MCADIR
C
C     REMARQUE:
C     ---------
C     A LA FIN DE L'EXECUTION XET N'EST PAS OBLIGATOIREMENT COMPATIBLE AVEC
C     LES EVENTUELLES CONTRAINTES.
C     SI NC N'EST PAS NUL,IL EST NECESSAIRE APRES L'EXECUTION DE CALXET
C     D'APPELER XETM.
C
C
      SUBROUTINE CALXET(XET,AH,AR)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON/A/C(15),D(15),X(15),AL(15),XMAX(15),IVL(15)
      COMMON/B/TE,PASIN,EPSCON,EPS,FMAX
      COMMON/C/INEW,NPLUSP,NC
      DIMENSION XET(1),AR(15,15),AH(1),T1(15,15),U(15),IP(15)
      N=NPLUSP-NC
      NP1=N+1
      DO 2 I=1,NP1
      MAO=0
      DO 1 J=1,NPLUSP
      IF(IVL(J).EQ.1)GOTO1
      MAO=MAO+1
      T1(MAO,I)=AR(I+1,J)-AR(I,J)
    1 CONTINUE
```



```
    2 CONTINUE
      DO 3 I=1,N
    3 AL(I)=-T1(I,NP1)
      CALL MCADIR(T1,AL,15,N,N,D,C,U,IP)
      AL(NP1)=1.D0
      IF(IP(1).EQ.0)GOTO5
    4 INEW=1
      RETURN
    5 S=0.D0
      DO 6 I=1,NP1
    6 S=S+AH(I)*AL(I)
      IF(DABS(S).LE.(0.1D-29))GOTO4
      DO 8 I=1,NPLUSP
      XET(I)=0.D0
      IF(IVL(I).EQ.1)GOTO8
      DO 7 J=1,NP1
    7 XET(I)=XET(I)+AL(J)*AR(J,I)*AH(J)
      XET(I)=XET(I)/S
    8 CONTINUE
      INEW=0
      RETURN
      END
```

```
C                          ----------
C                          !CALVA(X)!
C                          ----------
C
C
C              DONNEES:X,NC,IVL,ITERMA,EPSCON
C              --------
C
C              RESULTATS:X,INEW
C              ----------
C
C
C     CE SOUS-PROGRAMME CALCULE LES VARIABLES LIEES A PARTIR DES VARIABLES
C     LIBRES PAR LA METHODE DE NEWTON.
C
C     PARAMETRES:
C     -----------
C
C     X:TABLEAU SUR LEQUEL S EFFECTUE L ELIMINATION.
C     NC:NOMBRE DE CONTRAINTES UTILES.(BILATERALES+UNILATERALES SATUREES)
C     IVL:TABLEAU ENTIER DEFINISSANT LE TYPE DE CHAQUE VARIABLE:
C              *SI IVL(I)=1     X(I) EST LIEE.
C              *SI IVL(I)=0     X(I) EST LIBRE.
C     ITERMA:NOMBRE MAXIMUN D'ITERATIONS TOLEREES DANS LA METHODE D'ELIMINATION
C     NUMERIQUE.
C     EPSCON:DONNE LA PRECISION DE L ELIMINATION.LE SYSTEME DES CONTRAINTES
C     EST VERIFIE A EPSCON PRES.
C     INEW:INDICATEUR D INCIDENT:
C                         *SI INEW=0 L ELIMINATION NUMERIQUE EST CORRECTE.
C                         *SI INEW=1 IL Y A INCIDENT
C
C     SOUS-PROGRAMMES APPELES:
C     ------------------------
C
C     COSA      DEPCO      MCADIR
C
C
      SUBROUTINE CALVA(X)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON ITERMA
      COMMON/A/C(15),D(15),Z(15),AL(15),XMAX(15),IVL(15)
      COMMON/B/TE,PASIN,EPSCON,EPS,FMAX
      COMMON/C/INEW,NPLUSP,NC
      DIMENSION X(1)
      DIMENSION G1(3,15),F(3),U1(3),U2(3),U3(3),IP(3),U(3,3)
      IT=0
      IF(NC.EQ.0)GOTO8
    1 CALL COSA(X,F)
      CALL DEPCO(X,G1)
      M=0
      DO 3 I=1,NPLUSP
      IF(IVL(I).EQ.0)GOTO3
      M=M+1
      DO 2 J=1,NC
    2 U(J,M)=G1(J,I)
    3 CONTINUE
      IT=IT+1
      IF(IT.LE.ITERMA)GOTO5
    4 INEW=1
```

```
      RETURN
    5 CALL MCADIR(U,F,3,NC,NC,U1,U2,U3,IP)
      IF(IP(1).NE.0)GOTO4
      M=0
      DO 6 I=1,NPLUSP
      IF(IVL(I).NE.1)GOTO6
      M=M+1
      X(I)=X(I)-F(M)
    6 CONTINUE
      CALL COSA(X,U1)
      DO 7 I=1,NC
      IF(DABS(U1(I)).LT.EPSCON)GOTO7
      GOTO1
    7 CONTINUE
    8 INEW=0
      RETURN
      END
```



```
C
C     CE SOUS-PROGRAMME CALCULE LES VALEURS DES CONTRAINTES EN X ET LES RANGE
C     DANS LE TABLEAU G.
C
      SUBROUTINE COSA(X,G)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DIMENSION X(1),G(1)
      G(1)=-X(1)*X(1)-X(2)*X(2)-X(3)*X(3)-X(4)*X(4)-X(1)+X(2)-X(3)
     1+X(4)+8.D0
      G(2)=-2.D0*X(1)*X(1)-X(2)*X(2)-X(3)*X(3)-2.D0*X(1)+X(2)+X(4)+5.D0
      RETURN
      END
```

```
C                          --------
C                          !XREDUI!
C                          --------
C
C               DONNEES:PASIN,IVL,XMAX,D,NC,EPS,NPLUSP
C               --------
C
C               RESULTATS:XMAX,FMAX,INEW,TE
C               ----------
C
C
C     CE SOUS-PROGRAMME CHERCHE SUR LA DIRECTION DE MONTEE DEFINIE PAR LE
C     GRADIENT REDUIT UN VECTEUR MEILLEUR QUE XMAX,DERNIER ITERE OBTENU.
C         EN CAS DE SUCCES,IL RECHERCHE ENSUITE LE PAS OPTIMUN H* SELON LA
C     TECHNIQUE DECRITE DANS LA PREMIERE PARTIE.
C
C
C     PARAMETRES:
C     -----------
C     PASIN:VALEUR DU PAS DE LA METHODE DU GRADIENT EN DEBUT DE PROGRAMME.
C     IVL:TABLEAU INDIQUANT LA NATURE DES DIVERSES VARIABLES.
C               *SI IVL(I)=0      X(I)EST LIBRE.
C               *SI IVL(I)=1      X(I) EST LIEE.
C     XMAX:TABLEAU CONTENANT EN DEBUT DE PROGRAMME LES COORDONNEES DU VECTEUR
C          INITIAL ET EN FIN DE PROGRAMME CELLES DU VECTEUR ITERE.
C     D:TABLEAU DU GRADIENT REDUIT.
C     NC:NOMBRE DE CONTRAINTES.
C     EPS:VALEUR MINIMUN DU PAS TOLEREE.
C     NPLUSP:NOMBRE DE VARIABLES.
C     FMAX:VALEUR DE LA FONCTION AU POINT XMAX.
C     INEW:INDICATEUR D INCIDENT:
C               *SI INEW=0      IL Y A AMELIORATION.
C               *SI INEW=1      IL Y A INCIDENT. IL EST IMPOSSIBLE D AMELIORER
C                               LE DERNIER ITERE.
C     TE:VALEUR DU PAS UTILISEE POUR LA CALCUL DU NOUVEAU VECTEUR XMAX.
C        CETTE VALEUR SERA UTILISEE PAR L ACCELERATION DE CONVERGENCE.
C
C
C     SOUS-PROGRAMMES APPELES:
C     ------------------------
C     CALVA      F
C
C

      SUBROUTINE XREDUI
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON ITERMA
      COMMON/R/TE,PASIN,EPSCON,EPS,FMAX
      COMMON/C/INEW,NPLUSP,NC
      COMMON/A/L(15),D(15),X(15),AL(15),XMAX(15),IVL(15)
      TE=PASIN
    1 M=0
      DO 2 I=1,NPLUSP
      IF(IVL(I).EQ.1)GOTO2
      M=M+1
      X(I)=XMAX(I)+TE*D(M)
    2 CONTINUE
      IF(NC.EQ.0)GOTO4
```



```
      CALL CALVA(X)
      IF(INEW.EQ.0)GOTO4
    3 TE=TE/2.D0
      IF(TE.GT.EPS)GOTO1
      INEW=1
      RETURN
    4 FGR=F(X)
      IF(FGR.LT.FMAX)GOTO3
      EE=2.*TE
      M=0
      DO 5 I=1,NPLUSP
      IF(IVL(I).EQ.1)GOTO5
      M=M+1
      AL(I)=XMAX(I)+EE*D(M)
    5 CONTINUE
      IF(NC.EQ.0)GOTO6
      CALL CALVA(AL)
      IF(INEW.NE.0)GOTO10
    6 F2=F(AL)
      AA=FMAX-2.*FGR+F2
      IF(AA.GE.(0.D0))GOTO9
      HET=-(TE/2.D0)*(-3.D0*FMAX+4.D0*FGR-F2)/AA
      M=0
      DO 7 I=1,NPLUSP
      IF(IVL(I).EQ.1)GOTO7
      M=M+1
      C(I)=XMAX(I)+HET*D(M)
    7 CONTINUE
      IF(NC.EQ.0)GOTO8
      CALL CALVA(C)
      IF(INEW.EQ.1)GOTO9
    8 F3=F(C)
      IF(FGR.LE.F3)GOTO15
    9 IF(FGR.LE.F2)GOTO13
   10 DO 11 I=1,NPLUSP
   11 XMAX(I)=X(I)
      FMAX=FGR
   12 INEW=0
      RETURN
   13 DO 14 I=1,NPLUSP
      X(I)=AL(I)
   14 XMAX(I)=X(I)
      FMAX=F2
      TE=EE
      GOTO12
   15 IF(F2.GT.F3)GOTO13
      DO 16 I=1,NPLUSP
      X(I)=C(I)
   16 XMAX(I)=C(I)
      FMAX=F3
      TE=HET
      GOTO12
      END
```

```
C                          -----------
C                          !XETM(XET)!
C                          -----------
C
C              DONNEES:XET,NPLUSP,IVL,XMAX
C              --------
C
C              RESULTATS:XET,XMAX,FMAX,INEW
C              ----------
C
C
C
C     CE SOUS-PROGRAMME ,COMPLEMENT DU SOUS-PROGRAMME CALXET REPREND
C     LE VECTEUR XET DE CE DERNIER,LE REND COMPATIBLE AVEC LES CONTRAINTES
C     ET VERIFIE QUE F(XET) AMELIORE LE DERNIER ITERE OBTENU.
C
C     PARAMETRES:
C     -----------
C     XET:TABLEAU X*
C     NPLUSP:NOMBRE DE VARIABLES.
C     IVL:TABLEAU ENTIER INDIQUANT LA NATURE DES DIVERSES VARIABLES.
C              *SI IVL(I)=0   X(I) EST LIBRE.
C              *SI IVL(I)=1   X(I) EST LIEE.
C     XMAX:TABLEAU CONTENANT LE DERNIER ITERE.
C     FMAX:VALEUR DE LA FONCTION EN XMAX.
C     INEW:INDICATEUR D INCIDENT.
C              *SI INEW=0 , LE PROGRAMME REMPLACE XMAX ET FMAX
C                         PAR XET ET F(XET)
C              *SI INEW=1 ,IL Y A RETOUR AU PROGRAMME APPELANT SANS
C                         AUCUNE MODIFICATION.
C
C     SOUS-PROGRAMMES APPELES:
C     ------------------------
C     CALVA      F
C
C
      SUBROUTINE XETM(XET)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON ITERMA
      COMMON/B/TE,PASIN,EPSCON,EPS,FMAX
      COMMON/C/INEW,NPLUSP,NC
      COMMON/A/C(15),D(15),X(15),AL(15),XMAX(15),IVL(15)
      DOUBLE PRECISION XET(1)
      DO 1 I=1,NPLUSP
      IF(IVL(I).EQ.0)GOTO1
      XET(I)=XMAX(I)
    1 CONTINUE
      CALL CALVA(XET)
      IF(INEW.NE.0)RETURN
      FE=F(XET)
      IF(FE.GT.FMAX)GOTO2
      INEW=1
      RETURN
    2 DO 3 I=1,NPLUSP
      X(I)=XET(I)
    3 XMAX(I)=X(I)
      FMAX=FE
      INEW=0
      RETURN
      END
```

NOM DE L'ETUDIANT : ROCHER Marc

NATURE DE LA THESE : DOCTORAT DE SPECIALITE

VU, APPROUVE

& PERMIS D'IMPRIMER

NANCY, le 14 novembre 1975

LE PRESIDENT DE L'UNIVERSITE DE NANCY I

J.R. HELLOT